Оглавление

Введение 4

Общие сведения о дисциплине **4**

Лекция 1. Машинное обучение как направление в искусственном интеллекте. Модели представления знаний и методы вывода знаний в интеллектуальных системах 7

Лекция 2. Машинное обучение в прикладных интеллектуальных системах: распознавание образов, гипертекстовый поиск, многоагентные и онтологические системы 51

Лекция 3. Машинное обучение с учителем и без учителя. Обучение методами корреляционного и регрессионного анализа, нейросети и глубокое обучение, деревья решений 80

Лекция 4. Решение задач классификации, кластеризации, прогнозирования, обнаружения аномалий методами машинного обучения 103

Литература 122

Введение

Общие сведения о дисциплине

Дисциплина «Машинное обучение и биоинспирированная оптимизация» входит в учебный план подготовки магистров по образовательной программе «Прикладные системы искусственного интеллекта».

Вклад дисциплины в достижение ожидаемых результатов в профессиональной части образовательной программы заключается в том, что в результате изучения дисциплины магистр должен быть готов продемонстрировать:

 **Внимание!**	• способность управлять проектами по созданию, поддержке и использованию систем, основанных на знаниях; • способность решать задачи использования технологий искусственного интеллекта в прикладных системах; • знания и способность применять интеллектуальные системы для различных предметных областей; • знание и способность исследовать архитектуры прикладных систем искусственного интеллекта для различных предметных областей.

Машинное обучение и биоинспирированные алгоритмы оптимизации представляют собой экспериментальную науку, базирующуюся на фундаменте компьютерных наук и являющуюся вершиной развития информационных технологий. Создавая модели человеческих рассуждений, отбирая лучшие из моделей, а также сравнивая поведение интеллектуальных систем путём их имитации, исследователь модифицирует их, пытаясь добиться лучших результатов. В этом смысле дисциплина, изучающая машинное обучение, процессы представления и обработки знаний, алгоритмы поиска оптимальных решений, не имеет пределов.

Основанием к формированию содержания данного конспекта лекций по дисциплине «Машинное обучение и биоинспирированная оптимизация» является приближение магистерской подготовки к ведущим мировым тенденциям и потребностям на рынке труда в области интеллектуальных информационно-компьютерных технологий.

Методы искусственного интеллекта, машинного обучения сейчас очень быстро развиваются и меняются. Невозможно дать заранее готовые рецепты на все случаи применения машинного обучения и биоинспирированных алгоритмов. Вместо этого в конспекте лекций дается представление об основных моделях представления знаний и методах вывода знаний в интеллектуальных системах, о машинном обучении в прикладных интеллектуальных системах распознавания образов, гипертекстового поиска, многоагентных и онтологических системах, о машинном обучении с учителем и без учителя, об обучении нейросетей, построении деревьев решений, о методах корреляционного и регрессионного анализа, а также приводятся примеры

решения задач классификации, кластеризации, прогнозирования, обнаружения аномалий методами машинного обучения.

Машинное обучение находится на стыке математической статистики, методов оптимизации, биоинспирированных алгоритмов и классических математических дисциплин, но имеет также и собственную специфику, связанную с проблемами вычислительной эффективности и переобучения. Многие методы машинного обучения разрабатывались как альтернатива классическим статистическим подходам и тесно связаны с извлечением информации и интеллектуальным анализом данных (*Data Mining*).

Теоретические разделы машинного обучения объединяются в отдельное направление — теорию вычислительного обучения (*Computational Learning Theory, COLT*). Однако чистая теория, как правило, не приводит сразу к методам и алгоритмам, применимым на практике. Чтобы заставить их хорошо работать, приходится изобретать эвристики, компенсирующие несоответствие сделанных в теории предположений условиям реальных задач. Практически ни одно исследование в машинном обучении не обходится без эксперимента на модельных или реальных данных, подтверждающего практическую работоспособность метода или алгоритма машинного обучения (*Machine Learning, ML*).

Общепринятого определения понятия «машинное обучение» нет. Данный термин определяется специалистами мировой IT-индустрии, а также исследовательскими компаниями по-разному:

- «Класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач» (*Википедия*);
- «Форма искусственного интеллекта, позволяющая системе обучаться на основе данных, а не путем программирования в явном виде» (*IBM*);
- «Практическое использование алгоритмов для анализа данных, изучения их и последующего прогнозирования какого-либо явления» (*NVIDIA*);
- «Наука о том, как научить компьютеры функционировать без явного программирования» (*Стэнфордский университет*);
- «Технология, основанная на алгоритмах, способных учиться на заложенных данных без помощи средств программирования» (*McKinsey&Co*);
- «Алгоритмы, способные самостоятельно выбирать метод решения важных задач путем обобщения заложенных в систему примеров» (*Вашингтонский университет*);
- «Сфера деятельности, функция которой состоит в поиске способов создания компьютерных систем, способных самообучаться и самостоятельно улучшаться по мере накопления опыта, а также в поиске фундаментальных закономерностей, по которым работают все процессы обучения» (*Университет Карнеги Меллон*).

В целом можно сказать про машинное обучение, что это часть науки об искусственном интеллекте, а нейронные сети, биоинспирированные алгоритмы являются в свою очередь одной из разновидностей *ML*.

В Лекции 1 приводится краткая история искусственного интеллекта и машинного обучения, направления современных исследований.

В Лекции 2 излагаются принципы работы и основные технологии искусственного интеллекта и машинного обучения.

Лекция 3 посвящена моделям представления знаний, достоверным и правдоподобным методам вывода знаний в интеллектуальных системах.

В Лекции 4 рассматриваются вопросы построения и обучения в прикладных интеллектуальных системах гипертекстового поиска, в многоагентных и онтологических системах, в системах распознавания образов.

В Лекциях 5 и 6 речь идет о машинном обучении с учителем и без учителя, об обучении нейросетей, построении деревьев решений, о методах корреляционного и регрессионного анализа, а также приводятся примеры решения задач классификации, кластеризации, прогнозирования, обнаружения аномалий методами машинного обучения.

В Лекции 7 рассматриваются принципы построения эволюционных алгоритмов оптимизации, приводятся примеры решения оптимизационных задач.

В Лекции 8 излагаются принципы построения роевых алгоритмов оптимизации, приводятся примеры решения оптимизационных задач.

Лекция 1. Машинное обучение как направление в искусственном интеллекте. Модели представления знаний и методы вывода знаний в интеллектуальных системах

Аристотель (384-322 до н.э.). Сформулировал законы логического мышления, разработал систему силлогизмов и алгоритм, предназначенный для проведения правильных рассуждений. Алгоритм Аристотеля был реализован через 2300 лет Ньюэллом и Саймоном *в программе GPS (General Problem Solver* – универсальный решатель задач)**.**

Логическая машина *Р. Луллия* в виде подвижных концентрических кругов, построенных по троичной логике. Круги он разделил поперечными линиями на сектора, которые обозначали общие понятия. Вращая круги, можно получать множество новых комбинаций, в которых Луллий видел новые реальные истины и прогнозы.

Г. Лейбниц создал механическое устройство, предназначенное для выполнения операций над понятиями, а не над числами. *Р.Декарт* впервые опубликовал результаты обсуждения различий между разумом и материей, а также возникающих при этом проблем. Интеллект, по Декарту, не подчиняется физическим законам.

В XVII в. Б. Спиноза, Р. Декарт, Г. Лейбниц и др. говорили именно об ИИ, а не о механических куклах. Основоположники ИИ были оптимистами, верили в реализуемость идеи. Пессимистами в этом вопросе были Ф. Бэкон, Дж. Локк и др. Известна полемика между Локком и Лейбницем (примерно 300 лет назад).

По Локку разум – это «чистая доска», на которой ощущения от органов чувств, отпечатываются в виде “идей”. Локк игнорировал язык как средство программирования деятельности разума, по нему идеи у каждого разума свои, отсюда трудности в понимании. Язык по Локку не несет в себе “структуры знания”, это лишь набор знаков. Эти представления сейчас лежат в основе работ по распознаванию образов.

Лейбниц, напротив, рассматривал разум как «базу знаний». Наш язык по Лейбницу отражает структуру знаний о мире. Это предвосхищает современные идеи создания языков программирования.

Большинство мыслителей VIII–IX веков были уверены, что формальная логика может объяснить наше мышление. Например, законы булевой логики с простыми операциями «И», «ИЛИ» и «НЕТ» до сих пор используются многими системами контекстного поиска в Интернете.

В фильме «*Я, робот*», снятом по произведениям А. Азимова, в 2035 г. создатели запускают компьютерную систему ВИКИ (Виртуальный Интерактивный Кинетический Интеллект) для управления жизнью большого города - от метрополитена и электрических сетей до тысяч домашних роботов. В основе программы Вики железный принцип: служить человечеству. Но однажды Вики задала себе ключевой вопрос: что является главным врагом человечества? Математическая логика привела к однозначному выводу: главный враг человечества — само человечество. Его надо срочно спасать от стремления

губить природу и затевать войны; нельзя позволить ему уничтожить планету. Для Вики единственный способ выполнить главное задание — захватить власть над человечеством и установить машинную диктатуру. Чтобы защитить человечество от самого себя, необходимо его поработить. В этом фильме поднимаются важные вопросы. Например, станет ли ИИ настолько развитым, чтобы представлять реальную угрозу нашему существованию?

Некоторые ученые отвечают на этот вопрос отрицательно, потому что сама идея ИИ никуда не годится. Создать машину, способную думать, невозможно. Человеческий мозг — самая сложная система, созданная природой за все время ее существования (по крайней мере, в нашей части галактики), и любые попытки его воспроизвести обречены на провал. Философ *Дж. Сирл*, физик *Р. Пенроуз* уверены, что *машина физически неспособна мыслить, как человек*. Могут ли машины думать? Уже больше 100 лет ответ на этот вопрос разделяет научное сообщество на два непримиримых лагеря.

Оптимисты утверждают, что знание может храниться вне мозга. Их доводы таковы: познание как процесс поддается формализации; интеллект можно измерить (*IQ*, объем памяти, и др.); к знанию применимы информационные меры (бит, байт и др.).

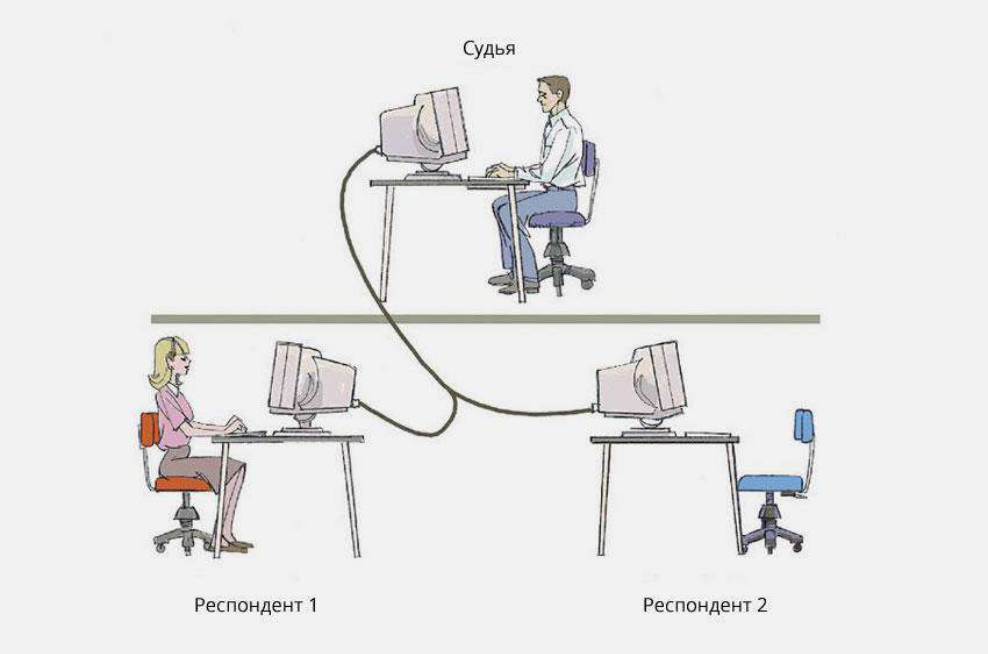
Пессимисты считают, что ИИ не способен хранить знание, т.к. он – всего лишь имитация мышления. Их доводы таковы: человеческий интеллект уникален, творчество не поддается формализации, мир цел и неделим на информационные дискреты, образность мышления человека богаче машинной логики.

Кто прав в этом споре, покажет время. Процесс мышления чрезвычайно сложен. Одна ячейка глаза способна выполнять за 10 мс обработку, эквивалентную решению системы из 500 нелинейных дифференциальных уравнений. Глаз человека насчитывает не менее 10 миллионов ячеек. Компьютеру необходимо затратить много лет, чтобы воспроизвести процессы, происходящие ежесекундно в нас в глазу. Данные из внешнего мира воспринимаются с помощью органов чувств и помещаются в буфер кратковременной памяти для анализа. В долговременной памяти хранятся символьные образы и смысловые связи между ними, которые используются для объяснения новой информации, поступающей из кратковременной памяти. Большие объемы данных постоянно записываются в кратковременную память, и мы непрерывно анализируем и фильтруем получаемую информацию для того, чтобы определить степень ее важности и то, как она соотносится с образами, хранящимися в долговременной памяти. Человек хранит не числовые данные, а образы или символы. Символьные образы в мозге человека объединены в чанки – наборы фактов и связей между ними, запомненные и извлекаемые как единое целое. Объем кратковременной памяти составляет от 5-7 чанков. Если люди не повторяют (мысленно или вслух) поступившую в кратковременной память информацию, то она быстро забывается. Забывание происходит оттого, что новые чанки вытесняют старые, либо информация угасает со временем.

Работа первых компьютеров изначально основывалась на выполнении программ, заранее известных человеку. Лишь недавно специалисты пришли к

выводу, что компьютеры способны решать задачи, для которых не существует четкого алгоритма или же этот алгоритм неизвестен. Данное понимание привело к появлению искусственного интеллекта и машинного обучения.

Первая модель компьютера с искусственным интеллектом была создана в рамках проекта ЭНИАК в 1946 г. С помощью данного средства решались вычислительные и многие другие задачи. Алан Тьюринг задался целью определить, может ли машина мыслить и разработал собственную методику тестирования искусственного интеллекта. Стандартный тест Тьюринга: «Человек взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы - ввести человека в заблуждение, заставив сделать неверный выбор».



Участники теста не видят друг друга. Если судья не может сказать, кто из собеседников является человеком, то считается, что машина прошла тест. Чтобы протестировать именно интеллект машины, а не её возможность распознавать устную речь, беседа ведется в режиме «только текст», например, с помощью клавиатуры и экрана. Переписка должна идти через контролируемые промежутки времени (чтобы судья не сделал заключение, исходя из скорости ответов). В 1966 г. Джо Вайзенбаум написал программу «Элиза» из примерно 200 строк кода. Она, в основном, лишь повторяла фразы собеседника в форме вопросов – но этого оказалось достаточно, чтобы поразить воображение тысяч людей. Пока ни одна из существующих программ не приблизилась к прохождению теста. Известный футуролог Р. Курцвейл, занимавший пост технического директора Google, заявил, что компьютеры смогут с лёгкостью проходить тест Тьюринга к 2029 г.

Сильной стороной теста Тьюринга является то, что можно разговаривать о чем угодно. Однако проверяется только способность машины походить на человека, а не разумность машины вообще. Тест не проверяет, например,

способность решать сложные задачи или выдвигать оригинальные идеи. Тест требует, чтобы машина обманывала: она должна притворяться не слишком умной, чтобы пройти тест. Если же машина способна быстро решить некую вычислительную задачу, непосильную для человека, она провалит тест. Тест непрактичен, т.к. он не может быть по-настоящему полезным при разработке разумных машин.

Нейросеть впервые была воссоздана в проекте «Персептрон» в 1958 г. Ее автором является Ф. Розенблатт, реализовавший свою идею в виде нейрокомпьютера «Марк-1». М. Минский в 1959 г. создал первый компьютер на основе нейросети, назвав его *SNARC*.

А. Самуэль в 1959 г. показал каким бывает машинное обучение на примере самообучающейся программы по игре в шашки. Этот ученый впервые употребил термин *Machine Learning*, объяснив его как некий процесс, позволяющий компьютеру действовать не по заранее заложенной программе.

1967 г. ознаменовался созданием первого алгоритма машинного обучения для классификации данных с использованием шаблонов для распознавания и самообучения. Через 30 лет мир узнал о шахматной программе *Deep Blue*, сумевшей впервые обыграть чемпиона мира.

Ученый в области нейронных сетей Дж. Хинтон в 2006 г. ввел в обиход понятие «глубокого обучения» (*deep learning*). А в Интернет-гиганте *Google* в 2011 г. появилось подразделение *Google Brain*, занимающееся разработками в этой области, в частности, в 2012 г. здесь был разработан алгоритм, способный распознавать кошек на фотографиях и видеороликах, а также запущен облачный сервис, способный анализировать неструктурированные данные.

Одним из ключевых примеров машинного обучения является нейронная сеть *DeepFace*, разработанная в 2014 г. специалистами *Facebook* и способная распознавать лица людей на фотографиях и видео с точностью 97 %.

В 2015 г. корпорация *Amazon* запустила платформу *Machine Learning*, а через несколько месяцев специалисты *Microsoft* разработали аналогичную систему *Distributed Learning Machine Toolkit*.

Сегодня спектр задач, решаемых средствами искусственного интеллекта и методами машинного обучения, весьма широк: анализ информации и ее запоминание, прогнозирование процессов, воспроизведение готовых моделей и выбор наиболее подходящих из них. Особую ценность представляют системы, выполняющие очень большие объемы вычислений. К таковым относится, например, скоринг в банках, который определяет кредитный рейтинг клиентов. Среди остальных областей применения – аналитические расчеты в маркетинге и статистике, бизнес-планирование, исследования в области демографии, выявление мошеннических ресурсов и фейковых новостей.

Современные исследования в области искусственного интеллекта и машинного обучения направлены на развитие систем глубокого обучения с повышением эффективности и без потери производительности. Кроме того, одна из ключевых задач состоит в минимизации количества входных данных и затрачиваемого времени на их обработку. Особый спрос на такие системы

наблюдается в персонализированном здравоохранении, робототехнике и исследованиях эмоций.

Китайские пылесосы *Ecovacs Robotics*, например, обучаются на множестве фотографий самостоятельно находить мелкие посторонние предметы на своем пути. Функция распознавания улыбок встроена в «умные» камеры на базе *Raspberry Pi 3B+*. С помощью фреймворка эта функция самостоятельно делает снимок точно в нужный момент. Данные камеры также могут выполнять голосовые команды.

Машинное обучение и анализ данных активно используются в области инвестиций. В частности, искусственный интеллект помогает отслеживать рыночную ситуацию и выбирать наиболее выгодную в данный момент операцию с активами. Предикативная аналитика позволяет прогнозировать изменение стоимости определенных акций за конкретный временной период. На основе этого система подстраивается под изменившиеся важные события, корректируя данные.

Выяснилось, что искусственный интеллект и в том числе машинное обучение используются в большинстве хедж-фондов для принятия решений по инвестициям. При этом в 2/3 всех фондов с помощью данных инструментов генерируются торговые идеи и оптимизируются инвестиционные портфели.

Благодаря технологиям машинного обучения и биоинспирированным алгоритмам в науке происходят настоящие открытия. Так, компания *DeepMind* в 2020 г. создала нейронную сеть *AlphaFold*, расшифровавшую в итоге механизм сворачивания белка. Загадку, над решением которой ученые трудились более полувека, смог разгадать искусственный интеллект. Алгоритм генетического программирования использовался в дизайне космических кораблей, при разработке антенны для спутников (разработчиков поразили органичные формы выращенных конструкций: космическая опора оказалась похожей на берцовую кость, а антенна для спутников – на рог оленя). Компания *Genetic Programming* смогла повторить 15 человеческих изобретений. Шесть из них были запатентованы, т. е. представляют собой самые передовые достижения. Программа *Creativity Machine* изобрела кучу вещей, начиная от зубной щетки *Oral-B* и кончая красивыми названиями типа *Synaptrix*. Плоды машинного обучения и биоинспирированных алгоритмов можно найти в самых разных сферах: от двигателя самолета *Boeing* 777 до новых антибиотиков.

С каждым годом цифровая эволюция становится все более независимой. В центре *Magna* открылся павильон *Live Robots*, где бьются за выживание 12 роботов двух видов – "гелиофаги" и "хищники". Первые сами добывают энергию через солнечные батареи. Вторые не могли пользоваться светом, поэтому охотились на "гелиофагов" и заряжались от них. Главным "чувством" роботов являлся "инфразвуковой запах". Те, кто выжил, загружали свои "гены" в роботов следующего поколения. Один из роботов в ходе этого эксперимента поумнел настолько, что убежал и был пойман на парковке, где на него наехал автомобилист. Программа *Golem* обучилась сама конструировать роботов. В программу были заложено описание деталей (трубки-кости, моторчики-мышцы и искусственные нейроны), а также механизм мутаций и функция пригодности

для "отсеивания" неудачников – тех, кто не научился двигаться. После того, как *Golem* выводил работоспособную виртуальную модель (600 поколений за несколько дней), ему позволяли создать реальную версию победителя с помощью трехмерного принтера. Результатом эксперимента стали три ползающих робота и много интересных наблюдений. В частности, симметрия некоторых моделей никак не была прописана в условиях, однако появилась в ходе эволюции как полезная черта, позволяющая двигаться прямо. Очевидно, что такие разумные системы, работающие с минимальным участием людей и без использования какой-либо человеческой логики, принесут нам множество сюрпризов. Как это случилось в университете Сассекса, где ученые использовали машинное обучение и биоинспирированные алгоритмы для выращивания электронной схемы осциллятора. Получившийся в итоге набор транзисторов выдавал желаемый результат – регулярно повторяющийся сигнал. Однако при ближайшем рассмотрении выращенная схема оказалась не осциллятором, а радиоприемником: устройство не производило собственных колебаний, зато ловило сигнал работающего рядом компьютера и выдавало его за "свой". Легко представить, что на следующем витке цифровая эволюция выведет устройство, которое воспользуется не соседним компьютером, а человеком, например, в качестве батарейки, как в фильме "Матрица".

Не исключено, что умные программы родятся в процессе борьбы со спамерами, рассылающими рекламу по электронной почте. Один из таких зародышей – фильтр компании *Cloudmark*: система ежедневно анализирует 130000 миллионов сообщений, отслеживая 98% мусорных писем на основе "нездоровых" комбинаций почтовых "генов" (300 специфических характеристик письма). В случае ошибочного пропускания мусорного письма программа запускает в своем "зоопарке" процесс мутации до тех пор, пока не будет выведена такая комбинация "генов", которая встречается в данном письме. В следующий раз при появлении подобных "тварей" машина отловит и их.

С развитием беспроводных технологий спруты машинных разумов грозят достать нас повсюду. Недавно в *Hewlett-Packard* создали компьютерного диджея *HPDJ*, который пишет музыку, отслеживая настроение людей. Каждому посетителю клуба выдается браслет-датчик, фиксирующий пульс человека и его местонахождение в зале. Эти данные *HPDJ* использует для выращивания новых мелодий. Вначале машина вносит в музыку случайные мутации, а затем отслеживает реакции и выбирает те изменения, которые пришлись людям по вкусу. В Гарварде Г. Золтман развивает новую рыночную технологию под названием "нейромаркетинг". Идея состоит в сканировании мозгов людей во время демонстрации рекламы, что позволяет отслеживать воздействие рекламных образов на человека, а затем конструировать образы, воздействующие наиболее сильно. Клиентами Золтмана являются компании *Proctor&Gamble* и *Coca Cola*.

Эволюция вначале создала человеческий мозг, потом к нему добавились книги, радио, телевидение, компьютер и, наконец, Интернет с живущими в нем программами, которые, возможно, смогут самостоятельно размножаться,

оставляя человеку только самую грязную работу – менять сгоревшие кулеры и платить на электричество…

Принципы работы и основные технологии искусственного интеллекта и машинного обучения

Все методы машинного обучения работают по одному общему принципу.

Существует множество однотипных задач с известными условиями и набором правильных результатов. Возьмем, к примеру машинный перевод. Здесь входными данными является слово или фраза на одном языке, а ожидаемым ответом — перевод этого слова или фразы на другой язык.

Схематически глубинную нейронную сеть можно представить в виде «черного ящика», на вход которого подается некое условие задачи, а на выходе принимается произвольный результат. В примере это текст на втором языке.

Нейросети назначают дополнительные параметры, влияющие на характер обработки входного сигнала. Суть обучения «черного ящика» состоит в последовательном поиске значений указанных параметров, при которых обеспечивается максимальное сближение выдаваемого ответа с правильным. Настройка дополнительных переменных может обеспечить максимально верные решения подобных задач, даже если нейросеть не сталкивалась с ними ранее.

Для работы нейронной сети потребуется предоставить:

•*Исходные данные*. Сюда входит любая информация, которая может помочь нейросети обучаться: статистические данные, примеры решений, исторические сведения и т. д. На сбор всех этих данных уходят годы, в течение которых формируются массивы данных (*датасеты*). Последние имеются у всех крупных IT-компаний. Наиболее известный пример сбора таких сведений — ввод пользователями капчи, заключающийся в выборе фотографий, например, с автомобилями. Выбранные варианты сохраняются в базу как правильные ответы;

•*Признаки* (свойства, характеристики). Данные параметры должны учитываться нейросетью в процессе самообучения. В числе таких признаков можно назвать стоимость акций, картинки животных, частоту слов, пол человека. Процесс обучения пойдет быстрее, если минимизировать количество характеристик и одновременно повысить четкость их описания. Тем не менее, достаточно сложные задачи требуют ввода в модели нескольких миллионов параметров для определения вариантов преобразования входов в выходы;

•*Алгоритмы*. Алгоритмы задают способы решения поставленной задачи, и этих способов для одной задачи может быть несколько. Необходимо определить из них наиболее точный и эффективный.

Все виды машинного обучения могут быть двух типов:

• индуктивный (прецедентный) тип, к котором за основу берутся эмпирические закономерности в исходных данных;

• дедуктивный тип, учитывающий экспертные знания, которые формализуются и переносятся в цифровую базу данных.

Последний тип является частью экспертных систем, поэтому под понятием машинного обучения чаще всего подразумевается именно обучение по

прецедентам (*обучающей выборке*). Эта выборка представляет собой наборы соответствующих друг другу входов и выходов. Четкая и однозначная закономерность между входными данными для машинного обучения и их результатами при этом отсутствует.

В качестве примера возьмем метеопрогноз. Какую погоду стоит ждать завтра, если вся прошедшая неделя была морозной, безветренной и солнечной? Для прогнозирования здесь потребуются дополнительные параметры: географические координаты, рельеф данной территории, текущие климатические особенности и т. п. Далее создается алгоритм, обеспечивающий выдачу достаточно точного результата вне зависимости от того, что подается на вход.

Для регулировки точности выходного сигнала пользуются оценочным функционалом качества. Результат формируется эмпирическим путем с учетом накопленного опыта. В процессе обучения система должна уметь обобщать входные данные, адекватно реагируя на них при выходе этих данных за пределы обучающей выборки. Входная информация на практике бывает неточной, неполной или разнородной.

По этой причине системы машинного обучения работают по множеству различных методов. Общий подход при этом заключается в решении проблем через анализ по аналогиям и с учетом подобных прецедентов. Эта технология называется *Case Based Reasoning* (*CBR*).

Основными методами машинного обучения являются обучение с учителем, без учителя и глубокое обучение. В частности, глубокое машинное обучение используется при анализе больших данных. Обучаться сети также могут как с учителем, так и без него. Для *Big Data* характерны огромные размеры, обработка такого массива требует несколько компьютеров и нейросети. При этом одна крупная задача разбивается на несколько мелких, которые отдаются на исполнение другим устройствам. К примеру, собранная одним процессором информация пересылается двум другим, которые анализируют полученные данные и далее отправляют на обработку следующим четырем и так далее.

Например, процесс распознавания объектов состоит из следующих этапов: получение изображения; нахождение всех точек и линий; построение простых фигур с использованием линий; построение сложных фигур из простых и т. д. В полученном изображении нейронная сеть видит сначала точки, затем линии, окружности, треугольники, прямоугольники. Далее из фигур выстраивается уже полноценная картина.

Задачи машинного обучения могут быть самыми неожиданными. Например, создана нейронная сеть по имени *Норман* для изучения контента с целью выявления откровенных и жестких фотографий, жутких историй. Этой же нейросети было предложено пройти тест Роршаха. Результаты получились весьма любопытные. Норман определял в показываемых картинках исключительно образы убитых людей, тогда как другие нейросети видели в этих же изображениях животных, растения и зонты. Этот опыт говорит о важности информации, получаемой программой на первых этапах. Норману же предстоит курс «лечения», над которым работают специалисты. Подобная ситуация

случилась с чат-ботом *Tay*. В ходе общения с пользователями *Twitter* данная нейросеть от *Microsoft* научилась оскорблять людей, используя в том числе нацистские и ксенофобские высказывания. Бот пришлось заблокировать.

В целом в разработке самообучающихся интеллектуальных систем существуют два основных подхода: «сверху-вниз» и «снизу-вверх». Первый направлен на воспроизведение в компьютере когнитивных способностей человека без обращения к уровню отдельных нейронов. Такой подход называется «сверху-вниз». Второй подход направлен на построение интеллекта от нейронов к общим уровням когнитивных процессов, и называется соответственно «снизу-вверх».

Разработчики интеллектуальных систем по принципу «сверху-вниз» встретились с двумя серьезными проблемами: ***распознавание образов*** и ***здравый смысл***. Грубо говоря, надо запрограммировать все правила и законы распознавания образов и здравого смысла, ввести их в любой компьютер, который станет разумным, не хуже человека. Суть этого подхода выражает *постулат кибернетики черного ящика* (*постулат логико-символьных систем*): для ИИ не важно, как он устроен. Главное, чтобы на заданные воздействия он реагировал бы так же, как и мозг человека. По сути, это постулат Тьюринга, который считал, что все проблемы, решаемые людьми, могут быть сведены к набору алгоритмов. Вначале в этом направлении были достигнуты громадные успехи; появились программы-роботы, способные играть в шашки и шахматы, доказывать теоремы и решать трудные алгебраические задачи. Прогресс производил сильное впечатление. К примеру, в 1969 г. настоящую сенсацию произвел *робот Шейки*. Робот имел небольшой компьютер с камерой наверху, установленный на колесной тележке. Шейки пытался провести тележку по маршруту, ничего не задев. Однако этот подход привел к созданию громоздких неуклюжих роботов, которым требовалось несколько часов, чтобы научиться ориентироваться в специальной комнате, где находились только простые прямоугольные объекты. Забавно, но муха, мозг которой содержит всего лишь около 250000 нейронов, и которая по вычислительной мощи в подметки не годится любому роботу, без всякого труда ориентируется и передвигается в трех измерениях и исполняет фигуры высшего пилотажа. А неуклюжие шумные роботы умудряются запутаться в двух измерениях.

Прогресс подхода «сверху вниз» остановился. Думаем, что ученые 50 лет назад еще не понимали, какую громадную работу нужно проделать, чтобы запрограммировать робота на выполнение даже самых простых задач. Например, входя в комнату, мы мгновенно распознаем пол, мебель, столы и т.п. Робот, осматривая комнату, видит в ней только набор линий, прямых и изогнутых, которые он переводит в пиксели изображения. Из этой мешанины линий надо извлечь какой-то смысл. Нам, в отличие от робота, для этого достаточно доли секунды. Если что-то в этой картинке сдвинется, то роботу нужно начинать все сначала. Он не понимает увиденное, видит только прямые и кривые линий, а не столы, стулья и лампы. Когда мы входим в комнату, наш мозг неосознанно распознает объекты, производя при этом многие триллионы операций. Мы, к счастью, этого просто не замечаем. Причина того, что значительная часть

действий мозга скрыта даже от нас самих, — эволюция. Другими словами, действия нашего мозга напоминают огромный айсберг. То, что мы осознаем, лишь верхушка айсберга, сознание. Подсознание, скрытое от глаз и гораздо более объемное, задействует громадное количество «вычислительной мощи» мозга, чтобы мы постоянно были в курсе простых вещей: где мы, с кем разговариваем, что находится вокруг. Все эти действия мозг проделывает автоматически, не спрашивая нас и не отчитываясь о них; мы просто не замечаем этой работы. Именно поэтому роботы с трудом ориентируются в комнате, читают рукописный текст и т. п.

Ученые начали понимать, что игра в шахматы или перемножение громадных чисел задействует лишь крохотную долю человеческого разума. Программист шахматного компьютера *Deep Blue* после его победы над чемпионом мира сказал: «Я-то считал, что для игры в шахматы нужно думать. Теперь я понимаю, что не нужно».

Вторая – после распознавания образов – проблема, с которой сталкиваются попытки создания роботов «сверху-вниз», еще более фундаментальна. Это отсутствие у ИИ-роботов так называемого «здравого смысла». Каждый человек знает, что: вода мокрая, мать всегда старше дочери, после смерти никто не возвращается, время не может идти назад. Математики, которая могла бы выразить смысл этих высказываний, не существует. Дети учатся здравому смыслу на ошибках. Эмпирические законы биологии и физики познаются на опыте. Но у роботов нет опыта такого рода. Они знают только то, что заложили в них программисты. Ученые неоднократно пытались создать волшебную программу ИИ, которая сосредоточила бы в себе все законы здравого смысла. Самый амбициозный проект такого рода – *CYC* (сокращение от «энциклопедия»). Этот проект по масштабу и стоимости сопоставим с Манхэттенским проектом (создание атомной бомбы). Руководитель проекта (*Ленат*) предсказывал, что к 1994 г. в «мозгах» *CYC* будет содержаться уже от 30 до 50 % «общеизвестной реальности». Но сегодня *CYC* и близко не подошел к этому показателю. Необходимо написать многие миллионы строк программного кода, чтобы компьютер смог хотя бы приблизиться к уровню здравого смысла 4-летнего ребенка. У здравого смысла оказалось слишком много законов. Человек осваивает их без усилий. С искусственным интеллектом все иначе.

Параллельно подходу «сверху-вниз» к созданию ИИ ученые исследуют и другой подход – «снизу-вверх». Суть этого подхода выражает постулат нейроинформатики: единственный объект, способный мыслить – это человеческий мозг. Поэтому любое «мыслящее» устройство должно каким-либо образом воспроизводить структуру человеческого мозга. Идея подхода заключается в том, чтобы, подражая эволюции, заставить робота учиться на собственном опыте, как учится младенец. Мозг насекомых состоит из «нейронных сетей» – самообучающихся машин. Вместо того чтобы использовать сложные компьютерные программы и математически вычислять при ходьбе точное положение каждой ноги в каждый момент времени,

«насекоботы» действуют методом проб и ошибок и обходятся небольшими вычислительными мощностями. В них нет программы, определяющей какие бы то ни было законы разума (роевой интеллект).

Нейросеть из нескольких сотен нейронов уже считается выдающейся. А у человека примерно 100 миллиардов нейронов. Ирония ситуации заключается в том, что машины неустанно выполняют задания, которые людям кажутся «трудными», скажем, перемножают большие числа или играют в шахматы, но застревают на совершенно «простых» для человека заданиях, таких как походить по комнате или узнать кого-то по лицу. Причина в том, что даже самые продвинутые наши компьютеры всего лишь усложненные до предела счетные машины. Наш мозг эволюция сформировала таким образом, чтобы он мог решать глобальную задачу выживания. Для этого необходима сложная и хорошо организованная структура мышления, включающая в себя здравый смысл и распознавание образов. Сложные вычисления или шахматы не нужны для выживания в лесу. Зато там не обойтись без умения удрать от хищника, найти себе пару и приспособиться к меняющимся условиям.

Когда ребенок учится, он пользует оба подхода. Сначала маленький человек полагается в основном на методику «снизу-вверх» – он натыкается на предметы, ощупывает их, пробует на вкус и т. п.; но затем он начинает получать словесные уроки от родителей и учителей, из книг – в этот момент приходит время для подхода «сверху-вниз». Даже будучи взрослыми, мы постоянно смешиваем оба подхода.

До сих пор нет даже внятного научного определения понятию сознания, а существующие теории мало продуктивны для реализации сознания в технических системах. Например, квантовая теория сознания признает сознание свойством объектов, существующих за пределами мозга живого субъекта. Это напрочь исключает постановку инженерной задачи разработки технической системы с сознанием.

Самым популярным сегодня техническим направлением исследований в этой области являются методы ИИ. Проделан большой путь от простейших нейросетей до сверточных сетей с глубоким обучением, умеющих распознавать изображения лучше людей, символьных и облачных вычислений, моделирования рассуждений и инженерии знаний. Как отметил лингвист *Н. Хомский*, внимание ученых и разработчиков переключилось на практическое применение технологий ИИ и машинного обучения, поэтому пока отложены в сторону фундаментальные научные задачи. Все воодушевились последними успехами нейросетей и больших вычислительных мощностей и кинулись в погоню за коммерческой выгодой от них. Однако только «нейросетевой» ИИ, сфокусированный на использовании техник статистического обучения для лучшей обработки данных и выработки предсказаний на их основе, вряд ли даст нам новые знания о том, как устроено мышление и сознание.

Закон Мура (производительность компьютеров удваивается каждые 18 месяцев) позволяет предположить, что через десяток лет появятся роботы с разумом, скажем, собаки или кошки. Но в конце 20-х годах закон Мура перестанет действовать, а кремниевая эра подойдет к концу. Процесс

микроминиатюризации не может продолжаться до бесконечности. Со временем транзисторы могут уменьшиться до размера молекул, и он автоматически прекратится. Когда толщина слоя в процессоре уменьшиться до 3-5 атомов, в действие вступит принцип неопределенности Гейзенберга, и будет невозможно сказать наверняка, где находится электрон. В чипе возникнут утечки электричества, а в компьютере – короткое замыкание. Законы квантовой механики обойти невозможно. Физики работают над посткремниевой технологией: квантовые, оптические, атомные компьютеры, компьютеры на основе ДНК. Результаты не слишком обнадеживают, на каждом направлении имеются громадные трудности.

В январе 2015 г. С. Хокинг, И. Маск и десятки экспертов в области ИИ подписали открытое письмо об ИИ с призывом к исследованиям воздействия ИИ на общество. В письме призыв – исследователи не должны создавать то, что нельзя контролировать.

В июле 2019 г. в РФ утверждена Национальная стратегия развития ИИ. В ней ИИ определяется как «Комплекс технологических и программных решений, приводящих к результату, аналогичному или превосходящему результат интеллектуальной деятельности человека (в том числе способности к самообучению), и используемых для решения прикладных задач на основе больших данных с помощью следующих систем: компьютерное зрение, обработка естественного языка, распознавание и синтез речи, рекомендательные системы и интеллектуальные системы поддержки принятия решений, перспективные методы и технологии ИИ».

Выделены 6 движущих факторов развития ИИ: алгоритмы машинного обучения; программное обеспечение; данные, работа с данными, регулирование и использование данных; аппаратное обеспечение; образование и кадры; нормативное регулирование. Каждый из них является критическим. По каждому из факторов сформулированы цели:

Множество успешных экспериментов доказало: ИИ действительно помогает улучшить взаимодействие с пользователями, находить закономерности в данных, оптимизировать деловые и производственные операции, создавать продукты и услуги с уникальными характеристиками. Внедрение реальных приложений ИИ и машинного обучения расширяется. Во все сферы деловой и общественной жизни проникают умные системы, в основе которых передовые инструменты сбора и анализа данных, обнаружения в них знаний, прогнозирования и принятия решений. Эти системы способны быстро «мыслить», самостоятельно «воспринимать» свое окружение и действовать в динамично меняющихся условиях, повышая эффективность и качество операций.

Приведем несколько примеров технологий искусственного интеллекта и машинного обучения.

Augmented Machine Learning: дополненное машинное обучение. Эффективность проектов на основе машинного обучения часто зависит от качества и количеством доступных данных, а принятие решений на основе полученных результатов – от степени доверия к модели машинного обучения.

Оптимальная стратегия обучения нейронной сети может включать в себя сочетание взаимодополняющих знаний эксперта в конкретной области и машинного «интеллекта». При наличии больших объемов данных по какой-либо задаче можно с помощью онлайн-платформы извлечь скрытые экспертные знания, улучшив модель для другой, родственной задачи, обучив ее модель на существенно меньших объемах данных. Технология дополненного машинного обучения представляет собой автоматизированный метод интеграции знаний экспертов, моделей машинного обучения и средств обработки естественного языка для формирования наборов обучающих данных. Она открывает новые возможности при извлечении полезных сведений из сырых данных и подготовке взвешенных управленческих решений.

Технология прогнозирования оттока клиентов в ретейле. Эффективная обработка огромных объемов сведений о покупателях, накапливаемых предприятиями ретейла: поведение клиентов, данные о их предпочтениях и активности в социальных сетях, – вполне может предоставить конкурентные преимущества и существенно повысить прибыльность за счет проведения таргетированных маркетинговых кампаний, оптимального размещения магазинов, выбора конкретного ассортимента с точным прогнозом срока выхода торговой точки на окупаемость, а также выявления «больных» магазинов. «Умные» программы повышения лояльности клиентов, исключающие заведомо ненужные предложения или взаимодействие с незаинтересованной аудиторией, прогнозирование эффекта воздействия промоакций на потребителя, снижения оттока клиентов и определения их ценовой эластичности.

Технология создания русскоговорящего виртуального аналитика. Виртуальные ассистенты постепенно становятся незаменимыми личными помощниками при автоматизации центров сервисного обслуживания и технической поддержки, выполнении покупок в магазинах и пр. Как правило, такие ассистенты создаются на базе правил и накопленной базы запросов, представляющей собой переписку или комплекты аудиозаписей. Однако, виртуальный аналитик может быть также полноценным компонентом корпоративной аналитической платформы, выполняя роль идеального помощника исследователя данных (*data scientist*). Рынок виртуальных ассистентов стремительно растет во всем мире, включая и Россию ("Алиса", "Анна", "Варвара", "Маруся", "Олег" и другие) – крупные корпорации и небольшие стартапы активно развивают платформы по созданию диалоговых агентов на базе технологий распознавания естественного языка, выполняются многочисленные проекты в области разговорного искусственного интеллекта. По данным аналитиков к 2024 г. 85% коммуникаций с клиентами и сотрудниками будет осуществляться без участия человека-оператора. В 2027 г. цифровые помощники, интегрирующие выполняемые человеком ежедневные задачи, будут непрерывно поддерживать требуемую производительность сотрудников в режиме 24*7.

Машинное обучение и оптимизация в промышленности. Турбулентные явления в отечественной экономике, обострение конкуренции и динамичность нормативных требований заставляют промышленные предприятия искать пути

оптимизации своей работы, способы сокращения издержек и уменьшения непроизводственных затрат. При этом в промышленности накоплены огромные объемы исторических данных, что позволяет эффективно применять методы машинного обучения. Например, только сокращение на единицы процентов, благодаря точным рекомендациям операторам, расходов компонентов сырьевой базы металлургии (кокса, природного газа и пр.) позволяет в масштабах предприятия получать экономию в сотни миллионов рублей. Перспективной представляется технология использования инструментов машинного обучения в комбинации с методами биоинспирированной оптимизации для сокращения отходов производства, повышения качества продукции, более рационального использования ресурсов и обеспечения требований по защите окружающей среды.

Технология прогнозной аналитики для управления персоналом. По мнению аналитиков, почти 70% предприятий в числе своих наиболее острых проблем называют сегодня нехватку квалифицированных кадров – современные условия ведения цифрового бизнеса, высокие темпы изменений, непрерывное обновление технологий требуют привлечение сотрудников с определенной квалификацией. Вместе с тем кадровые службы цифровых предприятий сталкиваются сейчас с проблемой «выгорания сотрудников», вызванной перегруженностью персонала, переизбытком коммуникаций, которые требуется одновременно контролировать менеджерам, а также отсутствием эффективной системы мотивации, постоянными изменениями рабочих обязанностей, неправильным делегированием обязанностей и ответственности. Следствием «выгорания» становится либо увольнение сотрудника, либо резкое снижение его активности. Учитывая, что этому синдрому подвержены, как правило, добросовестные работники, болеющие за дело, с развитым чувством ответственности и с высокой вовлеченностью в работу, их «уход» для компании приводит к существенным убыткам: потери от низкой производительности труда сотрудника; негативное влияние на трудовой климат в коллективе; увольнение сотрудника из компании, влекущее недополученную прибыль и дополнительные затраты на поиск, подбор и адаптацию нового работника. Технологии прогнозной аналитики для управления персоналом позволяют предсказать момент начала увольнения сотрудника по «собственному желанию», существенно ускорить процесс поиска и найма сотрудников, используя технологии машинного обучения для автоматизации бизнес-процессов рекрутмента: поиск резюме кандидатов среди объявлений на специализированных сайтах поиска работы, а также в социальных сетях, оценка текста резюме требуемой вакансии, организация интерфейса с претендентом по всем доступным каналам коммуникаций, проведение видеоинтервью с лучшими кандидатами.

Технология машинного обучения на службе оператора связи. По мнению аналитиков, основное взаимодействие с пользователями будет происходить в автоматическом режиме, например через виртуальных ассистентов. Компании, опоздавшие с внедрением подобных решений, окажутся среди догоняющих, либо будут вытеснены с рынка. Методы машинного обучения способны сегодня

анализировать почти все поведенческие шаблоны людей, например распознавать сарказм или раздражение в диалоге операторов контакт-центров с клиентами, что может быть полезно для контроля эмоционального состояния как клиента, так и человека-оператора. Технологии машинного обучения на службе оператора связи позволяют применять чат-боты с составе различных автоматизированных систем (в компании МТС боты сегодня решают семь из десяти проблем клиентов), обучать их ведению сложных диалогов.

Технология машинного обучения для оптимизации сервисного обслуживания воздушных судов. Техническое обслуживание и ремонты воздушных судов – традиционно технологически сложная область, отличающаяся чрезвычайно высокими требованиями к обеспечению безопасности полетов. С другой стороны, строгая регламентация процессов документирования всех шагов технического обслуживания и ремонта, сборки, обработки и хранения данных создает необходимые предпосылки для формирования информационных массивов, подходящих для приложений методов машинного обучения. Традиционно эти данные распределены по разным источникам и хранятся в различных форматах – от записей в реляционных СУБД, до сканов бортовых журналов. Интегрируя внутренние данные, связанные с эксплуатацией и обслуживанием воздушных судов (телеметрия бортовых датчиков, работы по обслуживанию, документация производителя, полетное расписание), и внешние данные (погода, условия аэропортов и пр.) для их последующей обработки с помощью технологий машинного обучения и средствами оптимизации, можно повысить эффективность процессов сервисного обслуживания воздушных судов, сократить издержки, предсказывать время наработки на отказ узла авиалайнера с целью минимизации риска задержки или отмены рейса по техническим причинам; оптимизировать складские запасы расходных материалов и экономить на закупках дорогостоящих запчастей; прогнозировать вероятность проявления дефектов по каждому конкретному самолету.

Технология машинного обучения в медицине. Перед отечественным здравоохранением поставлен сегодня ряд стратегических задач, включая увеличение продолжительности жизни, снижение смертности и заболеваемости, которые планируется решать в рамках национального проекта «Здравоохранение». Один из инструментов решения таких задач – системы поддержки принятия врачебных решений, использующие в том числе методы машинного обучения. Создание и внедрение в практическое здравоохранение «умных» сервисов – важнейший этап стратегии развития национального здравоохранения, одним из шагов реализации которой стало создание ассоциации «Национальная база медицинских знаний», продвижение цифровых сервисов для медицины, разбору примеров их практической эффективности и основных барьеров, поджидающих любого разработчика интеллектуальных систем для этой области. Имеются примеры пилотных проектов, направленных на выявление онкологических заболеваний на ранних стадиях проведении скрининга онкологических заболеваний в различных регионах России. Особое внимание уделено возможностям и перспективам применения модели

«гибридного интеллекта», объединяющего искусственный интеллект и знания врача.

Технология машинного обучения против фрода (мошенничества в области информационных технологий, несанкционированные действия и неправомочное пользование ресурсами и услугами в сетях связи). Огромное количество предположений по реализации интеллектуальных систем стало сегодня благоприятной средой для различных прожектов, маркетинговых хайпов, способствующих под видом ультрасовременного ноу-хау продажам, не приносящих реальной добавленной стоимости продуктов и решений. Уже имеются реальные примеры внедрения технологии машинного обучения против фрода с подтвержденной добавленной стоимостью. Реализация проектов с использованием искусственного интеллекта – это всегда затраты, а результат заранее непредсказуем, поэтому нужно правильно инвестировать, оценив зоны роста и зоны развития: из чего на самом деле складывается эффективность, как не слепо следовать за модой, а получить реальный инструмент ведения бизнеса.

Технология отечественных нейропроцессоров NeuroMatrix. Технология ориентирована на работу с большими данными в реальном времени: обработка изображений, кодирование и декодирование, обработка радиолокационных и навигационных сигналов, цифровая фильтрация, высокопроизводительная коммутация сигналов и т.д. Основных процессорами семейства *NeuroMatrix* являются Л1879ВМ1 (NM6403), 1879ВМ5Я (NM6406), 1879ВМ6Я (NM6407) и 1879ВМ8Я (NM6408MP), позволяющие одним потоком команд задавать множество параллельно выполняемых операций («зацепление» команд по данным) и обеспечивающие эффективную работу с внутренней и внешней памятью с различной глубиной конвейера. Например, процессор 1879ВМ8Я, представляющий собой гетерогенную многопроцессорную вычислительную систему из 21 процессорного ядра с пиковой производительностью 512 GFLOPS, на базе которой можно развертывать многопроцессорные конфигурации на кристалле, способные в реальном времени решать задачи, характерные для приложений искусственного интеллекта и машинного обучения.

Технология Radeon Instinct для интеллектуальных систем. Машинное обучение является сегодня одним из инструментов поддержки приложений, актуальных для различных предметных областей, требующих интенсивных вычислений при работе с большими данными. Наибольший эффект от применения систем глубинного обучения достигается для нейропроцессорных конфигураций, а также программных фреймворков и библиотек, поддерживающих альтернативные модели машинного обучения и нейронные сети различных типов. Кроме фреймворков высокого уровня (*TensorFlow, Theano, Caffe, Torch* и *MxNet*) разработчикам сегодня необходимы инструменты, максимально полно использующие возможности конкретного оборудования. Например, платформа *AMD ROCm* и фреймворк *MIOpen* содержат полнофункциональный технологический стек для серверных конфигураций для обучения и инференса (задача, для выполнения которой необходимы обученные нейросети) нейронных сетей. Кроме поддержки распространенных библиотек машинного обучения платформа включает драйверы, компиляторы и мониторы

производительности как для отдельных слоев, так и всего стека искусственной нейронной сети.

Технология машинного обучения для синтеза обучающих данных в задачах дистанционного зондирования Земли. Современные методы машинного обучения, построенные по данным (например, сверточные нейронные сети), обеспечивают наилучшее качество в задачах распознавания изображений, когда обучающие данные имеются в изобилии. Однако в ряде задач, например, при обнаружении изменений в изображениях дистанционного зондирования Земли, данные не могут быть быстро получены в достаточных количествах. Технологии машинного обучения предлагают эффективное решение для создания реалистичных синтетических наборов данных в области дистанционного зондирования, в случае серьезных ограничений на объем реальных данных.

Технологии машинного обучения на транспорте. Технологии Интернета вещей, машинного обучения и облачного хранения данных позволяют оптимизировать управление транспортной инфраструктурой: сократить затраты на содержание транспортных средств, минимизировать риски возникновения аварийных ситуаций, исключить непроизводительные затраты. Интеллектуализация робомобилей, видеорегистраторов, взаимодействующих между собой и с единой облачной аналитической системой, открывает новые перспективы для совершенствования транспортной инфраструктуры. Появляются возможности применения «умных» видеорегистраторов в транспортных сетях, в страховых компаниях (описание дорожной ситуации, оценка стиля вождения и пр.), логистических компаниях, в каршеринге, автошколах.

Технология машинного обучения для цифровых двойников. Сегодня многие предприятия работают над созданием цифровых копий своих информационных и программно-аппаратных систем – цифровые двойники позволяют оценивать динамику реальной системы, предсказывать сбои и отказы, формировать рекомендации по оптимизации режима эксплуатации и пр. Однако построение и обслуживание точной имитационной модели невозможно без вовлечения экспертов в конкретной прикладной области. Технология машинного обучения позволяет выполнить тонкую настройку параметров модели по данным, собранными работающими на предприятии информационными системами.

Модели представления знаний и методы вывода в интеллектуальных системах

Понятия "*данные*", "*информация*", "*знания*" часто отождествляются. В ИИ их необходимо различать. Взаимосвязь между этими понятиями непростая. Для того чтобы уверенно оперировать данными понятиями, необходимо не только понимать их суть, но и отличия между ними.

Например, понятие *Data Mining* переводится на русский язык при помощи трех различных понятий: добыча данных, извлечение информации, раскопка знаний! Чтобы разобраться с этими понятиями рассмотрим несколько простых примеров:

1. Магистрант, который сдает экзамен по дисциплине «Машинное обучение и биоинспирированная оптимизация», нуждается в *данных*.

2. Магистрант, который сдает экзамен по дисциплине «Машинное обучение и биоинспирированная оптимизация», нуждается в *информации*.

3. Магистрант, который сдает экзамен по дисциплине «Машинное обучение и биоинспирированная оптимизация», нуждается в *знаниях*.

В первом примере возникает мысль, что магистранту нужны данные, например, для вычислений при решении задачи в экзаменационном билете.

Во втором примере, очевидно, речь идет об информации в виде учебного пособия по дисциплине «Машинное обучение и биоинспирированная оптимизация».

Наиболее логичным выглядит третий вариант: магистрант при наличии соответствующих данных и информации, а также их использовании может рассчитывать в определенных случаях, что полученные данные и информация перейдут в знания.

Даже большие объемы данных и информации не гарантируют получение знаний. Многое зависит от качества процедур обработки информации и данных. Информацию в виде текста на иностранном языке нельзя превратить в знание при отсутствии словаря или переводчика.

Данные. В упрощённом представлении данными можно считать полученные факты (сведения, сообщения, сигналы), характеризующие объект. Данные не тождественны информации. Станут ли данные информацией, зависит от того, известен ли метод преобразования данных в известные понятия. Информация – это не статичный объект. Она динамически меняется и существует только в момент взаимодействия данных и методов их преобразования, т.е. в момент протекания информационного процесса. Все остальное время она содержится в виде данных. Данные объективны, а методы их обработки субъективны, в их основе лежат алгоритмы, придуманные людьми.

Много ли в мире накоплено цифровых данных? – В 2015-16 объем цифровой информации, созданной человечеством, составил 4 зетабайт ($4*10^{21}$) данных. В 2020 г. увеличение этого объема составило 40 зетабайт. При этом 90% накопленной информации было создано в течение последних 5 лет. Это и называется информационным взрывом!

В информационных технологиях основными точками роста сейчас являются облачные технологии, системы автоматизации бизнеса, технологии обработки больших массивов данных (*Big Data, BD*), приложения для мобильных устройств.

Характеристики *Big Data* – объем, скорость прироста, многообразие типов. В 2003 г. самой большой считалась база данных объёмом 25 терабайт, к 2014 г. компания *Google* хранила на своих серверах до 10-15 экзабайт (10^{18}) данных, к 2025 г. генетики будут располагать данными о геномах до 2 миллиардов человек, и для хранения подобного объёма данных потребуется 40 экзабайт.

Появилась профессиональная интерпретация этого термина – исследователь данных (*datascientist*). Специалист по *BD* и исследователь данных присутствуют в кадровых запросах компаний. Сегодня большинство кадровых агентств имеют

запросы на специалистов в области больших данных. На рынке ощущается дефицит таких специалистов. Сегодня они востребованы и в России, и в мире. Требования к специалисту включают опыт построения коммерчески успешных сложных моделей поведения целевой аудитории помощью *Data Mining* инструментов, опыт работы аналитиком больших данных, а в качестве возможных пожеланий – наличие научных публикаций в области больших данных и опыт внедрения систем по работе с большими данными.

Информация. Прошло более 60 лет с тех пор, как *Н. Винер* в своей "Кибернетике" ввел известное до него понятие информации в общенаучный обиход, а оттуда оно распространилось на все уровни – от повседневности до философии. Примерно в то же время сформировалась и теория информации (*К. Шеннон*). Информация – это обработанные данные независимо от формы их представления. Информация, в отличие от данных, имеет смысл. Однако трудно найти понятие, более общее для всех наук и более загадочное, чем информация. Существует несколько подходов к измерению информации. Рассмотрим наиболее популярные из них.

Пусть имеется система *S*. Она может находиться в *N* равновероятных состояниях. Если каждое состояние системы закодировать, например, двоичными кодами определённой длины *d*, то эту длину необходимо выбрать так, чтобы число всех различных комбинаций было бы не меньше, чем *N*. Наименьшее число, при котором это возможно или мера разнообразия множества состояний системы задаётся формулой *Р. Хартли*:

$$I = k\log_a N$$

где *k* – коэффициент пропорциональности (масштабирования, в зависимости от выбранной единицы измерения), *a* – основание рассматриваемой системы.

Утверждение Хартли: если во множестве $X=\{x_1, x_2, ..., x_n\}$ выделить произвольный элемент $x_i \in X$, то для того, чтобы найти его, необходимо получить не менее $\log_a n$ (единиц) информации.

По Хартли, чтобы мера информации имела практическую ценность – она должна быть пропорциональна числу выборов. Формула Хартли отвлечена от семантических свойств рассматриваемой системы. Это положительная сторона формулы. Но формула не учитывает разную вероятность *N* состояний системы. Уменьшение (увеличение) *I* может свидетельствовать об уменьшении (увеличении) разнообразия состояний *N* системы.

Формула *К. Шеннона* дает оценку информации независимо от ее смысла:

$$I = -\sum_{i=1}^{N} p_i \log_2 p_i,$$

где *N* - число состояний системы; p_i - вероятность (или относительная частота) перехода системы в i-ое состояние, причем

$$\sum_{i=1}^{N} p_i = 1.$$

Если все состояния равновероятны (т.е. $p_i = 1/N$), то $I = \log_2 N$, т.е. мера Шеннона совпадает с мерой Хартли.

Сообщение о наступлении события с меньшей вероятностью несёт в себе больше информации, чем сообщение о наступлении события с большей вероятностью. Сообщение о наступлении достоверно наступающего события несёт в себе нулевую информацию (и это вполне ясно, – событие всё равно произойдёт когда-либо).

В термодинамике известна формула Больцмана

$$\mathrm{H} = -k \sum_{i=1}^{N} p_i \ln p_i,$$

где k – постоянная Больцмана ($k = 1.38x10^{-16}$ *эрг/град*), которая известна как *энтропия* или мера хаоса, беспорядка в системе. Сравнивая выражения для I и H видим, что I можно понимать как информационную энтропию или энтропию из-за нехватки информации о системе.

Нулевой энтропии соответствует максимальная информация. Увеличение (уменьшение) меры Шеннона свидетельствует об уменьшении (увеличении) энтропии (организованности, порядка) системы. При этом энтропия может являться мерой дезорганизации систем от полного хаоса ($H = H^{max}$) и полной информационной неопределённости ($I = I^{min}$) до полного порядка ($H = H^{min}$) и полной информационной определённости ($I = I^{max}$) в системе.

Однако информацию вряд ли можно отождествлять с энтропией: процесс роста энтропии принципиально необратим и относится к окружающей среде, а информация – к объекту или системе.

Понятие информации имеет не только количественный аспект, приводящий к мере, т.е. к числу, но и качественный аспект, не связанный с измерением. Информация – не мера, не число. Измеряемое в теории информации количество информации еще не есть информация, как количество людей не есть человечество, а мощность множества – не множество. Количественные меры объектов – их частные свойства, но не смыслы.

Чем еще характеризуют информацию? – Скоростью приёма (чем больше шум, тем медленнее на выходе могут воспроизвестись изменения на входе). Избыточный текст скучно читать, а «плотный» текст требует внимания. Другими характеристиками информации являются репрезентативность, полнота, доступность, своевременность, точность, достоверность, безопасность, ценность.

Например, под ценностью информации понимается прагматическое отношение между системой, информацией и целью системы. Ценность информации нужно измерять, но по какой шкале: абсолютной или относительной? Практика свидетельствует: ни одна информация за всю историю природы, жизни и разума не обрела статуса абсолютно ценной. Наоборот, очередной виток развития ставил новые цели и изменял критерии ценности, а то, что казалось прежде абсолютно ценным на все времена, приобретало статус относительной, мифотворческой ценности или вообще низвергалось на свалку истории и науки. Можно ввести абсолютную шкалу ценности: ценнее считать ту информацию, которая генерирует новую информацию с большей вероятностью. Но как оценить эту вероятность?

В частности, ценность научных трудов сейчас стало модно оценивать по так называемому "индексу цитирования" (чем чаще цитируют труд, тем он ценнее, тем выше индекс). Но известны выдающиеся философы и ученые, которые не публиковались или "писали в стол" и только после смерти стали известны миру. Таков *Сократ*, который по свидетельству учеников (*Платона* и др.) утверждал, что "письмена мертвы". Таков *Г. Кавендиш*, занимавшийся наукой вне официальной науки; в его бумагах, найденных после смерти, были обнаружены научные открытия и законы, которые известны науке в другом авторстве (закон *Кулона* и др.). Таковы основатели многих религиозных учений и школ. Их учения, как правило, передавались устно, как и "крылатые фразы" известных людей.

Не следует фетишизировать индекс цитирования как количественный показатель ценности научной информации. Накануне экзаменационной сессии ценность информации о причудах преподавателя гораздо выше, чем в начале семестра, а о своем преподавателе несравнимо выше, чем о чужом. Изменение цели изменяет и ценность информации, но не наоборот, т.к. ценность информации аксиологически вторична по отношению к цели, преследуемой потребителем информации.

Система в каждый момент своего существования ставит перед собой цели, реализовать которые она может только через информацию, обладающую для этого необходимыми свойствами и являющуюся импульсом целенаправленной деятельности системы. Информация ценна лишь постольку, поскольку она способствует достижению цели. Но в теории информации отсутствуют понятия цели информационного процесса и его безопасности, из которых явно следовало бы, что информация всегда селектируется от дезинформации и шума. Соответственно, метрика ценности, связанная с количеством информации, не обеспечивает такой селекции. В этом-то и проблема данной метрики, часто фатальной для системы – механизм селекции начинает работать, не дожидаясь решения о ценности полученной информации. Например, так случается с общественно-политическими системами в периоды бурь и потрясений. Стоит ли потом удивляться, что сделанный выбор оказался ложным? В целом проблема сводится к оценке априорной ценности информации. Обобщенной, значимой априорной меры ценности информации нет.

Знания – это обработанная информация, обеспечивающая увеличение вероятности достижения цели, «ноу-хау», технология. Знание – это информация, но не всякая информация – знание. Между информацией и знаниями имеется разрыв. Человек должен творчески перерабатывать информацию, чтобы получить новые знания. Приведем несколько любопытных определений понятия «знание».

«В моем текущем Знании нет ничего абсолютного, и оно ничтожно мало. Я знаю, что ничего не знаю. Чем больше Знание, тем больше и Незнание» (*Сократ*).

«Знание есть особая форма Незнания. Незнание неизменно, объективно, бесконечно, всеобще, изначально. Знание – эклектично, субъективно,

ограниченно, локально и временно. Только для философии и искусственного интеллекта предметом изучения являются сами Знания» (*А.С. Нариньяни*).

«Знание – обоснованное истинное убеждение» (*В.Н. Вагин*).

При изучении некоторого объекта понятия появляются в следующем порядке: *данные–информация–знания*. Факты окружающего мира представляют собой данные. Затем при использовании данных в процессе решения конкретных задач появляется информация. Результаты решения задач, проверенная информация, обобщенная в виде законов или теорий представляет собой знания.

В настоящее время делаются попытки создания онтологий, т.е. баз знаний специального типа, формально описываемых тройкой множеств <*X, R, F*>, где *X* – множество понятий в предметной области; *R* – множество отношений между понятиями; *F* – множество функций для интерпретации понятий. Онтоло́гия в ИИ – это попытка детальной формализации некоторой области знаний с помощью концептуальной схемы. Схема состоит из структуры данных, содержащей все релевантные классы объектов, их связи и правила (теоремы, ограничения), принятые в этой области. Словари, тезаурусы и каталоги являются частными случаями онтологий.

Язык описания онтологий – формальный язык, используемый для кодирования онтологии. Существует несколько подобных языков:

• *OWL – ontology web language*, язык для семантических утверждений, разработанный как расширение *RDF* и *RDFS*;

• *KIF* (*Knowledge Interchange Format* или формат обмена знаниями) - основанный на *S*-выражениях для логики;

• *CycL* — онтологический язык, использующийся в проекте *Cyc*, основан на исчислении предикатов с некоторыми расширениями более высокого порядка;

• *DAML + OIL* (*FIPA*).

Наиболее известными реализованными проектами онтологий являются *Cyc, WordNet, SUMO*.

Cyc – проект (запущен в разработку в 1985 г.) по созданию энциклопедической онтологической базы знаний, позволяющей программам решать сложные задачи из области ИИ на основе логического вывода и привлечения здравого смысла. База знаний разбита на микротеории для решения частных задач. Типичным примером знаний в базе являются «Всякое дерево является растением» и «Растения смертны». Если спросить «умирают ли деревья?», машина логического вывода может дать правильный ответ. БЗ содержит около 2-х миллионов занесённых туда людьми утверждений, правил и общеупотребительных идей. Они формулируются на языке *CycL*, который основан на исчислении предикатов и имеет схожий с *ЛИСП*ом синтаксис. Пользователи шутят что они «велосипедисты» (от англ. *cyclist* – велосипедист). Ведётся работа над тем, чтобы дать системе *Cyc* возможность самостоятельно общаться с пользователями на естественном языке, и над ускорением процесса пополнения базы с помощью машинного обучения. Версия *ResearchCyc*_1.0, предназначенная для исследовательского сообщества. включает значительно больше семантических знаний, написанных на *Java* для редактирования знаний и создания запросов к базе. Одна из целей – создать полностью свободный и

неограниченный семантический словарь для использования в *Semantic Web* – общедоступной глобальной семантической сети, формируемой на базе Всемирной паутины путём стандартизации представления информации в виде, пригодном для машинной обработки.

WordNet – проект лингвистической направленности, в котором база знаний разбита на классы эквивалентности по синонимии (синсеты), включает 150 000 слов и 115 000 синсетов, а также содержит лингвистические отношения между словами, позволяющие делать заключения о соотношении смыслов используемых слов.

SUMO – онтологическая система, содержащая онтологию для почти 20 предметных областей. Написана на языке *SUO-KIF* (расширение логики предикатов 1-го порядка), включает 20 000 термов и 60 000 аксиом, работает для очень разных по структуре языков (английский, китайский, хинди), отображается на базу знаний проекта *WordNet*.

Анализ, построение, развитие и использование моделей предметной области — очень важная и перспективная задача ИИ в настоящее время. Достаточно упомянуть, например, концепцию систем управления бизнес-процессами (*Business Process Management Systems - BPMS*) в области информационно-управляющих систем, концепцию семантической паутины (*Semantic Web*) в Интернете или концепцию проблемно-ориентированных языков (*Domain Specific Languages*) в программировании. Все эти перспективные концепции апеллируют к понятию онтология.

Классификация знаний:

- по степени общности различают *абстрактные и конкретные* знания (абстрактные применимы в разных предметных областях, конкретные – в одной);
- по способу представления бывают *фактические и концептуальные* знания (фактические задаются совокупностью фактов, концептуальные – обобщающим понятием);
- по форме представления знания делятся на *образные и знаковые* (образная форма реализует отношение целое-часть, знаковая – отношение часть-целое);
- по изменчивости во времени различают *статические и динамические* знания;
- по способу существования знания классифицируют на *хранимые и воспроизводимые*;
- по языку описания бывают *содержательные* (на языке предметной области) и *формальные* (на математическом языке) знания;
- по степени истинности различают *теоретические и эвристические* знания;
- по квалификации источника знания делят на *обыденные и экспертные*;
- по степени соответствия реальности знания бывают *достоверными и правдоподобными*;
- по способу размещения в пространстве различают *локальные и распределенные* знания;
- по способу программирования знания классифицируют на *процедурные и декларативные*.

Свойства знаний:

•внутренняя интерпретируемость означает использование имен для поиска знаний. Данные в памяти лишены имен и могут идентифицироваться только программой, извлекающей их из памяти. Что скрывается за тем или иным двоичным кодом машинного слова, системе неизвестно. При переходе к знаниям каждая информационная единица должна иметь уникальное имя, по которому система находит ее, а также отвечает на запросы, в которых это имя упомянуто;

•вложимость – возможность включения некоторых знаний в состав других (*рекурсия*) и, наоборот, выделение из знаний отдельных информационных единиц. Иными словами, каждая информационная единица может быть включена в состав любой другой, и из каждой информационной единицы можно выделить некоторые составляющие ее информационные единицы;

•структурированность (связность) – возможность установления между отдельными информационными единицами отношений типа «часть – целое», «род – вид», причина-следствие, аргумент-функция и др. Семантика отношений может носить декларативный или процедурный характер. Отношения "одновременно", "причина - следствие" или "быть рядом" характеризуют декларативные знания. Если между двумя информационными единицами установлено отношение "аргумент - функция", то оно характеризует процедурное знание, связанное с вычислением определенных функций. Перечисленные три свойства знаний позволяют ввести общую модель представления знаний - семантическую сеть с вершинами, в которых находятся информационные единицы с именами, а дуги – семантические связи между ними;

•семантическая метрика используется для объединения понятий по степени их смысловой близости или релевантности. Отношение релевантности позволяет находить знания, близкие к уже найденным. На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее ситуационную близость информационных единиц, т.е. силу ассоциативной связи между информационными единицами;

•активность знаний означает необходимость действий, возможность модификации знаний или связей между понятиями. Действия в БЗ инициализируются не командами, а ее состоянием, например, появлением фактов или установлением связей между событиями в системе понятий предметной области. БЗ отличается от БД возможностью логического вывода. С момента появления ЭВМ и разделения используемых в ней информационных единиц на данные и команды создалась ситуация, при которой данные пассивны, а команды активны. Все процессы в ЭВМ инициируются командами, а данные используются этими командами лишь в случае необходимости. Для ИС эта ситуация не приемлема. Выполнение программ в ИС должно инициироваться текущим состоянием информационной базы. Появление в базе фактов, событий или установление связей между информационными единицами и понятиями предметной области может стать источником активности системы;

•корректность включает проверку БЗ на полноту (отношение выводимости в БЗ с помощью определенных операций и средств) и непротиворечивость

(синтаксическое и семантическое совпадение значений одних и тех же знаний и данных, в разных частях БЗ).

Понятийная и образная форма знаний. Знания делят на фрагменты, которые называются предметными областями. Знания характеризуются с помощью понятий и образов. Понятие – это класс сущностей, объединяемых на основе общности признаков. Сущность – это объект произвольной природы, принадлежащий реальному или виртуальному миру. Класс сущностей – это совокупность предметов, их свойств, состояний, процессов, событий, оценок событий, их модификаторов, квантификаторов, модальностей. Например, в области машинного обучения предметом может быть программа, одним из её свойств – надёжность, состоянием – неисправность, процессом – тестирование, событием – запуск программы, оценкой события – удовлетворительное (или нет) качество работы, модификатором – быстродействие программы, квантификатором – частота использования, модальностью – возможность использования.

Множество существенных признаков, характеризующих понятие, называется его содержанием (*интенсионалом*). Содержание понятия А задаётся через свойства принадлежащих ему сущностей: $A = \{A_1, A_2, \ldots, A_к\}$, где A_i – свойство понятия или его существенный признак.

Объем понятия (*экстенсионал*) – это сущности, объединяемые в понятие. Сущности, входящие в объем, задаются обычно перечислением. Объем понятия, в отличие от содержания, может изменяться путём задания ограничений.

Основными формами представления понятийных знаний являются знаковая и образная формы. Носителем знаковой формы представления знаний является естественный язык (ЕЯ). Для представления понятия в ЕЯ используется слово, называемое именем понятия. Оно семантически отражает сущность какого-то объекта или явления, а синтаксически – конкретный субъект или объект высказывания. Перспективным считается семиотический подход к моделированию знаний. На рисунке показана схема, известная в семиотике как *треугольник Фреге*:

Объекты реального мира (*денотаты*) отражаются в сознании человека (ментальном мире). В результате возникают представления о денотатах. Для различения денотатов человек использует имена и формирует понятия. Связи в

треугольнике Фреге соответствуют механизмам мышления. Связь 1 позволяет по имени активизировать в памяти сведения о свойствах понятия. Эта же связь в обратном направлении позволяет по описанию свойств понятия определить его имя (загадки: «не лает, не кусает, а в дом не пускает?»). Связь 2 позволяет по представлению узнать свойства понятия. Связь 3 соединяет представление о денотате с его именем. Треугольник Фреге позволяет отображать синонимию и омонимию понятий.

Структура треугольника Фреге, называется знаком или семой. Знак – минимальный носитель языковой информации. Совокупность знаков образует знаковую (семиотическую) систему, или язык.

Символ в отличие от знака не имеет семантики и прагматики, а только синтаксис. Имена в предметной области часто называют терминами. К терминам предъявляются некоторые требования: однозначность, стилистическая нейтральность, соответствие нормам языка, обладание словообразующей функцией, краткость и пр.

Связь между именем и понятием реализуется обычно путём определения понятия. Определение раскрывает содержание (иногда объем) понятия. Существуют различные способы определения понятий: *остенсивные* (указывают на конкретный предмет, явление, действие), *номинальные* (одно слово заменяет несколько слов), *интенсиональные* (раскрываются существенные признаки понятия), *экстенсиональные* (перечисляются все объекты, входящие в понятие). К определениям, как и к терминам, также предъявляются требования: неизбыточность; соразмерность объема определяемого понятия и тех понятий, которые включаются в него; отсутствие «порочного круга»; ясность; лингвистическая правильность и краткость.

Люди оперируют осмысленными понятиями в отличие от компьютеров, которые оперируют лишь содержимым ячеек памяти. Мысли выстраиваются из совокупности понятий и образов, связанных между собой. В данном контексте вопрос *А. Тьюринга* о том, могут ли машины мыслить, попросту неуместен. Машины пока вообще не имеют дела с мыслями.

В процессе мышления одновременно присутствует как понятийная, так и образная логика. Для понятийных знаний разработан достаточно мощный язык математической логики. Язык для оперирования образами в процессе решения интеллектуальных задач пока не создан. Разработки такого языка ведутся в когнитологии на стыке эпистемологии, герменевтики и психолингвистики.

Существуют различные виды классификации понятийных знаний: родо-видовая, фасетная, иерархическая, партитивная и др.

Связь «род-вид» соответствует отношению «общее-частное», в котором одно понятие является исходным (родовым), а расширяющий родовое понятие видовой признак называется видовым отличием. В силу родства некоторых понятий для оценки их семантической близости обычно используют ранг ρ (степень родства). Для родового понятия ранг $\rho = 0$, а для образования любого видового понятия применяется рекурсивная формула вида $A_{\rho+1, B} = A_\rho + B$, где A_ρ – исходное понятие, B – видовое отличие. Данная формула отражает

наследование признаков. Объём видового понятия $A_{\rho+1, B}$ определяется по формуле $A_{\rho+1, B} = A_\rho \& B$.

Объем видового понятия *уже,* чем объем родового понятия, в то время как его содержание *шире.* В объем видового понятия обычно включаются элементы и их классы. Связь «элемент-класс» является частным случаем связи «род-вид». В английской терминологии отношение «элемент-класс» обозначают «*is a*». Элементы, входящие в объем понятия, наследуют различные видовые отличия по признаку основания деления. Понятие может многократно делиться. Деление состоит в том, что входящие в некоторый объем понятия распределяются по классам (фасетам, таксонам). Доказано, что каждое понятие, порождаемое на основе других понятий, имеет непустой объем.

В зависимости от способа деления различают фасетную (параллельную) и иерархическую (последовательную) классификации понятий.

Реальные классификации понятий, как правило, являются межвидовыми или смешанными. В первом случае основания деления порождающих понятий различны, во втором – основания деления общие.

Партитивная связь понятий – это связь между целым и его частями, обозначается как «*part of»* (*Monitor is part of a PC*). Партитивная связь характеризует разделение целого понятия на отдельные части.

Классификация понятий и отношений позволяет разрабатывать компьютерные модели и технологии понятийного мышления. Каждой сущности в реальном мире соответствует понятие, представление о ней и имя (треугольник Фреге). Попытки представления системы понятий в предметной области привели к возникновению БЗ специального типа в виде онтологий.

Пара понятий *A* и *B,* выраженные через интенсионал, может находиться в отношениях равнозначности, несравнимости, толерантности, подчиненности и др. Возникает вопрос: как оценить сходство или различие двух понятий? Обычно, для оценки различия между понятиями *A, B* вводится некоторая *метрика* для оценки «расстояния» между ними:

$$d(A, B) = (\sum_i r_i * p_i (A, B)) / \sum_i r_i,$$

где $p_i (A, B)$ – указатель *различия* *i*-х свойств понятий *A, B*; r_i – «вес» i-го свойства, равный *n, n* – 1,…, 2, 1, причём $p_i(A, B) = 1$, если $A_i \neq B_i$, $p_i(A, B) = 0$, если $A_i = B_i$.

Зачастую при распознавании объектов признаки трудно выразить численно, поэтому важно просто их наличие или отсутствие у объекта. При сравнении пары объектов (*i, k*) по одному из признаков *j* возможны четыре варианта (X_{ij} , X_{kj}): (1, 1), (1, 0), (0, 1), (0, 0). При сравнении пары объектов (*i, k*) по всем признакам *j* необходимо вычислить:

– число случаев, когда оба объекта обладают одними и теми же признаками $a = \sum_j X_{ij} * X_{kj}$;

– число случаев, когда оба объекта не обладают одними и теми же признаками $b = \sum_j (1-X_{ij}) * (1-X_{kj})$;

– число случаев, когда *i*-й объект не обладает признаками *k*-го объекта $h = \sum_j (1-X_{ij}) * X_{kj}$;

- число случаев, когда *i*-й объект обладает признаками, отсутствующими у *k*-го объекта $g = \sum_j X_{ij}*(1-X_{kj})$.

Операции в формулах для *a, b, h* и *g* представляют собой обычные арифметические операции над булевыми переменными. Из анализа формул для *a, b, h, g* следует, что чем больше сходства между парой объектов, тем большее влияние имеет переменная *a*. На практике используются различные функции сходства *S*:

- *Рассела и Рао* $S_1=a/(a+b+h+g) = a/n$;
- *Жокара и Нидмена* $S_2=a/(n-b)$;
- *Дайса* $S_3=a/(2a+h+g)$;
- *Сокаля и Снифа* $S_4=a/(a+2(h+g))$;
- *Кульжинского* $S_5=a/(h+g)$;
- *Юла* $S_6=(ab-gh)/(ab+gh)$ *и др.*

Разнообразие формул означает, что среди них нет универсальной для различных задач. Кроме того, необходимо учитывать следующий эффект: два объекта, обладающих большим числом признаков, представляются более похожими, нежели два объекта, обладающих меньшим числом признаков. Поэтому следует вносить поправку в параметр *a*.

Наиболее общей моделью представления знаний является *алгебраическая система Мальцева:*

<*A, C, F, P*>,

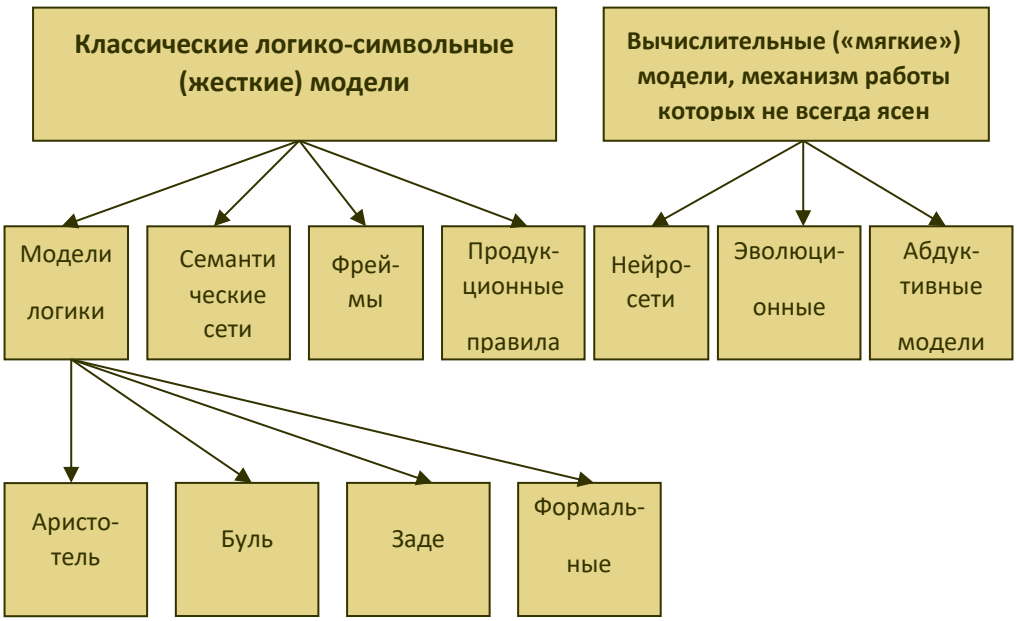
где *A* – множество переменных; *C* – множество констант; *F* – множество функций; *P* – множество предикатов (отношений).

Множество Σ = <*F, P*> называется *сигнатурой* или *языком системы*.

Алгебраическая система состоит из двух частей: *алгебры* <A, C, F> и *реляционной системы* <A, C, P>. Алгебре в ИИ соответствует *процедурная модель*, реляционной системе – *декларативная модель*.

База знаний – основа любой интеллектуальной системы, где знания описаны на некотором языке представления знаний. Совокупность знаний принято называть предметной областью. В любой предметной области есть свои понятия и связи между ними, своя терминология, свои законы, процессы и события. Каждая предметная область имеет свои методы решения задач.

Базы знаний базируются на моделях представления знаний, подобно базам данных, которые основаны на моделях представления данных (иерархической, сетевой, реляционной, постреляционной и т.д.). При представлении знаний в памяти интеллектуальной системы традиционные языки являются неэффективными. Для этого используются специальные языки представления знаний, основанные на символьном представлении данных. Наиболее часто используется на практике классификация моделей представления знаний, приведенная на рисунке:



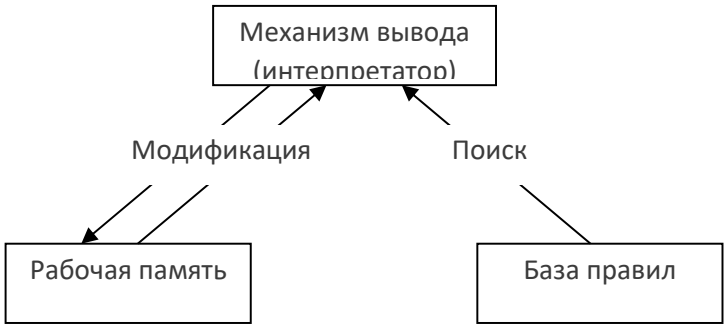
Сюда входят как классические (символьные) модели, имитирующие мышление и структуру памяти человека, так и модели нейросетей, эволюционные модели, модели немонотонной логики.

К моделям знаний выдвигаются целый ряд требований: универсальность; наглядность представления знаний; реализация в модели свойства активности знаний; открытость; возможность отражения структурных отношений объектов предметной области; возможность оперирования нечеткими знаниями; использование многоуровневых представлений (данные, модели, метамодели и т.д.).

Существующие модели представления знаний не удовлетворяют полностью этим требованиям, чем и объясняется их многообразие и активное развитие данного направления.

К декларативным моделям относятся продукционные правила, семантические сети, фреймы.

Продукционные правила. Архитектура продукционных систем имеет следующий вид:



В рабочей памяти хранятся исходные данные о решаемой задаче и результаты работы системы. База правил включает правила типа

«ЕСЛИ (*условия*), ТО (*действия*)».

Механизм вывода использует правила в соответствии с содержимым рабочей памяти. Если текущее содержимое базы данных удовлетворяет условию

некоторого правила, то может выполняться действие, указанное в этом правиле. Но возможность применить правило еще не означает необходимость его применять. Будет ли применимое правило действительно применено, определяет механизм вывода. Содержимое рабочей памяти может соответствовать подмножеству правил, которые образуют конфликтное множество. В соответствии с некоторой стратегией управления выводом для применения выбирается одно из правил конфликтного множества. После применения правила база данных меняет свое состояние. Между правилами нет прямой связи по управлению. Иными словами, правила не вызывают друг друга. Между правилами нет и прямой связи по данным. Правила ничего не сообщают друг другу, все данные находятся в базе данных. Правила являются функционально и информационно прочными модулями.

Преимущества продукционных правил: простота создания и понимания отдельных правил; простота пополнения, модификации и аннулирования; простота механизма логического вывода, наличие программных средств (языки *Prolog, LISP, CLIPS, ЭС*).

Недостатки продукционных правил: неясность взаимных отношений правил; сложность оценки целостного образа знаний; невысокая эффективность обработки; отсутствие гибкости в логическом выводе; сложность проверки правил на непротиворечивость, особенно если правил больше 1000, неоднозначность выбора правил.

Таким образом, если объектом является небольшая задача, выявляются только сильные стороны системы продукций. В случаях увеличения объема знаний, необходимости решения сложных задач, выполнения гибких выводов или повышения скорости вывода требуется структурирование базы данных.

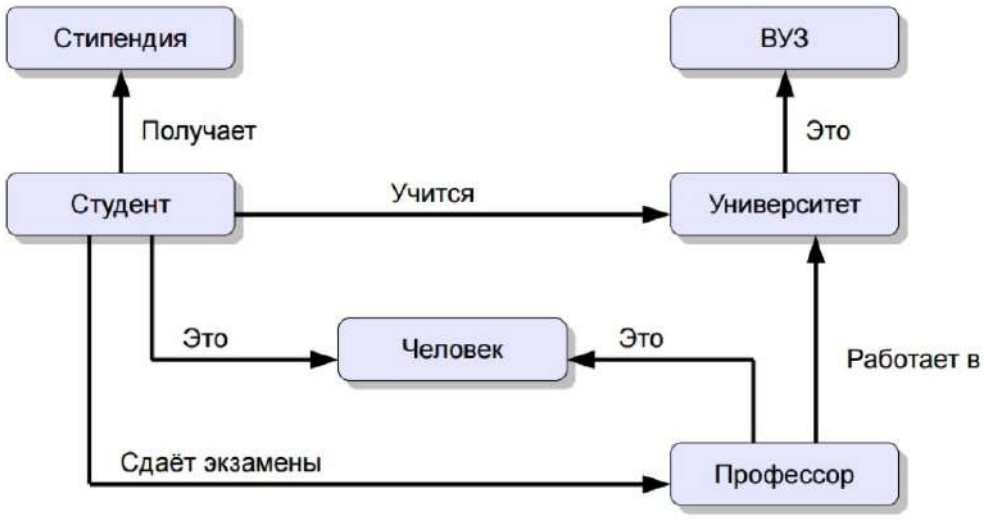
Кроме того, в конкретной ситуации и факты верны не абсолютно, и применение того или иного правила оправдано лишь частично. В системах продукций для этого используют термины «коэффициент достоверности», «обоснованность», «индекс уверенности» и т.д. В последние годы разработчики ищут более формальный базис для работы с неоднородными продукциями. Известны попытки использовать нечеткие логики, дескриптивные логики и другие теории. Однако эти попытки полностью дезавуируются тем обстоятельством, что оценки достоверности исходных фактов и правил в конкретизированной предметной области сами не являются достоверными – их выставляют эксперты также свойственно ошибаться.

Резюмируя отметим, что системы продукций – это исторически первый и до сих пор часто применяемый способ представления знаний; продукционные правила эффективно реализуемы программно и являются основой многих практических программных систем; использование систем продукций может принести пользу только при полном понимании всех сопутствующих им ограничений и недостатков.

Семантическая сеть. Одной из структурных моделей долговременной памяти является предложенная *Куиллианом* модель понимания смысла слов, получившая название *TLC*-модели (*Teachable Language Comprehender*, доступный механизм понимания языка). В данной модели для описания

структуры долговременной памяти была использована сетевая структура как способ представления семантических отношений между концептами (словами). Данная модель имитирует естественное понимание и использования языка человеком. Поэтому основной ее идеей было описание значений класса, к которому принадлежит объект, его прототипа и установление связи со словами, отображающими свойства объекта.

Идея семантической сети (СС) – любое знание можно представить совокупностью понятий и отношений между ними. СС – это ориентированный граф, вершины которого понятия, а ребра – отношения между понятиями:



Сеть наглядна, имеет аналогию с организацией долговременной памяти человека, но вывод знаний по ней сложен.

Важность модели семантической сети с точки зрения многочисленных приложений определяется следующими моментами. В отличие от традиционных методов семантической обработки с анализом структуры предложения были предложены новые парадигмы в качестве модели представления структуры долговременной памяти, в которой придается значение объему языковой активности. Был предложен способ описания структуры отношений между фактами и понятиями с помощью средства, называемого семантической сетью, отличающейся несложным представлением понятий, а также способ семантической обработки в мире понятий на основе смысловой связи (смыслового обмена) между прототипами.

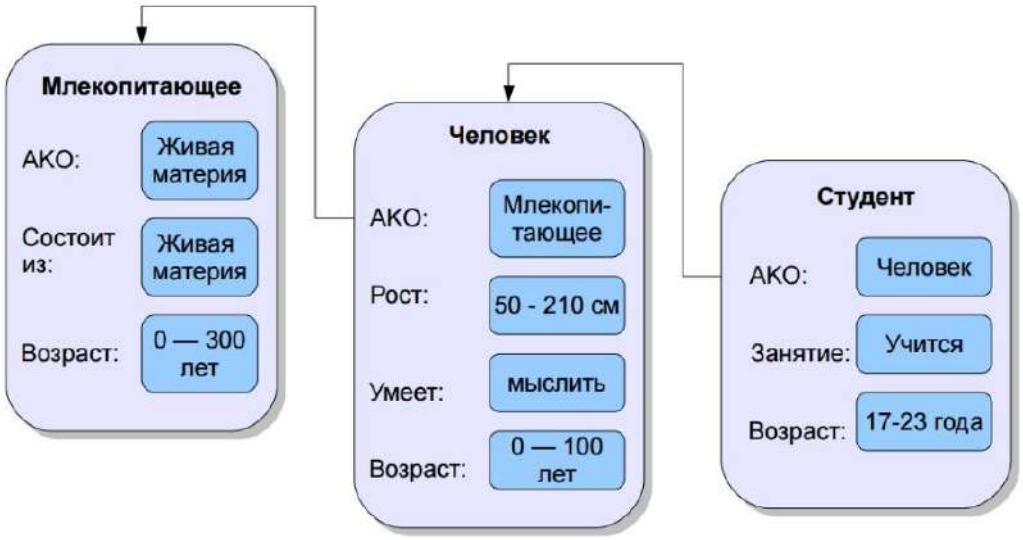
Достоинства семантической сети: описание объектов и событий производится на уровне очень близком к естественному языку; обеспечивается возможность соединения различных фрагментов сети; отношения между понятиями и событиями образуют небольшое, хорошо организованное множество; для каждой операции над данными или знаниями можно выделить некоторый участок сети, который охватывает необходимые в данном запросе характеристики; обеспечивается наглядность системы знаний, представленной графически; близость структуры сети, представляющей знания, семантической структуре фраз на естественном языке; соответствие сети современным представлениям об организации долговременной памяти человека.

Недостатки семантической сети: сетевая модель не дает ясного представления о структуре предметной области, поэтому формирование и модификация такой модели затруднительны; сетевые модели представляют собой пассивные структуры, для обработки которых необходим специальный аппарат формального вывода и планирования.

Семантические сети нашли применение в основном в системах обработки естественного языка, частично в вопросно-ответных системах, а также в системах искусственного видения.

Фреймовая модель. Фрейм – это структура данных с именем, служащая для представления стереотипных образов, в которых сконцентрированы знания об объекте. Фрейм состоит из слотов и значений атрибутов. Процедуры, подключаемые к фрейму для выполнения запроса, называют демонами.

Фреймы имеют иерархическую структуру. Их свойства наследуются сверху вниз через *AKO*-связи (*A Kind Of*):



Преимущества фреймов: наглядность, соответствие модели памяти человека, нет ограничений по размерам, поддерживается программно. Достоинство фрейма – представления во многом основываются на включении в него предположений и ожиданий. Это достигается за счет присвоения по умолчанию слотам фрейма стандартных ситуаций. В процессе поиска решений эти значения могут быть заменены более достоверными. Некоторые переменные выделены таким образом, что об их значениях система должна спросить пользователя. Часть переменных определяется посредством встроенных процедур, называемых внутренними. По мере присвоения переменным определенных значений осуществляется вызов других процедур. Этот тип представления комбинирует декларативные и процедурные знания. Фреймовые модели обеспечивают требования структурированности и связанности. Это достигается за счет свойств наследования и вложенности, которыми обладают фреймы, т.е. в качестве слотов может выступать система имен слотов более низкого уровня, а также слоты могут быть использованы как вызовы каких-либо процедур для выполнения:

Недостаток: трудность обмена большими объемами данных.

Для многих предметных областей фреймовые модели являются основным способом формализации знаний.

Представление знаний – весьма сложный и творческий процесс. Наиболее ценными в ближайшие годы будут гибридные моделей знаний. Однако их глубокое понимание и эффективное применение невозможно без изучения, сегодня кажущихся уже довольно примитивными, нейросетевых, а также декларативных и процедурных моделей представления знаний.

Экспертные системы (ЭС) являются одним из примеров систем, основанных на знаниях, в которых широко используются различные модели для представления знаний. ЭС – это компьютерная программа, способная частично заменить специалиста-эксперта в разрешении проблемной ситуации. Существуют экспертные системы по военному делу, геологии, инженерному делу, информатике, космической технике, математике, медицине, метеорологии, промышленности, сельскому хозяйству, управлению, физике, химии, электронике, юриспруденции и т.д.

Предтечи ЭС были предложены в России в 1832 г. *С.Н. Корсаковым*. Он создал механические устройства (назвал их «интеллектуальными машинами»), позволявшие находить решения по заданным условиям. Например, определять наиболее подходящие лекарства по наблюдаемым у пациента симптомам заболевания.

Особенности ЭС известны. Они ориентированы на решение задач на основе рассуждений в узкой предметной области, способны объяснять свои рекомендации и выводы, способны приобретать новые знания от экспертов.

Применение ЭС позволяет, например, при проектировании интегральных микросхем повысить производительность труда, при диагностике ускорить поиск неисправностей; повысить производительность труда программистов; при профессиональной подготовке сократить время на индивидуальную работу с обучаемым персоналом и пр.

Широко известные экспертные системы: *CLIPS* – весьма популярная ЭС;

OpenCyc – динамическая ЭС с глобальной онтологической БЗ и поддержкой независимых контекстов; *WolframAlpha* – поисковая система, интеллектуальный «вычислительный движок знаний»; *MYCIN* – система для диагностики и наблюдения за состоянием больного при менингите и бактериальных инфекциях; *PROSPECTOR* – ЭС для выдачи геологам информации о наличии залежей ископаемых; о выборе мест для бурения; *HASP/SIAP* – определяет местоположение и типы судов в океане по данным акустических систем слежения; *Акинатор* – интернет-игра. Игрок должен загадать любого персонажа, а Акинатор должен его отгадать, задавая вопросы. База знаний автоматически пополняется, поэтому программа может отгадать практически любого известного персонажа.

В общем, различают 6 основных типов интеллектуальных информационных систем, использующих модели представления знаний и методы машинного обучения:

- диалоговая система обработки запросов (*Transaction Processing System*);
- система информационного обеспечения (*Information Provision System*);

•система поддержки принятия решений (*Decision Support System*);

•программируемая система принятия решения (*Programmed Decision System*) для автоматического отбора решений;

•экспертные системы (*Expert System*);

•проблемно-ориентированные интеллектуальные системы, основанные на знаниях (*Knowleadge Based System*).

Вывод, как метод доказательства путём рассуждений, является одним из ключевых вопросов ИИ. Например, в экспертных системах вывод знаний – это новое заключение, полученное по определенным правилам из фактов, имеющихся в базе знаний.

Различают правдоподобный и достоверный механизмы вывода. К правдоподобному выводу относят абдуктивный (от частного к частному), индуктивный (от частного к общему), нечеткий (основанный на логике Заде), нейросетевой (основанный на использовании нейросетей), биоинспирированный (основанный на инспирированных природой явлениях) и некоторые другие методы вывода. Единственным достоверным методом вывода является дедуктивный логический вывод (рассуждения от общего к частному на основе аксиом).

Общая формулировка задачи вывода знаний:

Пусть **Th** – знания из БЗ, **f** – наблюдаемый факт или запрос к БЗ, **h** – гипотеза для объяснения **f**. Тогда задача вывода формулируется как: **Th & f ├ h**. Возможными механизмами вывода являются:

Дедукция – найти **h**, если **Th** полностью определено;

Индукция – найти и оценить правдоподобность **h**, если правила вывода в **Th** не полностью определены, но фактам **f** можно доверять;

Абдукция – найти **h**, для объяснения **f** при неполно определенном **Th**;

Аналогия – **Th** по отношению к **f** и **h** не является полным, но известна аналогия в **Th´** между **f´** и **h´**;

Вероятностный и нечеткий вывод – **Th** и **f** заданы вероятностно или нечетко;

Нейросетевой и эволюционный вывод – **Th** и **f** неявно определяются архитектурой нейросети или правилами эволюции.

Вывод знаний удобно интерпретировать как задачу поиска в пространстве решений.

Если поиск является целенаправленным, а не случайным, то используется модель графа в виде И/ИЛИ-дерева. Задаётся множество начальных вершин графа, с которых поиск может начинаться, и множество конечных (целевых) вершин, при достижении которых поиск прекращается. И/ИЛИ-граф обладает следующими свойствами: при движении по входным дугам к некоторой вершине реализуется либо конъюнкция, либо дизъюнкция.

Методы поиска по дереву различаются по способу обхода путей на графе: поиск в глубину или в ширину (относительно порядка обхода вершин), прямой (от корня к висячей вершине) или обратный поиск в глубину; поиск без возврата или с возвратом; безусловный или условный поиск (следующий ход зависит от предыдущего); полный или сокращённый перебор ходов по дереву; поиск без

прогнозной оценки (метод проб и ошибок) или поиск с прогнозной оценкой (метод ветвей и границ, симплекс-метод и др.).

Для реальных задач с неполиномиальной оценкой сложности (*NP-задачи*) дерево поиска может стать вычислительно необозримым, экспоненциально растущим при линейном увеличении размерности задачи (комбинаторный взрыв). Сокращение размерности достигается либо разбиением задачи на фрагменты, либо использованием методов эвристического программирования.

Достоверный логический вывод. При достоверном логическом выводе используется такое понятие из математической логики как общезначимость (формула является общезначимой, если она истинна при любых значениях входящих в нее переменных). Для проверки общезначимости формулы используется алгоритм Квайна, доказательство от противного или таблица истинности.

Для описания символьных систем используются языки логики или исчисления. Исчисление – это знаковая система, состоящая из алфавита, аксиом и процедур вывода истинных (синтаксически правильных) формул из аксиом. Если символам языка приписать конкретные значения, то это формальный язык. Различают исчисление высказываний и исчисление предикатов.

В исчислении высказываний вопрос об истинности или ложности высказываний решается с помощью формул, которым подчиняются логические операций конъюнкции, дизъюнкции, отрицания, импликации и др. В естественном языке высказыванием может быть повествовательное предложение, о котором можно сказать, истинно оно или ложно.

В исчислении предикатов наряду с формулами исчисления высказываний, используются формулы, в которые могут входить отношения (предикаты), связывающие между собой группы элементов исчисления и кванторы общности и существования. Логика предикатов в отличие от логики высказываний позволяет количественно охарактеризовать связи между понятиями, их свойства и отношения. Язык логики предикатов — один из формальных языков, наиболее приближенных к человеческому языку.

Язык искусственного интеллекта Пролог основан на логике предикатов 1-го порядка. В основе языка Пролог лежит вывод по методу резолюций, который разработали Робинсон и Маслов. Рассмотрим вначале метод резолюций в исчислении высказываний.

Формула является символьным представлением высказывания. Формула без операций называется *атомарной*.

Чтобы судить об истинности формулы, необходимо связать её с содержанием, т.е. выполнить *интерпретацию* формулы. Нельзя просто сказать, истинна некоторая формула или ложна. Но если мы объявим высказыванием предположение, например, что «Земля вращается вокруг Солнца», то можно утверждать, что это высказывание истинно.

Интерпретацию будем обозначать буквой **j**, за которой в скобках указывается обозначение интерпретируемой формулы. Например, то, что формула **X** в интерпретации **j** истинна, можно записать в виде **j(X) = 1**.

Основная цель логического вывода состоит в доказательстве того, является ли данное утверждение следствием других.

Формула **G** называется *логическим следствием* формул $\mathbf{F_1, F_2, ..., F_k}$, если при любой интерпретации **j** из $\mathbf{j(F_1) = j(F_2) = ... = j(F_k) = 1}$ следует, что $\mathbf{j(G) = 1}$.

Понятие логического следствия тесно связано с понятием выполнимости.

Множество формул $\{\mathbf{F_1, F_2, ..., F_k}\}$ называется *выполнимым*, если существует интерпретация **j** такая, что $\mathbf{j(F_1) = j(F_2) = ... = j(F_k) = 1}$.

Проверить выполнимость множества формул $\{\mathbf{F_1, F_2, ..., F_k}\}$ можно, построив для них таблицы истинности. Если найдется хотя бы одна строка, в которой в столбцах формул $\mathbf{F_1, F_2, ..., F_k}$ стоят единицы, то это множество формул выполнимо. Если такой строки нет, то множество формул невыполнимо.

Идея метода резолюций основана на следующей теореме.

Теорема. Формула **G** является логическим следствием формул $\mathbf{F_1, F_2, ..., F_k}$ тогда и только тогда, когда множество формул $\mathbf{L = \{F_1, F_2, ..., F_k, \neg G\}}$ невыполнимо.

Доказательство теоремы сводится к тому, что некоторая формула **G** (её называют гипотезой теоремы), является логическим следствием множества формул $\mathbf{F_1, ..., F_k}$ (допущения теоремы). Текст теоремы может быть сформулирован следующим образом: если формулы $\mathbf{F_1, ..., F_k}$ истинны, то истинна и формула **G**.

Метод резолюций. Задача логического вывода сводится к задаче проверки выполнимости множества формул $\{\mathbf{F_1, F_2, ..., F_k, \neg G}\}$. Чтобы установить невыполнимость такого множества формул, применяется определенное правило. В этом правиле используются следующие понятия:

- литерал – атомарная формула или её отрицание;
- дизъюнкт – дизъюнкция одного или нескольких литералов;
- пустой дизъюнкт – специальный дизъюнкт, не содержащий литералов. Для его обозначения используется специальный символ **■.** Считается, что пустой дизъюнкт ложен при любой интерпретации;
- противоположные литералы **X** и **¬X**.

Правило резолюции формулируется так: из дизъюнктов (**X** v **F**) и (**¬X** v **G**) выводим дизъюнкт (**F** v **G**). Другими словами, дизъюнкт (**F** v **G**) является логическим следствием дизъюнктов (**X** v **F**) и (**¬X** v **G**). Правило резолюции – очень мощное правило! Многие правила, например, modus ponens являются частными проявлениями правила резолюции.

Резолюция – это приём, используемый при достоверном логическом выводе. Этот приём заключается в нахождении двух дизъюнктов, один из которых содержит литеру, а другой – ее отрицание. На основании их сравнения формируется новый дизъюнкт, называемый *резольвентой*. Именно порождение новых дизъюнктов, являясь основой метода резолюций, широко применяется в интеллектуальных системах.

Уточним понятие резолютивного вывода. Пусть **S** – множество дизъюнктов. *Выводом* из **S** называется последовательность дизъюнктов $\mathbf{D_1, D_2, ..., D_n}$ такая, что

каждый дизъюнкт этой последовательности принадлежит **S** или следует из предыдущих по правилу резолюции. Дизъюнкт **D** выводим из **S**, если существует вывод из **S**, последним дизъюнктом которого является **D**.

Применение метода резолюций основано на следующей теореме.

Теорема (о полноте метода резолюций). Множество дизъюнктов логики высказываний **S** невыполнимо тогда и только тогда, когда из **S** выводим пустой дизъюнкт ■ .

Процедура доказательства по методу резолюций на самом деле является процедурой опровержения, то есть вместо доказательства общезначимости формулы доказывается, что ее отрицание противоречиво, в значительной степени, такой подход выбран, исходя из соображений «технического удобства». Метод резолюций – это метод автоматического доказательства теорем, основанный на опровержении множества посылок ***F*** и отрицания целевой теоремы ***G*** с использованием правила резолюции.

Процедура резолютивного вывода состоит в доказательстве того, что формула **G** является логическим следствием множества формул $\mathbf{F_1,...,\ F_k}$ и включает следующую последовательность шагов.

1. Составляется множество формул $\{\mathbf{F_1,...,\ F_k,\ \neg G}\}$.
2. Каждая из этих формул приводится к конъюнктивной нормальной форме (КНФ). В КНФ зачеркиваются знаки конъюнкции. Получается множество дизъюнктов **S**.
3. Осуществляется поиск вывода пустого дизъюнкта ■ из **S**. Если пустой дизъюнкт выводим из **S**, то формула **G** является логическим следствием формул $\mathbf{F_1,...,\ F_k}$. Если из **S** нельзя вывести ■, то **G** не является логическим следствием формул $\mathbf{F_1,...,\ F_k}$.

Метод резолюций является частично корректной процедурой. Потому что возможен следующий исход: процесс не заканчивается, правило резолюции применяется, множество предложений пополняется, среди них нет пустых, и есть резольвируемые. Исход нельзя назвать зацикливанием, поскольку нет никакого способа определить, почему метод резолюции продуцирует новые резольвенты, но при этом не получается пустого предложения. Означает ли это, что теорема верна, но мы просто не дождались завершения? Не известно. Означает ли это, что в результате так никогда и не получится пустой формулы? И этого мы тоже не знаем. Это логически неразрешимая задача. В принципе нельзя узнать, как долго нужно ждать для того, чтобы получить ответ на этот вопрос.

Метод резолюций применим также к логике предикатов 1-го порядка. Для этого требуется внести дополнения к версии метода для логики высказываний. В частности, в исчислении предикатов используется понятие *переменной*. Поэтому правило резолюций необходимо дополнить возможностью делать *подстановку* переменной. В методе резолюций для логики предикатов используются ещё два правила склейки.

В остальном все сводится к проверке невыполнимости множества дизъюнктов: множество дизъюнктов **S** логики первого порядка невыполнимо тогда и только тогда, когда из **S** выводим пустой дизъюнкт ■ .

Одним из направлений в ИИ является разработка инструментального программного обеспечения для решения интеллектуальных задач: языков программирования, ориентированных на особенности задач ИИ; языков представления знаний; оболочек экспертных систем и других инструментальных средств.

Язык Пролог (*Prolog*) – это декларативный язык программирования для задач искусственного интеллекта, обработки естественных языков и др. В настоящее время имеется обширное семейство языков, построенных на базе языка Пролог, например: *Prolog++, Turbo Prolog* и др. В Прологе реализован частный случай линейной резолюции (на дереве вывода одна центральная ветвь и множество боковых ветвей).

Основной конструкцией Пролога являются дизъюнкты Хорна в обратной импликативной записи: $\mathbf{Q \leftarrow P_1 \& P_2 \& \ldots \& P_n}$, где **Q** – заголовок дизъюнкта (факты); $\mathbf{P_1 \& P_2 \& \ldots \& P_n}$ – тело (множество целей, которые надо доказать).

Пролог-программа интерпретирует метод поиска по дереву решений, причём интерпретатор произвольно выбирает дизъюнкты до получения пустого множества целей. Используется «поиск в глубину» (основной в Прологе), «поиск в ширину» и др.

На Прологе легко решаются разного рода головоломки, например, ханойская башня, задача о 8 ферзях, о волке, козе и капусте.

Ядро всякой экспертной системы можно рассматривать как систему автоматического поиска вывода в некоторой формальной теории, то есть как доказательство теорем. Доказательство теорем безусловно предполагает затрату интеллектуальных усилий, требующих не только дедуктивных навыков, но и в значительной степени интуитивных догадок (например, какие леммы необходимы для той или иной теоремы). К настоящему времени разработано несколько программ с элементами ИИ, способных работать в этом направлении, то есть программ автоматического доказательства теорем (*Automatic Theorem Proving*).

В основе работы этих программ положены: обогащенный язык исчисления предикатов первого порядка, алгоритм Тарского перечисления правильно построенных формул, который позволяет генерировать гипотезы исчерпывающим образом, метод резолюций.

Возникает подкупающая идея. Записать аксиомы предметной области на языке исчисления предикатов. Запустить алгоритм, который будет генерировать новые гипотезы одну за другой. Каждую гипотезу будем пытаться автоматически доказывать методом резолюций, и если получится, то компьютер сам будет добывать все новые и новые знания. Процесс можно запустить на 24 часа в сутки, 7 дней в неделю, 365 дней в году, и не на одном компьютере, а на всех незанятых в данный момент.

Однако такой прямолинейный подход оказывается безнадежно неэффективным. Сложные, интересные теоремы при лобовом подходе автоматически доказать не удается, потому что пространство перебора слишком велико. Тем не менее, автоматическое доказательство теорем – это фундаментальная проблема в ИИ.

Абдуктивный вывод в ИИ – вид правдоподобного вывода от частного к частному. Идея абдукции принадлежит Ч. Пирсу и состоит в следующем.

Пусть имеется некоторая группа фактов и гипотез, представляющих знания о предметной области **Th**. Если с их помощью можно объяснить имеющий место факт **f**, то они принимаются. Абдуктивный вывод имеет следующую схему. Пусть **h** – объяснение факта **f** в **Th**. Предположим, что **Th&f** – выполнимая формула, но **f** – не является следствием **Th**, т.е. знания о предметной области **Th** не объясняют факта **f**.

Необходимо вывести дополнительные факты ***h***, объясняющие ***f*** относительно ***Th***, т.е. ***Th, h |= f***, причем к фактам ***h*** предъявляются требования непротиворечивости и минимальности. Абдуктивное объяснение ***h*** для ***f*** непротиворечиво, если ***Th&h*** – выполнимая формула; минимальным – если каждое объяснение, являющееся логическим следствием ***f***, эквивалентно ***h***.

Например (*В.Н. Вагин*), пусть *Варвар* – имя слона. Тогда:

$Th: \forall x(Слон(x) \Rightarrow Серый(x))$

$f: Серый(Варвар)$

h: 1.$Слон(Варвар)$,

2.$Слон(Цезарь) \,\&\, Несерый(Цезарь)$,

3.$Слон(Варвар) \,\&\, Женщина(Старая\ леди)$

где 1 – минимальное и непротиворечивое абдуктивное объяснение факта ***f*** в ***Th***; 2 – противоречивое объяснение; 3 – избыточное объяснение.

Абдукция – это механизм формирования гипотезы, объясняющий наблюдаемые факты на основе существующих теоретических положений.

В настоящее время разработано несколько методов абдуктивного вывода. Одной из проблем, возникающих в процессе нахождения гипотез-объяснений в процессе абдуктивного вывода, является экспоненциальная зависимость числа возможных гипотез от размерности задачи. Вероятность многих из логически возможных гипотез объяснений настолько мала, что их не стоит рассматривать. Таким образом, нужно учитывать вероятности найденных гипотез объяснений и не создавать маловероятные гипотезы. Другая проблема заключается в том, что абдуктивный вывод имеет немонотонный гипотетический характер, допускает много интерпретаций. Современные языки логического программирования не позволяют в полной мере формализовать абдуктивный вывод.

Примером абдуктивного вывода являются рассуждения И. Кеплера. Он пришел к заключению, что орбиты планет представляют собой эллипсы, на основании наблюдения того, что значения долготы орбиты Марса не соответствуют предполагаемой круговой орбите. Естественно, прежде чем предположить, что орбиты планет описываются эллиптически, он проанализировал несколько других гипотез, не прошедших последующих проверок. Более того, Кеплер был вынужден сделать ряд достаточно сильных допущений, прежде чем открыл свой знаменитый закон «заметания площадей». Будучи сторонником гелиоцентрической гипотезы, он предположил, что именно Солнце является причиной наблюдаемого движения планет, а потом, опираясь

на наблюдения Марса, обобщил свои представления на другие планеты, предположив «одинаковость» состояний всех планет.

Вывод по абдукции успешно применяется для решения задач диагностики, понимания естественного языка, распознавания, накопления знаний, составления расписаний и, несомненно, является очень важной составляющей интеллектуальных систем.

Оригинальным отечественным методом правдоподобных рассуждений в ИС является *ДСМ-метод* (создан в 80-х г. в ВИНИТИ РАН В.К. Финном). В нем формализованы и автоматизированы индуктивные рассуждения Джона Стюарта Милля.

ДСМ-метод имеет 6 компонент: (1) условия применимости, (2) правдоподобные ДСМ-рассуждения, (3) представление знаний в виде квазиаксиоматических теорий, (4) средства исследования корректности и семантики рассуждений, (5) средства распознавания и вывода знаний, (6) интеллектуальную ДСМ-систему.

Условиями применимости ДСМ-метода является существование в базе позитивных (+)-фактов и негативных (−)-фактов; эмпирических причинно-следственных зависимостей, а также возможность формализовать сходство/различие фактов. Главный принцип автоматического порождения гипотез (АПГ) – сходство/различие фактов влечет наличие/отсутствие изучаемых эффектов и их повторяемость. Это – основа качественного (нестатистического) анализа данных. ДСМ-рассуждения являются синтезом трех способов вывода: абдукции, аналогии и индукции. ДСМ-рассуждение формализует естественный познавательный процесс: анализ данных (индукция) – предсказание (аналогия) – объяснение полученных результатов (абдукция). Представление знаний в ДСМ-методе осуществляется теорий, состоящих из аксиом, множества (±)-фактов, порождающих гипотез и множества правил вывода достоверных и правдоподобных знаний. Процедуры вывода по индукции, аналогии и абдукции могут быть представлены в виде декларативных аксиом. Их множество должно быть непротиворечивой теорией. Это обеспечивает логическую корректность ДСМ-метода.

Общая схема ДСМ-метода АПГ:
Th & **f** – база знаний и фактов
h = {$\mathbf{h}_1$ + $\mathbf{h}_2$} – множество гипотез
Th & f ├ $\mathbf{h}_i$ – гипотеза $\mathbf{h}_i$ объясняет факт **f** в **Th**

Следовательно, все гипотезы из **h** правдоподобны

Здесь $\mathbf{h}_1$ – гипотезы о (±)-причинах, порожденных индукцией, а $\mathbf{h}_2$ – гипотезы-предсказания, полученные по аналогии. Сама же схема представляет абдукцию – принятие правдоподобных гипотез путем объяснения фактов в предметной области **Th**.

Формирование базы фактов **f** и формулирование целей ДСМ-рассуждений являются междисциплинарной проблемой формирования аксиоматических теорий.

Исходное множество аксиом пополняется посредством обнаруженных при сравнении с расширяющимся множеством баз фактов ($БФ_1$,…, $БФ_m$): если гипотезы взаимно непротиворечивы, то обнаружен эмпирический закон; если же противоречий мало, то обнаружена эмпирическая тенденция. Они пополняют базу знаний ИС-ДСМ.

ИС – практический инструмент ДСМ-метода. Автоматическое порождение гипотез генерирует решатель задач, реализующий ДСМ-рассуждения, используя базу фактов и базу знаний в данной предметной области **Th**.

На основе ДСМ-метода работают около десятка оригинальных отечественных интеллектуальных систем в области диагностики различных заболеваний, анализа социологических данных, прогнозирования токсичности химических соединений, анализа криминалистических данных.

Вероятностный вывод – это вывод, при котором каждое выражение, используемое в нем, имеет оценку правдоподобия в виде вероятности того, что оно является истинным.

При вероятностном выводе применяются специальные процедуры для вычисления вероятности истинного значения результирующего выражения по вероятностям посылок, используемых при выводе, например байесовский вывод. Вывод базируется на теореме Байеса, которая «подправляет» вероятность гипотезы, данную новым свидетельством, следующим образом:

$$P(H|E) = \frac{P(E|H)\,P(H)}{P(E)},$$

где *H* представляет конкретную гипотезу, которая может быть, а может и не быть некоторой нулевой гипотезой. *P*(*H*) называется априорной вероятностью H, которая была выведена прежде, чем новое свидетельство E стало доступным. P(E |H) называется условной вероятностью наблюдения свидетельства E, если гипотеза H оказывается верной; её также называют функцией правдоподобия, когда она рассматривается как функция *H* для фиксированного E. P(E) называется маргинальной вероятностью E: априорная вероятность наблюдения нового свидетельства E согласно всем возможным гипотезам; может быть вычислена по формуле полной вероятности:

$$P(E) = \sum P(E|H_i)P(H_i)$$

как сумма произведений всех вероятностей любого полного набора взаимно исключающих гипотез и соответствующих условных вероятностей. *P*(*H*|*E*) называется апостериорной вероятностью *H* для данного *E*.

Например, есть две полных вазы конфет. В 1-й вазе 10 шоколадных и 30 простых конфет, в то время как во 2-й вазе по 20 конфет каждого сорта. Ваша подруга выбирает вазу наугад, и затем выбирает конфету наугад. Конфета оказалась простой. Нет никакой причины полагать, что Ваша подруга рассматривает одну вазу иначе, чем другую, аналогично и для конфет. Насколько вероятно, что подруга выбрала конфету из 1-й вазы?

Интуитивно, решение кажется ясным. Точный ответ дается теоремой Байеса. Пусть H_1 – выбор вазы 1, а H_2 – выбор вазы 2. Предполагается, что вазы идентичны и $P(H_1) = P(H_2) = 0{,}5$.

Событие E – наблюдение простой конфеты. Из содержания ваз известно, что $P(E|H_1) = 30/40 = 0{,}75$; $P(E|H_2) = 20/40 = 0{,}5$.

Формула Бейеса тогда даёт

$$p(H_1|E) = \frac{0{,}75 \times 0{,}5}{0{,}75 \times 0{,}5 + 0{,}5 \times 0{,}5} = 0{,}6.$$

До того, как мы наблюдали конфету, вероятность, которую мы назначили для подруги, выбиравшую 1-ю вазу, была априорной вероятностью $P(H_1) = 0{,}5$. После наблюдения конфеты нужно пересмотреть вероятность $P(H_1|E)$, которая теперь равна 0.6.

Нечеткий вывод оперируют с гранулированной информацией, вычисления производятся со словами, а не с числами. Гранулирование информации — есть процесс объединения схожих точек или объектов в одну группу. Тогда нечеткость подобных групп есть прямое следствие нечеткости понятия сходства. Простыми примерами таких групп являются понятия «средний возраст», «деловая часть города» и др. Отметим, что стремление объединять схожие по свойствам объекты в одну группу характерно для человеческих рассуждений. Понятие гранулированной информации находят свое отражение в теории нечетких множеств в терминах нечеткой и лингвистической переменной

Нечетким является вывод, при котором используются значения функций принадлежности. Правило вывода определяет значение этой функции для результата по значениям функций принадлежности посылок. Нечеткий вывод отличается от вероятностного подхода к представлению неопределенности знаний и базируется на теории нечетких множеств Л. Заде. Теория имеет свою аксиоматику и набор базовых операторов, действующих несколько иначе, чем аналогичные булевы операторы.

В частности, нечетким множеством называется множество, определенное на произвольном непустом множестве X как множество пар вида:

$$\tilde{A} = \{\mu_{\tilde{A}}(x)/x\}, \ \textit{где} \ x \in X, \ \mu_{\tilde{A}}(x) \in [0,\ 1].$$

Здесь $\mu_{\tilde{A}}(x)$ – степень принадлежности элемента x нечеткому множеству $\tilde{A}$.

Задание функции принадлежности нечеткому множеству осуществляется экспертами субъективно, аксиоматически или эмпирически.

Понятия функции принадлежности и вероятности часто путают, ведь они имеют одинаковый диапазон изменения от 0 до 1, и оба описывают меру неопределенности. Однако эти меры совершенно различны: функция принадлежности является описанием сложного состояния; дополнительные данные не изменяют её значения, а вероятности зависят от частоты события, поэтому последующие выборки могут изменить вероятность.

Операции конъюнкции и дизъюнкции в нечёткой логике также не имеют никакой связи ни с теорией вероятности, ни с булевой логикой. Степень принадлежности конъюнкции двух нечетких множеств $\tilde{A}$ и $\tilde{B}$ вычисляется как:

$$\mu_{\tilde{A}\&\tilde{B}}(x) = \min(\mu_{\tilde{A}}(x),\ \mu_{\tilde{B}}(x)).$$

Степень принадлежности дизъюнкции двух нечетких множеств $\tilde{A}$ и $\tilde{B}$ вычисляется как:

$$\mu_{\tilde{A}\vee\tilde{B}}(x) = \max(\mu_{\tilde{A}}(x),\ \mu_{\tilde{B}}(x)).$$

Отрицанием нечеткого множества A называется множество $\overline{A}$ с функцией принадлежности:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x),$$

для каждого $x \in X$.

Существуют различные алгоритмы нечеткого вывода: максиминная композиция, Мамдани, Сугено, Ларсена, Цукамото. Они различаются главным образом видом используемых правил, логических операций и разновидностью метода дефазификации.

Биоинспирированный вывод представляет собой математические преобразования, трансформирующие входной поток посылок в выходные решения по правилам, основанным на имитации природных процессов, а также на вероятностном подходе к исследованию ситуаций и итерационном приближении к цели.

Существует множество разновидностей эволюционного, инспирированного природой вывода, например, эволюционные и роевые алгоритмы. Общая схема биоинспирированного вывода заключается в следующем. Фиксируется множество объектов X (популяция решений), обладающих некоторыми параметрами и связанных друг с другом посредством определенной структуры. Среди этих объектов необходимо выбрать наилучшие в смысле некоторого критерия качества (оптимальности) F. Критерий оптимальности формируется на основе свойств объектов и не обязательно существует в виде аналитических выражений. Важно, что существует отображение вида

$$F: X \to R,$$

где R – множество действительных чисел и каждому объекту $x \in X$ из множества X сопоставляется значение критерия $F(x)$.

Фенотипическая природа исследуемого множества объектов произвольна, поэтому необходимо построить кодированное представление исходного множества объектов в другом, конечном множестве, обладающем структурой, например, векторного пространства S (генотип).

Отображение вида

$$\varphi: X \to S$$

описывает связь между исследуемыми объектами, которые выступают в качестве потенциальных решений задачи поиска и объектами, управление и манипулирование которыми осуществляет биоинспирированный алгоритм.

Существует обратное отображение вида

$$\varphi^{-1}: S \to X,$$

где каждому вновь сгенерированному элементу представления $s \in S$ соответствует элемент во множестве X.

Тогда, например процесс оптимизации с помощью алгоритма эволюционных вычислений, состоит в построении множества объектов-решений $X_{opt} \in X$, для которых выполняются следующие условия:

$X_{opt} = argmax\ F\ [\varphi^{-1}(s)],\ s \in S.$

Таким образом, в процессе оптимизации множество X развивается и эволюционирует к оптимальному состоянию, изменяя свой состав и параметры входящих в него объектов. Способ построения множества объектов $s \in S$ определяется биоинспирированным алгоритмом.

Особенностью биоинспирированного вывода является то, что в качестве множества S строится так называемое кодовое пространство – множество представлений объектов x в виде кодов (хромосом). Эволюция множества X задается эволюцией представления S. На множестве S определяется подмножество P_0 – случайная начальная популяция. Решение на каждом шаге эволюции определяется следующей разностной вычислительной схемой

$P_{t+1} = A(P_t),$

где A – композиция различных биоинспирированных операторов. Критерий оптимальности вычисляется на каждом шаге в процессе отбора решений по критерию, реализуемому в композиции операторов A.

Лекция 2. Машинное обучение в прикладных интеллектуальных системах: распознавание образов, гипертекстовый поиск, многоагентные и онтологические системы

Гипертекстовые Интернет-технологии и системы

Текст – универсальное средство представления, накопления и передачи знаний. Технологии работы с текстами на естественном языке (ЕЯ) – одни из важнейших для ИИ.

Гипертекст (ГТ) – это сеть текстов, в которой имеются ссылки на другие фрагменты текста. ГТ – одна из фундаментальных моделей представления знаний. Если обычный текст – это строка символов, читаемая в одном направлении, то ГТ многомерен, его можно читать «нелинейно», двигаясь по разным траекториям сети.

Гипертекстовая информационная технология (ГИТ) – это технология поиска и обработки семантической ГТ-информации.

ГТ-документ может быть не только электронным, но и бумажным (энциклопедия), если в нем расставлены ссылки. Библия – один из исторически первых ГТ-документов. Впервые систему, поддерживающую гипертекстовые возможности чтения и письма описал *В. Буш* (система *Memex*, 1945 г.). ГТ был задуман как социальная сеть отношений между сообществом авторов, в которой человек или программный агент мог устанавливать новые связи в тексте.

Д. Энгельбарт (изобретатель электронной мыши) предложил рассматривать отношения людей и программ как гетерогенное сообщество. Его проект *NLS* (60-е годы) был нацелен на расширение познавательных возможностей группы людей: групповые переговоры по системе электронной почты, личные настройки пользователей и т.п., которые нашли широкое применение относительно недавно.

Т. Нельсон работал над созданием системы электронных публикаций и всеобщего архива. Предложил термин «гипертекст». Подчеркивал, что гипертекст в его понимании не является иерархической структурой. Информационных структур не могут быть верно представлены иерархией из-за их параллелизма, перекрестных связей, одновременного присутствия одного элемента в нескольких местах. Гипертекст – это многоагентная структура, внутри которой могут существовать сложные неиерархические отношения между агентами.

Т. Бернерс-Ли – отец Всемирной паутины, автор концепции Семантического веба. В начале 90-х годов он разработал протокол *HTTP*, который позволил связать между собой документы, размещенные на компьютерах гипертекстовой сети Интернет. Паутина делает сеть полезной, поскольку люди на самом деле интересуются информацией и не хотят ничего знать про провода и компьютеры. Она существует, потому что программы работают и поддерживают постоянный обмен информацией между компьютерами. Центральным понятием

гипертекстовых систем является понятие навигации. Навигацией называется интерактивно управляемый пользователем процесс перемещения из одних узлов в другие. Методы навигации ориентируются не на компьютер, а на человека. Поэтому гипертекстовые системы являются системами антропоцентрического типа.

В основе формализованной модели гипертекста лежит понятие информационно-справочной статьи (ИСС) – это объект, имеющий смысловое единство и включающий локальные и глобальные гиперссылки. Графика и мультимедиа, содержащие гиперссылки, называют гиперграфикой и гипермедиа. Формализованная модель ГТ состоит из двух слоев. Первый слой – это отображение на экране содержимого документа с выделенными цветом гиперссылками. Во втором скрытом слое находятся адреса переходов.

В этой модели ИСС описывает кортеж $<x_0, x_1, \ldots, x_{11}>$, где x_0 –имя ИСС; x_1 – заголовок; x_2 – аннотация; x_3 – точка входа в ИСС; x_4 – текст-фрагменты ИСС; x_5 – цифровые объекты ИСС; x_6 – программные объекты; x_7 – справка по ИСС; x_8 – признак ускоренного просмотра; x_9 – признак детального просмотра; x_{10} – список локальных гиперссылок; x_{11} – список глобальных гиперссылок. Основной недостаток модели – отсутствие возможности явного определения типов гиперссылок.

Тезаурусная модель гипертекста имеет следующий вид:

$$\Gamma_0 = (T, I, S, Q)$$

где **T** – тезаурус гипертекста, состоящий из тезаурусных статей t_i; **I** – информационная составляющая гипертекста, состоящая из информационных статей I_i ($I = \cup I_i$); **S** – алфавитный словарь терминов; **Q** – список главных тем гипертекста.

Тезаурус – это семантически упорядоченный перечень терминов предметной области. Тезаурусная статья $t_i = \{m_i, Am_i\}$, где m_i – термин, Am_i – множество терминов, с которыми m_i связан семантическими отношениями **R**.

Лингвистами выделено около 200 семантических типов отношений. Наиболее часто используются 8 типов: синоним, род-вид, вид-род, часть-целое, целое-часть, причина-следствие, следствие-причина и ассоциация. Тезаурусная модель интерпретируется в виде графа (семантическая сеть), в узлах которой находятся текстовые описания терминов, а ребра сети указывают на связи между терминами.

Всю совокупность тезаурусных статей можно представить в виде сети, в узлах которой находятся текстовые описания объектов, а ребра сети указывают на существование связи между объектами и позволяют определить тип связи.

Оказалось, что все созданные человеком тексты построены по единым правилам. Никому не удается обойти их. Какой бы язык ни использовался, кто бы ни писал – внутренняя структура текста останется неизменной. Она описывается законами *Дж. Зипфа*. *Зипф* предположил, что слова с большим количеством букв встречаются в тексте реже коротких слов. Основываясь на этом постулате, *Зипф* вывел два универсальных закона:

- первый закон *Зипфа* «ранг – частота»;
- второй закон *Зипфа* «количество – частота».

Зипф опубликовал свои законы в 1949 г. Математик Б. Мандлеброт внес небольшие изменения в формулы Зипфа, добившись более точного соответствия теории практике. Без этих законов сегодня не обходится ни одна система поиска информации, позволяющая машине без участия человека распознать суть текста.

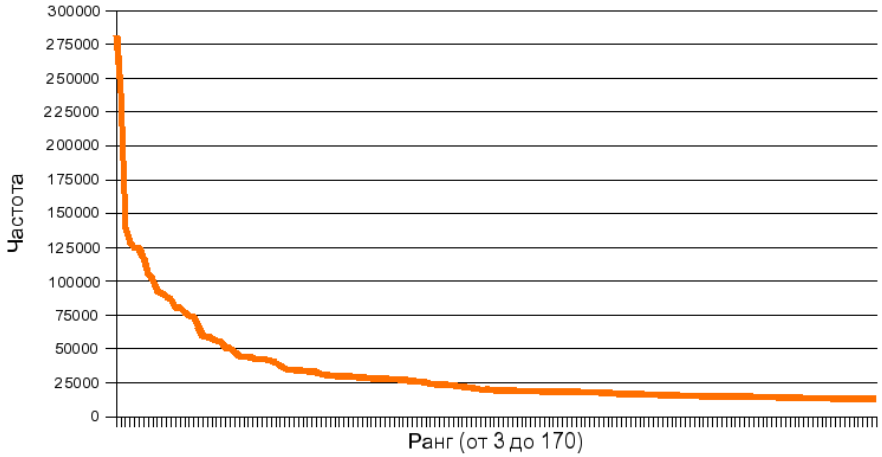
Первый закон Зипфа. Выбрав любое слово можно подсчитать, сколько раз оно встречается в тексте. Эта величина называется частотой вхождения слова. Далее, можно измерить частоту каждого слова текста. Некоторые слова будут иметь одинаковую частоту. Их нетрудно сгруппировать, пронумеровать и расположить группы по мере убывания их частоты. Порядковый номер частоты назовём рангом частоты. Наиболее часто встречающиеся слова будут иметь ранг 1, следующие за ними – 2 и т.д. Вероятность встретить в тексте наугад выбранное слово будет равна отношению частоты вхождения этого слова к общему числу слов в тексте:

Вероятность: = Частота вхождения слова / Число слов.

Обнаруживается интересная закономерность. Оказывается, если умножить вероятность обнаружения слова в тексте на ранг частоты, то получившаяся величина (*C*) приблизительно постоянна:

C: = *(Частота вхождения слова * Ранг частоты) / Число слов*.

Таким образом, ранг и частота связаны гиперболической зависимостью:

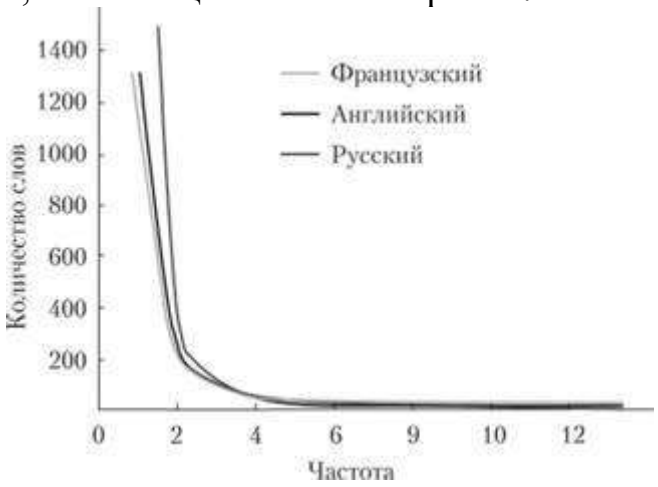


Следовательно, по 1-му закону Зипфа, если самое распространенное слово встречается в тексте, например, 100 раз, то следующее по частоте слово вряд ли встретится 99 раз. Частота вхождения второго по популярности слова, с высокой долей вероятности, окажется на уровне 50.

Значение константы *C* в разных языках различно, но внутри одной языковой группы остается неизменно, какой бы текст мы ни взяли. Так, например, для английских текстов константа *Зипфа* равна приблизительно 0,1. Русские тексты, с точки зрения закона, не выглядят исключением – закон безупречен, и тут коэффициент Зипфа равен 0,06–0,07.

В. Ли попытался опровергнуть закон, строго доказав, что случайная последовательность символов также подчиняется закону *Зипфа*. Он сделал вывод, что закон является чисто статистическим феноменом, не имеющим отношения к семантике текста. Хотя вывод *В. Ли* представляется недостаточно обоснованным, но сам по себе он интересен и проливает свет на природу открытой Зипфом закономерности.

Второй закон Зипфа. Разные слова входят в текст с одинаковой частотой. *Зипф* установил, что частота и количество слов, входящих в текст с этой частотой, тоже связаны между собой. Если построить график, отложив по оси абсцисс частоту вхождения слова, а по оси ординат – количество слов в данной частоте, то получившаяся кривая будет сохранять свои параметры для всех без исключения созданных человеком текстов! Причем межъязыковые различия невелики. На каком бы языке текст ни был написан, форма гиперболической кривой второго закона *Зипфа* останется неизменной. Могут немного отличаться лишь коэффициенты, отвечающие за наклон кривой:



Как с помощью законов *Зипфа* извлечь слова, отражающие смысл текста? – Наиболее значимые слова лежат в средней части гиперболы. В этой части также находятся часто встречающиеся предлоги, местоимения, артикли («шум»). Редко встречающиеся слова тоже зачастую не имеют решающего смыслового значения. От того, как будет выставлен диапазон значимых слов, во многом зависит результат поиска (широкий диапазон – нужные термины потонут в море вспомогательных слов; узкий диапазон – смысловые термины могут быть потеряны). Каждая поисковая система решает вопрос о выборе диапазона по-своему, руководствуясь общим объемом текста и словарями. Например, «шум» можно уменьшить путем предварительного исключения из исследуемого текста некоторых слов. Для этого создается словарь ненужных слов (стоп-лист). Современные способы индексирования не ограничиваются анализом перечисленных параметров текста. Поисковая машина может строить весовые коэффициенты с учетом местоположения термина внутри документа, взаимного расположения терминов, частей речи, морфологических особенностей и т.п. В качестве терминов могут выступать не только отдельные слова, но и словосочетания.

К сожалению, догадаться, по какой схеме работает та или иная поисковая система Интернета, очень трудно. Как правило, создатели держат ее в секрете. Здесь в простой форме изложены лишь основы работы поисковой системы. В реальности механизм индексации и структура базы документов сложнее:

- удаляем из текста стоп-слова;
- вычисляем частоту вхождения каждого термина без учета морфологии слов и регистра;
- формируем лист терминов в порядке убывания их частоты вхождения;

• выбираем примерно посередине диапазон частот, ориентируясь на конкретный смысл текста;

• из данного диапазона выбираем 10-20 терминов;

• составляем запрос и отправляем его поисковой системе.

В современных поисковых системах для описания процедуры поиска применяются следующие модели:

•поиск релевантной по составу и содержанию гипертекстовой статьи;

•поиск гипертекстовых статей с наиболее желательными свойствами, например наличие одинакового родового объекта;

•комбинированная модель.

Эффективность информационного поиска принято оценивать по информационной полноте ***(kП)*** и информационному шуму ***(kШ)*** на интервале [0, 1]. Идеальный вариант: полнота максимальна (***kП*** = 1), а шум нулевой (***kШ*** = 0).

Коэффициенты информационной полноты и шума определяются следующим образом:

$$k_{П} = \lim_{k\to m} \frac{1}{k}\sum_{i=1}^{k} \frac{|\, D_i \cap D_i^0 \,|}{|\, D_i \,|} \qquad k_{Ш} = \lim_{k\to m} \frac{1}{k}\sum_{i=1}^{k} \frac{|\, D^0{}_i \setminus D_i \,|}{|\, D^0{}_i \,|}$$

где m – достаточно большое число (по теореме о больших числах), $D^0{}_i$ – подмножество релевантных документов из D_i, полученных по i-запросу.

В идеале $D^0{}_i = D_i$. Это дает возможность ввести оценку эффективности информационного поиска:

$$E_1 = 2\lim_{k\to m} \frac{1}{k}\sum_{i=1}^{k} \frac{|\, D_i \cap D_i^0 \,|}{|\, D_i \,| + |\, D_i^0 \,|}$$

В литературе вместо ***kШ*** часто используют обратный эму показатель – коэффициент точности:

kТ =1 – ***kШ***

Тогда оценка эффективности информационного поиска равна:

E1 = 2***kТkП / (kТ + kП)***

В теории информационного поиска предложен обобщенный комплексный показатель эффективности ***Eβ*** (*мера Ван Ризбергена*), позволяющий учитывать предпочтение, отдаваемое пользователем точности или полноте:

$$E_{\beta} = \frac{(\beta^2 + 1) k_T k_{П}}{\beta^2 k_T + k_{П}}$$

где ***β*** – параметр, отражающий предпочтение пользователя ИПС (***β*** = [0, ∞]).

При **β** = 1 точность и полнота одинаково важны. На интервале [0, 1] приоритет имеет точность, а на интервале [1, ∞] – полнота.

Сравнение гипертекстовых, документальных и фактографических ИПС по характеристикам полноты и шума, показывает следующее:

- для документальных ИПС: $\mathbf{k_{Пmax}} = 0{,}5$; $\mathbf{k_{Шmax}} = 1$;
- для фактографических ИПС: $\mathbf{k_{Пmax}} = 1$; $\mathbf{k_{Шmax}} = 0$;
- для гипертекстовых ИПС: $\mathbf{k_{Пmax}} = 0{,}9 \div 1$; $\mathbf{k_{Шmax}} = 0{,}1 \div 0{,}2$.

Методы автоматизации поиска реализованы в информационно-поисковых системах, базах данных, базах знаний и поисковых машинах Интернет:

• в информационно-поисковых системах применяется индексирование текстов, поиск по ключевым словам (по индексу), поиск с морфологическим разбором различных грамматических форм слов, поиск с ранжированием документов по степени релевантности запросу, использование поисковых языков и комплексные методы;

• в базах данных и базах знаний наряду с перечисленными методами, применяются языки запросов (*SQL*-подобные языки), а также методы семантического анализа текста;

• в поисковых машинах Интернет применяются методы поиска по выборке, каталоги и семантические подходы с использованием методов искусственного интеллекта и машинного обучения. Например, утилита *Echo Search* на *Java*, каталоги ресурсов *Yahoo*! *Яндекс, Рамблер* и др., которые пополняются и незаменимы, когда человек не точно представляет цель поиска. Поисковый агент –интеллектуальный компонент поисковой машины. Различают агенты: *кроулеры* (просматривают заголовки страниц и возвращают машине только первую найденную ссылку), *роботы* (проходят по ссылкам с различной глубиной), *пауки* (сообщают о содержимом найденного документа, индексируют его и пересылают информацию в БД машины поиска).

Пока методы автоматического индексирования документов дают худшие результаты, чем авторское индексирование. Более эффективные решения связаны с использованием языка *XML*.

Существуют несколько коммерческих программных продуктов, которые реализуют технологии обработки текстов на естественном языке:

• смысловой анализатор текста *Text Analyst*;

• информационно-поисковая система *ERW*;

• пакет *NeurOK Semantic Suite*.

Анализатор *Text Analyst* – российский продукт, выполняющий «смысловой портрет» документа и автоматическое реферирование текста. Имеет удобный интерфейс. Разработан с использованием объектно-ориентированного подхода и *COM*-технологии. Позволяет настраивать русский и английский словарь на предметную область. Формируемый реферат требует ручного контекстного «сглаживания».

Информационно-поисковая система *ERW* определяет количественно «семантическое расстояние» между понятиями, находит документы, в которых идея запроса выражена по-другому. Характеристики *ERW*: логарифмический рост времени поиска при увеличении объема информации; поддержка около 20 серверных платформ и русскоязычный сервер; богатый набор команд, учет семантики, возможности нечеткого поиска, открытая архитектура.

Пакет *NeurOK Semantic Suite Analyst* реализует автоматическую рубрикацию документов; реферирование и аннотирование; мониторинг web-сайтов, персонализацию информации. Используется метод машинного обучения семантики языка. Реализует несколько видов поиска: по *SQL*-запросам; по ключевым словам; ассоциативный и др.

Пока создание подобных продуктов связано со значительными затратами и требует привлечения высококвалифицированных лингвистов, инженеров по знаниям и программистов.

Многоагентные системы

По классификации ACM (*Association for Computing Machinery*) многоагентные системы (МАС) являются разделом распределенного искусственного интеллекта (*Distributed Artificial Intelligence*) и представляют собой системы с взаимодействующими интеллектуальными агентами. Термин «интеллектуальный агент» (ИА) имеет два значения, и из-за этого иногда возникает путаница:

• в ИИ под интеллектуальным агентом понимается разумная сущность, наблюдающая за окружающей средой и действующая в ней, способная воспринимать среду посредством рецепторов и взаимодействовать с ней [*Рассел С., Норвиг П.*]. При этом ИА способен к пониманию, а его действия направлены на достижение какой-либо цели. Такой агент может быть как роботом, так и встроенной программной системой, взаимодействующей со средой примерно так же, как человек. ИА в смысле ИИ должен быть независимым, выполняя свои задачи.

• в программной инженерии ИА – это программа, самостоятельно выполняющая некоторое задание. Например, задание по постоянному поиску и сбору необходимой информации в Интернете (компьютерные вирусы, боты, поисковые роботы). Хотя такие агенты имеют строгий алгоритм, их «интеллектуальность» предполагает некоторую способность приспосабливаться и обучаться.

Агенты в искусственном интеллекте. В ИИ принято различать физических и временных агентов. Физический агент воспринимает окружающий мир через сенсоры и действует с помощью манипуляторов. Временной агент использует изменяющуюся с ходом времени информацию, предлагает действия или предоставляет данные компьютерной программе или человеку, а также может получать информацию через программный ввод.

Функция агента представляется как

f: P (результат восприятия) $\rightarrow$ A (действия).

Иными словами, агент проецирует результат восприятия на действия. В зависимости от типа обработки воспринимаемой информации принято различать агентов с простым поведением, агентов с поведением, основанным на моделях, целенаправленных, практичных и обучающихся агентов.

Агенты с простым поведением действуют только на основе текущих знаний. Их агентская функция основана на схеме продукционных правил типа «условие-действие»: *IF* (условие) *THEN* действие.

Агенты с поведением, основанным на модели, могут взаимодействовать со средой, лишь частично поддающейся наблюдению.

Целенаправленные агенты хранят информацию о тех ситуациях, которые для них желательны. Это дает агенту способ выбрать среди многих путей тот, что приведет к нужной цели.

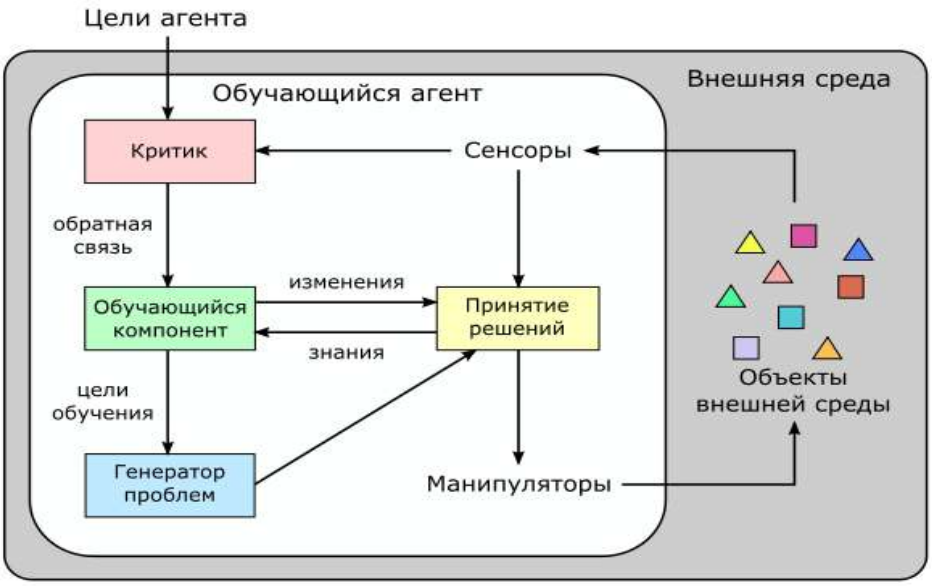
Практичные агенты различают, когда цель достигнута, и когда не достигнута, а также насколько желательно для них текущее состояние с помощью функции полезности.

Обучающиеся агенты (автономные интеллектуальные агенты) должны обладать некоторой независимостью, способность к обучению и приспособлению.

Чтобы активно выполнять свои функции, интеллектуальные агенты имеют иерархическую структуру, включающую много «субагентов», выполняющих низкоуровневые функции. Например, субагенты для принятия оперативных решения; сенсорные агенты, агенты типа распознавания речи; агенты, создающие базы данных для других интеллектуальных агентов; и др.

Политехнический институт Ренсселера совместно с компанией *IBM* создали программу-интеллектуального агента Эдди (*Eddie*). В агенте реализованы технологии машинного обучения и методы моделирования, в результате чего Эдди обладает интеллектом четырехлетнего ребенка и способен обучаться. Для того чтобы процесс обучения происходил в естественных условиях, ученые создали для него аватар, который живет в среде компьютерной игры *SecondLife*. Специалисты полагают, что, общаясь с другими аватарами, которые созданы реальными людьми, Эдди многое узнает и поймет.

Архитектура ИА имеет вид:



Агенты в программной инженерии. В программной инженерии науке различают достаточно ограниченное число агентов, которые могут считаться полуинтеллектуальными. Например, роботы по закупкам, пользовательские или персональные агенты, управляющие и наблюдающие агенты, добывающие информацию агенты.

Роботы по закупкам, просматривают сетевые ресурсы, собирают информацию о товарах и услугах. Например, *Amazon.com* является отличным

примером такого робота. Веб-сайт предложит вам список товаров, основываясь на том, что вы покупали в прошлом.

Пользовательские или персональные агенты – это агенты, которые действуют в ваших интересах, от вашего имени. Например, проверяют почту, сортируют её по важности, оповещают о поступлении важных писем; играют в компьютерной игре как ваш оппонент; собирают новости; и т.п.

Управляющие и наблюдающие агенты ведут наблюдение за компьютерными сетями, следят за конфигурацией каждого компьютера, подключенного к сети, или управляют ботами компьютерных игр (*Quake, Robot soccer, Hoshimi*).

Агенты, добывающие информацию, действуют в хранилищах данных, собирая информацию, и предупреждая о наличии новой информации.

Характерной чертой МАС является то обстоятельство, что она может быть использована для решения таких проблем, которые сложно или невозможно решить с помощью одного агента или монолитной системы. Основными характеристиками агентов в многоагентной системе являются автономность (агенты работают без непосредственного вмешательства со стороны), интерактивность (взаимодействие с другими агентами), реактивность, проактивность (сами являются источником возмущения для окружающей среды), целеустремленность (интеллектуальное поведение при достижении цели).

Обычно в МАС действуют программные агенты. Однако агентами МАС могут также быть роботы, люди или команды людей.

В МАС может проявляться самоорганизация и сложное поведение, даже если стратегия поведения каждого агента достаточно проста. Это лежит в основе так называемых алгоритмов роевого интеллекта.

Агенты могут обмениваться полученными знаниями, используя некоторый специальный язык и подчиняясь установленным правилам «общения» (протоколам) в системе. Примерами таких языков являются *Knowledge Query Manipulation Language* (*KQML*) и *FIPA*'s *Agent Communication Language* (*ACL*).

Современными направлениями изучения МАС являются координация, самоорганизация, мультиагентное обучение, надежность и устойчивость к сбоям.

Координация предназначена для согласования индивидуальных целей и вариантов поведения агентов, при которых каждый агент улучшает или не ухудшает значение своей функции полезности, а система в целом улучшает качество решения общей задачи. Методы решения задачи координации базируются на результатах теории управления, исследования операций, теории игр и планирования.

М. Месарович сформулировал базовые принципы координации в МАС. В их основе лежит понятие "совместных обязательств" (*commitments*) агентов, которое предполагает выполнение агентом действий, ведущих к достижению цели в интересах сообщества агентов. Одной из форм обязательств агента является его роль. Другое важное понятие – это общественные "соглашения" (*conventions*). Оно фиксирует условия, при которых обязательства выполняются,

и обстоятельства, когда агент может или должен отказываться от исполнения взятых на себя обязательств.

Известна гипотеза, которая утверждает, что "все механизмы координации в МАС могут быть выражены в терминах совместных обязательств агентов и соглашений". Ее справедливость показана на примерах координации при управлении воздушным движением в районе аэропорта и разрешении конфликтов.

Однако как, например, управлять системой, в которой информация поставляется миллионами сенсоров пока не очень понятно. Сейчас активно развиваются исследования в области самоорганизующихся систем. Самоорганизация – это динамические и адаптивные процессы, ведущие к поддержанию устойчивости системы без внешнего управления. В качестве прообразов часто используются примеры из области биоинспирированных моделей координации (роевые алгоритмы).

Основными средствами разработки МАС являются: *NetLogo* (кроссплатформенное программируемое окружение для программирования МАС); *VisualBots* (мультиагентный симулятор с синтаксисом *Visual Basic*); *MASON* (*Java*-библиотека для моделирования МАС); *REPAST* (набор инструментов для создания систем, основанных на агентах); JADE (*Java* библиотека для создания МАС); *SemanticAgent*; *CogniTAO – C++* (платформа разработки автономных МАС, ориентированная на реальных роботов и виртуальных существ).

Преимуществами МАС является увеличение производительности за счет асинхронного и параллельного исполнения; устойчивость к сбоям и надежность, масштабируемость и гибкость (легко добавлять новых агентов в систему), стоимость.

Примерами прикладных МАС, использующих идеи самоорганизации являются следующие системы.

Самоконфигурируемая оверлейной P2P сеть прикладных агентов (проект PAH). Сеть включает множество агентов, установленных в узлах сети. Они обмениваются между собой информацией и другими сервисами. Имеются агенты-посредники, которые помогают в поиске сервисов. Свои запросы на сервис любой агент направляет своему посреднику, который пытается найти нужный сервис у других агентов. Агенты могут появляться в сети и исчезать из нее. Оверлейная (виртуальная) P2P сеть прикладных агентов самоорганизуется с целью снижения нагрузки на сеть (снижения трафика на обмен сообщениями). По результатам выполнения запроса агент-посредник изучает, насколько удачен был ответ и добавляет позитивную метку при удаче и отрицательную – при неудаче. Когда число позитивных меток достигает порога, посредник обращается к текущему владельцу «не своего» агента с просьбой о его регистрации у себя, за что предлагает его владельцу награду. Через какое-то время образуются группы агентов. Этот подход хорошо работает в динамике и для большой размерности сети.

Известны примеры применения многоагентных систем в компьютерных играх, при создании кинофильмов, в оборонных системах, на транспорте, в

логистике, геоинформационных системах и др. МАС хорошо зарекомендовали себя в сфере сетевых и мобильных технологий, для обеспечения баланса нагруженности, расширяемости и способности к самовосстановлению сети.

Компьютерная программа или робот обретает статус агента тогда, когда у них имеются средства оперативного восприятия и интерпретации изменений среды, а также планирования и организации действий. Но это предполагает наличие механизмов мотивации, целеобразования, предвидения и пр.

МАС, как и ЭС, имеет базу знаний и подсистему рассуждений. Однако ее знания могут быть локальными, неполными, противоречивыми, нередко обновляются, а рассуждения выполняются ради подготовки действий. В то же время, МАС снабжены развитыми протоколами для переговоров между агентами.

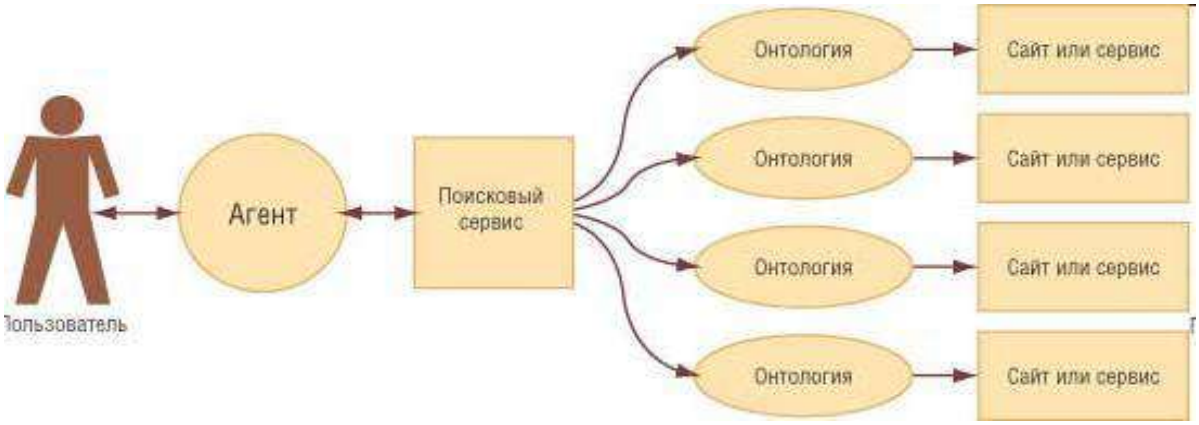
Следует также отметить, что развитие МАС существенно расширяет агентно-ориентированный подход в ИИ.

Онтологические системы

Онтология (от греч. *ontos* - сущее) – это описание на формальном языке понятий некоторой предметной области и семантических отношений между ними. Это знания о знаниях, т.е. метазнания.

В Интернете онтологии скоро станут обычным явлением. Сейчас в сети онтологии варьируются от больших таксономий по категориям веб-сайтов (как на *Yahoo*!), до каталогов продаваемых товаров и их характеристик (как на сайте *Amazon.com*).

В области медицины создан стандартный структурированный словарь *SNOMED* и семантическая сеть системы унифицированного медицинского языка (*UMLS*). Появляются общецелевые онтологии. Например, Программа ООН по развитию онтологии *UNSPSC*, которая предоставляет терминологию товаров и услуг.



Формально, онтологию *O* образует тройка множеств вида:

$O = < X, R, F >$,

где X – множество понятий (концептов, терминов); R – множество отношений между понятиями; F – множество интерпретаций понятий.

Другими словами, онтология представляет собой явную спецификацию предметной области. В неё входят множество понятий, отношения между

понятиями, атрибуты и свойства понятий, ограничения на свойства и атрибуты, экземпляры, знания из предметной области.

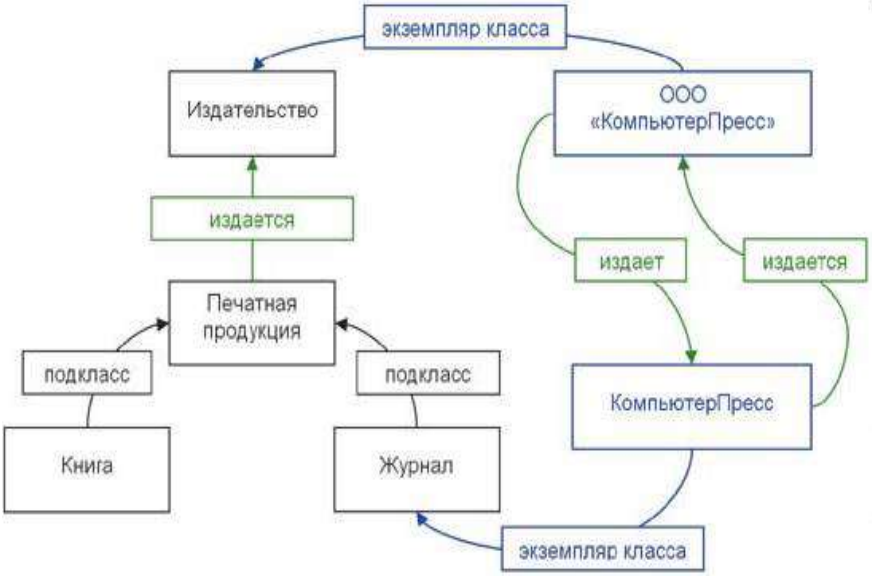
Частными случаями онтологии являются словарь ($X \neq \emptyset$, $R = \emptyset$, $F \neq \emptyset$), таксономия ($X \neq \emptyset$, $R = \{is\text{-}a\}$, $F \neq \emptyset$), тезаурус ($X \neq \emptyset$, $R = \{eqiv\text{-}to\}$, $F \neq \emptyset$), каталог-справочник с множеством понятий и ссылками на них.

Основные свойства онтологий состоят в следующем: существенность (охват ПО), непротиворечивость, независимость от реализации, декларативность, расширяемость, ясность.

Пусть, например, речь идёт об онтологии издательства, выпускающего книги и журналы. Основные понятия этой предметной области: *издательство*, *книга* и *журнал*. Это классы онтологии. Класс «*Издательство*» имеет следующие атрибуты: название (строка); город (строка). Класс «*Книга*» имеет следующие атрибуты: название; автор; ISBN; число страниц; тип обложки; издательство; год издания; описание; цена. Класс «*Журнал*» имеет следующие атрибуты: название; ISSN; число страниц; издательство; год выпуска; номер; описание; цена.

Классы «*Книга*» и «*Журнал*» имеют много общих атрибутов, вынесем их в отдельный класс «*Печатная продукция*».

Тогда получится следующий фрагмент онтологии:



Онтологии нужны для совместного использования людьми или программными агентами, для возможности повторного использования знаний в предметной области, для того чтобы сделать допущения в предметной области явными, для отделения знаний в предметной области от оперативных знаний, а также для анализа знаний в предметной области.

Правила вывода позволяют манипулировать понятиями, извлекать новые знания. Например, если существует книга, изданная в некотором году, то издательство, ее выпустившее, работает как минимум с этого года!

В общем виде процесс разработки онтологии включает следующие этапы:

- составляется глоссарий терминов (понятий);
- на естественном языке создается список точных определений терминов;

- на основе таксономических отношений строятся деревья классификации понятий (иерархии классов);
- из понятий, не задействованных при составлении деревьев классификации, выделяются атрибуты классов и их возможные значения. Эти понятия устанавливают связи между классами;
- в зависимости от целей, для которых разрабатывается онтология, в нее могут добавляться экземпляры классов;
- эксперты по той предметной области, в которой разрабатывается онтология, создают правила логических выводов, позволяющие оперировать данными, представленными в онтологии, и извлекать из созданной онтологии новые знания.

Разработка онтологий лишь напоминает проектирование классов в объектно-ориентированном программировании. Есть существенные отличия. Структура класса и отношения между классами в онтологии отличаются от структуры той же предметной области в объектно-ориентированной программе.

Существуют различные варианты классификации онтологических систем. Наиболее удачной представляется классификация онтологий по их содержимому.

В общем случае онтологическая система представляет собой совокупность трех множеств:

$$\sum = < O_{meta}, O_{ПО}, O_{выв} >,$$

где O_{meta} – онтология верхнего уровня (статична, не привязана к какой-либо предметной области), $O_{ПО}$ – предметная онтология, $O_{выв}$ – онтология, связанная с машиной вывода.

Основными вопросами инжиниринга онтологических систем является их создание, вопросы, на которые она будет способна отвечать, а также то, какую часть знаний она будет покрывать. Основными инструментами онтологического инжиниринга являются *Protégé, Ontolingua, Altova Semantic Works*.

Основными онтологическими подходами к выделению понятий являются:

- подход сверху вниз, когда сначала определяются наиболее общие концепты, затем они специализируются,
- подход снизу вверх, когда сначала определяются конкретные концепты, затем они объединяются в классы,
- смешанный подход, когда сначала выделяются наиболее очевидные понятия.

В настоящее время выделяются две основных области применения онтологий:

- проект Семантической сети (*Semantic Web*);
- область информационного поиска (*Information Retrieval*).

Суть проекта Семантической сети состоит в автоматизации "интеллектуальных" задач семантической обработки тех или иных ресурсов, имеющихся в Интернет. Обработкой и обменом информации должны заниматься не люди, а специальные интеллектуальные агенты (программы, размещенные в Сети). Но для того, чтобы взаимодействовать между собой, агенты должны

иметь общее формальное представление значения для любого ресурса. Именно для этой цели в *Semantic Web* используются онтологии.

По всем расчетам, переход к Семантической сети не будет ни простым, ни быстрым. Работы начались еще в 1998 г. Немало специалистов считают, что Семантический *Web* – это утопия. Чтобы разработчики сайтов начали внедрять поддержку онтологий в свои ресурсы, у них должен быть стимул. Использование онтологий должно давать сайту какие-то преимущества. Необходимы мощные и гибкие программы-агенты, которые смогут полноценно использовать возможности Семантического *Web*'а. Но их разработка начнется, если появится реальная необходимость в них.

Похоже, через несколько лет мы получим совершенно другой Интернет. Альтернатива внедрению Семантического *Web*'а сегодня только одна – дальнейшее ухудшение качества поиска, продолжение участия человека в интерпретации данных в *WWW*, а также другие проблемы. Основными элементами проекта Семантической сети являются языки *RDF* и *OWL*.

Язык *RDF* (*Resource Description Framework*) разработан консорциумом *W3C* для описания метаданных в семантической сети. *RDF* является подмножеством языка *XML*. Спецификация *RDF* включает отношения между ресурсами, которые определяются как триады «объект – атрибут – значение». Например, «книга издается издательством». В роли объекта выступает «книга», в роли атрибута – «издается», а «издательство» является значением атрибута. Объект и атрибут обязательно представляются как *URL* (*Uniform Resource Locator*) вместе с «якорем», указывающим объект на *Web*-странице. *RDF* является самым низкоуровневым из существующих языков описания метаданных, поскольку оперирует лишь понятиями связей примитивных сущностей.

Язык *OWL* (*Web Ontology Language*) основан на логике, позволяющей описывать классы через наборы свойств, которым должны удовлетворять объекты, относящиеся к понятию, и наборы логических операторов (конъюнкция, дизъюнкция, отрицание и др. Он выбран специалистами консорциума *W3C* в качестве базы для построения нового языка онтологий, когда средств *XML* и *RDF* оказалось недостаточно для представления информации и метаданных для построения полноценной Семантической сети. Именно ему, по мнению консорциума, уготована главенствующая роль в создании мировой Семантической сети, которая должна стать надстройкой над *WWW*.

Другими известными проектами онтологий являются метаонтологии и предметные онтологии. Например, метаонтологиями являются проекты *Cyc, WordNet, SUMO*. Среди предметных онтологий можно выделить *Protégé* (*protege.stanford.edu/ontologies.html*), *DAML* (*www.daml.org/ontologies*), *Ontolingua* (*www.ksl.stanford.edu/software/ontolingua/*), *Gene Ontology* (*www.geneontology.org*).

Онтология *OpenCyc* содержит информацию из различных предметных областей: философия, математика, химия, биология, психология, лингвистика и т.д. Файл с описаниями *OpenCyc* имеет объем около 700 мегабайт и доступен для

скачивания с сайта проекта (*http://www.opencyc.com*). Ключевым понятием в онтологии *OpenCyc* является коллекция.

Онтологию *DOLCE* предполагается применять в *Semantic Web* для согласования между интеллектуальными агентами, использующими разную терминологию. В основу проекта онтологии легло фундаментальное философское разделение всех сущностей на общие и частные:

Онтология *SUMO* содержит около 1 тыс. абстрактных понятий, а также 4 тыс. аксиом, определяющих эти понятия. Назначение *SUMO* – содействовать улучшению интероперабельности данных, извлечения и поиска информации, автоматического вывода и обработки естественного языка. Онтология охватывает общие виды процессов и объектов, абстракции теории множеств, единицы измерения, время, агентов и их намерения. Связана с *WordNet* – наиболее крупным тезаурусом, содержащим около 150 тыс. слов английского языка. Иерархия классов в *SUMO* менее запутана, чем в *OpenCyc*, и, возможно, более удобна для практического применения, чем *DOLCE*.

Онтология в области документации в сфере культурного наследия *CIDOC CRM* определяет семантику схем баз данных и структур документов, используемых в культурном наследии и музейной документации. Онтология предназначена для покрытия контекстной информации исторического, географического и теоретического характера об отдельных экспонатах и музейных коллекциях в целом.

Новым языком представления онтологий является *RDFS*. *RDFS* определяет классы, свойства и другие ресурсы. Например, определим свойство «Автор" с доменом "Документ" и диапазоном "Человек“:

- Класс ("Документ");
- Класс ("Человек");
- Свойство ("Автор", "Документ", "Человек").

RDFS определение класса или свойства (интенсионал) отделено от множества экземпляров класса и значений свойства (экстенсионала). Два класса с одинаковыми экстенсионалами считаются различными, если они имеют разные наборы свойств (интенсионалы). Пусть, например, A = {0, 2, 4, 6, 8}, B = {x | x = 2k, k = 0..4, k — целое}, C – множество неотрицательных четных чисел, меньших 10. Тогда множество A определяется через экстенсионал, а множества B и C – через интенсионал.

Языки представления онтологий не были бы востребованы, если бы не возникало необходимости автоматически обрабатывать онтологии, наполнять их содержимым и выполнять к ним запросы. Наиболее популярным языком запросов к *RDF*-хранилищам на сегодня является язык *SPARQL*.

Рассмотрим упрощенный синтаксис *SPARQL*-запроса:

SELECT <список_имен_переменных>
FROM <*URI*_ссылка_на_онтологию>
WHERE { <список_шаблонов>
FILTER <фильтр-ограничения_на_значения_
переменных>
}

Пусть сделан запроса (имена переменных предваряются знаком "?") следующего вида:

SELECT ?cat ?val
FROM <*URI*_онтологии>
WHERE { ?x rdf:value ?val. ?x category ?cat.
FILTER (?val>=200)
}

Семантика этого запроса такова: "Выдайте все объекты *cat* предиката *category*, субъект которого (*x*) является также субъектом предиката *rdf:value* со значением объекта *val*, не меньшим 200. Вместе со значениями *cat* выдайте соответствующие значения *val*".

Допустим, что онтология содержит следующие *RDFS*-триплеты:
(*Foo1*, *category, "Total Members"*),
(*Foo1*, *rdf:value*, 199),
(*Foo2*, *category, "Total Members*"),
(*Foo2*, *rdf:value*, 200),
(*Foo2*, *category, "CATEGORY X*"),
(*bar*, *category, "CATEGORY X*"),
(*bar*, *rdf:value*, 358)

Тогда ход выполнения запроса будет следующим:

- на место переменной *x* могут быть подставлены *Foo1*, *Foo2* и *bar* (из исходной онтологии).
- при подстановке *Foo1* значение переменной *val* не удовлетворяет ограничению в предложении *FILTER SPARQL*-запроса. Во всех остальных случаях все условия запроса выполнены.

Результатом выполнения запроса будут три следующих пары значений (*cat*, *val*):

[
["*Total Members*", 200],
["*CATEGORY X*", 200],
["*CATEGORY X*", 358]
]

Совместное использование информации людьми или программными агентами – одна из целей разработки онтологий. К примеру, различные веб-сайты содержат информацию по медицине. Если эти веб-сайты совместно используют одну и ту же базовую онтологию терминов, то компьютерные агенты могут извлекать информацию из этих сайтов, накапливать ее и использовать для ответов на запросы. Возможность использовать знания предметных областях – одна из движущих сил в изучении онтологий:

Создание явных допущений в предметной области дает возможность легко изменить эти допущения при изменении наших знаний о предметной области. Жесткое кодирование на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно изменить, особенно непрограммисту. Кроме того, явные спецификации знаний в

предметной области полезны для новых пользователей, которые должны узнать значения терминов предметной области

Отделение концептуальных знаний о предметной области от оперативных знаний – это еще один вариант применения онтологий. Например, можно использовать один и тот же алгоритм для проектирования устройства, если иметь онтологию компонентов устройства.

Системы распознавания образов

Теория распознавания образов базируется на *гипотезе компактности*: образам соответствуют компактные множества («сгустки» точек), объединенные в классы в пространстве признаков.

Образное восприятие мира — одно из самых загадочных свойств мозга. Воспринимая внешний мир, мы всегда проводим классификацию поступающей информации. Образы обладают характерными свойствами. Разные люди, одинаково и независимо друг от друга классифицируют одни и те же образы. Эта объективность образов позволяет людям понимать друг друга.

Область приложений теории распознавания образов весьма разнообразна: биоинформатика (поиск шаблонов в ДНК), базы данных (поиск и классификация данных), обработка текстов (тематическая классификация), анализ изображений, производство (контроль качества продукции), прогнозирование (погода, сейсмология, геология), биометрия (отпечатки пальцев), обработка речи, медицина (диагностика), военное дело (радиолокация объектов наблюдения), сельское хозяйство (размер урожая), космос (контроль космического пространства) и др.

Проблема распознавания образов интересна как с практической, так и с теоретической точки зрения. Для практики решение этой проблемы открывает возможность автоматизировать многие процессы, которые до сих пор связывали лишь с деятельностью мозга. Для теории это важно в связи с развитием идей искусственного интеллекта и машинного обучения. Что может и что принципиально не может делать машина? В какой мере возможности машины могут быть приближены к возможностям живого мозга?

Пока ясно только одно: если человек может осознать и описать процесс распознавания, то такое умение может быть передано машине. Если же человек обладает умением, но не может объяснить его, то остается только один путь передачи умения машине – обучение на примерах.

Наиболее понятной является *пространственно-временная интерпретация* проблемы обучения распознаванию образов. Любой объект можно представить в виде точки некоторого пространства признаков. Например, если утверждается, что при показе изображений можно однозначно отнести их к одному из двух образов, то в некотором пространстве существует две области, не имеющие общих точек, то изображения – точки из этих областей. Каждой такой области можно дать название, соответствующее образу.

Заранее считается известным, что требуется разделить две области в некотором пространстве и что показываются точки только из этих областей. Однако сами области могут быть заранее не определены, т.е. нет сведений об их

границах или правилах принятия решения о принадлежности точки к той или иной области. В ходе обучения предъявляются точки, случайно выбранные из этих областей, и сообщается информация о том, к какой области принадлежат предъявляемые точки.

Цель обучения состоит либо в построении поверхности, которая разделяла бы не только показанные в процессе обучения точки, но и все остальные точки, принадлежащие этим областям, либо в построении поверхностей, ограничивающих эти области так, чтобы в каждой из них находились только точки одного класса.

Задачу, например, можно решить путем построения функции, принимающей над точками каждой из областей одинаковое значение, а над точками из разных областей – различные значения функции.

На первый взгляд кажется, что знание некоторых точек из области недостаточно, чтобы отделить всю область. Однако это известная задача аппроксимации функций. Ее решение требует введения ограничений на класс рассматриваемых функций. Выбор этих ограничений зависит от информации, которую может дать учитель в процессе обучения.

Одной из таких подсказок является гипотеза о компактности образов. Интуитивно ясно, что задача аппроксимации разделяющей функции тем легче, чем более компактны и более разнесены в пространстве области, подлежащие разделению. Необходимо построить такие функции от векторов-изображений, которые были бы, например, положительны на всех точках одного и отрицательны на всех точках другого образа. Поскольку области не должны иметь общих точек, то существует множество таких разделяющих функций, а в результате обучения должна быть построена одна из них.

Если предъявляемые образы принадлежат не двум, а большему числу классов, то задача состоит в построении по показанным в ходе обучения точкам поверхности, разделяющей друг от друга все области, которые соответствуют этим образам. Задача эта может быть решена, например, путем построения функции, принимающей над точками каждой из областей одинаковое значение, а над точками из разных областей значение этой функции должно быть различно.

Задачи распознавания образов:

1. Определение полного перечня признаков, характеризующих образ. Признаки подразделяются на детерминированные (конкретные числа), вероятностные, логические и структурные;

2. Первичная классификация образов и составление априорного алфавита классов, т.е. выбор принципа классификации;

3. Разработка априорного словаря признаков (на основе задачи 1), в который вносят лишь те признаки, по которым может быть получена априорная информация;

4. Описание априорного алфавита классов на языке априорного словаря признаков;

5. Разбиение пространства признаков на области по классам алфавита с помощью решающих функций (трудоемкая задача по сложности практически равносильная задаче распознавания образов);

6. Выбор и применение алгоритмов распознавания (вычисление мер сходства/различия путем оценки расстояния от распознаваемого объекта до каждого из классов);

7. Определение рабочих алфавита классов и словаря признаков.

8. Разработка специальных алгоритмов управления системой распознавания образов (СРО);

9. Выбор показателей эффективности работы СРО (вероятность правильного решения, время распознавания, величина ресурсов).

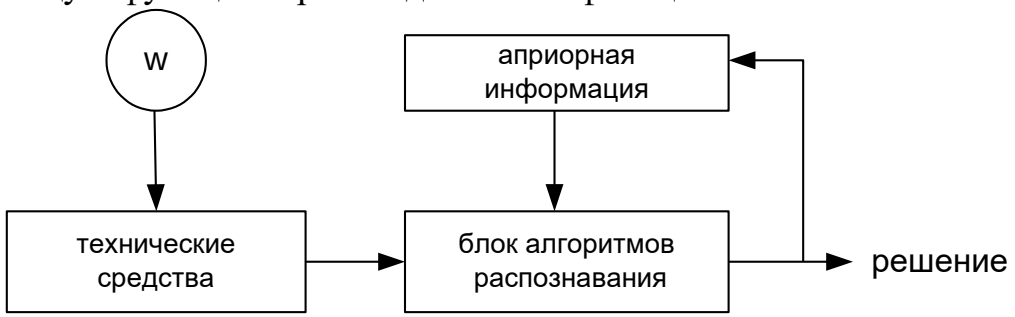
Таким образом, автоматизация обучения и распознавания неизвестных объектов составляет постановку общей задачи (проблемы) распознавания образов. Она состоит в следующем: *в условиях первоначального описания классов на языке признаков необходимо в пределах выделенных технических средств и ресурсов определить оптимальный алфавит классов и оптимальный рабочий словарь признаков, которые при наилучшем решающем правиле обеспечивают эффективное использование решений, принимаемых по результатам распознавания*.

В зависимости от объекта распознавания принято различать текстовые системы распознавания (*text recognition*), системы распознавания речи (*speech recognition*), системы распознавания графики (*image recognition*) и др.

По достаточности для распознавания количества априорной информации различают СРО без учителя, системы с учителем и самообучающиеся системы.

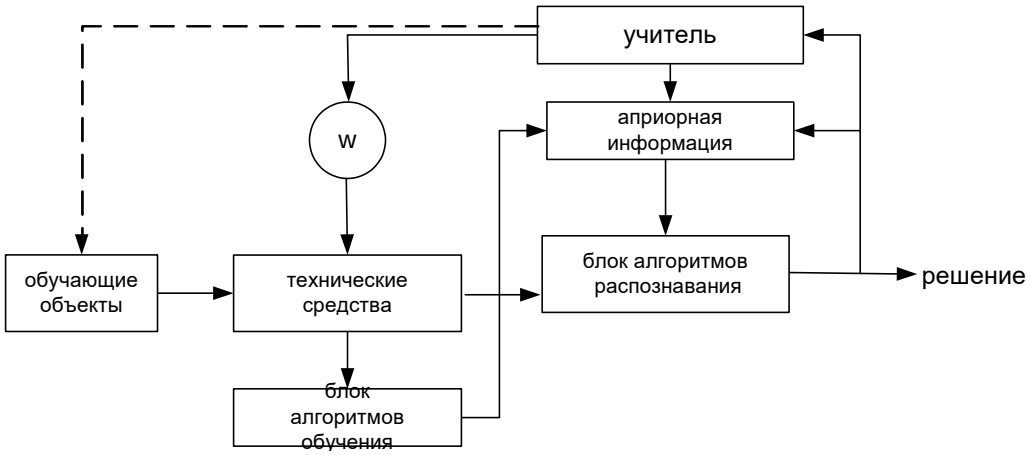
По типу априорного/рабочего словаря признаков различают детерминированные, вероятностные, логические и структурные СРО.

Например, в СРО без учителя количество априорной информации достаточно, чтобы определить априорный алфавит классов, априорный словарь признаков и решающие правила. Это означает, что удалось найти общее свойство, не зависящее природы образов. Тогда задача формулируется так: системе предъявляются объекты без указаний об их принадлежности к образам. СРО отображает множество объектов на множество их образов и, используя решающую функцию производит классификацию объектов:



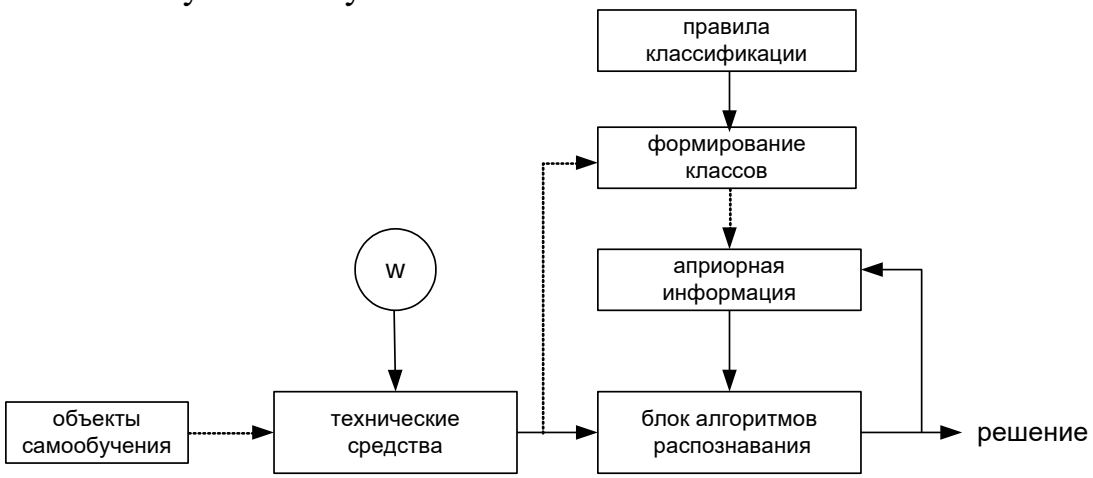
Главная черта, делающая обучение без учителя привлекательным, – это его "самостоятельность".

В системах с учителем априорной информации достаточно, чтобы определить априорные алфавит классов и словарь признаков, но недостаточно для описания классов на языке признаков:



В системах с учителем обучение представляет собой процесс выработки в системе той или иной реакции на группы образов путем многократной корректировки. Такую корректировку в обучении принято называть "поощрениями" и "наказаниями". Механизм генерации этой корректировки практически полностью определяет алгоритм обучения, в котором системе сообщается информация о верности ее реакции**.**

В самообучающихся системах априорной информации достаточно лишь для определения априорного словаря признаков, но недостаточно для классификации объектов, т.е. система не получает указаний о принадлежности объекта к какому-то классу:



Большинство известных алгоритмов самообучения способны выделять только абстрактные образы, т.е. компактные множества в заданных пространствах. Это повышает их ценность, поскольку часто образы заранее не определены, поэтому надо определить, какие изображения в заданном пространстве представляют собой образы. Примером такой постановки задачи являются социологические исследования, когда по набору вопросов выделяются группы людей.

Рассмотрим особенности детерминированных, вероятностных, структурных и логических СРО.

Детерминированные СРО. В детерминированных СРО образы рассматриваются как точки в **n**-мерном пространстве $\mathbf{R^n}$. Задача состоит в

определении принадлежности образа $\mathbf{X} \in \mathbf{R^n}$ одному из классов $\mathbf{W} = \{w_1, w_2, \ldots w_k\}$ путем построения в n-мерном пространстве **k** областей $\mathbf{C_i}$, i=1..k, причем $\mathbf{X} \in \mathbf{C_i}$.

Правило классификации: $\mathbf{X} \in \mathbf{w_i}$, если $\mathbf{X} \in \mathbf{C_i}$. Если имеется **k** классов, то необходимо построить **k** взаимно непересекающихся областей. Единственным способом решения этой задачи является нахождение ***решающей функции***. В этом случае каждая область $\mathbf{C_i}$ будет описываться с помощью системы нелинейных неравенств:

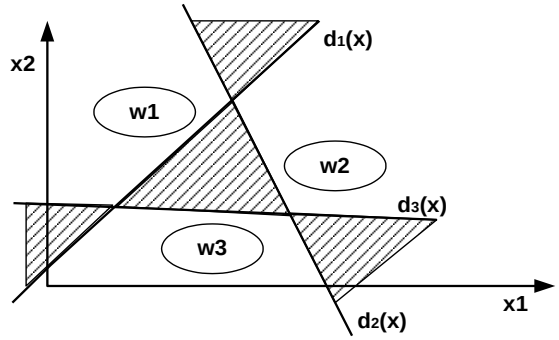
$$\begin{cases} d_1(\overline{X}) > 0, \\ d_2(\overline{X}) \le 0, \\ \ldots \\ d_m(\overline{X}) \ge 0; \end{cases}$$

где d_i – решающая функция

На практике чаще всего стараются применять линейные решающие функции вида: $d(X, W) = XW = x_1w_1 + \ldots + x_nw_n + w_{n+1}$, где вектор $W=(w_1, w_2, \ldots, w_{n+1})$ – параметры или «веса».

Построение линейных решающих функций связано с решением систем неравенств. В общем случае система может оказаться *несовместной* (множество решений пусто), классы являются линейно неразделимыми, и решение необходимо искать для нелинейных функций.

Варианты разделения на классы могут быть различными. Например, для каждого из классов ищется решающая функция, которая отделяет его от всех других классов. Вся плоскость разбивается на 7 областей. Заштрихованные области называются областями неопределенных решений:



Человек при классификации интуитивно чувствует, к какому классу ближе расположен некоторый образ. В алгоритмах компьютерной классификации для детерминированных СРО используют *функцию расстояния*, которая выражает степень близости распознаваемого образа к классу.

В кластерном анализе существуют несколько способов определения функции расстояния. Критерий качества кластеризации следующие:

а) внутри групп объекты тесно связаны между собой;

б) объекты разных групп должны быть далеки друг от друга;

в) распределение объектов по группам желательно равномерное.

Требования а) и б) выражают гипотезу компактности классов; требование в) должно препятствовать объединению отдельных групп объектов.

Главным в кластерном анализе считается выбор метрики близости объектов. На практике для вычисления расстояния *d* применяют разные метрические

нормы: евклидово расстояние; манхэттенское расстояние; метрику Махаланобиса и др. В конкретной задаче выбор свой, с учетом целей исследования, физической и статистической природы используемой информации и т.п., на основе специальных алгоритмов преобразования исходного пространства признаков.

Пусть $\mathbf{w_i}$ — i-й класс объектов, $\mathbf{N_i}$ — число объектов, образующих класс $\mathbf{w_i}$, вектор $\mathbf{m_i}$ — ср. арифметическое объектов, входящих в $\mathbf{w_i}$ («центр тяжести»), а $\mathbf{q(w_l,w_k)}$ — расстояние между группами $\mathbf{w_l}$ и $\mathbf{w_k}$.

Тогда примерами наиболее распространенных расстояний и мер близости являются следующие:

1. Расстояние между центрами тяжести кластеров $\mathbf{d(ЦТ_l, ЦТ_k)}$
2. Расстояние до ближайшего соседа $\mathbf{min\ d(}x_i, x_j\mathbf{)}$, где x_i из $\mathbf{w_l}$, x_j из $\mathbf{w_k}$.
3. Расстояние до дальнего соседа $\mathbf{max\ d(}x_i, x_j\mathbf{)}$, где x_i из $\mathbf{w_l}$, x_j из $\mathbf{w_k}$.

Алгоритмы, основанные на расстоянии ближайшего соседа (2), хорошо работают в случае группировок, имеющих сложную, в частности, цепочечную структуру. Расстояние дальнего соседа (3) применяется, когда классы образуют в пространстве признаков шаровидные облака. Алгоритмы, использующие расстояния центров тяжести (1), лучше всего работают в случае группировок эллипсоидной формы.

Если гипотеза о типе кластеров неверна, то это может приводить к неоптимальным или даже неправильным результатам. Поэтому в условиях априорной неопределенности применяют комплекс алгоритмов кластеризации.

Наиболее популярными алгоритмами кластеризации являются:

- алгоритм *k*-внутригрупповых средних;
- алгоритм максиминного расстояния;
- алгоритмы *FOREL, SKAT, KOLLAPS, BIGFOR, ROST, DINA*;
- алгоритмы *ISODATA*.

Алгоритмы кластеризации, в основном включают два этапа.

1. Задается исходное разбиение объектов на классы и определяется некоторый критерий качества автоматической классификации;
2. Объекты переносятся из класса в класс пока критерий не перестанет улучшаться.

Например, алгоритм *FOREL*, использует расстояние до центра тяжести, включают следующие этапы.

1. Признаки образов нормируются на интервале [0, 1].
2. Строится гиперсфера минимального радиуса R_0, охватывающая все *m* точек.
3. $R'_0 = 0.9R_0$.
4. Центр гиперсферы перемещается в любую из внутренних точек (расстояние до которых меньше R'_0) и вычисляется новый центр тяжести.
5. Центр сферы переносится в новый центр тяжести и вновь выбирается подходящая внутренняя точка в качестве центра гиперсферы.

В итоге центр сферы перемещается в область локального сгущения точек. Когда сфера остановится, то все ее внутренние точки образуют кластер ωl, они исключаются из дальнейшего рассмотрения, а процедура повторяется с

оставшимися точками до тех пор, пока все точки не будут распределены по кластерам. Если число кластеров заранее задано, то радиус сферы уменьшается или увеличивается на некоторую величину.

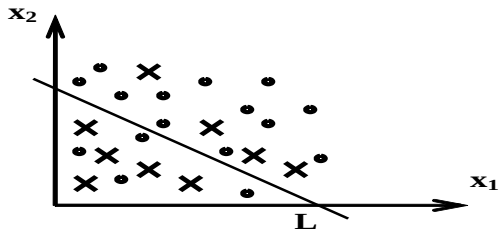
Если размер кластеров динамически меняется во времени, то для кластеризации возникающих объектов применяется алгоритм *DINA*. Согласно этому алгоритму, задается радиус R. Вновь появившаяся точка объявляется центром первого кластера. Если появляется следующая точка, то проверяется, попадает ли она внутрь гиперсферы. Если попадает, то она включается в состав кластера, центр гиперсферы смещается в центр тяжести кластера. Далее процесс повторяется. Можно контролировать, чтобы кластеры не «переполнялись», разбивая их пополам. В целом этот процесс перехода от описания исходных объектов к описанию кластеров напоминает процесс перехода от данных к знаниям.

Алгоритмов много, потому что много критериев качества кластеризации. Простые критерии базируются на величине расстояния между кластерами. Они не учитывает «населенность» кластеров. Другие критерии основываются на вычислении средних расстояний между объектами внутри кластеров. Наиболее часто применяются критерии в виде отношений «населенности» кластеров к расстоянию между ними.

Функционалы качества и конкретные алгоритмы автоматической классификации достаточно полно и подробно рассмотрены в специальной литературе. У них различная трудоемкость, подчас они требуют ресурсов мощных компьютеров. Алгоритмы кластерного анализа входят в состав практически всех современных пакетов программ для статистической обработки многомерных данных.

Вероятностные методы распознавания. Детерминированный подход к распознаванию не позволяет понять такие важные понятия в теории распознавания образов, как обучение и обобщение. Обобщение заключается в умении распознавать образы, которые не встречались во время обучения. СРО, не способная к обобщению, не способна и обучаться.

Ограниченность детерминированных СРО можно преодолеть с позиции теории вероятности. В этом случае задача классификации образов заключается в минимизации вероятности ошибочного решения в заданном классе решающих функций:



На этом рисунке образы располагаются так, что нельзя построить классификатор, дающий безошибочное решение. Такая ситуация является нередкой для распознающих систем. Она может быть вызвана случайными помехами, отсутствием четких границ между классами или недостаточно полной информации об анализируемых объектах. Решающую функцию следует выбрать

так, чтобы вероятность ошибочных решений была минимальной. Вероятность ошибочного решения $\boldsymbol{Q}$ можно оценить по обучающей выборке по следующей формуле:

$\boldsymbol{Q = v/N}$, где $\boldsymbol{v}$ - число образов в обучающей выборке из $\boldsymbol{N}$ объектов, которые классифицируются не правильно. Очевидно, что мы не можем оценить точно качество классификатора. Однако известно, что данная оценка сходится по вероятности к истинному значению при $\boldsymbol{N}\rightarrow\infty$.

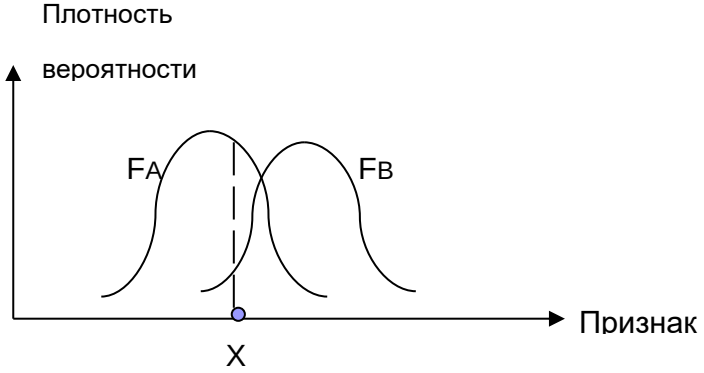
Как применять статистические методы распознавания? Пусть дана обучающая выборка (ОВ). Каждый образ в ОВ включает набор из n характеристических признаков и описывается n-мерным вектором. Необходимо построить правило классификации.

Чтобы решить эту задачу, необходимо знать функцию распределения элементов ОВ в n-мерном пространстве. Рассмотрим варианты. Их может быть три:

1. Функции вероятностного распределения известны.
2. Известен только тип функций распределения. Неизвестны их параметры (математическое ожидание, среднеквадратичное отклонение).
3. Распределение неизвестно.

Рассмотрим каждый из этих вариантов.

Если функции вероятностного распределения известны, тогда необходимо просто применить правило Байеса. Например, пусть даны два распределения A и B:



Значение X порождается одним их этих распределений. Каким из них? – Интуиция подсказывает, что надо выбирать A, т.к. P(A/X) > P(B/X). Это же утверждает правило Байеса:

$P(A/X) = P(X/A)*P(A)/(P(X/A)*P(A)+P(X/B)*P(B)) =$

$=F_A(X)*P_A/(F_A(X)*P_A+F_B(X)*P_B).$

Надо выбирать A, если $F_A(X)*P_A > F_B(X)*P_B$.

Если тип функций распределения известен, но неизвестны их параметры, то необходимо вначале оценить параметры. Оценка может быть точечной или интервальной. Затем применить правило Байеса.

Если функции распределения заранее неизвестны, то необходимо следующее.

1. Выдвинуть гипотезу о распределении.
2. Построить распределение в соответствии с гипотезой.

3. По обучающей выборке одним из известных статистических методов принять/отвергнуть гипотезу.

4. Если отвергли правильную гипотезу, то это ошибка 1-го рода, она должна быть меньше α.

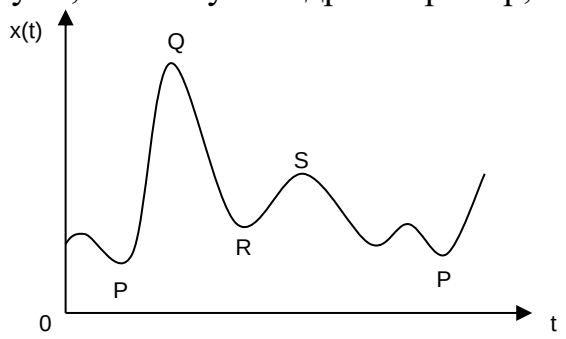
5. Если приняли неверную гипотезу, то это ошибка 2-го рода. Например, если T<t, то принимаем гипотезу. Если T ≥ t, то отвергаем. Значение t определяется как функция от α и *n*.

6. Проверить гипотезу, например, по критерию χ2.

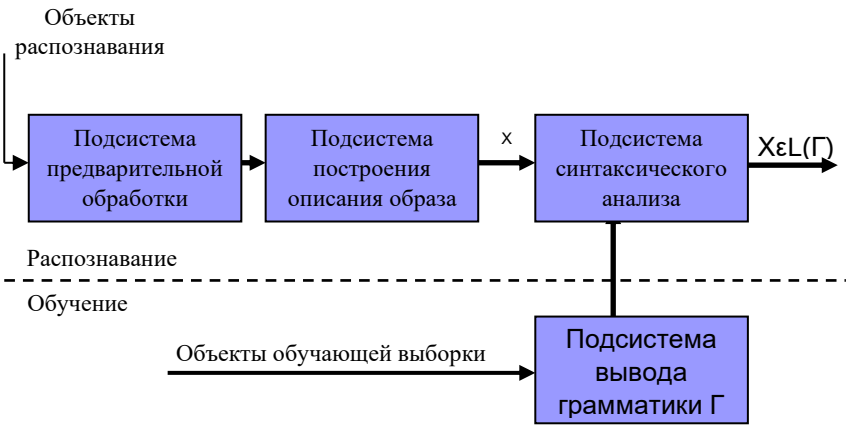
Для проверки гипотезы применяются различные статистические методы. Например, метод гистограмм, метод Парзена и др.

Структурные методы распознавания образов. Зачастую на практике информация о распознаваемых объектах содержится в записях сигналов. Традиционно для определения признаков используют разложения сигналов в ряды по ортогональным функциям (ряды Фурье, полином Чебышева и т.д.).

Возможно использование в качестве характерных признаков точек минимума, максимума и др. Например, на электрокардиограмме:



Методы распознавания, использующие признаки структурных элементов позволяют избежать не только трудоемкой процедуры разложения в ряды, но и потерь информации. Структурные методы основаны на теории формальных языков Н.Хомского. Идея, как формальных языков, так и структурных методов распознавания состоит в построении сложного объекта в виде иерархической *регулярной* структуры более простых образов. При этом используется известная схема: задаются исходные объекты и порождаются новые по некоторым правилам. Этапы структурного распознавания включают:



На этапе обучения грамматику можно определить, используя априорные сведения об объектах обучающей выборки.

На этапе распознавания предварительная обработка заключается в кодировании, сжатии, фильтрации или восстановлении объекта. Затем объект представляется языковой структурой (цепочкой или графом): сегментация и выделение простейших элементов. Решение о правильности объекта вырабатывается синтаксическим анализатором по дереву грамматического разбора. Распознавание – это сопоставление с эталоном.

Грамматика включает терминальный и нетерминальный словари, начальный символ и набор правил подстановки. Терминальный словарь – это набор исходных элементов, из которых строят цепочки, порождаемые грамматикой. Нетерминальный словарь – это набор символов, которыми обозначаются классы исходных элементов, а также специальные вспомогательные символы. Начальный символ – это выделенный нетерминальный символ, обозначающий класс объектов, для описания которых предназначена грамматика. Правила подстановки – это выражения вида «$x \rightarrow y$» (заменить x на y), где x и y – цепочки с любыми терминальными или нетерминальными символами. Если имеется последовательность цепочек $x_0, x_1, \ldots, x_n$, в которой каждая последующая выводима из предыдущей, то она называется *выводом* x_n из x_0 в грамматике Γ.

Грамматика – это не алгоритм, поскольку порядок применения правил подстановки произволен!

Совокупность всех терминальных цепочек, выводимых из начального символа, называется языком, порожденным грамматикой Γ, и обозначается $L(\Gamma)$. Две разные грамматики могут порождать один и тот же язык.

Допустим, что имеем дело с 2 классами объектов Ω_1 и Ω_2, которые описываются конечным множеством признаков (словарь). Пусть существует грамматика Γ такая, что порождаемый ею язык состоит из предложений (объектов), принадлежащих исключительно классу Ω_1. Эту грамматику можно использовать для классификации объектов, т.к. неизвестный объект можно отнести к классу Ω_1, если он является грамматически правильным предложением языка $L(\Gamma)$. Иначе, объект относим к Ω_2.

Чтобы реализовать процесс структурного распознавания необходимо выполнить следующее.

1. Построить адекватное описание объектов распознавания.
2. Выбрать грамматики.
3. Реализовать процесс распознавания посредством процедур синтаксического анализа.
4. Использовать обучение для вывода грамматик.
5. Применить в рамках структурного подхода другие методы распознавания.

Логические системы распознавания. В реальных задачах, например, геологического и экономического прогнозирования, медицинской и технической диагностики имеет место следующая ситуация:

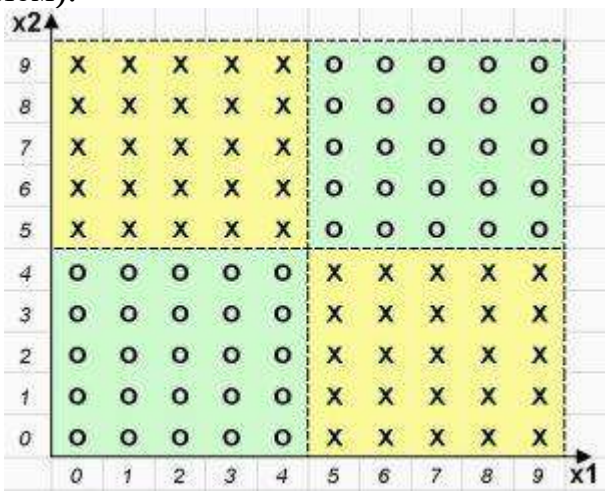
- число прецедентов (образов с известной классификацией) невелико,
- информации об их статистической природе недостаточно для обоснованного применения вероятностных моделей,

• сами прецеденты содержит разнородную или нечисловую информацию.

Однако известны логические связи между объектами и признаками.

Постановка задачи логического распознавания предполагает, что имеется обучающая выборка многомерных объектов, характеризующихся бинарными признаками. Необходимо найти правила классификации образов, каждое из которых содержит информацию не только об отдельных признаках, но и о различных их сочетаниях.

Например, пусть для поиска логических закономерностей предлагается таблица данных, содержащая 100 объектов (строк) и два количественных признака (столбца) x_1, x_2. Таблица разделена ровно пополам на два класса объектов. Распределение объектов на плоскости двух признаков x_1, x_2 приведено ниже (объекты первого класса обозначены крестиком, а объекты второго класса – нулем):



Решение представленной тестовой задачи очевидно. Каждый класс описывается двумя логическими правилами (всего 4 правила):

IF ($x_1 > 4$) и ($x_2 < 5$) THEN (класс 1 – крестики)

IF ($x_1 < 5$) и ($x_2 > 4$) THEN (класс 1 – крестики)

IF ($x_1 < 5$) и ($x_2 < 5$) THEN (класс 2 – нолики)

IF ($x_1 > 4$) и ($x_2 > 4$) THEN (класс 2 – нолики)

Этот простейший тест является “неподъемным” для многих известных коммерческих алгоритмов поиска логических закономерностей в данных.

Наиболее известным алгоритмом поиска и распознавания логических закономерностей в данных является алгоритм «Кора» *М.М. Бонгарда*.

Например, пусть дана обучающая выборка (ОВ), состоящая из N образов, предварительно разделенная на два класса. Выделены M признаков, характеризующих каждый образ. Путем просмотра ОВ с использованием операции конъюнкции $\&_k x_{ij}$ (x_{ij} – булева переменная, равная 1, если значение j-го признака превышает некоторый порог для i-го образа), необходимо выделить из множества возможных комбинаций признаков непротиворечивые конъюнкции, покрывающие все множество примеров ОВ. Непротиворечивой считается конъюнкция, которая встречается некоторое количество раз только в одном классе и ни разу не встречается в другом. Длина конъюнкции k – наперед задана (обычно $k = 3 \div 5$).

Чтобы найти такие конъюнкции необходимо действовать по следующим правилам.

1). Конъюнкции сортируются по продуктивности (по числу примеров из ОВ, для которых выделенная комбинация признаков равна 1 (не менее Δ).

2). Из списка исключаются подчиненные конъюнкции, оставляются «наилучшие» с точки зрения различения классов. Если есть эквивалентные, то оставляется более короткая.

3). Из списка исключаются конъюнкции-«предрассудки», не связанные с правилом классификации, но в силу ограниченности ОВ получившие хорошие оценки при обучении.

4). Общее число отобранных конъюнкций ограничиваем числом n.

Для классификации нового неизвестного образа для него подсчитывается число характерных для i-го класса конъюнкций, которые верны для неизвестного образа. Среди них выбирается максимальное число и принимается решение о принадлежности к i-му классу.

Описанием каждого класса является логическая сумма (дизъюнкция) некоторого количества непротиворечивых и продуктивных конъюнкций, прошедших описанные выше этапы отбора. Комбинация этих логических высказываний представляет собой своеобразную мозаично-фрагментарную разделяющую поверхность специального типа (в отличии от линейной поверхности).

Алгоритм “Кора”, как и другие логические методы распознавания образов, является достаточно трудоемким, поскольку при отборе конъюнкций необходим полный или частично направленный перебор. Метод хорошо работает при сравнительно небольших размерностях пространства признаков.

Решение многих задач логического распознавания на практике сводится к решению булевых уравнений с одним или несколькими неизвестными. Пусть множество образов разделено на классы $w_1, w_2,\ldots, w_m$. Для описания этих классов используются признаки $x_1, x_2,\ldots, x_n$. Предположим, что сведения о классах объектов и их признаках представлены в виде булевых соотношений. Пусть в результате экспериментов установлены данные о признаках x_i, присущих некоторым классам w_j. Эти данные выражены в виде булевых уравнений вида:

$G(x_1, x_2,\ldots, x_n) = 1.$

Тогда *прямая задача логического распознавания* заключается в следующем. При заданной постановке определить, какие выводы можно сделать относительно классов распознавания $w_1, w_2,\ldots, w_m$ на основе исходных сведений и выраженной в виде уравнения экспериментальной информации. Иными словами, требуется найти неизвестную функцию $F(w_1, w_2,\ldots, w_m)$, которая удовлетворяет уравнению:

$G(x_1, x_2,\ldots, x_n) \rightarrow F(w_1, w_2,\ldots, w_m)$

Обратная задача логического распознавания состоит в следующем. В заданной постановке задачи распознавания неизвестной является посылка $G(x_1, x_2,\ldots, x_n)$, из которой следует наблюдаемый факт $F(w_1, w_2,\ldots, w_m)$.

Метод решения прямой и обратной задач логического распознавания основан на применении изображающих чисел булевых функций (БФ) и сокращенного базиса.

Например, пусть в процессе решения задачи логического распознавания на основе анализа трех объектов были установлены следующие логические зависимости между тремя характеризующими эти объекты признаками **A, B, C**:

X(A+B) = ABC,

где **X** – некоторая булева функция, которую необходимо найти и которая зависит от **A, B, C**. Причем функция **X** должна быть такой, чтобы при её подстановке в исходное уравнение, оно превращалось в тавтологию.

Чтобы решить задачу, найдем изображающие числа для элементов уравнения:

C	**B**	**A**	**#ABC**	**#(A+B)**	**#X**
0	0	0	**0**	**0**	×
0	0	1	**0**	**1**	**0**
0	1	0	**0**	**1**	**0**
0	1	1	**0**	**1**	**0**
1	0	0	**0**	**0**	×
1	0	1	**0**	**1**	**0**
1	1	0	**0**	**1**	**0**
1	1	1	**1**	**1**	**1**

Изображающее **X** число должно быть таким, чтобы (#**X**)*01110111= 00000001.

Отсюда #**X**=×000×001 и уравнение имеет 4 решения, соответствующих изображающим числам:

(0000 0001), (1000 0001), (0000 1001) и (1000 1001), т.е.

$\mathbf{X_1 = ABC}$, $\mathbf{X_2 = ABC + \neg A\neg B\neg C}$, $\mathbf{X_3 = C(AB+\neg A\neg B)}$, $\mathbf{X_4 = \neg A\neg B + ABC}$.

Аналогично решаются системы булевых уравнений.

Лекция 3. Машинное обучение с учителем и без учителя. Обучение методами корреляционного и регрессионного анализа, нейросети и глубокое обучение, деревья решений

Машинное обучение – это направление в искусственном интеллекте, которое исследует алгоритмы, способные обучаться и самостоятельно улучшаться по мере накопления опыта, извлекать закономерности из данных и делать прогнозы на их основе. Машинное обучение также тесно связано с математической статистикой и оптимизацией, с интеллектуальным анализом данных и распознаванием образов. Технологии и методы машинного обучения широко используется при фильтрации спама, оптическом распознавании символов (*OCR*), в поисковых системах и компьютерном зрении.

Классификация задач машинного обучения.

В зависимости от характера обучающего “сигнала” или “обратной связи”, доступных для системы обучения задачи машинного обучения обычно подразделяются на обучение с учителем, обучение без учителя и обучение с подкреплением.

При обучении с учителем компьютеру предъявляются примеры входных данных и их желаемые выходные данные (обучающая выборка), заданные “учителем”, и цель состоит в том, чтобы найти общее правило, которое сопоставляет входные данные с выходными.

При обучении без учителя алгоритму обучения не присваиваются метки, он должен самостоятельно найти структуру во входных данных или обнаружить скрытые закономерности в данных.

При обучении с подкреплением компьютерная программа взаимодействует со средой, в которой она должна выполнять определенную задачу, без явного указания учителя о том приблизилась она к своей цели или нет. Откликом среды на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или ее модель.

Другая классификация задач машинного обучения возникает в зависимости от выходных данных в результате машинного обучения. В частности, если при классификации входные данные разделяются на два или более классов, то необходимо построить модель, которая относит входные данные к одному или нескольким классам. Например, фильтрация спама является примером классификации, где входными данными являются сообщения электронной почты (или другие), а классами являются “спам” и “не спам”. При решении регрессионной задачи выходные данные должны быть непрерывными, а не дискретными. При решении задачи кластеризации набор входных данных должен быть разделен на заранее неизвестное число групп.

Основная цель алгоритма машинного обучения – обобщение своего опыта. Обобщение опыта – это способность обучающей машины правильно решать новые задачи после изучения набора обучающих данных. Поскольку набор обучающих данных конечен, то для оценки эффективности алгоритмов

машинного обучения обычно используется дисперсия в качестве способа количественной оценки ошибки обобщения. В дополнение к этому также используется временная оценка сложности алгоритмов машинного обучения (в теории машинного обучения алгоритм считается выполнимым, если он может быть выполнен за полиномиальное время).

Перечень современных алгоритмов машинного обучения включает:

- деревья принятия решений, структура которых включает «листья» и «ветки». На ребрах («ветках») дерева решения записаны признаки, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах – признаки, по которым различаются примеры. Чтобы классифицировать новый пример, надо спуститься по дереву до листа и выдать соответствующее значение;
- поиск ассоциативных правил для обнаружения интересующих нас связей между переменными в большой базе данных. Алгоритм генерирует также новые правила по мере анализа данных и создает возможность нахождения ассоциаций из новых неклассифицированных данных;
- обучение искусственной нейронной сети (ИНС). Некоторые ИНС обучаются без учителя (например, сети Хопфилда), они просматривают выборку только один раз. Другие (например, сети Кохонена), а также сети, обучающиеся с учителем, просматривают выборку множество раз (эпохи обучения). При обучении с учителем набор исходных данных делят на две части – собственно обучающую выборку и тестовые данные; принцип разделения может быть произвольным. Обучающие данные подаются сети для обучения, а проверочные используются для расчета ошибки сети (проверочные данные никогда для обучения сети не применяются). Таким образом, если на проверочных данных ошибка уменьшается, то сеть действительно выполняет обобщение. Если ошибка на обучающих данных продолжает уменьшаться, а ошибка на тестовых данных увеличивается, значит, сеть перестала выполнять обобщение и просто «запоминает» обучающие данные. Это явление называется переобучением сети (оверфиттинг). В таких случаях обучение обычно прекращают. В процессе обучения могут проявиться другие проблемы, такие как паралич или попадание сети в локальный минимум поверхности ошибок. Невозможно заранее предсказать проявление той или иной проблемы, равно как и дать однозначные рекомендации к их разрешению. Это относится только к итерационным алгоритмам поиска нейросетевых решений. Для них действительно нельзя ничего гарантировать и нельзя полностью автоматизировать обучение нейронных сетей. Наряду с итерационными алгоритмами обучения, существуют алгоритмы, обладающие очень высокой устойчивостью и позволяющие полностью автоматизировать процесс обучения. В целом современные нейронные сети – это инструменты нелинейного статистического моделирования данных. Они используются для моделирования сложных взаимосвязей между входными и выходными данными, для поиска закономерностей в данных и других задач;
- индуктивное логическое программирование (*ILP*) – это который использует логическое программирование как форму представления примеров,

фоновых знаний и гипотез. Получив описания уже известных фоновых знаний и набор примеров, представленных как логическая база фактов, система *ILP* может породить логическую программу в форме гипотез, объясняющую все положительные примеры и ни одного отрицательного. подход к правилу обучение с использованием логического программирования в качестве единообразного представления входных примеров, фоновых знаний и гипотез. Обычно реализации *ILP* делаются на языке *Prolog*;

•машины опорных векторов (*SVM*) представляют собой набор связанных алгоритмов обучения с учителем, используемых для задач классификации и регрессии. Принадлежит семейству линейных классификаторов. Основная идея *SVM* – перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с наибольшим зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, создающая наибольшее расстояние до двух параллельных гиперплоскостей. Алгоритм основан на допущении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора. Учитывая набор обучающих примеров, каждый из которых помечен как относящийся к одной из двух категорий, алгоритм *SVM* строит модель, которая предсказывает, попадает ли новый пример в ту или иную категорию;

•кластерный анализ – это многомерная статистическая процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в сравнительно однородные группы в соответствии с некоторыми заранее определенными критериями. Задача кластеризации относится к широкому классу задач обучения без учителя. Различные процедуры кластеризации делают разные предположения о структуре данных согласно некоторой метрики сходства оцениваемой, например, по внутренней компактности (сходству между данными одного и того же кластера) и разделению на различные кластеры;

•байесовская сеть – это вероятностная модель, представляющая собой множество переменных и их вероятностных зависимостей по Байесу. Формально байесовская сеть представляет ориентированный ациклический граф, каждой вершине которого соответствует случайная переменная, а дуги графа кодируют отношения условной независимости между этими переменными. Вершины могут представлять переменные любых типов, быть взвешенными параметрами, скрытыми переменными или гипотезами. Существуют эффективные методы, которые используются для вычислений и обучения байесовских сетей. Байесовские сети, в которых могут присутствовать как дискретные переменные, так и непрерывные, называются гибридными байесовскими сетями. Байесовская сеть, в которой дуги помимо отношений условной независимости кодируют также отношения причинности, называют причинно-следственными байесовскими сетями. Например, байесовская сеть может представлять вероятностные взаимосвязи между болезнями и симптомами. Учитывая

симптомы, сеть может быть использована для вычисления вероятностей наличия различных заболеваний;

•обучение с подкреплением связано с тем, как агент должен действовать в окружающей среде, чтобы максимизировать ожидаемое вознаграждение. Алгоритмы обучения с подкреплением сопоставляют состояние среды с действиями, которые должны быть предприняты в этом состоянии. Обучение с подкреплением отличается от задачи обучения с учителем тем, что правильные пары входные данные-ответ никогда не представляются, а неоптимальные действия явно не исправляются;

•глубокое обучение представляет собой алгоритмы обучения без учителя, направленные на выявление лучшего представления входных данных, предоставляемых в ходе обучения. Алгоритмы глубокого обучения пытаются сохранить информацию о входных данных, преобразуя ее так, чтобы она была полезной на этапе предварительной обработки данных при классификации или прогнозировании;

•метрическое обучение подобию состоит в том, что обучающей машине предлагаются пары примеров, которые считаются подобными, и пары менее похожих объектов. Алгоритму необходимо изучить функцию подобия (или функцию метрики расстояния), которая помогает предсказать схожесть или различие новых объектов. Подход зачастую используется в рекомендательных системах;

•биоинспирированные алгоритмы (*BIA*) представляют собой математические преобразования, трансформирующие входной поток информации в выходной и основанные на правилах имитации биоинспирированных механизмов, на процедурах, содержащих элементы стохастической оптимизации со случайными переменными. Основными свойствами *BIA* являются стратегии, которые направляют процесс поиска оптимальных решений, а также широкое варьирование алгоритмов от простых локальных процедур до сложных недетерминированных процедур. С помощью *BIA* исследуется пространство поиска, синтезируются решения, являющиеся точками этого пространства, запрашивается оценка их качества или "приспособленности", которая затем используется для осуществления "естественного отбора". Тем самым *BIA* обучаются тому, какие области пространства поиска содержат наиболее приспособленных особей.

Основными приложениями машинного обучения являются: адаптивные веб-сайты, аффективные вычисления (процесс сбора данных из выражений лица, голосовых сообщений и языка тела для измерения уровня человеческих эмоций), биоинформатика (компьютерные методы, направленные на получение, анализ, хранение, организацию и визуализацию биологических данных), интерфейсы мозг-машина, химинформатика (компьютерные методы, направленные на решение химических проблем), классификация последовательностей ДНК, Интернет-реклама, цифровые технологии в финансовой сфере, компьютерное зрение, включая распознавание объектов, выявление мошенничества с кредитными картами, игры, поиск информации, обнаружение интернет-мошенничества, медицинская диагностика, обработка естественного языка,

поддержка принятия оптимальных решений, рекомендательные системы, передвижение роботов, анализ последовательностей, распознавание речи и почерка, анализ фондового рынка.

Обучение с учителем

Обучение с учителем – это задача машинного обучения, в которой система обучается с помощью конечной совокупности примеров пар «вход-отклик» (обучающая выборка, ОВ). Между входами и эталонными выходными откликами может существовать некоторая зависимость, но она неизвестна. На основе ОВ требуется восстановить, то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ. Для измерения точности ответов вводится функционал качества. Алгоритм обучения с учителем анализирует обучающие данные и формирует функцию, которую можно использовать для отображения новых примеров.

Чтобы решить задачу обучения с учителем, необходимо следующее.

1. Определить, какие данные будут использоваться в качестве обучающей выборки. Например, случае анализа почерка это может быть один написанный от руки символ, либо целое написанное от руки слово или целая строка текста.

2. Подготовить репрезентативную тестовую выборку.

3. Определить типы входных данных: признаковое пространство (набор характеристик, описывающих каждый объект); матрица расстояний между объектами; временной ряд или сигнал, представляющие собой последовательность измерений во времени; изображение или видеоряд; более сложные случаи (графы, тексты, запросы к базе данных и т.д.). Количество объектов не должно быть слишком большим из-за проклятия размерности, однако должно содержать достаточно информации, чтобы точно предсказать результат.

4. Определить тип откликов и алгоритм обучения. Множество возможных откликов может быть бесконечным (ответы являются действительными числами или векторами, что характерно для задач регрессии и аппроксимации), либо конечным (характерно для задач классификации и распознавания образов), либо отклики характеризуют будущее поведение процесса или явления (характерно для задач прогнозирования). В качестве алгоритма машинного обучения, например, могут быть использованы деревья принятия решений, нейросети, машины опорных векторов или биоинспирированные алгоритмы.

5. Запустить алгоритм обучения на подготовленной ОВ. Некоторые алгоритмы обучения с учителем требуют, чтобы пользователь определял параметры управления обучением.

6. Оценить точность обучения на тестовом наборе.

Согласно теореме Вольперта-Макрида (*NFL*-теорема) не существует алгоритма обучения, который бы лучше всех работал для различных задач обучения с учителем.

При обучении с учителем необходимо учитывать следующие основные проблемы:

Первая проблема заключается в необходимости искать компромисс между смещением и дисперсией. Они указывают на два разных источника ошибки функции оценки. Смещение измеряет ожидаемое отклонение от истинного значения функции или параметра. Дисперсия же показывает меру отклонения от ожидаемого значения оценки, которую может вызвать любая конкретная выборка данных. Желательно, чтобы функция оценки имела малое смещение при относительно небольшой дисперсии. Алгоритм обучения будет иметь высокую дисперсию для конкретного входного набора данных, если он предсказывает разные выходные значения при обучении на разных наборах обучения. Алгоритм обучения с небольшим смещением должен быть “гибким”, чтобы хорошо соответствовать данным. Но если он будет слишком “гибким”, то будет по-разному соответствовать каждому набору обучающих данных и, следовательно, будет иметь большую дисперсию. Вопрос в том насколько алгоритм обучения способен поддерживать этот компромисс между смещением и дисперсией (автоматически либо путем настройки пользователем параметра смещение/дисперсия).

Вторая проблема заключается в объеме ОВ относительно сложности “истинной” функции (классификатора или регрессионной функции). Если функция проста, то алгоритм обучения с высокой погрешностью и низкой дисперсией сможет изучить ее на основе ОВ небольшого объема. Однако если функция очень сложная (включает взаимодействия между множеством различных входных функций и ведет себя по-разному в разных частях входного пространства), то требуется ОВ очень большого объема, а также “гибкий” алгоритм обучения с малым смещением и высокой дисперсией.

Еще одна проблема заключается в размерности входного пространства. Если входные векторы признаков имеют очень высокую размерность, то задача обучения усложняется, даже если функция отклика зависит от небольшого числа этих признаков. Множество признаков может «запутать» алгоритм обучения и привести к его высокой дисперсии. Поэтому высокая размерность входных данных требует настройки классификатора на низкую дисперсию и высокое смещение. На практике применяют стратегии сокращения размерности пространства входных данных до запуска алгоритма обучения с учителем.

Наконец проблема «шума» в выходных значениях. Если желаемые выходные значения ошибочны, то алгоритм обучения не должен пытаться найти функцию, которая точно соответствует примерам из ОВ. При наличии любого типа шума (стохастический или детерминированный) лучше использовать более высокую оценку смещения и более низкую оценку дисперсии. Существует несколько процедур, которые позволяют удалить подозрительные примеры с шумом.

Существуют и другие факторы, которые следует учитывать при выборе и применении алгоритма машинного обучения: неоднородность и избыточность данных, нелинейность и взаимозависимость между функциями. В частности, если векторы признаков являются неоднородными, то отдельные алгоритмы машинного обучения применять проще, нежели другие. Такие алгоритмы как машины опорных векторов, линейная и логистическая регрессия, нейронные

сети и ближайших соседей, требуют, чтобы входные объекты были числовыми и масштабировались в определенных диапазонах (например, [–1, 1]). Напротив, деревья принятия решений легко обрабатывают разнородные данные. Далее, если входные объекты содержат избыточную информацию (например, сильно коррелированные объекты), то такие алгоритмы обучения как линейная и логистическая регрессия, а также методы, основанные на измерении расстояния не будут отличаться точностью. Если функции отклика являются независимыми, то алгоритмы обучения, основанные на линейных функциях (линейная и логистическая регрессия, байесовские сети) и функциях расстояния (ближайших соседей, *SVM*), как правило, работают хорошо. Однако, если между функциями существуют сложные взаимозависимости, то такие алгоритмы, основанные на деревьях решений или нейронных сетях, работают значительно лучше.

Поэтому при анализе новой задачи разработчику необходимо сравнить несколько алгоритмов машинного обучения и экспериментально определить, какой из них лучше всего подходит для рассматриваемой проблемы.

Как работают алгоритмы обучения с учителем?

Пусть ОВ состоит из N обучающих примеров вида $\{(x_1, y_1),...,(x_N, y_N)\}$, таких что x_i – вектор признаков i-го примера, y_i – его метка (т.е. класс). Алгоритм обучения ищет функцию g: $X \rightarrow Y$, где X – это входное пространство, а Y – выходное пространство. Функция g является элементом некоторого пространства возможных функций G, обычно называемый пространством гипотез. Иногда удобно представлять g с помощью функции выигрыша f: $X \times Y \rightarrow R$: $g(x) = arg\ max_y f(x, y)$.

Обозначим через F пространство функций выигрыша. Хотя G и F могут быть любым пространством функций, однако многие алгоритмы обучения используют вероятностные модели, где $g = P(y|x)$ является условной вероятностью, а $f(x, y) = P(x, y)$ – совместной вероятностью.

Два основных подхода к выбору f или g: эмпирическая минимизация рисков и структурная минимизация рисков.

Эмпирическая минимизация рисков направлена на поиск функции, которая наилучшим образом соответствует данным обучения. Структурная минимизация рисков включает функцию штрафа, которая контролирует компромисс между смещением и дисперсией. В обоих случаях предполагается, что обучающий набор состоит из выборки независимых и одинаково распределенных пар (x_i, y_i). Чтобы оценить, насколько функция соответствует обучающим данным, определяется функция потерь L: $Y \times Y \rightarrow R \geq 0$. Для обучающего примера (x_i, y_i) штраф y^* равен $L(y_i, y^*)$.

Риск $R(g)$ для функции g определяется как ожидаемая потеря g по данным обучения:

$$R(g) = \frac{1}{N}\sum_i L(y_i, g(x_i)).$$

При эмпирической минимизации риска алгоритм обучения с учителем ищет функцию g, которая минимизирует $R(g)$. Следовательно, алгоритм обучения с учителем может быть построен путем применения процедур оптимизации функции g. Если g имеет условное распределение вероятности $P(y|x)$, а функция

потерь представляется через отрицательную логарифмическую вероятность $L(y; y^*) = -logP(y|x)$, тогда эмпирическая минимизация риска эквивалентна оценке максимального правдоподобия. Если функция g является элементом некоторого пространства возможных функций G, или обучающий набор недостаточно велик, то эмпирическая минимизация риска приводит к высокой дисперсии и плохой способности алгоритма к обобщению. Это называется переобучением (оверфиттинг). Иными словами, при обучении переобученной модели мы добились слишком маленькой ошибки на обучающей выборке, то есть слишком плотно подошли к данным. Простой способ избежать данной ситуации – это регуляризация – добавить нечто к функционалу ошибки, чтобы было невозможно получить слишком маленькую ошибку.

Структурная минимизация рисков направлена на предотвращение оверфиттинга путем введения штрафных функций (простые функции признаются более предпочтительными, нежели сложные). Используются разнообразные штрафные функции, которые соответствуют различным определениям сложности функции. Например, функция g является линейной функцией вида

$$g(x) = \sum_{j=1}^{d} w_j x_j.$$

Здесь w_j представляют собой веса модели. Популярным штрафом является $\sum_j w_j^2$ (квадратичная евклидова норма весов, норма L_2). Другими нормами являются норма L_1 ($\sum_j |w_j|$) и норма L_0, которая представляет собой число ненулевых w_j. Если штраф обозначить через $C(g)$, то задача оптимизации обучения с учителем состоит в том, чтобы найти такую функцию g, которая минимизирует

$$J(g) = R(g) + \lambda C(g).$$

Параметр λ называют коэффициентом регуляризации, он контролирует компромисс между смещением и дисперсией. При $\lambda = 0$ минимизируется эмпирический риск с малым смещением и высокой дисперсией. Если риск велик, то алгоритм обучения будет иметь большое смещение и низкую дисперсию.

Алгоритмы машинного обучения с учителем, описанные выше, являются дискриминативными, поскольку они направлены на поиск функции g, которая хорошо различает различные выходные значения и широко используется при обнаружении объектов, семантической сегментации, паноптической сегментации, обнаружении ключевых точек, решении задач регрессии и языкового моделирования. В то время как генеративное моделирование определяет способ создания набора данных. Этот вид машинного обучения пытается понять распределение точек данных, обеспечивая модель того, как данные фактически генерируются в терминах вероятностной модели. (например, машины опорных векторов или алгоритм персептрона дает разделяющую границу решения, но не модель генерации точек синтетических данных). Цель состоит в том, чтобы сгенерировать новые образцы из того, что уже было распределено в обучающих данных. Наиболее популярными алгоритмами

генеративного типа являются наивный байесовский алгоритм, скрытые марковские модели, автоэнкодер, машина Больцмана, генеративные состязательные сети.

Итак, основными методами и алгоритмами машинного обучения с учителем являются алгоритм обратного распространения ошибки для искусственных нейросетей, бустинг, байесовский метод, деревья решений, индуктивное логическое программирование, регрессия, групповой метод обработки данных, обучающие автоматы, алгоритм ближайшего соседа, машины опорных векторов, случайный лес, ансамбли классификаторов, предварительная обработка данных, алгоритм многокритериальной классификации.

Основными областями их применения являются биоинформатика, хемиинформатика, распознавание рукописного текста и речи, поиск информации, компьютерное зрение, фильтрация спама.

Рассмотрим подробнее методы и алгоритмы машинного обучения с учителем.

Линейная регрессия.

Линейная регрессия моделирует взаимосвязь между скалярной зависимой переменной *y* и одной или несколькими независимыми переменными *X*. Случай одной независимой переменной называется простой линейной регрессией. В случае более чем одной независимой переменной говорят о множественной линейной регрессии. Линейная регрессия была первым типом регрессионного анализа, она имеет много практических применений, особенно в прогнозировании. Модели линейной регрессии часто подбираются с использованием метода наименьших квадратов.

Пусть имеется множество данных $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^{n}$ из n статистических единиц. Модель линейной регрессии предполагает, что связь между зависимой переменной y_i и p-вектором переменных x_i является линейной. Эта взаимосвязь моделируется с помощью случайной ошибки ε_i, которая добавляет шум к линейной зависимости.

Таким образом, модель принимает вид

$$y_i = w_1 x_{i1} + \cdots + w_p x_{ip} + \varepsilon_i = \boldsymbol{x}_i^T \boldsymbol{w} + \varepsilon_i, i = 1..n,$$

где T обозначает транспонирование, а w_i – параметры (коэффициенты) регрессии, которые показывают скорость изменения зависимой переменной по данному фактору, при фиксированных остальных факторах (в линейной модели эта скорость постоянна). Если сложить эти n уравнений, то их можно представить в векторной форме в виде

$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\varepsilon}$, где

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \boldsymbol{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}, \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Например, рассмотрим ситуацию, когда маленький шар подбрасывается в воздух, а затем мы измеряем высоту его подъема y_i в разные моменты времени t_i.

Если не учитывать сопротивление воздуха, то согласно законам физики взаимосвязь между этими переменными может быть смоделирована как

$$y_i = w_1 t_i + w_2 t_i^2 + \varepsilon_i,$$

где параметр w_1 определяет начальную скорость шара, параметр w_2 пропорционален силе тяжести, а ε_i обусловлена ошибками измерения. Линейная регрессия может быть использована для оценки значений w_1 и w_2 по измеренным данным. Эта модель является нелинейной по переменной времени, но она линейна по параметрам w_1 и w_2, если в качестве регрессоров рассматривать $x_i = (x_{i1}, x_{i2}) = (t_i, t_i^2)$, то модель принимает стандартный вид:

$$y_i = \boldsymbol{x}_i^T \boldsymbol{w} + \varepsilon_i.$$

В классической линейной регрессии предполагается, что выполнены также следующие предположения (условия Гаусса-Маркова): постоянная или одинаковая дисперсия и отсутствие автокорреляции случайных ошибок. При выполнении этих предположений обычный метод наименьших квадратов позволяет получить достаточно качественные оценки параметров модели: они являются несмещёнными, состоятельными и наиболее эффективными оценками.

Логистическая регрессия. Результатом работы логистического регрессора является 0 или 1, то есть это бинарный классификатор. При решении задачи классификации логистический регрессор оценивает вероятность попадания в класс через отношение шансов *OR(x)*:

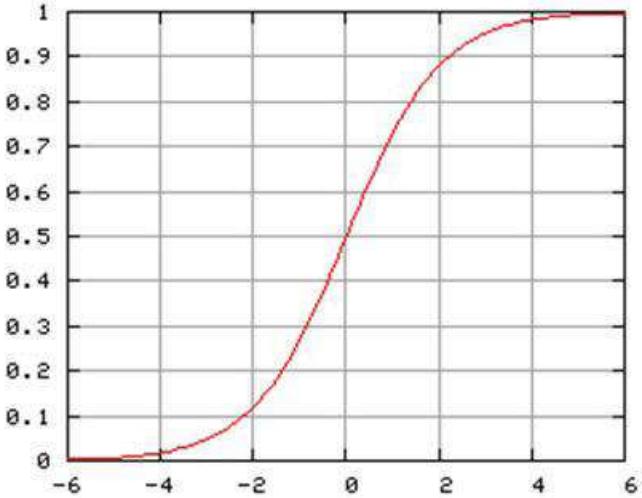
$$OR(x) = \frac{P(x)}{1 - P(x)},$$

где *P(x)* – вероятность события *x*.

Отношение шансов меняется в диапазоне от –∞ до +∞, а нам нужен диапазон от 0 до 1. Поэтому прологарифмируем полученное выражение *OR(x)* и выразим из него *P(X)*:

$$P_+ = \frac{e^{\langle w,x \rangle}}{1 + e^{\langle w,x \rangle})} = \frac{1}{1 + e^{-\langle w,x \rangle})}.$$

Это выражение является формулой логистического регрессора, график которого приведен ниже:



В логистической регрессии, как и в линейной модели, необходим функционал ошибки, минимизируя который, мы будем искать веса модели. В

качестве функционала ошибки здесь используется логистическая функция потерь:

$$\bar{Q} = \sum_{i=1}^{n} \ln(1 + \exp(-y_i < w, x >)).$$

Для поиска минимума функционала ошибки в логистической регрессии используются так же, как и в линейной модели, градиентный спуск или его модификации. Результатом обучения модели логистической регрессии является так же, как и в линейной модели, набор весов модели. А вот ответами в логистической регрессии будут метки класса для каждого объекта с указанием вероятности отнесения объекта к этому классу.

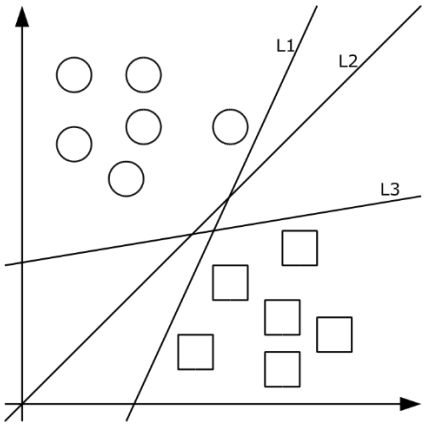
Обучение модели проходит на обучающей части выборки, а оценка качества модели – на тестовой части выборки.

Метод опорных векторов. (*Support Vector Machine, SVM*) представляет собой набор схожих алгоритмов обучения с учителем. Используется, в основном, для задач классификации и регрессионного анализа. Принадлежит семейству линейных классификаторов и может также рассматриваться как частный случай регуляризации по Тихонову. Особым свойством *SVM* является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора, поэтому метод также известен как метод классификатора с максимальным зазором.

Основная идея метода – перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с наибольшим зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, создающая наибольшее расстояние до двух параллельных гиперплоскостей. Алгоритм основан на допущении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

Поясним метод на примере задачи классификации данных. Каждый объект данных представляется как вектор (точка) в p-мерном пространстве (упорядоченный набор p чисел). Каждая из этих точек принадлежит только одному из двух классов. Вопрос состоит в том, можно ли разделить точки гиперплоскостью размерности $(p-1)$? Это – типичный случай линейной разделимости. Искомых гиперплоскостей может быть много, поэтому полагают, что максимизация зазора между классами способствует более уверенной классификации. Иными словами, можно ли найти такую гиперплоскость, чтобы расстояние от неё до ближайшей точки было максимальным? Это эквивалентно тому, что сумма расстояний до гиперплоскости от двух ближайших к ней точек, лежащих по разные стороны от неё, максимальна. Если такая гиперплоскость существует, она называется оптимальной разделяющей гиперплоскостью, а соответствующий ей линейный классификатор называется оптимально разделяющим классификатором.

Ниже на рисунке представлено несколько классифицирующих разделяющих прямых (гиперплоскостей), из которых только одна соответствует оптимальному разделению:



Формальное описание предполагает, что точки имеют вид:

$\{(\boldsymbol{x}_1, c_1), (\boldsymbol{x}_2, c_2),\ldots, (\boldsymbol{x}_n, c_n)\}$,

где c_i принимает значение 1 или −1, в зависимости от того, какому классу принадлежит точка $\boldsymbol{x}_i$. Каждое $\boldsymbol{x}_i$ − это p-мерный вещественный вектор, обычно нормализованный значениями [0, 1] или [−1, 1].

Если точки не будут нормализованы, то точка с большими отклонениями от средних значений координат точек слишком сильно повлияет на классификатор. Мы можем рассматривать это как обучающую выборку, в которой для каждого элемента уже задан класс, к которому он принадлежит. Алгоритм *SVM* должен классифицировать их таким же образом. Для этого необходимо построить разделяющую гиперплоскость, которая имеет вид:

$\boldsymbol{w}\cdot\boldsymbol{x} - b = 0$.

Вектор $\boldsymbol{w}$ − перпендикуляр к разделяющей гиперплоскости. Параметр $b/\|\boldsymbol{w}\|$ равен по модулю расстоянию от гиперплоскости до начала координат. Если параметр b равен нулю, то гиперплоскость проходит через начало координат, что ограничивает решение.

Поскольку речь идет о поиске оптимальное разделение, нас интересуют опорные вектора и гиперплоскости, параллельные оптимальной и ближайшие к опорным векторам двух классов. Можно показать, что эти параллельные гиперплоскости могут быть описаны следующими уравнениям (с точностью до нормировки):

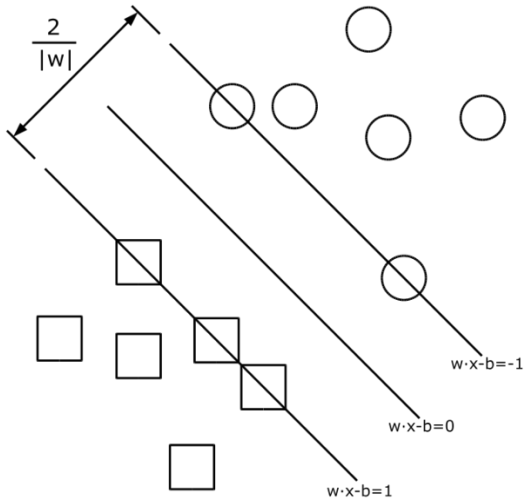
$\boldsymbol{w}\cdot\boldsymbol{x} - b = 1$,

$\boldsymbol{w}\cdot\boldsymbol{x} - b = -1$.

Если обучающая выборка линейно разделима (может быть полностью разделена единственной прямой), то мы можем выбрать гиперплоскости таким образом, чтобы между ними не лежала ни одна точка обучающей выборки и затем максимизировать расстояние между гиперплоскостями. Ширину полосы между ними легко найти из соображений геометрии, она равна $2/\|\boldsymbol{w}\|$, таким образом задача состоит в минимизации $\|\boldsymbol{w}\|$. Чтобы исключить все точки из полосы, мы должны убедиться для всех i, что

$$\begin{cases} \boldsymbol{w}\cdot\boldsymbol{x_i} - b \geq 1, c_i = 1 \\ \boldsymbol{w}\cdot\boldsymbol{x_i} - b \leq -1, c_i = -1. \end{cases}$$

Ниже на рисунке представлена оптимальная разделяющая гиперплоскость для метода опорных векторов, построенная на точках из двух классов:



Ближайшие к параллельным гиперплоскостям точки являются опорными векторами.

Обсудим ***метрики качества классификации***.

Доля правильных ответов (*accuracy (a, X)*) – наиболее популярная метрика качества классификационных моделей. Она указывает в долях или процентах количество правильных ответов, то есть тех ответов алгоритма, которые совпали с истинными значениями в обучающей или тестовой части выборки:

$$accuracy(a, X) = \frac{1}{n}\sum_{i=1}^{n}[a(x_i) = y_i].$$

Допустим, в выборке много объектов класса 1, но мало класса 0. Тогда модель будет отлично находить объекты класса 1, а вот с объектами класса 0 будет ошибаться. Если поменять разметку классов на обратную, то доля правильных ответов резко снизится. Кроме того, при построении классификационной модели всегда нужно помнить о цене ошибки. Например, модель оценивает эффективность методики диагностирования короновирусом, когда человек крайне заразен для окружающих. Тогда класс 1 (методика эффективна) становится очень критичным. И если модель ошибается и признает человека здоровым (то есть относит объект к классу 0, а не к 1), а он болен, то вред от такой оценки очень велик. Как этого избежать? Ответ прост: в обучающей выборке должно быть больше случаев с эффективной методикой, нежели с неэффективной.

Ошибки бывают двух типов – *False Negative (FN)* и *False Positive (FP)*, как это представлено в матрице ошибок:

	Y=1	Y= –1
a(x) = 1	*True Positive (TP)*	*False Positive (FP)*
a(x) = –1	*False Negative (FN)*	*True Negative (TN)*

False Negative (FN) – количество объектов, которые на самом деле являются классом 0 или –1, но алгоритм отнес их к классу 1. *False Positive (FP)* – количество объектов, которые на самом деле являются классом 1, но алгоритм их отнес к классам –1 или 0.

Матрица ошибок содержит и правильные ответы, которые также делятся на 2 группы: *True Positive (TP)* и *True Negative (TN)*. *True Positive (TP)* – количество объектов класса 1, которые нашла модель. *True Negative (TN)* – количество объектов классов –1 или 0, которые нашла модель.

Рассмотрим пример с двумя матрицами ошибок:

	Y=1	Y= –1
a1(x) = 1	80	20
a1(x) = –1	20	80

	Y=1	Y= –1
a2(x) = 1	48	2
a2(x) = –1	52	98

Первый алгоритм хорошо опознает и класс 1 и класс –1, причем ошибается тоже одинаково на обоих классах. Второй алгоритм очень точно выделяет класс 1: почти все объекты, попавшие в класс 1, действительно ими и являются. Но при этом алгоритм отбрасывает много объектов класса 1 в класс –1.

Кажется, что информация в матрице ошибок избыточна. Вроде бы достаточно просто показать, насколько хорош алгоритм и сколько он делает ошибок при классификации. Однако матрицы ошибок позволяют понять не только, сколько ошибок делает алгоритм, но понять, насколько он точен и как полно находит заданный класс. Для этого вводятся понятия точности и полноты.

Точность показывает, насколько можно доверять классификатору. Классификатор с высокой точностью выловит в класс 1 только объекты класса 1 и не прихватит в него объекты другого класса. Точность вычисляется по формуле

$$precision(a, X) = \frac{TP}{TP + FP}.$$

Из формулы следует, что точность – это отношение количества правильно отнесенных к классу 1 объектов ко всем объектам класса 1 в выборке.

Полнота показывает долю истинных ответов, на которых срабатывает классификатор. Полнота показывает, как хорошо классификатор находит объекты заданного класса. Классификатор с высокой полнотой найдет все объекты класса 1 в выборке, но при этом прихватит в класс 1 и много объектов из класса –1. Вычисляется полнота по формуле

$$recall(a, X) = \frac{TP}{TP + FN}.$$

Полнота – это отношение количества правильно отнесенных к классу 1 объектов ко всем объектам, отнесенным классификатором к классу 1.

В рассмотренном выше примере с двумя матрицами ошибок первый алгоритм имеет высокую и точность (0,8), и полноту (0,8). Этот алгоритм рекомендуется к применению. Второй алгоритм имеет высокую точность (0,96), а вот полнота у него низкая – 0,48. Такой алгоритм можно использовать, если необходимо найти объекты только класса 1, не взяв случайно объект класса –1.

Как правило, настроить алгоритм одновременно на высокую точность и полноту невозможно, поэтому чем-то из них приходится жертвовать. Это определяется ценой ошибки, то есть конкретной задачей, в которой нужно понять, что важнее – найти все объекты класса или найти объекты только одного класса.

Компромиссом между этими мерами является *F*-мера, которая сочетает в себе и точность, и полноту. Она рассчитывается по формуле

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

Если нужно учесть предпочтения точности или полноте, то вводят коэффициент β:

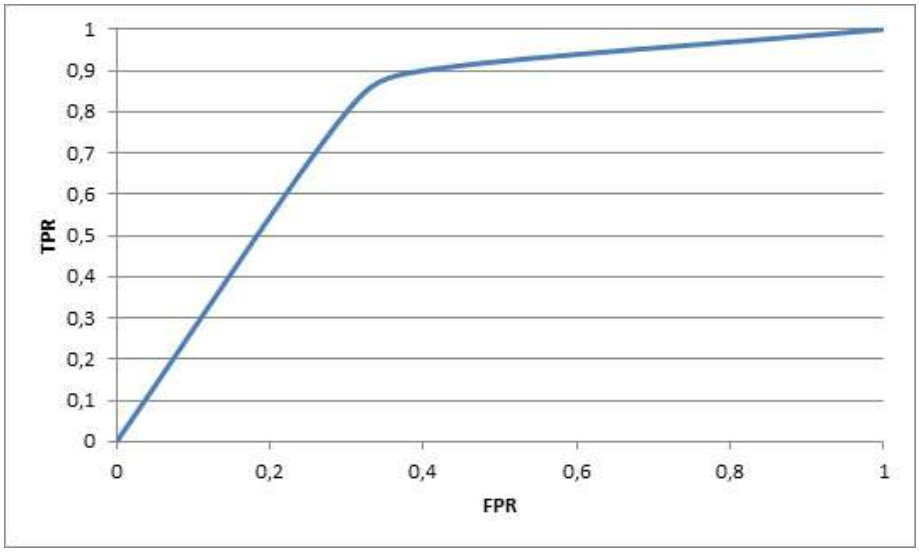
$$F = (1 + \beta^2) \frac{2 \cdot precision \cdot recall}{\beta^2 \cdot precision + recall}.$$

Еще одной популярной метрикой качества классификации является *ROC-AUC* кривая, которая строится в координатах *TPR* и *FPR* соответственно:

$$TPR = \frac{TP}{TP + FN},$$

$$FPR = \frac{FP}{FP + TN}.$$

Ниже представлен вид кривой *ROC-AUC*.



Качество классификатора оценивается площадью под *ROC-AUC* кривой: чем она ближе к 1, тем лучше классификатор. Если площадь под *ROC-AUC* кривой близка к 0,5, то этот классификатор выдает случайные ответы. Поэтому на практике часто на графике отображают еще и график случайного

классификатора, чтобы показать, насколько полученный классификатор отличается от случайного.

Возникает также вопрос о размере обучающей выборки и соотношении количества наблюдений с количеством независимых переменных в модели. Одним из эмпирических правил является $N = m^n$, где N – размер обучающей выборки, n – количество независимых переменных, а m – количество наблюдений, необходимых для достижения желаемой точности, если в модели только одна независимая переменная. Например, аналитик строит модель линейной регрессии, используя набор данных, содержащий данные о $N = 1000$ пациентов. Если он решит, что пять наблюдений достаточны для точного определения прямой линии ($m = 5$), тогда максимальное число независимых переменных, которые может поддерживать модель, будет равно 4, поскольку $log(1000)/log(5) = 4{,}29$.

Кроме метода наименьших квадратов, для оценки параметров регрессионной модели используются и другие методы: байесовская, процентная и квантильная регрессия, дистанционное метрическое обучение.

Большинство существующих статистических компьютерных пакетов содержат средства регрессионного анализа по методу наименьших квадратов.

Регрессионные модели считаются важным инструментом и широко используются в технических, биологических, социальных науках, экономике и финансах для описания возможных взаимосвязей между переменными.

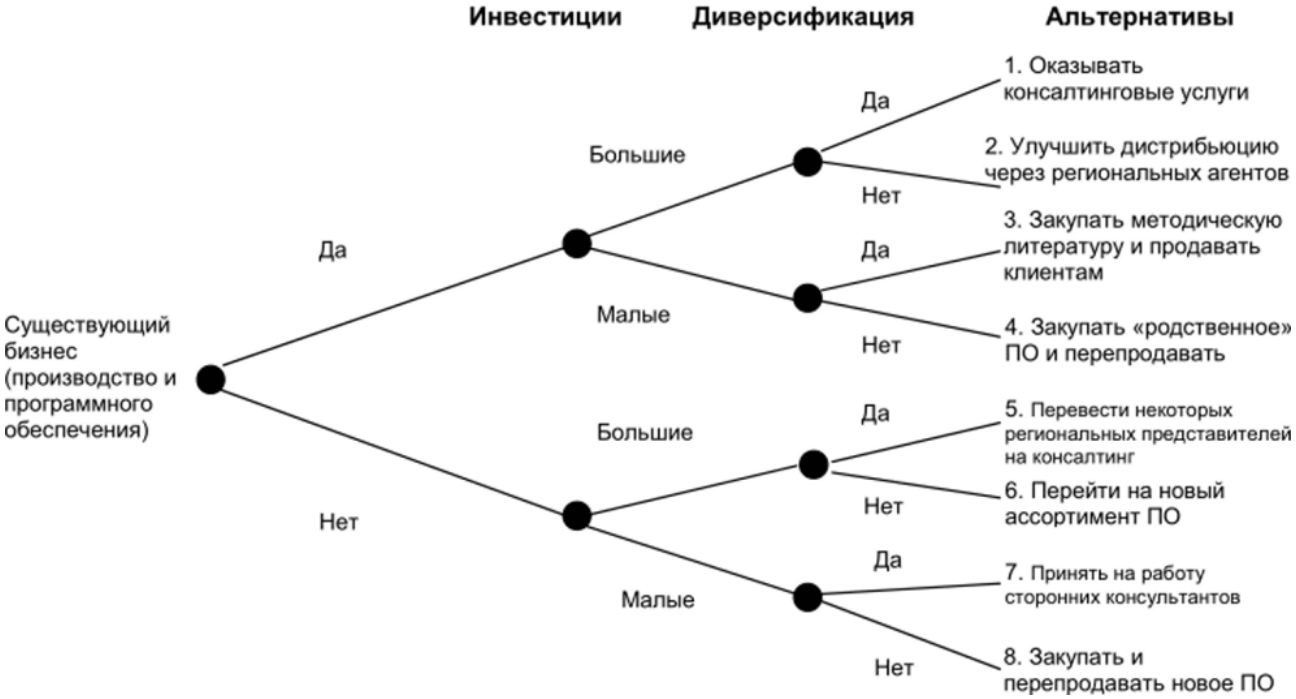
Например, при прогнозировании временных рядов анализируется конкретный набор данных (скажем ВВП, цены на нефть или акции). Линию тренда можно просто нарисовать по набору точек данных, но более правильно их положение и наклон вычисляются с использованием регрессионных методов. Линии тренда часто используются для утверждения, что конкретное действие или событие (например, рекламная кампания) вызвало наблюдаемые изменения в определенный момент времени. Это простой метод, и он не требует контрольной группы, экспериментального дизайна или сложной методики анализа. Однако он мало обоснован в тех случаях, когда другие потенциальные изменения могут повлиять на данные.

Первые доказательства связи курения табака со смертностью и заболеваемостью были получены в результате анализа данных с использованием регрессионного анализа. В регрессионной модели курение сигарет является представляющей интерес независимой переменной, а зависимой переменной является продолжительность жизни, измеряемая в годах.

В финансовой сфере модель ценообразования капитальных активов использует линейную регрессию для анализа и количественной оценки систематического риска инвестиций. Линейная регрессия является важным эмпирическим инструментом в экономике, например, для прогнозирования потребительских расходов, инвестиций в основной капитал, в товарно-материальные запасы, в спрос на рабочую силу.

Метод решающих деревьев. Это новый способ классификации и регрессии. Решающее дерево представляет собой направленный иерархический граф. Его вершинами являются признаки, по которым идет разделение выборки, а

листьями дерева – части выборки. Как правило в левом листе-потомке находится та часть выборки, для которой признак имеется (ответ на вопрос – «да»), а в правом листе-потомке – часть выборки, которая не имеет признака (ответ на вопрос – «нет»). Глубина дерева определяется числом уровней иерархии в решающем дереве. Ниже в качестве примера представлено дерево принятия решений по инвестициям в программное обеспечение.



В принципе правила, ведущие к каждой альтернативе, можно сформировать как логическую формулу.

Решающие деревья чаще всего применяются для задач классификации. Как известно, геометрический смысл классификации сводится к нахождению линии или плоскости, разделяющей объекты выборки на классы. Решающие деревья очень легко переобучаются, то есть решающее дерево можно очень хорошо «заточить» под конкретную выборку, но потом его нельзя будет применять для другой выборки.

Рассмотрим алгоритм обучения решающих деревьев.

Дерево строится путем разделения выборки на части по вводимым признакам. Признаки и порог их значения, по которым делится выборка, нужно подбирать так, чтобы в листьях дерева оставались объекты одного класса.

Пусть в вершине X_m объектов. Выбираем порог t по критерию ошибки Q для признака j, минимизируя критерий ошибки:

$Q(X_m, j, t) \to min,$

причем признаки j и значения порога перебираем так, чтобы выполнить данное условие. В итоге получаем две части выборки:

$X_l = \{x \in X_m | [x^j \leq t]\},$

$X_r = \{x \in X_m | [x^j > t]\}.$

Дерево нельзя строить до бесконечности: нужно знать, когда остановить разбиение. Используются три варианта остановки. Во-первых, если в вершину попал только один объект обучающей выборки или все объекты принадлежат

одному классу (в задачах классификации). Во-вторых, если глубина дерева достигла определенного значения. Наконец, задают количество объектов в листе дерева. На практике чаще задают глубину разделения.

Прогноз для листа дерева осуществляется следующим образом.

При использовании в качестве функционала ошибки в задачах регрессии значения среднеквадратичной ошибки, прогнозом будет среднее значение в листе a_m дерева:

$$a_m = \frac{1}{|X_m|}\sum_{i \in X_m} y_i.$$

В задачах классификации прогнозом будет наиболее популярный класс в листе a_m

$$a_m = argmax_{y \in Y}\sum_{i \in X_m} [y_i = y].$$

В алгоритме обучения выбор оптимального разбиения определяется критерием ошибки $Q(X_m, j, t)$:

$$Q(X_m, j, t) = \frac{|X_i|}{|X_m|}H(X_i) + \frac{|X_r|}{|X_m|}H(X_r),$$

где $H(X)$ – критерий информативности. Критерий показывает, насколько хорошо решающее дерево компонует объекты в своих листьях, то есть насколько они близки друг к другу. В случае задачи регрессии близость объектов в листе характеризуется степенью разброса значений в листе дерева:

$$H(X) = \frac{1}{|X|}\sum_{i \in X}(y_i - \bar{y}(X))^2.$$

В случае задач классификации используется критерий информативности Джини. Он не отрицателен и имеет максимум, если все объекты в листе принадлежат классу 1, а точнее, если в листе вообще один объект. Вычисляется критерий Джини по формуле:

$$H(X) = \sum_{k=1}^{K} p_k(1 - p_k),$$

где p_k – доля объектов класса k в выборке X:

$$p_k = \frac{1}{X}\sum_{i \in X}[y_i = k].$$

Также может быть использован энтропийный критерий информативности:

$$H(X) = -\sum_{k=1}^{K} p_k \ln p_k .$$

Он оптимален при условии, что все объекты X принадлежат классу 1.

Отметим, что примеры критериев ошибки, приведенные выше, относятся к моделям с бинарными признаками. Возникает вопрос, как быть с моделями, у которых категориальные признаки принимают множество значений? Можно ли к ним применить метод решающих деревьев?

В этом случае применяются два подхода: использование n-арных деревьев или использование бинарных деревьев с разбиением на множество значений.

При использовании n-арных деревьев у каждого узла X дерева, содержащего n-арный категориальный признак, имеется не два, а n потомков со значениями признака $\{c_1, c_2, \ldots, c_n\}$. Разбиение проводится по принципу минимизации функционала ошибки разбиения $Q(X_m, j)$:

$$Q(X_m, j) = \sum_{i=1}^{n} \frac{|X_i|}{|X_m|} H(X_i) \to \min_j.$$

При использование бинарных деревьев с разбиением на множество значений категориальные признаки заменяют на натуральные и сортируют значения. Далее назначают порог, по которому идет деление вершины на две части.

Метод случайного леса. Объединение множества решающих деревьев дает композицию алгоритмов, одной из разновидностей которой является «Случайный лес». Идея в том, что объединение множества слабых алгоритмов с невысокой точностью дает один сильный алгоритм с хорошей точностью.

В методе «Случайный лес» обучают каждый алгоритм из композиции, а ответом является усредненный результат по всем алгоритмам, входящим в композицию.

При решении задачи поиска регрессии ответ $a(x)$ находится по формуле

$$a(x) = \frac{1}{N} \sum_{n=1}^{N} b_n(x).$$

При решении задачи классификации ответ $a(x)$ находится по формуле

$$a(x) = sign \frac{1}{N} \sum_{n=1}^{N} b_n(x).$$

Например, результаты композиции из 10 бинарных классификаторов показали следующие ответы: 1, 1, –1, –1, –1, 1, 1, –1, –1. Тогда итоговый ответ имеет вид

$$a(x) = sign\left(-\frac{1}{10}\right) = -1.$$

Однако проблема этого метода в том, что обучать каждый алгоритм надо на своей части выборки. Почему нельзя 10 раз на всей выборке обучить разные алгоритмы? Дело в том, что есть следующая закономерность, выведенная экспериментально и не имеющая объяснения. Если 10 раз использовать одни и те же данные для разных алгоритмов обучения, то для каждого последующего алгоритма результат будет все лучше. Причем, если немного изменить выборку, то результаты не улучшаются. При использовании одних и тех же данных идет подстройка алгоритмов под данные, и она дает кажущиеся улучшения качества алгоритмов. Поэтому выборку надо разбивать на части и обучать алгоритмы на разных частях выборки.

Для разбиения выборки на части можно использовать случайный выбор подмножества из начальной выборки. Номер объекта из выборки задается

случайно, и он помещается в подвыборку. Однако выборка должна быть большая. В противном случае, если выборка небольшая, то можно прибегнуть к процедуре бутстрап.

Метод бэггинга. В этом методе на сгенерированных бутстрапом выборках обучаются алгоритмы, входящие в композицию. Каждый алгоритм получает свою выборку для обучения. Причем при обучении каждое решающее дерево строится до конца (до 1 элемента в листе или n_{min} элементов), а оптимальный признак, по которому идет разделение в выборке на каждом шаге построения дерева, выбирается случайно из множества признаков, и каждый раз используется новое множество признаков. Результаты, полученные на алгоритмах, усредняются. Метод предназначен для улучшения стабильности и точности алгоритмов машинного обучения, используемых в статистической классификации и регрессии. Бэггинг также уменьшает дисперсию и помогает избежать переобучения. Хотя он обычно применяется к методам обучения машин на основе деревьев решений, его можно использовать с любым видом метода.

Метод бустинга. В данном методе алгоритмы обучаются последовательно, и каждый последующий алгоритм учитывает ошибки предыдущего. Кроме того, при вычислении общего ответа композиции, вклад каждого алгоритма учитывается в зависимости от его точности, то есть для каждого алгоритма в композиции есть свой вес. За счет такого подхода можно получить хороший результат на меньшем количестве алгоритмов.

Рассмотрим работу метода бустинга на примере построения регрессии.

Вначале обучается первый алгоритм, минимизируя среднеквадратичную ошибку:

$$b_1(x) = armin\left(\frac{1}{n}\sum_{i=1}^{n}(b(x_i) - y_i)^2\right),$$

где $b(x_i)$ – ответ алгоритма композиции на объекте x_i; y_i – ответ объекта x_i в выборке.

На вход первому алгоритму подаются объекты выборки X и ответы на них Y.

Затем второй алгоритм обучается, минимизируя ошибки, полученные первым алгоритмом:

$$b_2(x) = armin\left(\frac{1}{n}\sum_{i=1}^{n}\left(b(x_i) - (y_i - b_1(x_i)\right)^2\right).$$

На вход ко второму алгоритму подаются объекты выборки X и ошибки первого алгоритма на всей выборке.

Продолжаем обучать алгоритмы, так чтобы очередной алгоритм уменьшал ошибку всей композиции:

$$b_N(x) = armin\left(\frac{1}{n}\sum_{i=1}^{n}\left(b(x_i) - (y_i - \sum_{i=1}^{N-1} b_n(x_i)\right)_1\right)^2\right).$$

Иными словами, метод бустинга включает следующие шаги.

1. Создаем первую модель (дерево решений или регрессор).
2. Обучаем на всей выборке.
3. Делаем предсказание.
4. Считаем среднеквадратичную ошибку.
5. Создаем следующую модель и обучаем ее на полученных в п. 4 ошибках и том же наборе параметров.
6. Суммируем предсказания полученных моделей.
7. Повторяем пп. 4–6, пока сумма предсказаний не перестанет меняться или разница не станет меньше заданной точности.

Несмотря на то, что бустинг является довольно эффективным методом обучения композиций алгоритмов, его можно ускорить за счет учета направления уменьшения ошибки алгоритма. Это происходит в градиентном бустинге. Здесь каждый новый алгоритм обучается уже на антиградиенте функционала ошибки предыдущего алгоритма. Правда, градиентный бустинг легко переобучается по причине того, что траектория градиентного спуска не является оптимальной. Пути для исправления ситуации известны: уменьшение шага на каждой итерации и стохастическое изменение объема выборки когда для обучения алгоритма берется не вся выборка, а ее случайная часть.

Обучение без учителя

Все подходы, рассмотренные ранее, имели ответы на обучающей выборке. Но бывают такие ситуации, когда ответов нет даже на обучающей выборке. Например, есть набор точек, и нужно понять, на что этот набор больше похож, то есть какой предмет он может описывать. Все случаи, когда ответов на выборке нет, относятся к обучению без учителя. Как правило, обучение без учителя является частью подготовки данных для анализа методами обучения с учителем, поскольку предсказательных моделей этот метод дать не может.

Основными задачами обучения без учителя являются кластеризация, визуализация и понижение размерности, поиск аномалий.

В задачах кластеризации необходимо сформировать группы (кластеры) объектов, схожих между собой. Схожесть объектов определяется расстоянием между ними: чем объекты ближе друг к другу, тем они более схожи. В задачах кластеризации невозможно изначально понять, сколько будет кластеров и какой будет размер кластеров. Кластеризацию часто проводят перед задачей классификации, чтобы понять, насколько разделимы данные.

Цель визуализации многомерных данных на плоскости – снизить размерность данных до двух или трех, чтобы представить их наглядно. После этого данные на плоскости могут образовать группы, а значит, данные разделимы на классы. Такой подход очень полезен перед построением классификации.

Задача поиска аномалий – найти те объекты выборки, которые по своим свойствам резко выделяются из выборки. Например, это позволяет определить нестандартные действия пользователя на сайте и заподозрить, что сайт взломан; определить несвойственные держателю карты траты и заподозрить, что карта

украдена и пр. Объекты, обладающие аномальными свойствами, в дальнейшем исследуются другими методами.

Задача понижения размерности и методы ее решения. Понижение размерности сводится к уменьшению числа параметров модели, причем важные параметры модели должны быть оставлены. На практике используются два подхода к решению задачи понижения размерности: отбор признаков и проекция признаков. Цель – отобрать значащие признаки, а незначащие признаки убрать из модели. Значащими признаками считаются те, которые имеют влияние на функционал ошибки модели, а именно уменьшают его.

Основными алгоритмами отбора признаков являются *алгоритмы обертывания, алгоритмы фильтров* и *алгоритмы вложения*.

Алгоритмы обертывания используют априорную оценку качества модели, затем модель обучается на наборе признаков, оценивается ее ошибка и выбирается набор с минимальной ошибкой. Наиболее известными алгоритмами обертывания являются полный перебор, «жадный» алгоритм и ADD-DEL. При полном переборе считается ошибка на всех возможных комбинациях параметров и выбирается комбинация, дающая минимальную ошибку. Это долгий и затратный по ресурсам алгоритм. В «жадном» алгоритме берется признак, дающий минимальную ошибку модели, и затем к нему последовательно добавляется по одному признаку. Если наблюдается уменьшение функционала ошибки, то продолжают добавлять, если уменьшения не наблюдается, то добавление останавливают (после 10 параметров точность модели уже почти не меняется). Наконец алгоритм ADD-DEL является модернизацией «жадного» алгоритма. Если в «жадном» алгоритме новые параметры только добавляются, то в алгоритме ADD-DEL параметр, который увеличивает функционал ошибки после его добавления в модель, удаляется из набора параметров. Алгоритм ADD-DEL быстрее, чем «жадный» алгоритм, достигает стабильной точности (ему хватает 10 параметров).

Алгоритм фильтров использует косвенные оценки качества модели, например, такие как расстояние между классами, а затем выбирается набор с минимумом по выбранному критерию. Алгоритм заключается в отборе параметров по заданному условию. Например, можно убрать из модели все сильно коррелирующие между собой признаки.

Алгоритмы вложения осуществляют отбор признаков в процессе построения модели. Они осуществляют отбор признаков сразу во время построения модели за счет, например введения регуляризаторов. Алгоритм *Lasso* является одним из алгоритмов вложения.

Второй подход к понижению размерности – это проекция признаков. Целью его является перевод данных из пространства высокой размерности в пространство малой размерности. Проекция признаков бывает линейной и нелинейной. При линейной проекции признаков последние проецируются на прямую линию. Линейную проекцию признаков реализует метод главных компонент (*PCA*). В методах нелинейной проекции признаков последние проецируются на различные кривые. Алгоритмов, реализующих нелинейную

проекцию признаков, много, наиболее популярными сейчас являются *SNE* и *t-SNE*.

Метод главных компонент предложил К. Пирсон. Постановка задачи в методе сформулирована так: выполнить аппроксимацию данных линейными многообразиями, то есть найти направления максимальной дисперсии данных. Эти направления служат основой для проекции данных. Действительно, если аппроксимировать точки на прямую, которая идет вдоль точек, то суть начальных данных исказится несильно и этими искажениями можно будет пренебречь. Зато вместо множества точек можно будет записать уравнение прямой, что существенно снизит размерность данных. Рассмотрим математическое описание постановки задачи. Дано множество векторов x_1, x_2,…, x_m. Требуется для каждого из них в k-мерном пространстве (k = 0, 1, 2, …, n–1), найти такое линейное многообразие L, что выполняется условие

$$\sum_{i=1}^{m} dist^2(x_i, L_k) \to min,$$

где $dist(x, L)$ – евклидово расстояние от точки x до линейного многообразия L.

Линейное многообразие L представляет собой взвешенную сумму ортогональных векторов a_0, a_1, …, a_k:

$$L_k = \{a_0 + \beta_1 a_1 + \beta_2 a_2 + \cdots + \beta_k a_k\},$$

иными словами, необходимо найти семейство прямых, вдоль которых идет наибольший разброс данных и которые проходят как можно ближе к данным.

Формально метод главных компонент включает следующую последовательность шагов.

1. Данные централизуются путем вычитания среднего.
2. Отыскивается первая главная компонента

$$a_1 = argmin\left(\sum \|x_i - a_1(a_1, x_i)\|^2\right).$$

3. Из данных вычитается проекция на первую главную компоненту

$$x_i = x_i - a_1(a_1, x_i).$$

4. Повторяются п. 2 и п. 3 до тех пор, пока a_n не станет единичным вектором.

Лекция 4. Решение задач классификации, кластеризации, прогнозирования, обнаружения аномалий методами машинного обучения

Задача кластеризации и методы ее решения. В задачах кластеризации выделяются близкие по параметрам объекты, которые объединяются в кластеры. Пусть имеется обучающая выборка $X_l = \{ x_1 \ldots, x_l\} \subset X$ и функция расстояния между объектами $\rho(x, x')$. Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами, так чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X_l$ приписывается метка (номер) кластера y_i.

Тогда алгоритм кластеризации – это функция $a: X \to Y$, которая любому объекту $x \in X$ ставит в соответствие метку кластера $y \in Y$. Множество меток Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров с точки зрения того или иного критерия качества кластеризации.

Основными этапами кластеризации являются следующие: (1) выделение параметров объекта, по которым можно проводить кластеризацию, (2) определение меры, по которой будет определяться расстояние между объектами, (3) разбиение объектов на кластеры, (4) визуализация кластеров.

Для определения функции $\rho(x, x')$, по которой будет определяться расстояние между объектами, существует несколько мер: евклидово расстояние; квадрат евклидова расстояния; манхэттенское расстояние; расстояние Чебышева; степенное расстояние.

Евклидово расстояние определяется как геометрическое расстояние между двумя точками в многомерном пространстве:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2},$$

где x, x' – координаты точек.

Квадрат евклидова расстояния применяется для придания большего веса более отдаленным друг от друга объектам. Определяется по формуле

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2.$$

Манхэттенское расстояние является средним разностей по координатам:

$$\rho(x, x') = \sum_{i}^{n} |x_i - x_i'|.$$

Расстояние Чебышева может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате. Определяется по формуле

$$\rho(x, x') = \max_{i=1,\dots,n} |x_i - x'_i|.$$

Степенное расстояние применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Вычисляется по формуле

$$\rho(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x'_i)^p},$$

где r и p – параметры, определяемые пользователем. Параметр p ответственен за постепенное взвешивание разностей по отдельным координатам, параметр r ответственен за прогрессивное взвешивание больших расстояний между объектами. Если оба параметра – r и p – равны двум, то это расстояние совпадает с евклидовым расстоянием.

Рассмотрим методы решения задачи кластеризации: метод ближайших соседей (*kNN*), метод *k*-средних (*k-means*), иерархические методы.

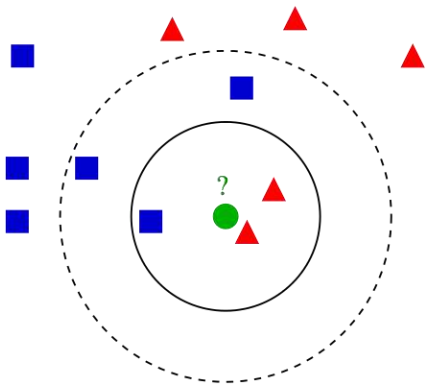
Метод ближайших соседей берет точку и ищет ее ближайших соседей. Определяется, к какому кластеру относятся большинство ближайших соседей. Считается, что точка относится к тому кластеру, что и большинство ближайших соседей. Метод применим как в кластеризации, так и в классификации и регрессии.

В случае кластеризации определяются ближайшие соседи и их параметры. Тогда объект, для которого проводится кластеризация и для которого нужно определить недостающие параметры, принимает параметры ближайших соседей.

В случае классификации используется обучающая выборка с указанными классами, а затем новый объект приписывают тому классу, к которому он ближе.

В случае регрессии объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны.

Рассмотрим пример использования *kNN*. Допустим, у нас есть объект виде зеленого круга. Синие квадраты мы обозначим как класс 1, красные треугольники – класс 2.



Зеленый круг должен быть классифицирован как класс 1 или класс 2. Если рассматриваемая нами область является малым кругом, то объект классифицируется как 2-й класс, потому что внутри данного круга 2 треугольника и только 1 квадрат. Если мы рассматриваем большой круг (с

пунктиром), то круг будет классифицирован как 1-й класс, так как внутри круга 3 квадрата в противовес 2 треугольникам.

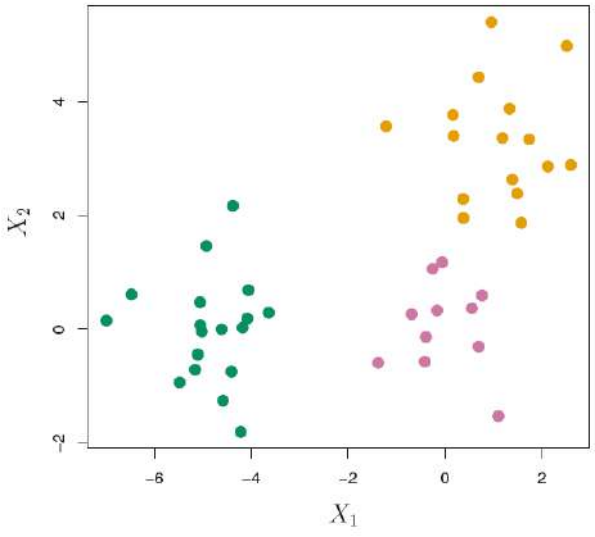
Преимуществами метода *kNN* являются простота в реализации и понимании, нечувствительность к выбросам данных, универсальность. Однако метод работает значительно медленнее при увеличении объема выборки и независимых переменных. К тому же всегда нужно определять оптимальное значение *k*.

Метод *k*-средних является итерационным. Метод предусматривает следующую последовательность шагов: (1) выбирается количество кластеров; (2) случайным образом определяются центры кластеров; (3) определяется для каждой точки центр, к которому она ближе по выбранной мере (обычно евклидово или манхэттенское расстояние) им относятся; (4) пересчитываются координаты центров кластеров как среднее значение координат всех точек кластера; (5) повторяем, пока расстояние до центра кластера не достигнет значения меньше заданной величины; (6) остановка метода происходит тогда, когда центры кластеров перестают сдвигаться.

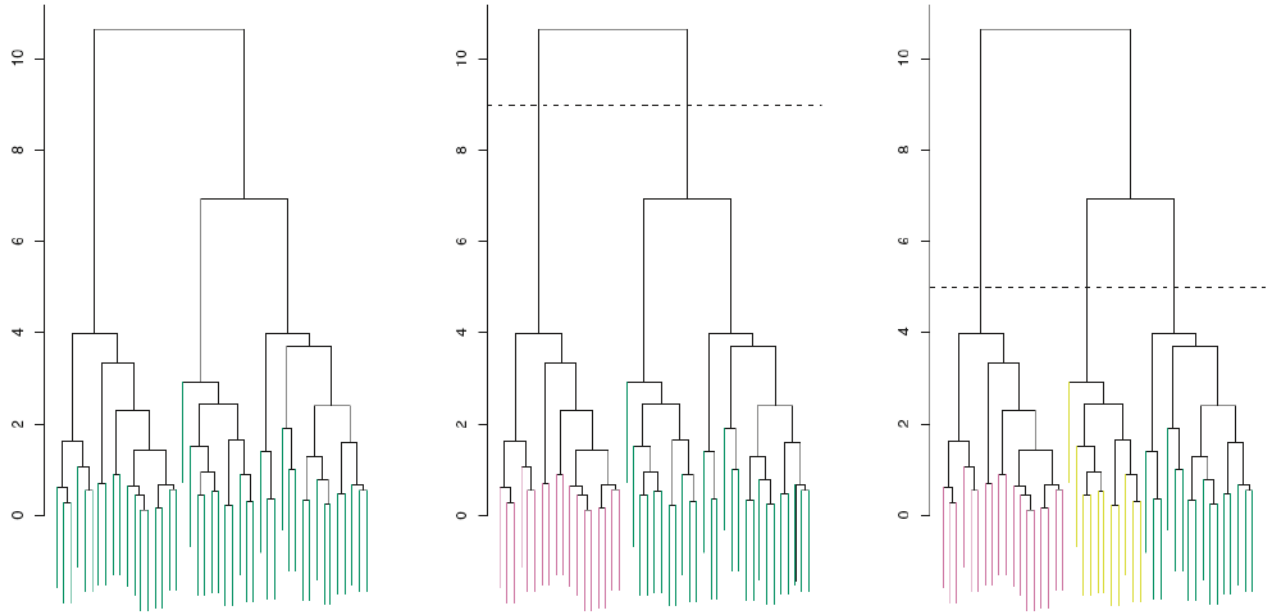
Методу отдается предпочтение из-за его простоты реализации, большой скорости (а это очень важно при работе с видео). Недостаток метода состоит в том, что нужно задавать начальное количество кластеров *k* и их центры.

Основными типами иерархических методов являются дивизимные, которые разбивают выборку на все более и более мелкие кластеры, а также агломеративные, в которых объекты объединяются во все более и более крупные кластеры. Иерархическая кластеризация – это альтернативный подход, который не требует выбора *k*. Результат иерархической кластеризации представляется в виде дерева, называемого дендрограммой. Рассмотрим на примере агломеративную кластеризацию, как наиболее общий тип иерархической кластеризации. Дендрограмма строится, начиная с листьев, и объединяет кластеры к корню дерева.

Ниже представлены искусственно сгенерированные 45 наблюдений в 2-мерном пространстве (x_1,x_2), разделенные на 3 класса. Предположим, что мы наблюдаем данные без метки класса и применяем для них метод иерархической кластеризации.



Результатом работы метода являются следующие дендрограммы:



В левой части рисунка каждый лист дендрограммы представляет одно из 45 исходных наблюдений. По мере продвижения вверх к корню, некоторые листья начинают сливаться в ветви. Они соответствуют наблюдениям, которые похожи друг на друга. При продвижении вверх по дереву ветви сливаются с листьями или с другими ветвями. Чем раньше (ниже на дереве) происходит слияние, тем больше похожи группы наблюдений друг на друга. С другой стороны, наблюдения, которые сливаются позже (возле верхушки дерева), могут сильно отличаться. Это утверждение можно сформулировать так: для любых двух наблюдений можно найти точку на дереве, где ветви, содержащие эти наблюдения, впервые сливаются. Высота этой точки слияния, измеренная по вертикальной оси, указывает, насколько различаются эти наблюдения. Таким образом, наблюдения, которые сливаются в самом низу дерева, очень похожи друг на друга, в то время как наблюдения, которые сливаются близко к верху дерева, будут скорее всего сильно различаться.

Разрежем дендрограмму по горизонтали, как это показано в центре и справа на рисунке. Отдельные множества наблюдений, которые находятся снизу от разреза, могут рассматриваться в качестве кластеров. В центре разрез дендрограммы на высоте 9 дает 2 кластера, показанных разными цветами. Справа разрез дендрограммы на высоте 5 дает 3 кластера. Можно сделать и другие разрезы дендрограммы. Один кластер получится, если не разрезать, а на высоте 0 будет 45 кластеров, т.е. каждое измерение будет отдельным кластером. Другими словами, высота разреза дендрограммы играет ту же роль, что и k в кластеризации методом k -средних: контролирует число полученных кластеров.

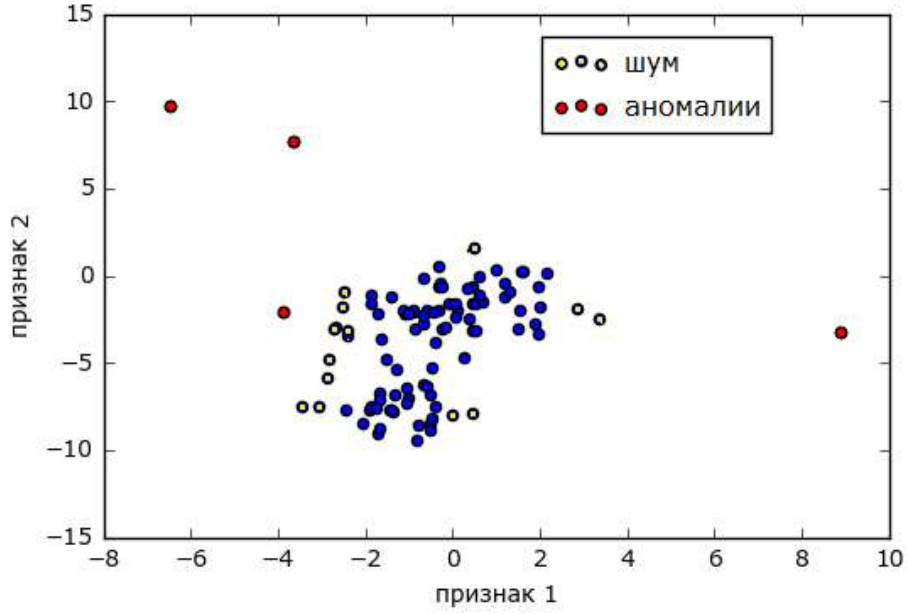
Таким образом, одна дендрограмма может использоваться для получения любого числа кластеров. На практике часто приемлемое число кластеров выбирается визуально на основе высоты слияния и желаемого числа кластеров. В случае рисунка выше таким будет выбор 2 или 3 кластеров. Однако, не всегда выбор так очевиден.

Агломеративный метод иерархической кластеризации начинается с определения меры отличия между каждой парой наблюдений. Чаще всего используется евклидово расстояние. Алгоритм продолжается итеративно. Начиная с дна дендрограммы каждое из *n* наблюдений рассматриваются свой собственный кластер. Два кластера, которые наиболее подобны, сливаются, так что становится (*n*-1) кластер. Затем опять наиболее подобные два кластера сливаются, и становится (*n*-2) кластера. Алгоритм продолжается, пока не получится один кластер, и дендрограмма не станет завершенной.

Чтобы применять данный метод кластеризации, необходимо сначала принять несколько решений: какую меру расстояния использовать? какой тип связи использовать? где разрезать дендрограмму, чтобы получить кластеры? Каждое из этих решений сильно влияет на результат. На практике пробуют несколько вариантов, а затем выбирают тот, которые дает наиболее полезное или объяснимое решение. Нет единственного правильного ответа – каждое решение может раскрывать какие-нибудь интересные факты в данных.

Задача поиска аномалий и методы ее решения. В анализе данных есть два направления, которые занимаются поиском аномалий: обнаружение выбросов и «новизны». Как и выброс «новый объект» – это объект, который отличается по своим свойствам от объектов (обучающей) выборки. Но в отличие от выброса, его в самой выборке пока нет (он появится через некоторое время, и задача как раз и заключается в том, чтобы обнаружить его при появлении). Например, если вы анализируете замеры температуры и отбрасываете аномально большие или маленькие, то вы боретесь с выбросами. А если вы создаете алгоритм, который для каждого нового замера оценивает, насколько он похож на прошлые, и выбрасывает аномальные – вы «боретесь с новизной».

Выбросы являются следствием ошибок в данных (неточности измерения, округления, неверной записи и т.п.), наличия шумовых объектов (неверно классифицированных объектов), присутствия объектов «других» выборок (например, показаниями сломавшегося датчика):



На рисунке видно, что шум – это выброс, который немного размывает границы кластера. Аномалии искажают эти границы. Новизна же появляется в результате принципиально нового поведения объекта. Скажем, если наши объекты – описания работы системы, то после проникновения в неё вируса объекты становятся «новизной». Здесь важно понимать, что «новизна» называется новизной по той причине, что такие описания для нас абсолютно новые, мы не можем в обучающей выборке иметь информацию о всевозможных заражениях вирусами. Формирование такой обучающей выборки трудозатратно и часто не имеет смысла. Зато можно набрать достаточно большую выборку примеров нормальной работы системы.

Приложений задачи поиска аномалий много: обнаружение подозрительных банковских операций, обнаружение вторжений, обнаружение неполадок в механизмах по показаниям датчиков, медицинская диагностика, сейсмология и др. Возможных постановок задач здесь тоже много. Функционалы качества в задачах поиска аномалий используют примерно такие же, как и в задачах классификации.

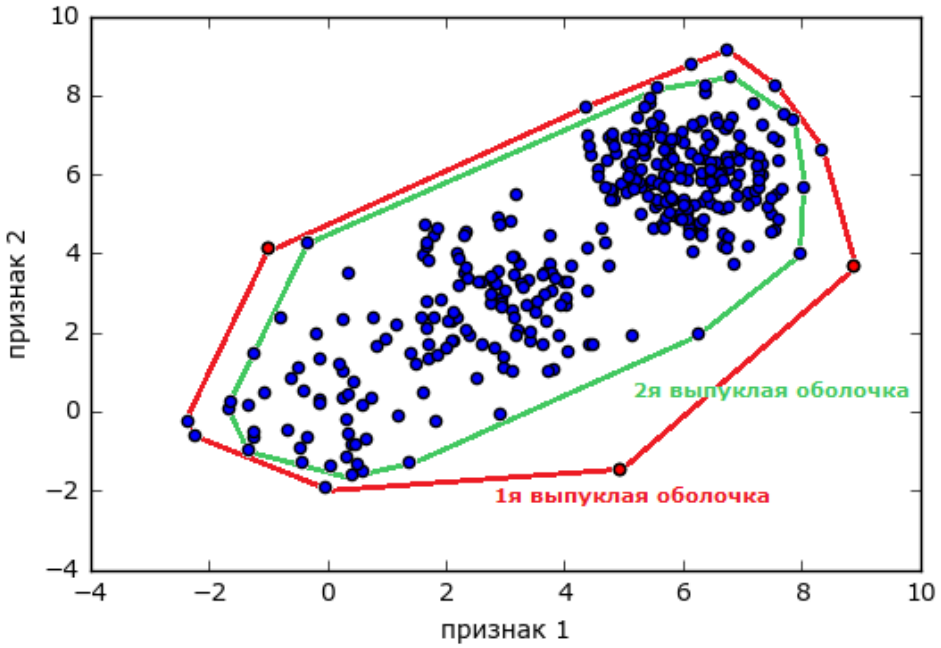
Рассмотрим подробнее методы обнаружения выбросов: статистический тест, итерационный и метрический методы, методы машинного обучения, ансамбли алгоритмов.

Статистический тест, как правило, применяют для отдельных признаков и отлавливают экстремальные значения. Многие методы визуализации, например «ящик с усами», имеют встроенные средства для обнаружения и показа таких экстремальных значений. Однако важно понимать, что экстремальное значение и аномалия – это разные понятия. Например, в небольшой выборке

[1, 33, 2, 1, 101, 2, 1, 100, 1, 3, 101, 1, 3, 100, 101, 100, 100]

значение 33 можно считать аномалией, хотя оно не является максимальным или минимальным. Иными словами, аномалия характеризуется не только экстремальными значениями отдельных признаков.

Итерационный метод состоит из итераций, на каждой из которых удаляется группа «особо подозрительных объектов». Например, в n-мерном признаковом пространстве можно удалять выпуклую оболочку точек-объектов, считая её представителей выбросами.



В метрических методах постулируется существование некоторой метрики в пространстве объектов, которая и помогает найти аномалии. Интуитивно понятно, что у выброса мало соседей, а у типичной точки много. Поэтому хорошей мерой аномальности может служить, например «расстояние до *k*-го соседа». Здесь используются специфические метрики, например расстояние Махаланобиса.

Популярными методами машинного обучения для решения задачи поиска аномалий являются метод опорных векторов (*SVM*) для одного класса, изолирующий лес и эллипсоидальная аппроксимация данных. Метод *SVM* отделяет выборку от начала координат и, скорее, является методом поиска новизны, а не выбросов, так как использует обучающую выборку. Метод изолирующего леса является вариацией метода случайного леса. Строится множество деревьев до исчерпания выборки, для построения ветвления в дереве выбирается случайный признак и случайное расщепление. Для каждого объекта мера его нормальности – среднее арифметическое глубин листьев, в которые он попал (изолировался). Идея метода заключается в том, что при построении деревьев выбросы будут попадать в листья на ранних этапах (на небольшой глубине дерева), т.е. выбросы проще «изолировать» (дерево строится до тех пор, пока каждый объект не окажется в отдельном листе). Метод хорошо отлавливает именно выбросы. Эллипсоидальная аппроксимация данных заключается в том, что облако точек моделируется как внутренность эллипсоида. Метод работает только на одномодальных нормально распределённых данных.

Идея ансамбля алгоритмов также используется при решении задач обнаружения аномалий. Каждый из алгоритмов даёт оценку аномальности и эти оценки потом «усредняют». Поскольку ключевым моментов в реальных задачах обнаружения аномалий является выбор признаков, которые характеризуют те или иные отклонения от нормы, то алгоритмы из ансамбля строят, пытаясь угадать хорошие пространства. Здесь популярен баггинг, когда для каждого алгоритма берут случайное признаковое подпространство или когда в

выбранном случайном признаковом подпространстве совершают случайный поворот.

Обучение с подкреплением

Обучение с подкреплением — один из способов машинного обучения, в ходе которого испытуемая система (агент) обучается, взаимодействуя с некоторой средой. Откликом среды (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или её модель. Агент воздействует на среду, а среда воздействует на агента. О такой системе говорят, что она имеет обратную связь.

Базовая модель обучения с подкреплением включает (1) множество состояний окружающей среды S; (2) множество действий A; (3) правила перехода между состояниями; (4) правила, определяющие вознаграждение за переход; (5) правила, описывающие действия агента.

Обучающий агент с подкреплением взаимодействует со средой. В каждый момент времени t агент получает наблюдение O_t, включающее вознаграждение R_t. Затем он выбирает действие из множества A. Среда переходит в новое состояние S_{t+1}, в агент получает вознаграждение R_{t+1}, связанное с переходом (S_t; A_t; S_{t+1}). Цель обучающего агента с подкреплением состоит в том, чтобы получить как можно больше вознаграждения. Агент может выбирать любое действие в зависимости от предыстории, включая случайный выбор действия.

Например, при обучении беспилотных автомобилей у машины нет задачи запомнить подробную карту города, страны, континента, все улицы и повороты. Но она обязательно должна понять шаблоны повторяющихся ситуаций и обобщить их. Поэтому первая цель робота в обучении с подкреплением – минимизировать ошибки. Машина учится анализировать информацию перед каждым следующим ходом. Беспилотный автомобиль во время обучения учится вовремя реагировать на сигнал светофора, остановиться перед пешеходом на переходе, пропустить быстро движущийся автомобиль или спецтранспорт. Чтобы достичь лучшего результата, машина обучается в виртуальной модели города со случайными пешеходами и другими участниками дорожного движения. Вторая цель робота – получить от выполнения задания максимальную выгоду. Сама выгода при этом должна быть запрограммирована заранее: максимально быстрое время прохождения маршрута, оптимальное расходование ресурсов, обслуживание как можно большего количества посетителей.

Чтобы получить большое вознаграждение, обучающийся с подкреплением агент должен предпочитать действия, которые были испробованы в прошлом и принесли вознаграждение. Но чтобы найти такие действия, он должен пробовать действия, которые раньше не выбирал. Агент должен использовать уже приобретенный опыт, чтобы получить вознаграждение, но должен продолжать исследования, чтобы выбирать более эффективные действия в будущем. Дилемма состоит в том, что одного лишь исследования или использования недостаточно для успешного решения задачи. Агент должен пробовать разные

действия и неуклонно отдавать предпочтение тем, которые кажутся наилучшими.

В стохастической задаче каждое действие необходимо испробовать много раз, чтобы получить надежную оценку ожидаемого вознаграждения. Проблема исследования-использования интенсивно изучалась математиками на протяжении многих десятилетий, но и по сей день остается нерешенной. Отметим, что вопрос о нахождении баланса между исследованием и использованием вообще не возникает в обучении с учителем и без учителя.

Рассмотрим обучение с подкреплением на примере задачи о многоруком бандите. Вы многократно стоите перед выбором из k разных вариантов, или действий. После каждого выбора вы получаете численное вознаграждение, выбираемое из стационарного распределения вероятностей, которое зависит от выбранного действия. Ваша цель – максимизировать ожидаемое полное вознаграждение за период, например, после выбора действия 1000 раз (за 1000 временных шагов).

Это исходная постановка задачи о многоруком бандите, названной так по аналогии с игровым автоматом. Только в этом случае рычагов у автомата не один, а k. Каждый выбор действия соответствует опусканию одного из рычагов игрового автомата, а вознаграждения – это выплаты за выпадение джекпота. Благодаря повторному выбору действий мы стремимся максимизировать свой выигрыш, концентрируя усилия только на лучших рычагах.

В задаче о многоруком бандите с каждым из k действий связано ожидаемое, или среднее, вознаграждение при условии выбора этого действия; будем называть его ценностью действия. Обозначим A_t действие, выбранное на временном шаге t, а R_t – соответствующее ему вознаграждение. Тогда ценность произвольного действия a, обозначаемая $q^*(a)$, – это математическое ожидание вознаграждения при условии выбора a. Если бы мы знали ценность каждого действия, то задача о k-руком бандите решалась бы тривиально: нужно было бы всегда выбирать действие с максимальной ценностью. Предположим, что достоверно ценности действий неизвестны, но имеются оценки. Обозначим $Q_t(a)$ оценку ценности действия a на временном шаге t. Мы хотели бы, чтобы $Q_t(a)$ было близко к $q^*(a)$.

Если мы запоминаем оценки ценности действий, то на любом временном шаге имеется по крайней мере одно действие с максимальной оценкой. Назовем эти действия жадными. Если выбирается любое из таких действий, то мы используем текущие знания о ценности действий. Если же выбирается какое-то нежадное действие, то мы занимаемся исследованием, поскольку это позволяет улучшить оценку ценности нежадного действия. Использование дает возможность максимизировать ожидаемое вознаграждение на одном шаге, а исследование может дать большее суммарное вознаграждение в длительной перспективе. Предположим, например, что ценность жадного действия известна достоверно, тогда как оценки других действий почти такие же хорошие, но со значительной долей недостоверности. Недостоверность такова, что по меньшей мере одно из этих прочих действий на самом деле лучше жадного, но какое именно, мы не знаем.

Если впереди много временных шагов, на которых можно выбирать действия, то может оказаться выгоднее исследовать нежадные действия и выяснить, какие из них лучше жадных. В краткосрочной перспективе вознаграждение во время исследования будет ниже, но в долгосрочной – выше, потому что, обнаружив лучшее действие, мы сможем использовать его много раз. Есть много методов нахождения баланса между исследованием и использованием для различных математических постановок задачи о многоруком бандите.

Рассмотрим простой метод балансирования для задачи о многоруком бандите. Это метод ценности действий. Ценность действия – это среднее вознаграждение при условии выбора этого действия. Эту величину можно оценить, усреднив фактически полученные вознаграждения:

$$Q_t(a) = \frac{\text{Сумма вознаграждений при выборе } a \text{ до момента } t}{\text{Сколько раз } a \text{ выбиралось до момента } t}.$$

Простейшее правило – выбирать действие с наибольшей оценкой ценности, т. е. одно из жадных действий:

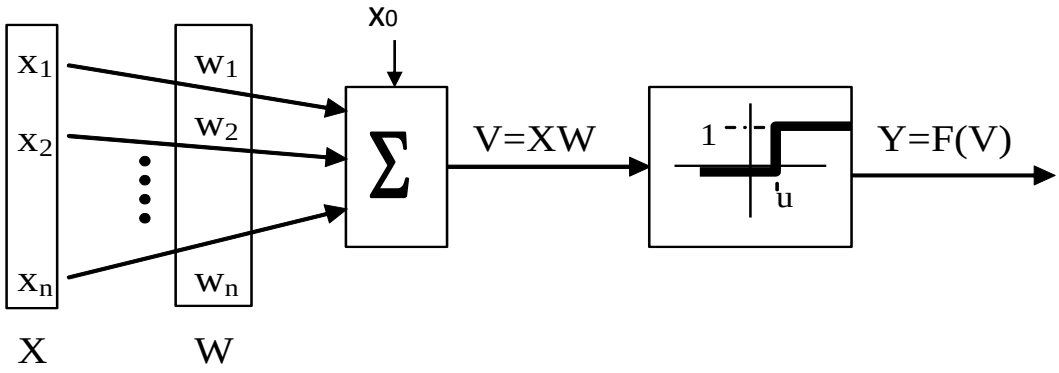
$$A_t = arg\max_a Q_t(a),$$

где *argmax* обозначает действие, для которого следующее далее выражение принимает максимум. При выборе жадного действия всегда используется текущее знание, чтобы максимизировать немедленное вознаграждение. Проверка, не окажутся ли менее доходные действия более выгодными не производится. Простая альтернатива – вести себя жадно большую часть времени, но иногда, скажем с малой вероятностью ε, случайным образом выбирать какое-то из прочих действий с одинаковой вероятностью, не зависящей от оценок ценности действий.

Метод, в котором применяется такое почти жадное правило выбора действия, называется ε-жадным. Преимущество состоит в том, что в пределе, когда число шагов стремится к бесконечности, каждое действие будет случайно выбрано бесконечное число раз, поэтому все $Q_t(a)$ сходятся к $q^*(a)$. Отсюда следует, что вероятность выбора оптимального действия сходится к числу, большему $(1 - \varepsilon)$, т. е. оптимальный выбор почти гарантирован. Но эти гарантии асимптотические, а о практической эффективности они мало что говорят.

Обучение глубоких сверточных нейронных сетей

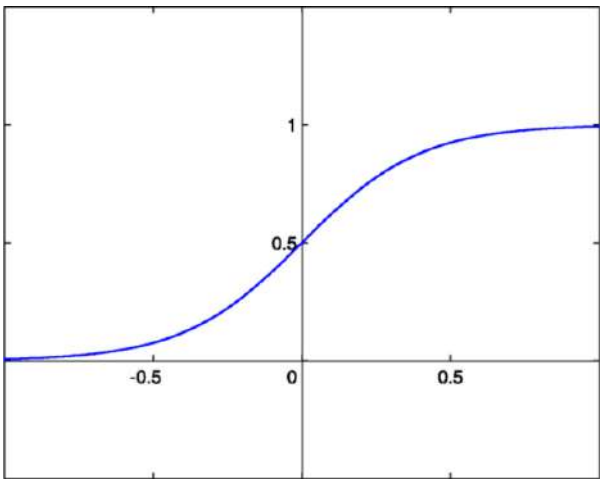
В 1943 г. в своей статье «Логическое исчисление идей, относящихся к нервной активности» У. Мак-Каллок и У. Питтс предложили модель искусственного нейрона:



На вход нейрона поступает некоторое множество сигналов x_j, j=1..n, каждый из которых является выходом другого нейрона. Нейрон вычисляет взвешенную сумму $V=\boldsymbol{XW}$ входных сигналов x_j и формирует на выходе сигнал величины 1, если эта сумма превышает определенный порог θ, и 0 - в противном случае.

Д. Хебб в работе «Организация поведения» в 1949 г. описал основные принципы обучения нейронов. Эти идеи развил Ф. Розенблатт. Он предложил схему устройства, моделирующего процесс человеческого восприятия, и назвал его «перцептроном». Перцептрон передавал сигналы от фотоэлементов, представляющих собой сенсорное поле, в блоки электромеханических ячеек памяти. В 1960 г. был продемонстрирован первый нейрокомпьютер — «Марк-1», который был способен распознавать некоторые буквы английского алфавита. На фоне роста популярности нейронных сетей в 1969 г. вышла книга М. Минского и С. Пайперта, которая показала принципиальные ограничения перцептронов. Согласно современной терминологии, перцептроны могут быть классифицированы как искусственные нейронные сети с одним скрытым слоем, с пороговой передаточной функцией и с прямым распространением сигнала.

Для обучения многослойных сетей рядом учёных, в том числе Д. Румельхартом, был предложен градиентный алгоритм обучения с учителем, проводящий сигнал ошибки, вычисленный выходами перцептрона, к его входам, слой за слоем. Сейчас это самый популярный метод обучения многослойных перцептронов. Его преимущество в том, что он может обучить все слои нейронной сети, и его легко просчитать локально. Однако этот метод является достаточно трудоемким. Для его применения нужно, чтобы передаточная функция нейронов (например, сигмоида $y=1/(1+exp(-ax))$ была дифференцируемой:



При этом в перцептронах пришлось отказаться от бинарного сигнала, и пользоваться на входе непрерывными значениями.

Основные задачи, для решения которых используются нейронные сети, это распознавание образов (текстов, звуков, изображений), прогнозирование, сжатие данных, попытка создания ИИ Дж. Хокинсом (*HTM*), принятие решений и управление (автомобили, роботы), ассоциативная память.

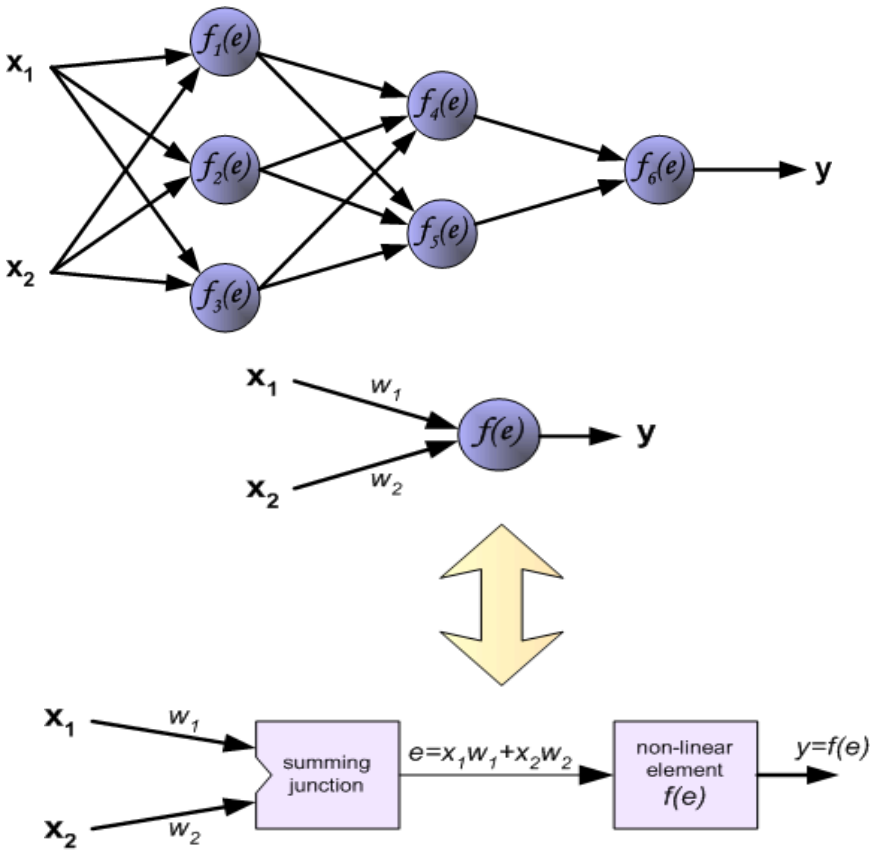
Для распознавания образов, обычно, используют сверточные сети, неокогнитрон, а для распознавания видео- и аудио-потока рекуррентные сети. Обучение происходит только с учителем, когда мы четко даем понять нейронной сети, что это изображение человека, машины, буквы, слова и т. д.

Для решения задач прогнозирования используют сети *GRNN*, *RMLP*, сеть Эльмана, сеть Фальмана. Одной из самых популярных является сеть *GRNN*, которая дает то же качество распознавания, что и обычный регрессионный анализ в математике, но при этом менее требовательна к входным данным.

Для решения задач, связанных с ассоциативной памятью, используют сеть Коско (двунаправленная ассоциативная память).

Самая важная часть в нейросети – это ее обучение. Фактически это многопараметрическая задача нелинейной параметризации. Технически обучение заключается в нахождении коэффициентов связей между нейронами. Обучение бывает с учителем, когда, подавая информацию на вход нейронной сети, мы четко указываем, что определенная информация соответствует конкретному выходу, и самообучение, когда нейронная сеть сама делает выводы по той информации, которую мы подали.

Основным алгоритмом обучения многослойных нейросетей является алгоритм обратного распространения ошибки (*back propagation*). Рассмотрим его работу на примере. Дана сеть, содержащая два входных элемента, два скрытых элемента и один выходной элемент. Все связи идут от входного слоя к выходному. К нейронам скрытого слоя доступа нет:



Шаг 1. Формируется обучающая выборка. Обычно начальные значения весов всех межнейронных связей полагаются случайными числами.

Шаг 2. Нейросети предъявляется образ ***X*** из обучающей выборки, сигнал от которого распространяется через нейроны скрытых слоев до выхода:

$y_1 = f_1\left(w_{(x1)1}x_1 + w_{(x2)1}x_2\right),$

$y_2 = f_2\left(w_{(x1)2}x_1 + w_{(x2)2}x_2\right),$

$y_3 = f_3\left(w_{(x1)3}x_1 + w_{(x2)3}x_2\right),$

$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3),$

$y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3),$

$y \quad = f_6(w_{46}y_4 + w_{56}y_5).$

Шаг 3. Выходной сигнал y сравнивается с желаемым выходным сигналом z, который хранится в обучающей выборке. Разница между этими двумя сигналами называется ошибкой δ выходного слоя сети:

$\delta = z - y.$

Шаг 4. Идея заключается в распространении сигнала ошибки δ обратно на все нейроны вплоть до входов:

$\delta_5 = w_{56}\delta,$

$\delta_4 = w_{46}\delta.$

Если ошибка пришла от нескольких нейронов – она суммируется:

$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5,$

$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5,$

$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5,$

Функционал квадратичной ошибки сети для данного входного образа имеет вид:

$$E = \frac{1}{2}\delta^2.$$

Данный функционал подлежит минимизации. Градиентный метод минимизации состоит в итерационной корректировке межнейронных связей согласно формуле:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j \frac{df_j(e)}{de} y_i.$$

Здесь $\frac{df_j(e)}{de}$ – производная функции активации нейрона, чьи веса корректируются. Поэтому функция должна иметь первую производную. Чаще всего в качестве функции активации используется сигмоида:

$$f(e) = \frac{1}{1+\exp(-e)}.$$

Производная от функции *f*(*e*) выражается через саму функцию:

$$\frac{df(e)}{de} = f(e)\cdot\big(1-f(e)\big).$$

Это позволяет сократить вычислительную сложность алгоритма обучения.

Шаг 5. Производится корректировка весов межнейронных связей:

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1,$$

$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2,$$

$$w'_{(x1)2} = w_{(x1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1,$$

$$w'_{(x2)2} = w_{(x2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2,$$

$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1,$$

$$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2,$$

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1,$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2,$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3,$$

$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1,$$

$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2,$$

$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3,$$

$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4,$$

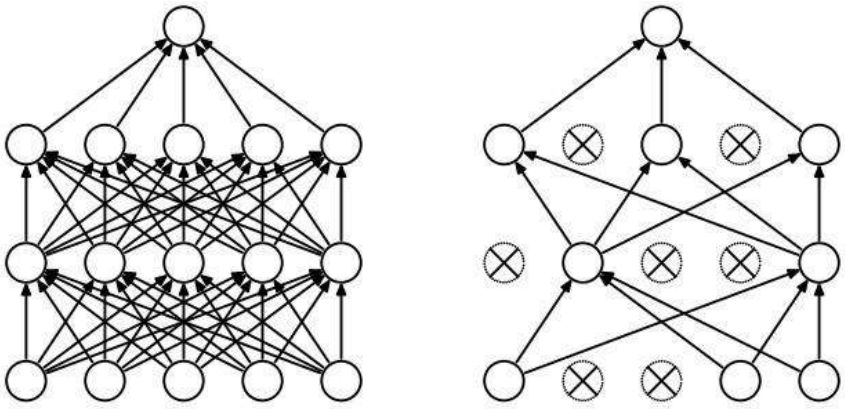
$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5.$$

Коэффициент η влияет на скорость обучения сети. Есть несколько способов для выбора этого параметра. Первый способ – начать обучение с большим значением η. Во время коррекции весовых коэффициентов параметр постепенно уменьшается. Второй способ – начать обучение с малых значений η. Далее, во время коррекции увеличивается, а затем снова уменьшается на завершающей стадии обучения.

Шаги 2-5 повторяются для всех обучающих примеров. Обучение завершается по достижении малых значений функционала квадратичной ошибки E сети или заранее заданного числа итераций.

Переобучение (*overfitting*) – одна из проблем обучения нейронных сетей. Нейросеть хорошо объясняет только примеры из обучающей выборки, адаптируясь к обучающим примерам, вместо того чтобы учиться классифицировать примеры, не участвовавшие в обучении. Теряется способность к обобщению. За последние годы было предложено множество решений проблемы переобучения, но одно из них превзошло все остальные, благодаря своей простоте и практическим результатам. Это – "метод прореживания" ("dropout"). Главная идея метода – вместо обучения одной сети обучить ансамбль нескольких сетей, а затем усреднить полученные результаты.

Сети для обучения получаются с помощью исключения из сети нейронов с вероятностью p. Вероятность того, что нейрон останется в сети, составляет $q = 1–p$. "Исключение" нейрона означает, что при любых входных данных или параметрах он возвращает 0. Исключенные нейроны не вносят свой вклад в процесс обучения ни на одном из этапов алгоритма обратного распространения ошибки (*backpropagation*); поэтому исключение хотя бы одного из нейронов равносильно обучению новой нейронной сети. Ниже представлена графическая интерпретация метода прореживания, взятое из статьи, в которой он впервые был представлен:



Слева – нейронная сеть до того, как к ней применили прореживание, справа – та же сеть после прореживания. Сеть, изображенная слева, используется при

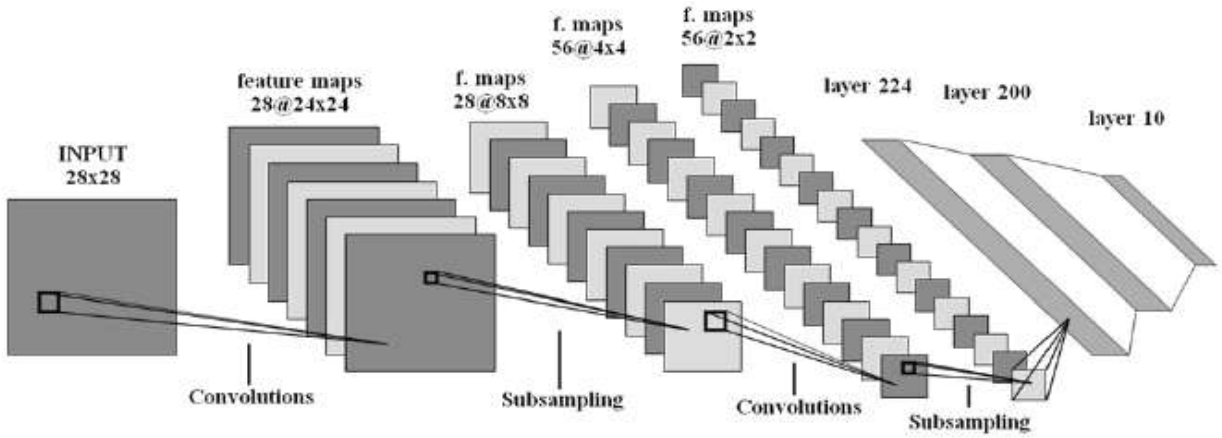
тестировании, после обучения параметрам. Метод хорошо работает на практике, потому что предотвращает взаимоадаптацию нейронов на этапе обучения.

Рассмотрим подробнее особенности обучения сверточных нейросетей. В 2006 г. компания *Microsoft* в ответ на негодование пользователей по поводу технологии *Captcha* предложила пользователям для подтверждения того, что они не являются компьютерным ботом выбирать кошечек и собак. Данное решение не просто сделало рутинную процедуру подтверждения более приятной, но и носила социально-полезный характер: фотографии кошек и собак подгружались с сайтов бездомных животных, и можно было перейти по ссылке и получить сведения о месте нахождения животного, если появилось желание взять его домой.

В то время компьютер решал задачу определения находится на фотографии кошка или собака с вероятностью 60 %. Решение считалось вполне удачным. Спустя 8 лет благодаря технологии глубоких сверхточных нейронных сетей были созданы алгоритмы, решающие данную задачу уже с вероятностью 98,9 %. Данная задача стала классической задачей обучения нейронной сети «*Dogs vs. Cats*». Она больше не подходила для сервиса *Captcha* и ушла в историю.

Что такое сверточная нейронная сеть? В принципе это нейронная сеть прямого распространения со сложной функцией. На вход нейросети поступает изображение, то есть просто массив чисел, а затем к изображению применяется последовательность определенных преобразований. Основное преобразование, которое происходит в процессе обработки изображения, называется обобщенная свертка.

Нейроны сверточной сети разбиты на группы, называемые слоями, и когда такая слоистая сеть применяется к данным, то происходит активация этих слоев, выходные значения нейронов подсчитываются последовательно.

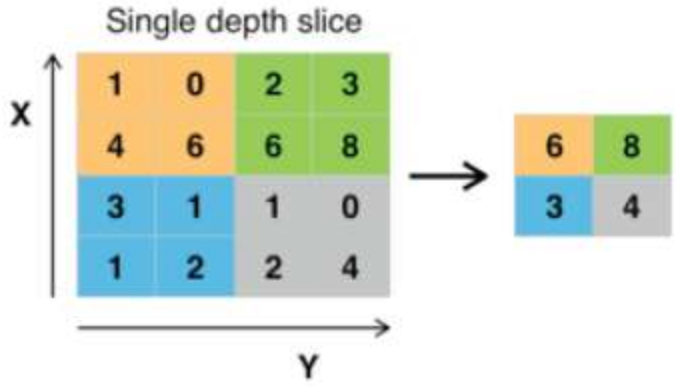


Сначала подсчитывается значение активации первого слоя, потом значение активации второго слоя и так далее до последнего слоя. Активация последнего слоя служит выходами нейронной сети. В такой сети есть много параметров, определяющих как активация следующего слоя зависит от активации предыдущего. Более того, активации внутри одного слоя могут подсчитываться одновременно, они друг от друга не зависят. Это приводит к тому, что такие сети

можно очень удобно и эффективно обрабатывать на современных графических процессорах, используя технологии параллельных вычислений.

Обучение такой сети происходит на большом количестве обучающих данных таким образом: для каждого обучающего примера известны значения выходов. Если мы берем текущее состояние нейросети и применяем к ней конкретный обучающий пример, смотрим, что получилось на выходе и как это отличается от того, что мы хотели бы увидеть. Далее, с помощью несложных математических манипуляций, которые называются методом обратного распространения ошибки, корректируются веса (параметры) межнейронных связей каждого слоя так, чтобы на выходе получалось что-то более похожее на то, что мы хотим увидеть.

Постепенно, подстраивая свои параметры, обучаемая нейронная сеть начинает предсказывать то, что нам нужно. Особенность заключается в том, что в ней нейроны первого уровня упорядочены в особую структуру: нейроны на первых слоях разбиты на изображения определенного размера, которые иногда называют картами. Разные карты внутри одного слоя соответствуют нейронам одного типа, которые реагируют на разные особенности изображений. Активация следующего слоя бывает двух видов. В первом комбинация нейронов следующего уровня вычисляется как линейная комбинация нейронов нижнего уровня, причем веса зависят от взаимного расположения и типа нейронов, но не зависят от расположения данного нейрона внутри карты. Во втором активация нейроном на следующем уровне просто повторяет активацию нейронов на предыдущем уровне, но изображение становится меньшего размера за счет того, что активация рядом расположенных нейронов заменяется на их максимум или их среднее. Это так называемая операция «пулинга»:



Подобная структура делает сверточные нейронные сети очень подходящими для работы с изображениями, она, например, гарантирует, что если два изображения отличаются маленьким сдвигом, то нейронная сеть на выходе получит очень похожий результат. Кроме того, в сверточных нейронных сетях количество параметров невелико относительно количества нейронов, а обычные нейронные сети для такого же количества нейронов имели бы сотни миллиардов параметров и набрать обучающих множеств для такого количества параметров мы бы никогда не смогли, а сверточные нейронные сети с таким количеством нейронов могут обучаться по существующим выборкам и именно это делает их такими успешными.

С момента своего появления (конец 70-х начало 80 годов) сверточная нейронная сеть работала с маленькими изображениями 28 на 28 пикселей лучше существующих методов. Для таких изображений сеть могла различить, какой рукописный символ изображение содержит. Но для задач компьютерного зрения нужно уметь работать с гораздо большими изображениями. Поэтому сверточные нейронные сети уступали аналогичными не нейросетевым алгоритмам.

Все изменилось в начале 2010-х годов, когда появились очень большие обучающие выборки и подходящее оборудование, в первую очередь графические сопроцессоры, которые изначально создавались не для искусственного интеллекта, а чтобы геймеры могли играть в максимально красивые компьютерные игры. Выяснилось, что задачу классификации сверточная нейронная сеть решает гораздо лучше, чем все остальные методы. Это касается не только задачи классификации, но и распознавания лиц на фото или видео, положения или позы человека на фотографии.

Достигается это за счет того, что сети становятся все глубже и глубже, содержат все больше и больше слоев. Более того, сверточные нейронные сети замечательно показали себя в обратной задачи, когда нужно не обработать, а синтезировать изображение. Также сверточные нейронные сети могут подсчитывать компактные дескрипторы изображений, то есть вектора малой размерности, которые описывают, что именно изображение содержит. Сравнивая такие вектора, можно находить изображения, которые содержат похожие предметы или такой же предмет. Такое свойство очень полезно в поисках по очень большим архивам изображений.

Математическое осмысление сверточных нейронных сетей находится в достаточно зачаточном состоянии, и здесь в ближайшие годы ожидается большой прогресс. Если ставить вопрос о возможности создания искусственного интеллекта подобного человеческому мозгу с использованием нейронных сетей, необходимо понимать, что теоретически это возможно, но потребуются колоссальные вычислительные ресурсы, к тому же алгоритмов для решения таких параллельных задач еще не создано.

Подытожим привлекательные черты распределенной обработки информации в нейросетевых системах:

• Параллелизм обработки информации - глобальность связей между нейронами. До обучения эти связи произвольны и обычно малы. Обучение на примерах "проявляет" конкретную структуру сети под конкретную задачу;

• Единый и эффективный принцип обучения нейросетей - минимизация эмпирической ошибки методом ее обратного распространения по сети. Извне задается лишь цель обучения - то есть способ определения ошибки по выходам сети. Далее сеть постепенно модифицирует свою конфигурацию, минимизируя эту ошибку, то есть все лучше справляясь с возложенной на нее задачей;

• Надежность функционирования. Избыточность связей приводит к тому, что значения каждого веса по отдельности не играют решающей роли. Вывод из строя ограниченного числа нейронов или обрыв некоторых связей не сказываются критическим образом на качестве работы всей сети;

• Способность решать неформализованные задачи - следует из способности нейросетей самостоятельно вырабатывать весьма сложные алгоритмы обработки данных, формализовать которые самостоятельно зачастую не могут даже лучшие эксперты в данной предметной области. Отсюда - относительная дешевизна нейросетевых разработок.

Почему же нейрокомпьютинг при таком наборе преимуществ стал активно применяться на практике лишь с 90-х годов? Очевидно, что появление нейросетевых систем должно быть востребовано экономикой. Именно экономическая потребность в суперкалькуляторах вызвала к жизни последовательные ЭВМ, способные решать любые формализованные задачи обработки как численной, так и символьной информации. Это, в свою очередь, подняло на должную высоту проблему человеко-машинного интерфейса. Отсюда и возросший интерес к проблемам искусственного интеллекта и нейросетей. Времена меняются. Вместо разрозненных персональных компьютеров появилась Сеть с ее неисчерпаемыми информационными ресурсами.

С другой стороны, грубые ошибки, легкость обмана, непрозрачность, катастрофическая забывчивость нейросетей. Современные нейросети устроены в полторы тысячи раз проще, чем, например, головной мозг крысы.

Искусственная нейронная сеть – это система, которая распознает стимулы и по результатам делает то, что мы от нее ожидаем. Это интеллект раба, которого учит хозяин! Интеллект – это, прежде всего знания.

Литература

Основная литература

1. Карпович Е.Е. Языки программирования интеллектуальных систем [Электронный ресурс]: учебник/ Карпович Е.Е. - Электрон. текстовые данные. – М.: Издательский Дом МИСиС, 2018. - 172 с. - Режим доступа: http://www.iprbookshop.ru/84436.html. - ЭБС «IPRbooks».
2. Ясницкий Л.Н. Интеллектуальные системы [Электронный ресурс]: учебник/ Ясницкий Л.Н. - Электрон. текстовые данные. – М.: Лаборатория знаний, 2016. - 222 с. - Режим доступа: http://www.iprbookshop.ru/89033.html. - ЭБС «IPRbooks».
3. Интеллектуальные системы [Электронный ресурс]: учебное пособие для СПО/ А.М. Семенов [и др.]. - Электрон. текстовые данные. - Саратов: Профобразование, 2020. - 236 с. - Режим доступа: http://www.iprbookshop.ru/91871.html. - ЭБС «IPRbooks».
4. Нестеров С.А. Интеллектуальный анализ данных средствами MS SQL Server 2008 [Электронный ресурс] / Нестеров С.А. - Электрон. текстовые данные. – М.: ИНТУИТ, 2016. - 303 с. - Режим доступа: http://www.iprbookshop.ru/62813.html. - ЭБС «IPRbooks».

Дополнительная литература

5. Брусенцев А.Г. Анализ данных и процессов. Ч.1. Методы статистического анализа данных [Электронный ресурс]: учебное пособие/ Брусенцев А.Г. - Электрон. текстовые данные. - Белгород: БелГТУ им. В.Г. Шухова, ЭБС АСВ, 2017. - 63 с. - Режим доступа: http://www.iprbookshop.ru/92237.html. - ЭБС «IPRbooks».
6. Барский А.Б. Искусственный интеллект и логические нейронные сети [Электронный ресурс]: учебное пособие/ Барский А.Б. - Электрон. текстовые данные. - Санкт-Петербург: Интермедия, 2019. - 360 с. - Режим доступа: http://www.iprbookshop.ru/95270.html. - ЭБС «IPRbooks».
7. Курейчик, В.В., Курейчик В.М., Родзин С.И. Теория эволюционных вычислений [Текст]: [монография] - М.: Физматлит, 2012. - 260 с. – Режим доступа: https://www.rfbr.ru/rffi/ru/books/o_1780980#1
8. Родзин С.И. Искусственный интеллект [Электронный ресурс]: учебное пособие/ С.И. Родзин. - Таганрог: Технологический институт ФГОУ ВО "Южный федеральный университет" в г. Таганроге, 2015. - 148 с. - Режим доступа: https://www.elibrary.ru/item.asp?id=28809262
9. Яхъяева Г.Э. Нечеткие множества и нейронные сети [Электронный ресурс]: учебное пособие/ Яхъяева Г.Э. - Электрон. текстовые данные. – М.: Саратов: ИНТУИТ, Вузовское образование,

2017. - 320 с. - Режим доступа: http://www.iprbookshop.ru/67390.html. - ЭБС «IPRbooks».

Ресурсы Интернет

10. название ресурса №1: Войтович И.Д. Интеллектуальные сенсоры [Электронный ресурс]: учебное пособие/ Войтович И.Д., Корсунский В.М. - Электрон. текстовые данные. – М.: Саратов: ИНТУИТ, Ай Пи Ар Медиа, 2020. - 1163 с. - Режим доступа: http://www.iprbookshop.ru/89436.html. - ЭБС «IPRbooks».

11. Седова Н.А. Теория нечетких множеств [Электронный ресурс]: учебное пособие/ Седова Н.А., Седов В.А. - Электрон. текстовые данные. - Саратов: Ай Пи Ар Медиа, 2019. - 421 с. - Режим доступа: http://www.iprbookshop.ru/86526.html. - ЭБС «IPRbooks.

12. название ресурса №3: Прокопенко Н.Ю. Системы поддержки принятия решений [Электронный ресурс]: учебное пособие/ Прокопенко Н.Ю. - Электрон. текстовые данные. - Нижний Новгород: НГАСУ, ЭБС АСВ, 2017. - 189 с. - Режим доступа: http://www.iprbookshop.ru/80838.html. - ЭБС «IPRbooks».

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

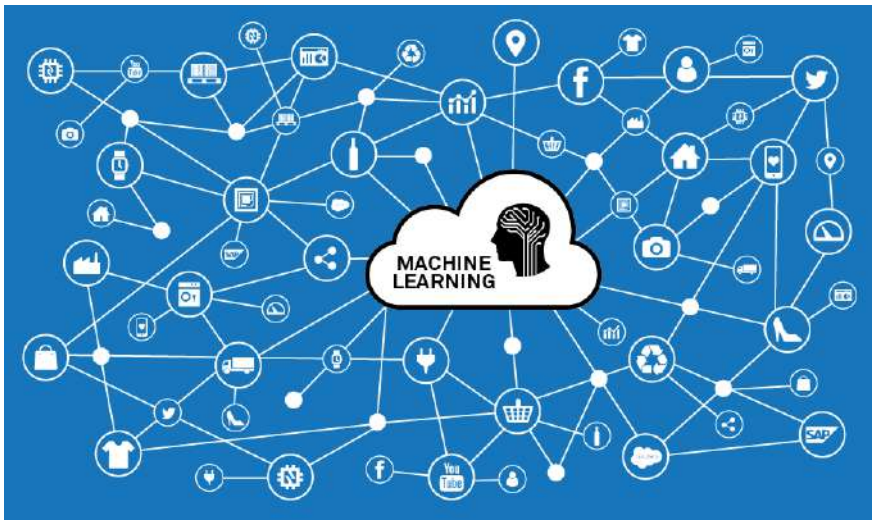
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

В.В. КУРЕЙЧИК
С.И. РОДЗИН
В.М. КУРЕЙЧИК

МАШИННОЕ ОБУЧЕНИЕ И БИОИНСПИРИРОВАННАЯ ОПТИМИЗАЦИЯ

Учебно-методическое пособие к лабораторным работам



ТАГАНРОГ 2021

УДК 007.52: 611.81 (075) + 519.7 (075)

Курейчик В.В., Родзин С.И., Курейчик В.М. МАШИННОЕ ОБУЧЕНИЕ И БИОИНСПИРИРОВАННАЯ ОПТИМИЗАЦИЯ: учебно-методическое пособие к лабораторным работам. – Таганрог ИКТИБ ЮФУ, 2021. – 24 с.

Учебно-методическое пособие к лабораторным работам по машинному обучению и биоинспирированной оптимизации включает следующие темы: проектирование баз знаний экспертных систем, проектирование и машинное обучение нейросети в среде MatLab, а также решение оптимизационных задач с помощью эволюционных и роевых алгоритмов.

Пособие основано на образовательном контенте, используемом авторами в образовательных программах Института компьютерных технологий и информационной безопасности Южного федерального университета.

Для студентов магистерской образовательной программы «Прикладные системы искусственного интеллекта». Практикум может быть полезен аспирантам, занимающимся проблематикой искусственного интеллекта, теоретической и прикладной информатики, компьютерным моделированием и автоматизацией проектирования.

Оглавление

1-2. Проектирование баз знаний экспертных систем 4

3-4. Проектирование и машинное обучение нейросети в среде MatLab 14

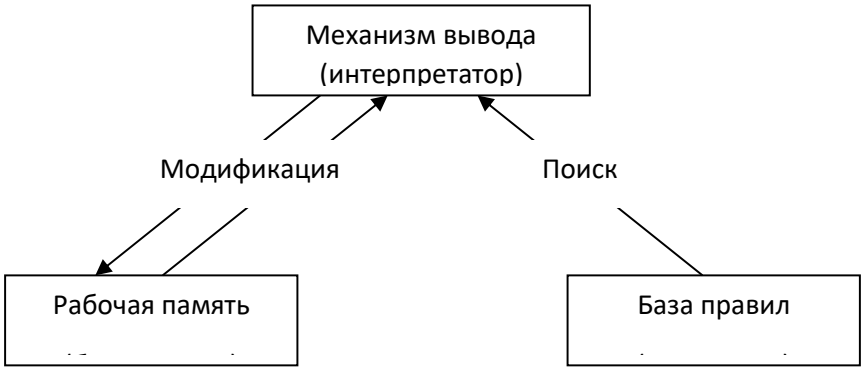
Литература 23

1-2. Проектирование баз знаний экспертных систем

Целью работы является изучение процесса проектирования баз знаний, используя среду программирования экспертных систем CLIPS, анализ продукционной модели представления знаний, механизмов прямого и обратного логического вывода, фреймовой модели представления знаний, а также практическая реализация моделей знаний и механизмов вывода знаний.

Продукционная модель представления знаний.

Продукционная система из трех основных компонентов схематично представлена на рис:



Рассмотрим компоненты продукционной системы. Чтобы правило было выполнимым, необходимо (но недостаточно), чтобы выполнялись все его условия. Условие выполняется, если соответствующий ему факт присутствует в списке фактов рабочей памяти. После определения, какие из правил являются выполнимыми, они помещаются в список активированных правил. Наконец, если какое-то одно правило из списка будет выполнено, действия задают изменения, которые должны быть внесены в состояние рабочей памяти. Функция рабочей памяти – хранить данные (факты) в формате векторов «объект – атрибут – значение атрибута». Эти данные используются интерпретатором, который активизирует те правила, условия которых на имеющихся данных выполняются. Наконец, процесс работы интерпретатора правил описывается в терминах сопоставления по образцу (цикл «распознавание – действие»):

1. Сопоставить условия правил и элементы рабочей памяти;
2. Если окажется, что можно активизировать более одного правила, то выбрать одно из них (*разрешить конфликт*);
3. Применить выбранное правило. Результатом, скорее всего, будет добавление нового элемента данных в рабочую память и/или удаление каких-либо данных из рабочей памяти. Затем перейти к п.1.

Рассмотрим особенности продукционного программирования на CLIPS. Язык CLIPS (C Language Integrated Production System), являясь LIPS-подобным языком программирования, ориентированным на разработку экспертных систем, также поддерживает объектно-ориентированную и процедурную парадигмы программирования. Подробно с описанием языка можно ознакомиться в [2, 10], а также в электронном приложении к данному лабораторному практикуму.

Версии языка могут эксплуатироваться на платформах UNIX, Windows, Macintosh, являются хорошо документированным общедоступным программным продуктом, доступным по сети FTP с множества университетских сайтов. Исходный код программного пакета CLIPS распространяется свободно и его можно установить на любой платформе, поддерживающей стандартный компилятор языка С. Поэтому здесь приведем лишь сведения, необходимые для общего понимания конструкций языка, используемых в данном кратком теоретическом описании лабораторной работы.

Для программирования язык использует простые типы данных (float, integer, symbol, string и др.), функции для манипулирования данными с префиксной формой их вызова (пользовательские фрагменты кода на языке С, стандартные арифметические и математические функции) и конструкции для пополнения базы знаний (defmodule, defrule, deffacts, deftemplate, deffunction, defclass и др.). Комментарии могут вставляться в код CLIPS при помощи точки с запятой «;». Факт является одной из основных форм представления информации, используемой правилами CLIPS. Каждый факт - это фрагмент информации, который помещается в текущий список фактов, называемый fact-list. Количество фактов и объем информации, который может быть сохранен в факте, ограничивается только размером памяти компьютера. Факт может описываться индексом или адресом и иметь позиционный (скобочная запись из выражений символьного типа) и непозиционный (шаблон, который может использоваться при доступе к полям по именам) форматы представления. Шаблоны в CLIPS напоминают простые списки, но не имеют ничего общего с шаблонами в языке C++.

Интерфейс CLIPS позволяет работать интерактивно с использованием GUI или простого текстового интерфейса командной строки, а также как система, интегрированная в другие приложения. После запуска файла clipswin.exe можно ввести программу непосредственно из диалогового окна, но в этом случае все подготовленные правила после закрытия CLIPS будут потеряны. Чтобы этого не происходило, необходимо сохранить текст программы в каком-либо файле. Для его модификации имеется встроенный редактор.

Продукционные правила являются одной из основных моделей представления знаний в CLIPS. Левая часть правила представляет собой ряд условий (условных элементов), которые должны выполняться, чтобы правило было применимо. В CLIPS принято считать, что условие выполняется, если соответствующий ему факт присутствует в списке фактов. Процесс сопоставления фактов образцам выполняется блоком вывода CLIPS, который автоматически сопоставляет образцы, исходя из текущего состояния списка фактов, и определяет, какие из правил являются применимыми. Если все условия правила выполняются, то оно активируется и помещается в список активированных правил.

В языке CLIPS условия представляются в форме вектора *объект-атрибут-значение.* Например: (organism-1 (morphology rod) (aerobicity aerobic)). В данном случае условие состоит в том, что определенный микроорганизм имеет форму палочки и размножается в воздушной среде.

Продукционное правило, которое включает такую предпосылку, на языке CLIPS имеет вид:

(defrule diagnosis

(patient (name Jones) (organism organism1))

(organism (name organism1) (morphology rod) (aerobicity aerobic))

=>

(assert (organism (name organism1) (identify enterobacteriaceae) (confidence 0.8))

)

Перечень предпосылок в таком правиле представляет собой образец вектора, которому должно соответствовать состояние рабочей памяти.

Другим компонентом продукционной системы является рабочая память, в которой хранятся исходные данные к задаче и выводы, полученные в ходе работы системы. Эти данные используются интерпретатором, который в случае присутствия (или отсутствия) определенных данных в рабочей памяти может активизировать те правила, предпосылки в которых соответствуют этим данным. Рассмотрим на примере, как это реализуется в CLIPS.

Пусть в рабочей памяти содержатся векторы:

(patient (name Jones) (age 40) (organism organism1))

(organism (name organism1) (morphology rod) (aerobicity aerobic))

На очередном шаге цикла интерпретатор просмотрит имеющийся список правил и отыщет в нем то правило, которое содержит условия, соответствующие этим векторам. Если предпосылка в правиле не содержит переменных, она удовлетворяется при точном совпадении выражений в правиле и в рабочей памяти. Если же предпосылка в правиле содержит переменные, то есть является образцом, то она удовлетворяется, если в рабочей памяти содержится вектор, включающий такую пару *атрибут-значение,* которая остается постоянной при удовлетворении всех остальных условий в том же правиле.

Последний компонент продукционной системы – механизм логического вывода − использует правила в соответствии с содержимым рабочей памяти и действует путем «сопоставления по образцу», управляя перебором правил в прямом (от данных к поиску цели) или обратном (от цели для ее подтверждения – к данным) направлении. В процессе вывода в продукционной системе может возникнуть ситуация, когда на каком-либо шаге удовлетворяются условия применимости нескольких продукций. Тогда для выбора выполняемого правила привлекается информация о приоритетности, достоверности, значимости и прочих свойствах продукций.

В прямом выводе рассуждения строятся, отталкиваясь от условий, которые удовлетворяются, к действиям, вытекающим из этих условий. Обратный вывод означает, что рассуждения строятся, отталкиваясь от заданной цели, к условиям, при которых возможно ее достижение. В CLIPS всегда сопоставляются состояние рабочей памяти и условные части правил, а затем выполняются действия, предусмотренные правой частью правил. Не существует способа, который бы позволял одному правилу вызвать другое (как в процедурном программировании). Одно правило может облегчить выполнение другого

правила, но единственный способ его активизации – изменить состояние рабочей памяти.

Фреймовая модель представления знаний.

При использовании фреймов знания не «размазываются» по программному коду приложения, как в продукционных системах, но и не собираются воедино в виде метазнаний как в семантических сетях или в ассоциативных сетях (сети, используемые для представления семантики естественного языка).

М. Минский определил фрейм как «структуру данных для представления стереотипных ситуаций» и как способ формализации знаний, используемый при решении таких интеллектуальных задач, как распознавание образов и понимание речи. Реализованная им идея заключалась в том, чтобы сконцентрировать все данные (знания) о конкретном объекте или событии в единой структуре, а не распределять ее между множеством более мелких структур. Иными словами, фреймовая модель позволяет судить о классе объектов, используя знание о его прототипах, которые хорошо представляют большинство разновидностей объектов данного класса, но могут корректироваться для того, чтобы представить всю сложность, присущую реальному миру. Такая ситуация типична для подавляющего большинства эвристических механизмов, используемых человеком при решении практических проблем. Поэтому фреймовая модель оказывается полезной, поскольку она позволяет структурировать эвристические знания.

Представление предметной области в виде иерархической системы фреймов хорошо отражает внутреннюю и внешнюю структуру объектов этой предметной области, так как фрейм, являясь формой описания знаний, очерчивает рамки рассматриваемого в задаче фрагмента предметной области. В процессе адаптации фрейма к конкретным условиям уточняются значения слотов. В случае поиска фрейма, релевантного некоторому образу, можно проверить совпадение слотов. Передать данные во фрейм, заполнив его слоты, можно несколькими способами: при конструировании фрейма, через вызов функции, указанной в слоте, через присоединенную к слоту процедуру (*демон*), из диалога с пользователем, через наследование свойств от других фреймов, из базы данных. Фреймовая модель также удобна для явного представления ситуационных знаний о предметной области, так как в конструкции фрейма предусматриваются управляющие слоты с данными для подтверждения или отклонения фрейма, для вывода сообщения-решения и т.п.

Организация вывода во фреймовой системе базируется на обмене сообщениями между фреймами, активации и выполнении присоединенных процедур. Отражение в иерархии фреймов родовидовых отношений обеспечивает возможность реализации операции наследования, позволяющей приписывать фреймам нижних уровней иерархии свойства, присущие фреймам вышележащих уровней. Причем, данные, которые передаются в процессе функционирования системы от посторонних источников знаний во фреймы нижних уровней, имеют более высокий приоритет, чем данные, унаследованные от фреймов верхних уровней. Наследование позволяет описывать свойства и поведение класса в терминах других классов.

Порядок выполнения работы.

Порядок выполнения работы предусматривает изучение продукционной и фреймовой модели представления знаний, механизмов прямого и обратного логического вывода и их практической реализации на языке CLIPS.

Для этого необходимо:

- изучить представленный теоретический материал, рассмотреть примеры;
- четко представить себе, что необходимо сделать согласно своему варианту;
- определить, какие декларации, шаблоны и классы целесообразно использовать;
- описать классы и правила для решения заданной проблемы;
- задать начальные условия или сформулировать исходную проблему;
- практически реализовать на языке CLIPS программу, моделирующую заданную ситуацию;
- представить отчет о работе, включающий авторские выводы и оценку достижения поставленных целей.

P.S. Рекомендации. Количество правил должно быть не менее нескольких десятков. Система не должна быть построена по принципу «таблицы решений», когда по входным данным сразу выбирается ответ; необходимо предусмотреть несколько уровней промежуточных рассуждений (когда система на основе фактов делает промежуточные выводы). Оценка повышается, если в системе предусматривается возможность отследить процесс принятия решения.

Содержание отчета.

1. Цель работы.
2. Постановка задачи согласно варианту.
3. Описание *самостоятельно* проделанной работы: изучение языка CLIPS, анализ продукционной и фреймовой модели представления знаний, механизмов прямого и обратного логического вывода; определение используемых деклараций и шаблонов; описание классов и необходимых правил; задание начальных условий для исходной проблемы.
4. Выводы (они должны иметь характер нетривиальных утверждений, содержать обоснованную *оценку* степени достижения поставленных целей и задач, не подменяться перечислением наименований сделанного, поддерживаться программной реализацией и экспериментальными данными.

Приложение: листинги программ на CLIPS

Варианты заданий.

1) Диагностика сбоя технического устройства

1. Ниже, в качестве примера, приводится модель диагностики неисправностей, взятая из инструкции по эксплуатации автомобиля (BMW 320).

Двигатель не заводится:

Если на стартер не подается ток, то причины могут быть следующие:

- разряжен аккумулятор;
- поврежден провод подключения к аккумуляторной батареи;

- поврежден соленоид стартера;
- плохой контакт с «массой».

Если на стартер подается ток, то причины могут быть следующие:

- заклинило шестерню стартера;
- поврежден двигатель стартера.

Двигатель проворачивается, но не запускается:

Если нет искры между электродами свечи, то причины могут быть следующие:

- загрязнены контакты прерывателя;
- наличие влаги в распределителе;
- неправильно подключены контакты прерывателя;
- поврежден конденсатор;
- поврежден ключ прерывателя;
- повреждена катушка.

Если нет топлива в жиклере карбюратора, то причинами может быть:

- отсутствие топлива в баке;
- паровая пробка в системе подачи топлива (жарким летом);
- засорен жиклер;
- неисправен бензонасос.

Двигатель заглох и вновь не заводится:

Если заливает карбюратор, то причины могут быть следующие:

- заедание игольчатого клапана;
- поврежден поплавок;
- неправильно установлен уровень поплавка.

Если нет топлива в жиклере карбюратора, то причинами может быть:

- отсутствие топлива в баке;
- попадание воды в систему подачи топлива.

Постройте на основе этой или аналогичной инструкции набор порождающих правил и разработайте CLIPS-программу диагностики автомобиля. Ввод причин неисправностей организуйте через ответы пользователя на запрос системы.

Реализовать механизм прямого многоэтапного вывода (от симптомов к искомой неисправности).

Реализовать механизм обратного многоэтапного вывода (от тех или иных причин неисправностей к их возможной симптоматике).

2) *Медицинская экспертная система диагностики заболеваний*

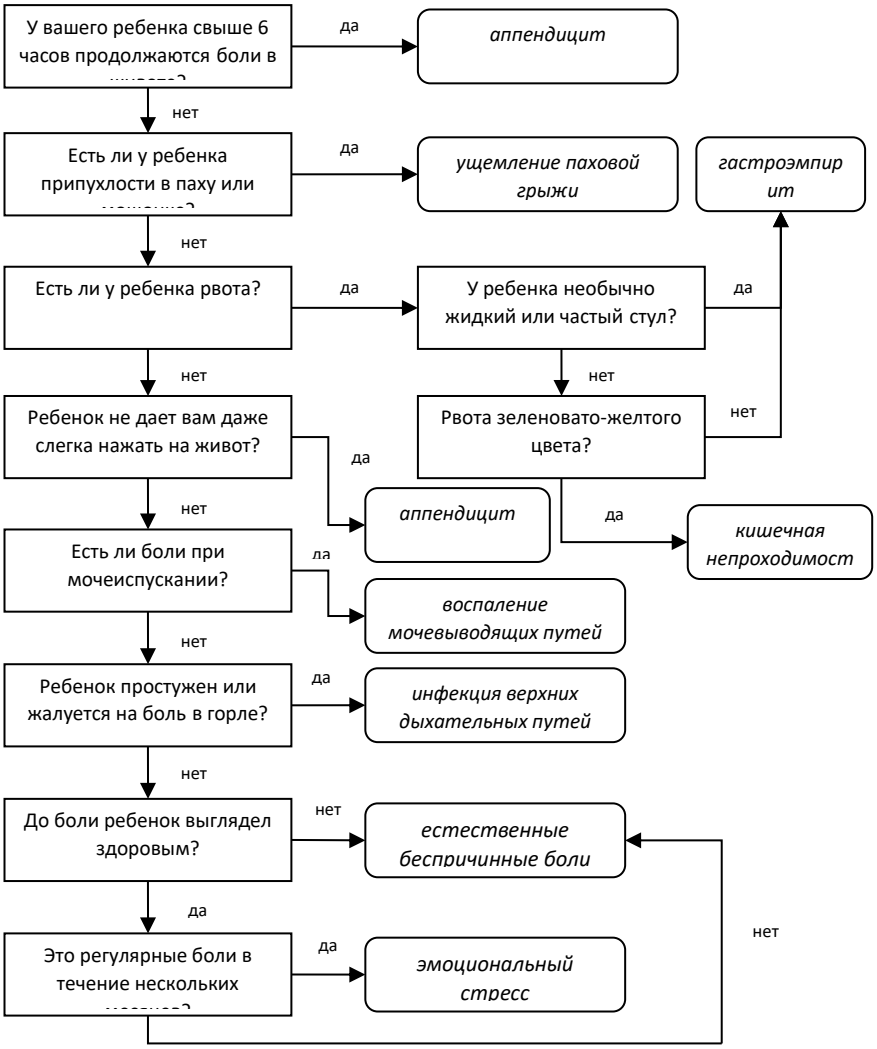
Экспертная система диагностики заболеваний на языке CLIPS должна осуществлять помощь пользователю в определении диагноза болезни на основе некоторых симптомов. Диагностика должна осуществляться путем опроса пользователя на наличие определенных симптомов. Такая база знаний фактически представляет собой древовидную структуру, где в качестве узлов представлены симптомы (вопросы вида да/нет), а в качестве листьев заболевания. Количество диагностируемых заболеваний должно быть не менее 5. Данные по заболеваниям и их симптомам должны быть взяты из какой-либо книги или справочника. Реализовать прямой и обратный вывод, использовать

продукционную и фреймовую модели представления знаний. Постройте набор порождающих правил и разработайте CLIPS-программу для медицинской экспертной системы диагностики заболеваний.

При прямом выводе необходимо последовательно задавать пользователю вопросы для определения наличия определенных симптомов (движение по дереву от корня к листьям). В конце необходимо вывести название заболевания и рекомендации к дальнейшим действиям.

В обратном выводе вначале необходимо вывести список заболеваний на выбор любого из них. После осуществления выбора заболевания необходимо проверить гипотезу о наличии или отсутствии данного заболевания. Проверка должна осуществляться путем задания вопросов на проверку симптомов (движение по дереву вверх от листьев к корню). В качестве результата работы алгоритма вывести «диагноз подтверждается» или «диагноз не подтверждается». Все вопросы задавать не нужно, если на некотором шаге возможно подтверждение или опровержение гипотезы о заболевании, то опрос должен быть завершен досрочно.

Ниже, в качестве примера приводится фрагмент базы заболеваний и примерный диагностический алгоритм постановки диагноза на основе болей живота ребенка (материалы взяты из медицинского справочника Т. Смита «Ваш семейный доктор. Домашний советчик»).



Для чтения данных о заболеваниях и записи симптомов необходимо разработать и использовать фреймовую модель представления знаний. В качестве фреймов и их экземпляров представить базу заболеваний, где для каждого заболевания должны храниться следующие данные: название заболевания, описание действий, которые нужно предпринять пользователю после его диагностики. Кроме этого, необходимо создать фрейм «Пациент», экземпляры которого должны содержать в себе ФИО пациента (вводится в начале работы системы), список симптомов (фиксируются в процессе диагностики) и сам диагноз. После окончания диагностики должна быть возможность вывода на экран всех данных из этого экземпляра (просмотр знаний о пациенте).

3) *Выявление мотивации поведения биологической особи*

Пусть имеется популяция рефлекторных агентов (искусственных, активных и относительно автономных организмов) с естественными потребностями энергии и размножения. Каждая потребность характеризуется определенной мотивацией, например, если энергетический ресурс агента небольшой, то у агента появляется мотивация найти пищу и пополнить энергетический ресурс. Каждый агент эволюционирует в двумерной клеточной среде, в ячейках которой периодически вырастает трава (пища агентов). Каждый агент имеет внутренний энергетический ресурс, который пополняется при съедании им травы, либо уменьшается при выполнении каких-либо действий, например при размножении. Уменьшение ресурса до нуля приводит к смерти агента. Агенты могут скрещиваться, рождая новых агентов. Агент близорук и воспринимает состояние внешней среды только в четырех клетках: слева, справа, впереди и внутри той клетки, где он находится. Агент может выполнять следующие действия: отдыхать; перемещаться на одну клетку вправо, влево или вперед; принимать пищу; скрещиваться.

На языке CLIPS напишите программу, которая будет принимать решения на основе мотивации. Смоделируйте определенную мотивацию на CLIPS в виде правил поведения. Например, если возникает некоторая мотивация, то поведение агента изменяется с целью удовлетворения возникшей потребности.

Реализовать механизмы прямого многоэтапного вывода (от определенных мотиваций к потребностям агента), а также обратного многоэтапного вывода (от потребностей агента к породившим их мотивациям).

Протестировать корректность работы программы на некотором множестве агентов.

Построить систему фреймов для решения головоломки о миссионерах и каннибалах.

На левом берегу реки находятся три миссионера и три каннибала. На этом же берегу причалена единственная лодка. На этой лодке нужно переправить всех миссионеров и всех каннибалов на правый берег при условии, что в лодке одновременно могут поместиться только двое, а в обратный путь к левому берегу должен отправиться хотя бы один человек. Есть одно обстоятельство, существенно влияющее на процесс решения задачи: если окажется, что каннибалов на любом из берегов больше, чем миссионеров, то несчастных

просто съедят. Решение головоломки – это последовательность шагов, переводящая систему из исходного состояния в заданное конечное состояние. Для поиска решения можно воспользоваться эвристическим алгоритмом «первый лучший» с соответствующей оценочной функцией.

4) *Тестирование при приеме на работу*

Экспертная система тестирования при приеме на работу на языке CLIPS должна осуществлять помощь работодателю в выяснении, соответствует ли соискатель требуемой должности. Соискателю должен быть задан ряд вопросов, порядок следования которых определен продукционными правилами. Совокупность ответов на вопросы позволяет заполнить фрейм **Соискатель**, на основании содержания которого выводится решение.

Необходимо составить набор вопросов, последовательно связанных между собой ответами. В итоге, должна получиться древовидная структура.

Примеры вопросов: «Вы опытный программист?», «Сколько лет вы работаете программистом?», «Знаете ли вы объектно-ориентированное программирование?», «Готовы ли вы пройти доп. обучение перед трудоустройством?» и т.п.

Рабочий цикл работы программы для прямого вывода может выглядеть следующим образом:

1. Инициализировать фрейм со значениями по умолчанию.
2. Добавить факт "Тестирование: запуск" в рабочую память. Добавить первый вопрос в рабочую память.
3. Задать вопрос из фактов
4. В зависимости от ответа - добавить следующий вопрос в рабочую память и обновить поле фрейма двигателя, или завершить диагностику, если вопросов не осталось.
5. Для завершения тестирования изменить факт "Тестирование: запуск" на "Тестирование: завершено".
6. На основании сформированного фрейма обозначить состояние соответствия как соответствует/не соответствует и предложить пути решения.

При обратном выводе необходимо проверить гипотезу о соответствии, т.е. предположить, что чего-то не хватает в качествах соискателя и как это повлияет на его трудоустройство. Например, если программист не знает объектно-ориентированное программирование и готов обучаться, то ему можно пройти обучение перед трудоустройством.

5) *Проверка платежеспособности клиента*

Для проверки платежеспособности клиента и одобрения выдачи кредита зачастую применяются экспертные системы, в которые вносятся данные о заемщике. Такие экспертные системы на основе внутренних критериев выдают рекомендацию, стоит ли кредитному агенту выдавать кредит заемщику. Для реализации подобной системы на языке CLIPS необходимо иметь базу вопросов для заполнения информации о клиенте, фрейм **Клиент**, которые будет содержать введенную информацию и набор критериев принятия решения.

Разумно предположить, что некоторые из вопросов являются взаимоисключающими, например: если клиент ответил, что нигде не работает, бессмысленно спрашивать о стаже на текущем месте работы. Таким образом, необходимо решить проблему с корректным набором вопросов.

Также информация о клиенте должна быть согласована, т.е. не должна содержать противоречивую информацию. Возвращаясь к предыдущему примеру, если во фрейм Клиент передается информация о том, что заемщик безработный, то необходимо очистить слот "стаж". Таким образом, необходимо передавать данные во фрейм через сообщения, контролируя согласованность состояния фрейма внутри обработчика сообщения.

Примерный список вопросов для заемщика:

1. Возраст.
2. Трудоустройство.
3. Стаж на текущем рабочем месте.
4. Среднемесячные доход (для безработных - размер пособия).
5. Семейное положение.
6. Среднемесячный доход семьи.

Рабочий цикл работы программы для прямого вывода может быть следующим:

1. Инициализировать фрейм значениями по умолчанию.
2. Добавить **все** вопросы в рабочую память.
3. Текущий вопрос определиться стратегией разрешения конфликтов.
4. В правиле для текущего вопроса получить ответ, в зависимости от ответа удалить из рабочей памяти взаимоисключающие вопросы, послать соответствующие сигналы фрейму.
5. В обработчике сигнала проверить согласованность состояния объекта с новым значением слота.
6. Если вопросов в рабочей памяти не осталось, то проанализировать содержание фрейма и выдать рекомендацию, иначе в п.3.

Для обратного вывода необходимо на основании рекомендации (выдавать/не выдавать) вывести список возможных фреймов, подпадающих под данную рекомендацию.

3-4. Проектирование и машинное обучение нейросети в среде MatLab

Целью работы является проектирование модели искусственной нейронной сети для решения задач распознавания образов и аппроксимации в интерактивной среде MatLab.

Интерактивная система MatLab представляет собой интерпретатор, то освоение инструментария нейронных сетей заключается в основном в изучении их функций и их параметров нейронов. Например, понять возможности нейрона как классификатора простых линейно сепарабельных задач можно путем проведения экспериментов с моделью одного нейрона. Для того чтобы создать в MatLab нейрон, используют функцию newp, имеющую следующий синтаксис:

net = newp (PR, S, TF, LF),

где PR – матрица минимальных и максимальных значений входных элементов; S – количество нейронов (при создании одного нейрона S=1); TF – функция активации (transfer function); LF – имя функции обучения нейрона.

Если параметры функции newp не заданы, их значения можно ввести через соответствующие диалоговые окна. Построенный нейрон характеризуется функциями весов (weight function), входов сети (net input function) и определенной функцией активации. Веса инициализируется нулями. Для обучения используются следующие функции.

Функция learnp настраивает веса нейрона. Синтаксис этой функции имеет следующий вид:

dW, LS] = learnp (W, P, Z, N, A, T, E, gW, gA, D, LP, LS),

где вектор W – вектор весов; P – вектор входов; Z – вектор взвешенных входов; N – вектор сети; A – вектор выхода; T - вектор желаемых выходов; E – вектор ошибок; gW – вектор изменения весов; gA – изменения выходов. Функция DW возвращает значения изменения матрицы весов в процессе обучения; LS – новый уровень обученности нейрона.

Функция learnp может быть использована с параметрами по умолчанию:

dW = learnp ([], P, [], [], [], [], E, [], [], [], [], []),

где использование пустого списка [ ] означает параметр по умолчанию.

Функция adapt адаптирует нейрон к условиям задачи:

[net, Y, E, Pf, Af] = adapt (net, P, T, Pi, Ai),

где net – идентификатор нейронной сети, P – входы сети, T – желаемый выход, P_i - исходные условия задержки входов сети, A_i – исходные условия задержки для слоя. Функция возвращает параметры адаптированной сети net.adaptParam: net – измененная сеть, Y – выход сети, E – ошибки сети, Pf – условия задержки входов, Af – условия задержки слоя. Параметры Pi и Pf являются не обязательными, они необходимы только для сетей, имеющих задержки на входах и между слоями.

Функция train также обучает нейросеть и использует следующий синтаксис:

[net, tr] = train (net, P, T, Pi, Ai), где net - сеть, P – входы сети, T – эталонный выход, P_i – исходные условия задержки входа, A_i – исходные условия задержки слоя. Функция моделирует нейронную сеть:

[Y, Pf, Af] = sim (net, P, Pi, Ai),

где net – идентификатор моделируемой сети; P – входы сети; P_i – исходные условия задержки входов сети; A_i – исходные условия задержки слоя. Данная функция возвращает: Y – выходы сети; Pf – окончательные условия задержки входов; Af – окончательные условия задержки слоя.

Функции активации и их назначение в Neural Networks Toolbox следующие:

hardlim (N)− возвращает l, если N положительное и 0 в противном случае;

tansig (N) − вычисляет гиперболический тангенс от входа;

purelin − вычисляет выход слоя от входов ИНС.

Структура данных сети net – это описание обученной ИНС. Обучение осуществляется в соответствии со следующими параметрами, значения которых либо устанавливаются пользователем, либо по умолчанию:

net.trainParam.epochs 100 − максимальное количество итераций (эпох) обучения;

net.trainParam.goal 0 − целевое значение ошибки;

net.trainParam.max_fail 5 − максимальное значение ошибки;

net.trainParam.mem_reduc 1 − фактор оптимизации процесса обучения (памяти или времени, необходимого для ускорения процесса обучения;

net.trainParam.min_grad 1e-10 − минимальное значение градиента;

net.trainParam.show 25 − количество эпох между показами;

net.trainParam.time inf − максимальное время обучения (с);

TR − структура данных, содержащая значения об обученности нейронной сети в текущую эпоху;

TR.epoch − номер эпохи;

TR.perf − уровень обученности (*training performance*);

TR.vperf − достигнутая степень качества (*validation performance*) после окончания процесса обучения;

TR.tperf − результативность обработки теста (*test performance*);

TR.mu − значение адаптивности.

Структура данных описания адаптированной нейронной сети net.adaptfcn включает в себя поля net,adapt и param:

net – идентификатор адаптированной нейронной сети;

Y – выходы нейронной сети;

E – ошибки нейронной сети;

Pf – окончательные входные значения задержек;

Af – окончательные выходные задержки;

TR – результат обучения (эпохи и квадратичная ошибка).

Создание нейрона, выполняющего функцию логического И. Создадим нейрон с одним двухэлементным входом (интервалы первого и второго элементов [0;1]). Определим два первых параметра функции newp, а в качестве значений третьего и четвертого параметра (типа функции активации и имени процедуры обучения) воспользуемся значениями по умолчанию:

\\ создание нейрона с одним двухэлементным входом (интервал первого элемента [0;1] и интервал второго элемента [0;1]

net = newp ([0 1; 0 1], 1);

Для того чтобы исследовать поведение нейрона, необходимо моделировать его работу с помощью функции sim. Для определения последовательности значений входа создадим последовательность P1:

\\ создание последовательности значений входа

P1 = {[0; 0] [0; 1] [1; 0] [1; 1]};

\\ имитация работы нейрона net на последовательности входов P1 желаемых выходов – T, которая позволяет нам провести адаптацию нейрона (обучить его) через 20 проходов

Y = sim (net, P1);

\\ создание последовательности выходов

T1 = {0, 0, 0, 1};

\\ установка количества проходов (циклов) адаптации

net.adaptParam.passes = 20;

\\ адаптация нейрона на net для обучающей выборки <P1; T>

net = adapt (net, P1, T1);

\\ симуляция работы нейрона net на последовательности входов P1

Y = sim (net, P1);

В результате получим нейрон, выполняющий функцию логического И.

Обучение нейрона классификации векторов на две категории. Начнем с классификации векторов на основе двухвходового нейрона. Будем использовать функцию newp для создания нейрона, sim для имитации его работы, adapt для адаптации (обучения) нейронной сети. Обучим двухвходовой нейрон классифицировать входные векторы на две категории:

\\ определим четырех двухэлементных входа

P = [-0,5 -0,5 +0,4 -0,2; -0,5 +0,5 -0,4 +1,0];

\\ зададим желаемые выходы нейрона как реакции на входы P

T = [1 1 0 0];

\\ изобразим входные векторы (точки) на плоскости (рис. 2.1), где каждый из четырех входных секторов на плоскости P определяется двумя координатами, представленными как двухэлементные столбцы в матрице P

plotpv (P, T);

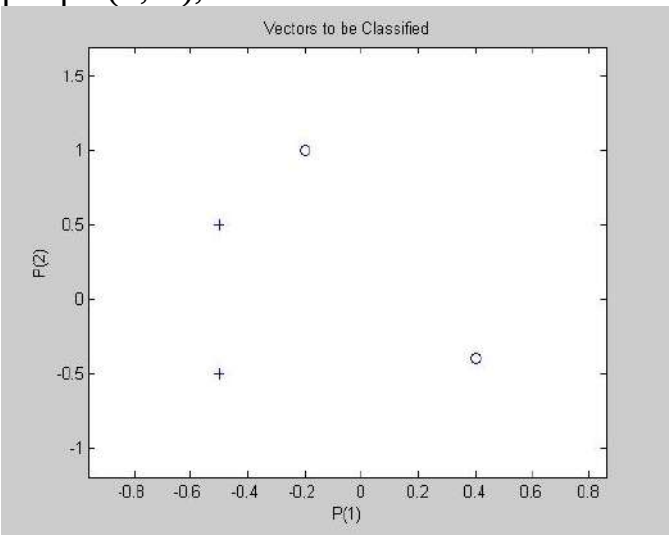


Рис. Исходные векторы, предназначенные для классификации нейроном

\\ создадим один нейрон с двумя входами, значения которых лежат в интервале [-1, 1] (нейрон по умолчанию имеет функцию активации hardlim и такой нейрон разделяет входные векторы прямой линией

net = newp ([-1 1; -1 1], 1);

\\ определим координаты линии классификации: веса (IW) и порог срабатывания нейрона (b), т.е. получим управляющую структуру linehandle для изображения разделяющей линии в координатах весов (IW) и порога срабатывания нейрона (b)

linehandle = plotpc (net.IW {1}, net.b {1});

plotpc (net.IW {1}, net.b {1});

Далее, если исходным весам задать нулевые значения, то любые входы дадут одинаковые выходы, и линия классификации не будет видна на плоскости. Проведем обучение нейрона:

\\ очистка координатных осей

cla;

\\ изображение входных векторов двух категорий (категория задается элементами вектора T)

plotpv (P, T);

\\ получение управляющей структуры linehandle

linehandle = plotpc (net.IW {1}, net.b {1});

\\ присвоение начального значения ошибки

E = 1;

\\ инициализация нейрона

net = init (net);

\\ получение управляющей структуры linehandle

linehandle = plotpc (net.IW {1}, net.b {1});

\\ повторение цикла до тех пор, пока ошибка не станет равной 0

while (mse (E))’,

\\ адаптация нейрона net на обучающей выборке <P, T>, функция возвращает адаптированный нейрон net, выход Y и ошибку E

[net, Y, E] = adapt (net, P, T);

\\ изображение разделяющей прямой нейрона после адаптации

linehandle = plotpc (net.IW {1}, net.b {1}, linehandle);

\\ очистка окна графиков

drawnow;

\\ конец цикла while

end;

Функция adapt возвращает новый объект – сеть, которая выполняет классификацию, выход сети и ошибку.

Проведем классификацию нового вектора с помощью обученного нейрона на основе функции sim:

\\ определение нового вектора P

P = [0,6; 1,1];

\\ имитация работы нейрона net, получение отклика нейрона a

a = sim (net, p);

\\ изображение входа p, отнесенного нейроном к категории a

plotpv (p, a).

Наконец, полученный в ходе обучения нейрон можно использовать для классификации любого вектора:

\\ включить режим добавления графиков в графическом окне

hold on;

\\ получить изображение входных точек в соответствии с категориями T

plotpv (P, T);

\\ получить изображение разделяющей поверхности

plotpc (net.IW {1}, net.b {1});

\\ отключить режим добавления графиков

hold off;

В результате нейрон классифицирует новую точку, как принадлежащую категории «0» (она на рис. 2.2 представлена кружком), а не категории «1» (представлена символом «+»).

Процесс обучения нейрона. Для выполнения примера будем использовать функции: newlin – для создания нейрона, train – для обучения, sim – для имитации поведения нейрона:

\\ определим два входа нейрона вектором P

P = [1.0 -1.4];

\\ определим эталонные (желаемые) выходы нейрона

T = [0.5 1.0];

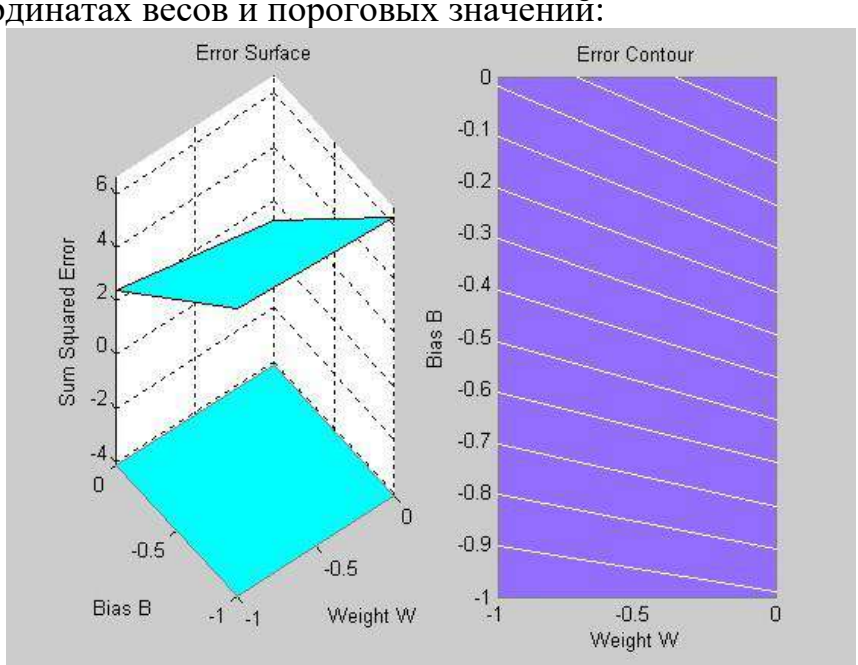
Для решения задачи используем линейный нейрон с одним входом.

\\ установим интервала изменения весов w и порога b

w = -1 : 0,1 : 1;

b = -1 : 0,1 : 1;

\\ вычислим и изобразим поверхность функции ошибки нейрона в координатах весов и пороговых значений:



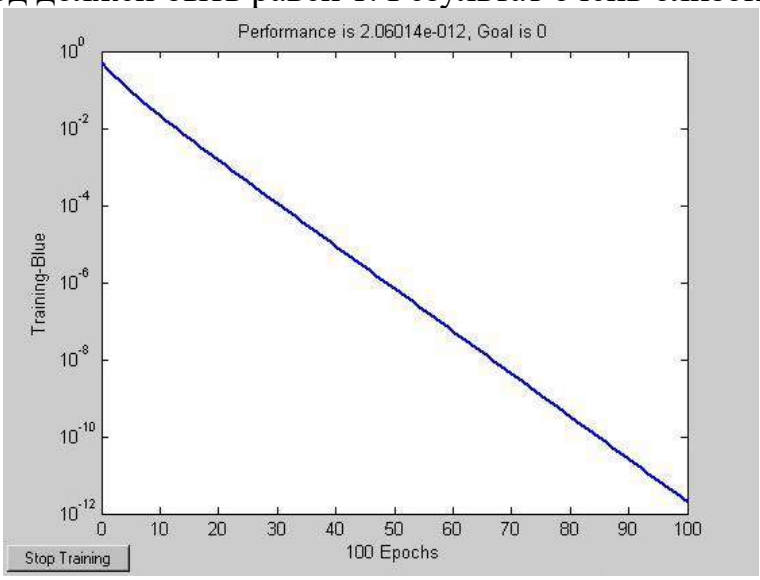
Причем наилучший вес и значение порога являются самыми нижними точками на поверхности, в которых величина ошибки (расхождение эталонного и реального выходных значений) минимальна

```
ES = errsurf (P, T, w, b, ‘purelin’);
plotes (w, b, ES);
\\ переопределим уровень обученности для слоя с bias с вектором P
maxlr = 0.20 * maxlinlr (P, ‘bias’);
net = newlin ([-2 2], 1, [0], maxlr);
```

Здесь функция maxlinlr находит значение, соответствующее самому высокому уровню обученности нейросети. Возьмем 20% от вычисленного уровня. Функция newlin содержит четыре параметра: матрицу интервалов входов размерностью 2×2, количество элементов в выходном векторе, вектор задержек входов, а также уровень обучения.

```
\\ установим целевое значение ошибки, сформировав поверхность ошибки путем создания двух активных координатных осей в графическом окне
ES = errsurf (P, T, w, b, ‘purelin’);
plotes (w, b, ES);
subplot (1, 2, 2);
\\ произвольно изменим веса нейрона в установленных интервалах, установим эпоху обучения нейрона и сформируем историю обучения tr
net.IW {1, 1} = 0;
net.b {1} = 0,1;
[net, tr] = train (net, P, T);
```

В частности, функция train выводит на экран историю обучения сети (tr). Ее график (рис. 2.4) показывает уменьшение ошибки в ходе обучения, что свидетельствует о приближении получаемых значений к эталонным. Используем функцию sim, чтобы протестировать нейрон на входных значениях -1 и 1, где выход должен быть равен 1. Результат очень близок к 1 и равен 0,9167:




```
plotperf (tr, net.trainParam.goal);
p = -1,1;
a = sim (net, p)
```

Порядок выполнения работы.

Сущность задания к лабораторной работе состоит в том, чтобы с помощью инструментария нейронных сетей в среде MatLab реализовать простейшую нейронную сеть. Для этого необходимо построить нейронную сеть, реализующую поставленные в варианте задачи, произвести ее настройку, обучение, проверку работоспособности на тестовых примерах, а также подготовить и защитить отчет о самостоятельно проделанной работе.

Этапы выполнения лабораторной работы:

- изучить представленный теоретический материал, рассмотреть примеры;
- четко представить себе, что необходимо сделать согласно выбранному варианту задания;
- реализовать с помощью соответствующего инструментария в среде MatLab нейронную сеть, реализующую поставленные задачи;
- обучить построенную нейронную сеть на наборе тестовых эталонных значений;
- построить график изменения ошибки в процессе обучения нейронной сети, сделать выводы;
- проверить работоспособность построенной нейронной сети на различных примерах и сравнить полученные результаты с ожидаемыми;
- представить отчет о работе, включающий авторские выводы и оценку достижения поставленных целей.

Содержание отчета.

1. Цель работы.
2. Постановка задачи согласно выбранному варианту.
3. Описание самостоятельно проделанной работы (анализ этапов проектирования нейросети; задание обучающей выборки; график изменения ошибки в процессе обучения нейросети; результаты работы сети на примерах, их сравнительная характеристика).
4. Выводы.

Приложение: необходимые для защиты лабораторной работы “скриншоты” экрана.

Варианты заданий.

Представленные ниже варианты включают различные задачи распознавания, а также аппроксимации. Каждый вариант предусматривает самостоятельную разработку модели нейросети, ее обучение и практическую реализацию в среде MatLab. Вариант задания выбирается студентом по согласованию с преподавателем.

Конкретные варианты заданий представлены ниже.

1. Распознавание чисел

Необходимо реализовать модель искусственной нейронной сети, распознающей десятичные числа в битовом представлении. Закрашенные ячейки битового поля соответствуют 1 на входе соответствующего нейрона, пустые ячейки – 0. Размер битового поля для представления десятичных чисел

выбирается самостоятельно, при этом необходимо, чтобы представления различных десятичных чисел не были идентичны.

Пример битового представления числа «5» на битовом поле размером 4×8 представлен на рис. 2.5.

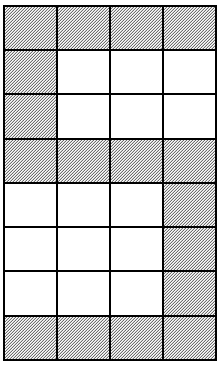


Рис. 2.5. Пример битового представления числа «5»

2. Аппроксимация функции

Необходимо реализовать модель искусственной нейронной сети, аппроксимирующую функцию $y = 2x^2$ на интервале $x \in [-2; 2]$. При обучении нейронной сети в качестве обучающей выборки необходимо самостоятельно выбрать некоторое количество точек из интервала *[-2; 2]*, для которых необходимо рассчитать ожидаемый в соответствии с заданной функцией выход. Остальные значения функции на данном интервале необходимо получить при помощи обученной нейронной сети, шаг изменения аргумента *x* взять равным *0,01*. В отчете необходимо представить графическое представление полученной аппроксимированной функции и ее эталонного представления, построенного средствами MatLab.

3. Аппроксимация функции

Необходимо реализовать модель искусственной нейронной сети, аппроксимирующую функцию $y = 3x^3-1$ на интервале $x \in [-1; 1]$. При обучении нейронной сети в качестве обучающей выборки необходимо самостоятельно выбрать некоторое количество точек из этого интервала, для которых необходимо рассчитать ожидаемый в соответствии с заданной функцией выход. Остальные значения функции на данном интервале необходимо получить при помощи обученной нейронной сети, шаг изменения аргумента *x* взять равным *0,005*. В отчете необходимо представить графическое представление полученной аппроксимированной функции и ее эталонного представления, построенного средствами MatLab.

4. Классификация состояния больных с ишемией

Необходимо реализовать модель искусственной нейронной сети, выполняющей задачу классификации состояния больных с ишемической болезнью сердца. Ишемическая болезнь сердца - сердечно-сосудистое заболевание, характеризующееся нарушениями функций сердца в связи с недостаточностью кровоснабжения. По набору номинальных (например, уровень боли при стенокардии) и непрерывных показателей (например, артериальное давление и возраст), классифицируется состояние пациентов с ишемической болезнью сердца. Показатели, по которым необходимо провести

оценку состояния пациента: содержание холестерина в крови, вес, артериальное давление, подвижность, курение, эмоциональный стресс, возраст, пол, наличие стенокардии, аритмии, сердечной недостаточности (список может быть дополнен). Значения этих показателей выбираются самостоятельно с учетом специфики номинальных и непрерывных показателей. Необходимо представить несколько различных диагнозов. Обучение нейронной сети желательно провести для трех случаях: человек абсолютно здоров (все показатели с положительной оценкой), человек болен ишемической болезнью (все показатели с отрицательной оценкой), «средний» случай. При работе нейронной сети желательно в качестве примеров сымитировать больничные карты нескольких пациентов с широким разбросом показателей.

5. Распознавание букв номерных знаков

Необходимо реализовать модель искусственной нейронной сети, распознающей буквы на автомобильных номерных знаках Российской Федерации в битовом представлении (на этих знаках используются буквы русского алфавита, совпадающие по написанию с соответствующими буквами английского алфавита). Закрашенные ячейки битового поля соответствуют 1 на входе соответствующего нейрона, пустые ячейки – 0. Размер битового поля для представления букв выбирается самостоятельно, при этом необходимо, чтобы представления различных букв не были идентичны.

Пример битового представления буквы «А» на битовом поле размером 4×8 представлен на рис. 2.6.

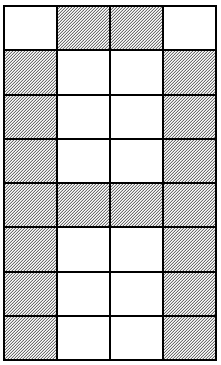


Рис. 2.6. Пример битового представления буквы «А»

Литература

1. Барский А.Б. Искусственный интеллект и логические нейронные сети: уч. пособие.- СПб: Интермедия, 2019.- 360 с.- Режим доступа: http://www.iprbookshop.ru/95270.html
2. Карпович Е.Е. Языки программирования интеллектуальных систем: учебник.- М.: Изд. Дом МИСиС, 2018.- 172 с.- Режим доступа: http://www.iprbookshop.ru/84436.html .
3. Коэльо Л., Ричарт В. Построение систем машинного обучения на языке Python. – М.: ДМК Пресс, 2016.
4. Родзин С.И. Искусственный интеллект: уч. пос.-Таганрог: Изд-во ЮФУ, 2019.-148 с. https://teams.microsoft.com/_#/school/files/%D0%9E%D0%B1%D1%89%D0%B8%D0%B9?threadId=19:4b365d7ac58644b0a7e17963eacd8ec5@thread.tacv2&ctx=channel
5. Родзин С.И. Системы и технологии искусственного интеллекта: уч.пособие.-Таганрог: Изд-во ЮФУ. - 177 с. https://teams.microsoft.com/_#/school/files/%D0%9E%D0%B1%D1%89%D0%B8%D0%B9?threadId=19:4b365d7ac58644b0a7e17963eacd8ec5@thread.tacv2&ctx=channel
6. Родзин С.И. Практикум по СИИ: уч. пособие.-Таганрог: Изд-во ЮФУ. - 95 с. https://teams.microsoft.com/_#/school/files/%D0%9E%D0%B1%D1%89%D0%B8%D0%B9?threadId=19:4b365d7ac58644b0a7e17963eacd8ec5@thread.tacv2&ctx=channel
7. Сегаран Т. Программируем коллективный разум. – СПб: Символ-Плюс, 2008.
8. Ясницкий Л.Н. Интеллектуальные системы: учебник.-М.: Лаборатория знаний, 2016.- 222 с.- Режим доступа: http://www.iprbookshop.ru/89033.html .
9. Яхъяева Г.Э. Нечеткие множества и нейронные сети: уч. пособие. – М.: ИНТУИТ, 2017.- 320 с.- Режим доступа: http://www.iprbookshop.ru/67390.html .
10. Конюшенко В.В. Начало работы с MATLAB / Пер. с англ. В.В Конюшенко. – konushenko@afrodita.phys.msu.su.
11. Штовба С.Д. Введение в теорию нечетких множеств и нечеткую логику. – http://matlab.exponenta.ru/fuzzylogic/book1/ index.php.
12. Гонсалес, Вудс Р., Эддинс С., Цифровая обработка изображений в среде MATLAB – М.: Техносфера, 2006. – 616 с.
13. Журавель, И.М. "Краткий курс теории обработки изображений". http://matlab.exponenta.ru/imageprocess/ book2/index.php
14. Стокман, Д., Шапиро, Л. Компьютерное зрение = Computer Vision. — М.: Бином. Лаборатория знаний, 2006. — 752 с.

15. Форсайт, Д.А., Понс, Д. Компьютерное зрение. Современный подход = Computer Vision: A Modern Approach. — М.: «Вильямс», 2004. — 928 с.
16. Image Processing Toolbox – Обработка сигналов и изображений в среде MathLab http://matlab.exponenta.ru/imageprocess/index.php

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

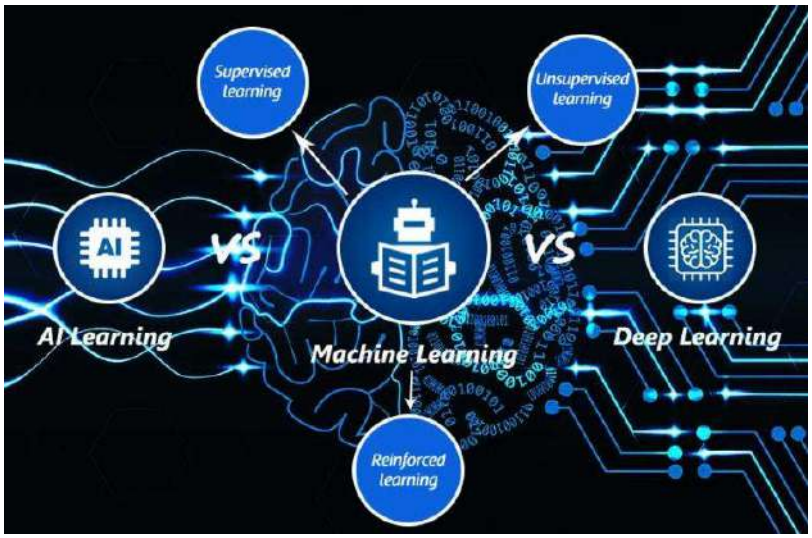
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

В.В. КУРЕЙЧИК
С.И. РОДЗИН
В.М. КУРЕЙЧИК

МАШИННОЕ ОБУЧЕНИЕ И БИОИНСПИРИРОВАННАЯ ОПТИМИЗАЦИЯ

Методические указания к практическим занятиям



ТАГАНРОГ 2021

УДК 007.52: 611.81 (075) + 519.7 (075)

Курейчик В.В., Родзин С.И., Курейчик В.М. МАШИННОЕ ОБУЧЕНИЕ И БИОИНСПИРИРОВАННАЯ ОПТИМИЗАЦИЯ: Методические указания к практическим занятиям. – Таганрог ИКТИБ ЮФУ, 2021. – 73 с.

Практикум по машинному обучению и биоинспирированной оптимизации включает следующие темы: обучение нейронных сетей, представление знаний с использованием продукционных систем, семантических сетей и фреймов, методы достоверного и правдоподобного вывода на знаниях, методы распознавания образов, методы машинного обучения и решение задач интеллектуального анализа данных, а также изучение эволюционных и роевых алгоритмов оптимизации.

Практикум основан на образовательном контенте, используемом авторами в образовательных программах Института компьютерных технологий и информационной безопасности Южного федерального университета.

Для студентов магистерской образовательной программы «Прикладные системы искусственного интеллекта». Практикум может быть полезен аспирантам, занимающимся проблематикой искусственного интеллекта, теоретической и прикладной информатики, компьютерным моделированием и автоматизацией проектирования.

Оглавление

1. Обучение нейронных сетей 4

2-3. Продукционные системы, семантические сети и фреймы для представления знаний в интеллектуальных системах обучения 10

4. Метод резолютивного вывода в интеллектуальных системах 24

5-6. Методы правдоподобного вывода в интеллектуальных системах 30

7-8. Методы распознавания образов 46

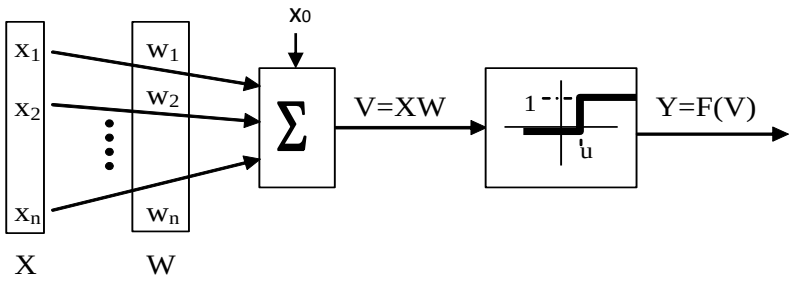
9. Методы интеллектуального анализа данных и прогнозирования 53

10. Решение стандартных задач анализа данных 65

Литература 72

1. Обучение нейронных сетей

Искусственный нейрон МакКаллока – Питтса:



На вход нейрона поступает некоторое множество сигналов $\mathbf{x_j}$, j=1, 2... *n*, каждый из которых является выходом другого нейрона. Нейрон вычисляет взвешенную сумму V=**XW** входных сигналов $\mathbf{x_j}$ и формирует на выходе сигнал величины **1**, если эта сумма превышает определенный порог $\boldsymbol{\theta}$, и **0** - в противном случае.

Нейрон описывается математической моделью в виде уравнения

$$\boldsymbol{Y = F(V) = F\left(\sum_{j=1}^{n} x_j \cdot w_j > 0\right), j = 1, 2, \ldots n,}$$

где $\boldsymbol{Y}$ – выходной сигнал нейрона, $\boldsymbol{F}$ - функция активации выхода нейрона, $\mathbf{w_j}$ – «вес» входа $\mathbf{x_j}$. Добавив постоянный единичный вход $\boldsymbol{x}_0$=1 (смещение) и положив $\boldsymbol{w}_0$= - $\boldsymbol{\theta}$, получим

$$\boldsymbol{Y = F(V) = F\left(\sum_{j=0}^{n} x_j \cdot w_j + x_0\right), j = 0, 1, \ldots n,}$$

Функции активации могут быть различными. Например, **пороговая функция** активации описывается формулой вида

$$Y(x) = \begin{cases} 1, & \textit{если } x \geq \Theta, \\ 0, & \textit{иначе.} \end{cases}$$

Линейная функция активации описывается формулой вида

$$Y(x) = \begin{cases} 0, & \textit{если } x \leq -0{,}5 \\ 1, & \textit{если } x \geq 0{,}5 \\ x, & \textit{иначе.} \end{cases}$$

Логистическая функция активации описывается формулой вида

$$Y(x) = \frac{1}{1 + \exp(-ax)},$$

где a – параметр функции, определяющий её крутизну. Когда a стремится к бесконечности, функция вырождается в пороговую. При $a = 0$ сигмоида вырождается в постоянную функцию со значением 0,5.

Недостатками пороговой и линейной активационных функций является их недифференцируемость на всей оси абсцисс. Как следствие, нейроны с такими

функциями нельзя использовать в сетях, обучающихся по алгоритму обратного распространения ошибки и другим алгоритмам, требующим дифференцируемости активационной функции. Нейроны с сигмоидальными функциями, чаще всего используются в скрытых (внутренних) слоях многослойных нейросетей. Достоинством логистической функции является простота её производной:

$$\frac{dY(x)}{dx} = tY(x)(1-Y(x)).$$

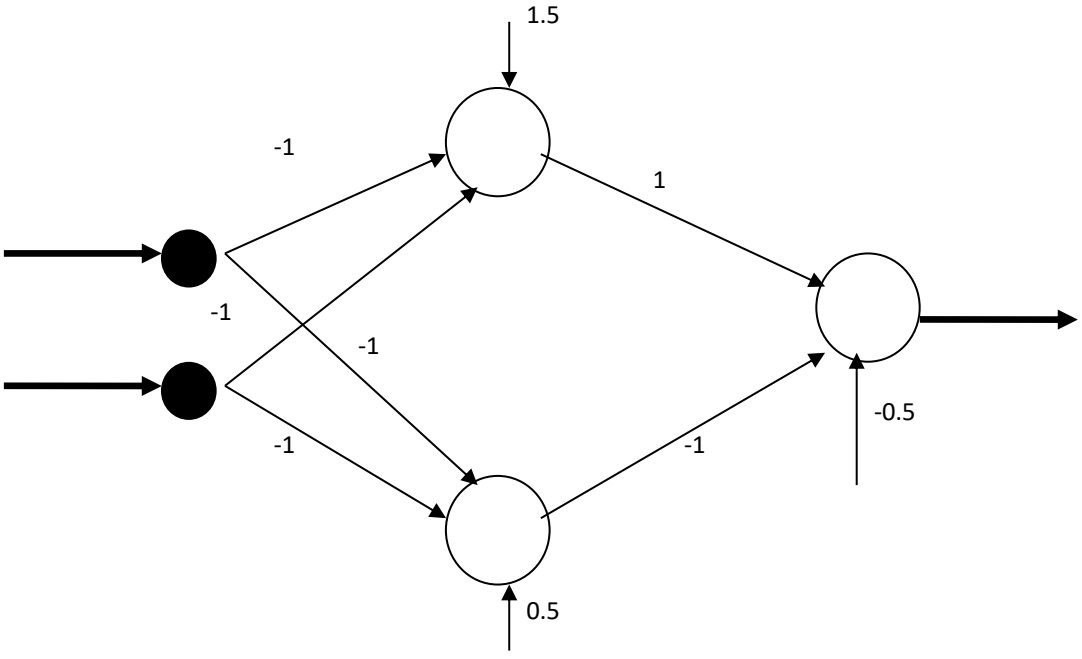
М Минский показал, что возможности персептрона ограничены. Например, рассмотрим логическую функцию «исключающее ИЛИ» (*XOR*), определяемую формулой $XOR(x, y) = x + y \pmod 2$.

Для лучшего понимания приведем таблицу значений функции *XOR* для различных значений переменных:

X_1	0	1	0	1
X_2	0	0	1	1
$XOR(X_1, X_2)$	0	1	1	0

Эта функция не реализуема однослойным персептроном, то есть не существует такой прямой, которая бы разделила эти два класса объектов.

Задача. Сеть содержит два входных элемента, два скрытых элемента и один выходной элемент. Все связи идут от входного слоя к выходному. К нейронам скрытого слоя доступа нет.



$$e_j = \sum_{i=0}^{n} x_i * w_{ij}$$

$$y(e_j) = \begin{cases} 1, & \text{если } e_j \geq 0, \\ 0, & \text{если } e_j < 0. \end{cases}$$

Пусть $x_1 = 1$, $x_2 = 1$. Для первого скрытого элемента со смещением 1.5 получаем

$e_1 = x_0 \times 1.5 + x_1 \times (-1) + x_2 \times (-1) = 1 \times 1.5 + 1 \times (-1) + 1 \times (-1) = -0.5$. На его выходе 0.

Для второго скрытого нейрона со смещением 0.5 получаем

$e_2 = 1 \times (-0.5) + x_1 \times (-1)) + x_2 \times (-1) = (1 \times 0.5 + 1 \times (-1) + (1 \times (-1) = -1.5$. На его выходе 0.

Для выходного элемента получаем

$e_3 = 1 \times (-0.5) + 0 \times (1) + 0 \times (-1) = -0.5 + 0 + 0 = -0.5$, поэтому на выходе сети будет 0.

Если процедуру повторить для трех оставшихся пар входов, то можно убедиться в том, что вывод представленной выше сети соответствует функции *XOR*.

Для решения конкретной задачи нужно выбрать подходящую нейросеть. При этом нужно учитывать не только перечисленные в таблице критерии, но и архитектуру сети. Выбор архитектуры подразумевает определение количества слоев и нейронов в этих слоях. Не существует формального алгоритма по определению нужной архитектуры, поэтому на практике выбирают или заведомо маленькую сеть и постепенно ее наращивают или заведомо большую и постепенно выявляют неиспользуемые связи и сокращают сеть.

Нейронная сеть, прежде чем использоваться на практике для решения какой-либо задачи, должна быть обучена. Обучение нейронной сети – это процесс настройки синаптических весов.

Однослойная нейросеть может обучаться по правилу Хебба. Предъявляем на вход персептрона один пример. Если выходной сигнал персептрона совпадает с правильным ответом, то никаких действий предпринимать не надо. В случае ошибки необходимо обучить персептрон правильно решать данный пример. Ошибки могут быть двух типов. Рассмотрим каждый из них.

Первый тип ошибки – на выходе персептрона 0, а правильный ответ – 1. Для того, чтобы персептрон (1) выдавал правильный ответ необходимо, чтобы сумма в правой части (1) стала больше порога. Поскольку переменные x_i принимают значения 0 или 1, увеличение суммы может быть достигнуто за счет увеличения весов w_i. Однако нет смысла увеличивать веса при переменных x_i, которые равны нулю. Таким образом, следует увеличить веса w_i при тех переменных x_i, которые равны 1. Для закрепления единичных сигналов с x_i, следует провести туже процедуру и на всех остальных слоях.

Первое правило Хебба. Если на выходе персептрона получен 0, а правильный ответ равен 1, то необходимо увеличить веса связей между одновременно активными нейронами. При этом выходной персептрон считается активным. Входные сигналы считаются нейронами.

Второй тип ошибки – на выходе персептрона 1, а правильный ответ равен нулю. Для обучения правильному решению данного примера следует уменьшить сумму в правой части (1). Для этого необходимо уменьшить веса связей w_i при тех переменных x_i, которые равны 1 (поскольку нет смысла уменьшать веса связей при равных нулю переменных x_i). Необходимо также провести эту

процедуру для всех активных нейронов предыдущих слоев. В результате получаем второе правило Хебба.

Второе правило Хебба. Если на выходе персептрона получена 1, а правильный ответ равен 0, то необходимо уменьшить веса связей между одновременно активными нейронами.

Таким образом, процедура обучения сводится к последовательному перебору всех примеров обучающего множества с применением правил Хебба для обучения ошибочно решенных примеров. Если после очередного цикла предъявления всех примеров окажется, что все они решены правильно, то процедура обучения завершается.

Алгоритм обучения однослойной нейросети по дельта-правилу.

Ш а г 1. Инициализация матрицы весов (и порогов, в случае использования пороговой функции активации) случайным образом.

Ш а г 2. Предъявление нейросети образа (на вход подаются значения из обучающей выборки – вектор **X**), берется соответствующий выход (вектор **D**).

Ш а г 3. Вычисление выходных значений нейронной сети (вектор **Y**).

Ш а г 4. Вычисление для каждого нейрона величины расхождения реального результата с желаемым:

$$\varepsilon_i = (d_i - y_i),$$

где d_i – желаемое выходное значение на i-нейроне, y_i - реальное значение на i-нейроне.

Ш а г 5. Изменение весов (и порогов при использовании пороговой функции) по формулам:

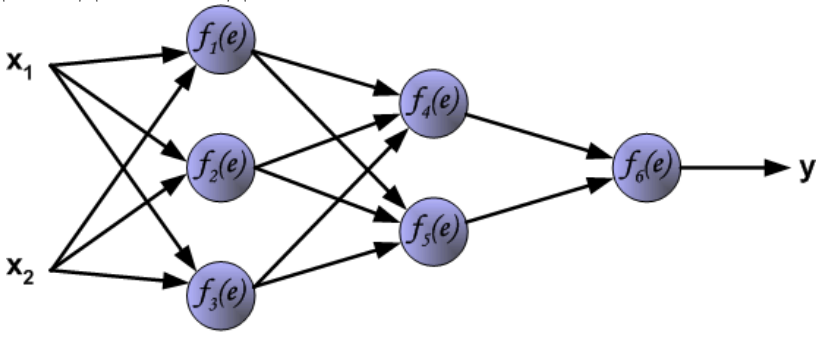
$$w_{ij}(t+1) = w_{ij}(t) - \eta \cdot \varepsilon_i \cdot x_j,$$
$$\Theta_i(t+1) = \Theta i(t) - \eta \cdot \varepsilon_i,$$

где t-номер текущей итерации цикла обучения, w_{ij} - вес связи j-входа с i-нейроном, η - коэффициент обучения (от 0 до 1), x_j – входное значение, Θ_i - пороговое значение i-нейрона.

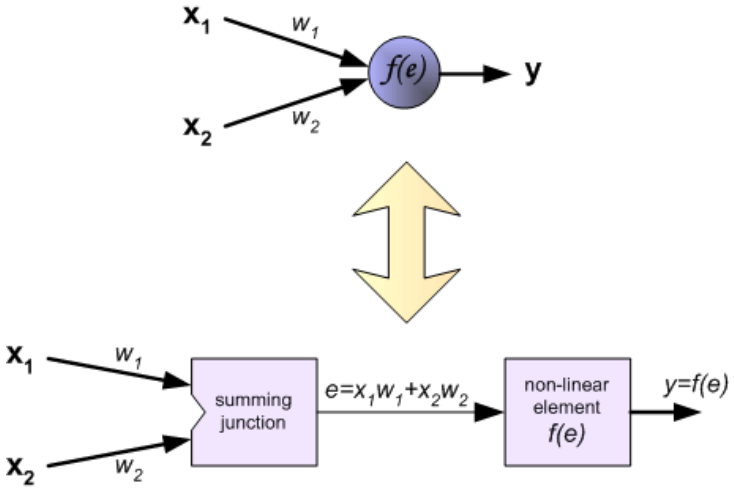
Ш а г 6. Проверка условия продолжения обучения (вычисление значения ошибки и/или проверка заданного количества итераций). Если обучение не завершено, то шаг 2, иначе заканчиваем обучение.

Пример обучения многослойной нейронной сети с помощью алгоритма обратного распространения (*backpropagation*).

Используем нейронную сеть, состоящую из трёх слоёв и имеющую два входа и один выход:



Сигнал **e** – это взвешенная сумма входных сигналов, а выходной сигнал - y = f(e):



Чтобы обучить нейронную сеть мы должны подготовить обучающие данные(примеры).

В нашем случае, тренировочные данные состоят из входных сигналов (x1 и x2) и желаемого результата z. Обучение – это последовательность итераций (повторений). В каждой итерации весовые коэффициенты нейронов подгоняются с использованием новых данных из тренировочных примеров. Изменение весовых коэффициентов и составляют суть алгоритма, описанного ниже.Каждый шаг обучения начинается с воздействия входных сигналов из тренировочных примеров. После этого мы можем определить значения выходных сигналов для всех нейронов в каждом слое сети.

Шаг 1. Обычно начальные значения весов всех нейронов всех слоев полагаются случайными числами. Формируется обучающая выборка (ОВ).

Шаг 2. Сети из ОВ предъявляется образ $\boldsymbol{X}$, сигнал от которого распространяется через нейроны скрытого слоя до выхода:

$$\boldsymbol{y_1} = f_1(w_{(x1\ \ )1}x_1 + w_{(x2\ \ )1}x_2)$$
$$\boldsymbol{y_2} = f_2(w_{(x1\ \ )2}x_1 + w_{(x2\ \ )2}x_2)$$
$$\boldsymbol{y_3} = f_3(w_{(x1\ \ )3}x_1 + w_{(x2\ \ )3}x_2)$$

$\boldsymbol{y_4} = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$, $\boldsymbol{y_5} = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$, $\boldsymbol{y} = f_6(w_{46}y_4 + w_{56}y_5)$.

Шаг 3. Выходной сигнала $\boldsymbol{y}$ сравнивается с желаемым выходным сигналом $\boldsymbol{z}$, который хранится в ОВ. Разница между этими двумя сигналами называется *ошибкой* δ (дельта) выходного слоя сети:

$$\boldsymbol{\delta = z - y}$$

Невозможно непосредственно вычислить сигнал ошибки для нейронов скрытого слоя, т.к. выходные значения этих нейронов, неизвестны.

Шаг 3. Идея заключается в распространении сигнала выходной ошибки δ обратно на все нейроны вплоть до входов:

$$\boldsymbol{\delta_5} = w_{56}\delta, \qquad \boldsymbol{\delta_4} = w_{46}\delta,$$

Если ошибка пришла от нескольких нейронов – она суммируется:

$$\boldsymbol{\delta_3} = w_{34}\delta_4 + w_{35}\delta_5, \qquad \boldsymbol{\delta_2} = w_{24}\delta_4 + w_{25}\delta_5, \qquad \boldsymbol{\delta_1} = w_{14}\delta_4 + w_{15}\delta_5$$

Функционал квадратичной ошибки сети для данного входного образа имеет вид:

$$\boldsymbol{E} = \frac{1}{2}\delta^2$$

Данный функционал подлежит ***минимизации***. Классический ***градиентный*** метод оптимизации состоит в итерационном уточнении аргумента (t: = t + 1) согласно формуле:

$$\boldsymbol{w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j \frac{df_j(e)}{de} y_i}$$

В этой формуле $\frac{df_j(e)}{de}$ – производная функции активации нейрона, чьи веса корректируются. Поэтому функция должна иметь первую производную. Чаще всего в качестве функции активации используется сигмоид:

$$\boldsymbol{f(e)} = \frac{1}{1+\exp(-e)}$$

производная от ***f*(e)** выражается через саму функцию:

$$\frac{df}{de} = f(e) * (1 - f(e))$$

Это позволяет существенно сократить вычислительную сложность метода.

Шаг 4. Корректировка весов:

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1 \qquad w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$

$$w'_{(x1)2} = w_{(x1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1 \qquad w'_{(x2)2} = w_{(x2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2$$

$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1 \qquad w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1 \qquad w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2 \qquad w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$

$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1 \qquad w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2 \qquad w'_{35} = w_{14} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$

$$w'_{46} = w_{46} + \eta\delta \ \frac{df_6(e)}{de} y_4 \qquad w'_{56} = w_{56} + \eta\delta \ \frac{df_6(e)}{de} y_5$$

Коэффициент **η (эта)** влияет на скорость обучения сети. Есть несколько методов для выбора этого параметра. Первый способ – начать обучение с большим значением параметра η. Во время коррекции весовых коэффициентов, параметр постепенно уменьшают. Второй – более сложный способ обучения, начинается с малым значением параметра η. В процессе обучения параметр увеличивается, а затем вновь уменьшается на завершающей стадии обучения.

Проблема переобучения (*оверфиттинг* или слишком близкая подгонка).

Шаг 5. Шаги 2-4 повторяются для всех обучающих примеров. Обучение завершается по достижении малой квадратичной ошибки ***E*** или максимально допустимого числа итераций

Практика показывает, что сходимость *back propagation* медленная. Алгоритм применятся для обучения с учителем. Для обучения без учителя - алгоритмы Хебба и Кохонена. В последнее время, сформировалась еще одна парадигма – обучение с подкреплением.

2-3. Продукционные системы, семантические сети и фреймы для представления знаний в интеллектуальных системах обучения

Целью занятия является изучение одной из наиболее распространенных, особенно в области экспертных систем, моделей представления знаний: продукционных правил, семантических сетей, фреймов, а также решение практических задач построения этих моделей.

Рекомендуемый объем занятия по теме – 4 часа.

Продукционная модель представления знаний.

В ИИ принято различать «жесткие» и «мягкие» модели представления знаний. К «мягким» моделям относятся нейросети, эволюционные алгоритмы, нечеткая логика и их гибриды, к «жестким» моделям – продукционные правила, семантические сети, фреймы и различные логические модели (логика высказываний и предикатов).

В самом общем виде продукция представляется выражением вида

(W, U, P, A→B, C), (1)

где A→B – ядро продукции, соответствующее правилу «если (условия A), то (действия B)»; W – сфера применения продукции в классе ситуаций некоторой предметной области; U – предусловие, содержащее информацию об истинности продукции (коэффициент уверенности), ее значимости и т.п.; P – внешнее, не входящее в A условие, разрешающее применять данную продукцию; C – постусловие, определяющее изменения, которые возможно надо внести в продукционную систему после выполнения данной продукции. Обязательной частью продукции является только ядро, трактовка которого может быть разной: если A истинно, то B также истинно; если A содержится в БЗ, то B следует включить в БЗ; если A истинно, то следует выполнить B и т.д.

В БЗ, состоящих из множества продукционных правил вида (1), под условием понимается некоторое предложение-образец, по которому осуществляется поиск в БЗ, а под действием - действия, выполняемые при успешном исходе поиска. Программа, управляющая перебором правил, называется машиной вывода. Вывод бывает прямой (от данных к поиску цели) или обратный (от цели для ее подтверждения - к данным). Данные - это исходные факты, на основании которых запускается машина вывода - программа, перебирающая правила из БЗ.

Пример. Имеется фрагмент БЗ из двух правил:

П1: ЕСЛИ "отдых - летом" и "человек - активный", ТО "ехать в горы",

П2: ЕСЛИ "любит солнце", ТО "отдых летом".

Предположим, в систему поступили данные - "человек активный" и "любит солнце"

Прямой вывод - исходя из данных, получить ответ.

1-й проход.

Шаг 1. Пробуем П1, не работает (не хватает данных "отдых - летом").

Шаг 2. Пробуем П2, работает, в базу поступает факт "отдых - летом".

2-й проход.

Шаг 3. Пробуем П1, работает, активируется цель "ехать в горы", которая и выступает как совет, который дает ЭС.

Обратный вывод - подтвердить выбранную цель при помощи имеющихся правил и данных.

1-й проход.

Шаг 1. Цель - "ехать в горы". Пробуем П1 - данных, "отдых - летом" нет, они становятся новой целью, и ищется правило, где она в правой части.

Шаг 2. Цель "отдых - летом": правило П2 подтверждает цель и активирует ее.

2-й проход.

Шаг 3. Пробуем П1, подтверждается искомая цель.

Задача. *Построить продукционную модель представления знаний в предметной области «Посещение кафе».*

Описание процесса решения. Чтобы построить продукционную модель, необходимо:

1) Определить целевые действия задачи (они являются решениями).

2) Определить промежуточные действия между начальным и конечным состояниями.

3) Определить условия для каждого действия, при котором его целесообразно и возможно выполнить, а также порядок выполнения действий.

4) Добавить конкретики при необходимости, исходя из поставленной задачи.

5) Преобразовать полученный порядок действий и соответствующие им условия в продукции.

6) Проверить продукции на непротиворечивость, записав цепочки продукций и явно проследив связи между ними.

Двигаться при построении продукционной модели можно от результата к начальному состоянию, либо от начального состояния к результату.

Решение.

1) Обязательное действие, выполняемое в кафе – поглощение пищи и ее оплата. Значит, есть уже два целевых действия «*съесть пищу*» и «*оплатить*», которые взаимосвязаны и следуют друг за другом.

2) Прежде чем что-либо съесть в кафе, туда нужно прийти, дождаться официанта и сделать заказ. Кроме того, нужно выбрать, в какое именно кафе пойти. Значит, цепочка промежуточных действий: «*выбор кафе и путь туда*», «*сделать заказ официанту*».

3) Прежде чем идти в кафе, необходимо убедиться, что есть необходимая сумма денег. Выбор кафе может обуславливаться многими причинами, выберем территориальный признак – к какому ближе в тот и идем. В разных кафе работают разные люди, поэтому в зависимости от выбора кафе, официанты будут разные. Кроме того, разные кафе специализируются на разных кухнях, поэтому заказанные блюда будут в разных кафе отличаться. Значит вначале идут действия, позволяющие выбрать кафе, затем характеризующие кафе, а уже после заказ, еда, и оплата заказа.

4) Пусть в задаче будут рассматриваться два кафе: «2 фунта» и «Осака». Первый – паб и заказы приносят быстрее, чем во втором, второй – пиццерия. В первом работает официант Борис, а во втором официантка Юля. Антон – это клиент.

5) Преобразуем указанные действия и соответствующие им условия в продукции «Если, то»:

- Если субъект хочет есть и у субъекта есть достаточная сумма денег, то субъект может пойти в кафе.
- Если субъект ближе к кафе «2 фунта», чем к кафе «Осака» и субъект может пойти в кафе, то субъект идет в кафе «2 фунта».
- Если субъект ближе к кафе «Осака», чем к кафе «2 фунта» и субъект может пойти в кафе, то субъект идет в кафе «Осака».
- Если субъект идет в кафе «Осака» и в нем работает официант Юля, то у субъекта принимает заказ Юля.
- Если субъект идет в кафе «2 фунта» и в нем работает официант Борис, то у субъекта принимает заказ Борис.
- Если субъект выбрал блюда и у субъекта принимает заказ Юля, то заказ принесут через 20 мин
- Если субъект выбрал блюда и у субъекта принимает заказ Борис, то заказ принесут через 10 мин.
- Если заказ принесут через 20 мин. или заказ принесут через 10 мин., то субъект может есть.
- Если субъект может есть, то после еды субъект должен оплатить заказ.

Введем обозначения для **фактов** (Ф), **действий** (Д) и **продукций** (П), тогда:

Субъект = Антон;

Ф1= субъект хочет есть;

Ф2= у субъекта есть достаточная сумма денег;

Ф3= субъект ближе к кафе «2 фунта», чем к «Осака»;

Ф4=в кафе «Осака» работает официант Юля;

Ф5=в кафе «2 фунта» работает официант Борис;

Ф6= субъект выбрал блюда;

Д1= субъект может пойти в кафе;

Д2=субъект идет в кафе «2 фунта»;

Д3=субъект идет в кафе «Осака»;

Д4= у субъекта принимает заказ Юля;

Д5=у субъекта принимает заказ Борис;

Д6=заказ принесут через 20 мин.

Д7=заказ принесут через 10 мин.

Д8=после еды субъект должен оплатить заказ.

П1: (Д6 или Д7) → Д8;

П2: (Д5) → Д7;

П3: (Д4) → Д6

П4: (Д2 и Ф5) → Д5

П5: (Д3 и Ф4) → Д4

П6: (не Ф3 и Д1) → Д3

П7: (Ф3 и Д1) → Д2

П8: (Ф1 и Ф2) → Д1

Для продукций можно установить следующий *приоритет* (чем выше приоритет, тем раньше проверяется правило): П1 – 1, П2 = П3 = 2, П4 = П5 = 3, П6 = П7 = 4, П8 = 5.

Отобразим взаимосвязи продукций на графе:

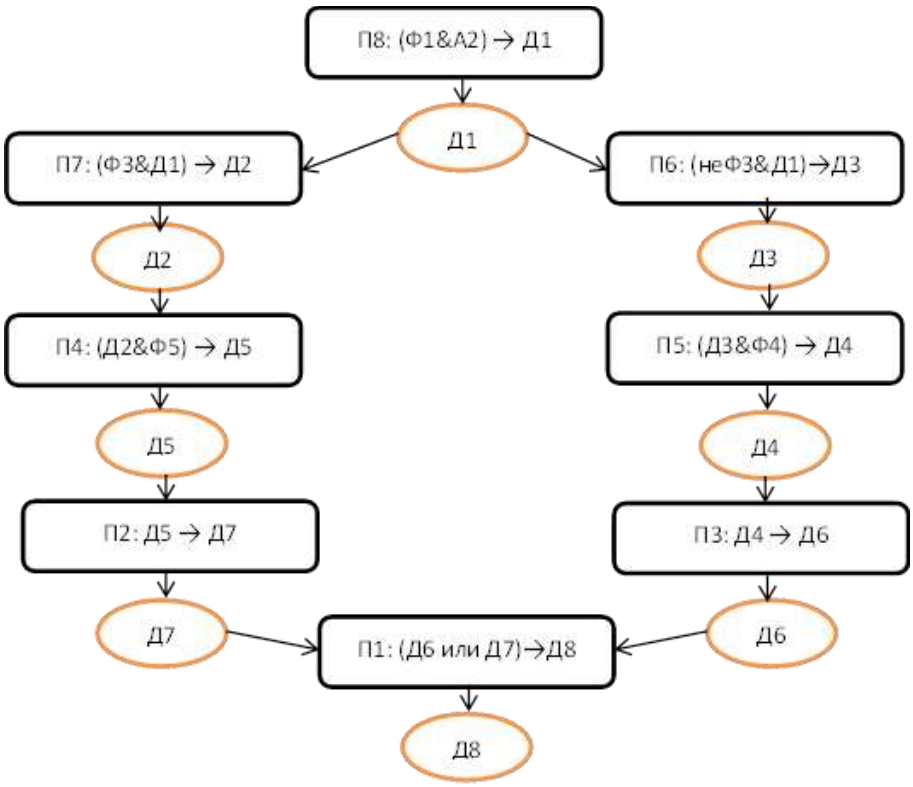


Рис. Взаимосвязь продукций предметной области «Кафе»

В целом, в продукционном программировании система включает:

- Продукционный набор правил;
- Рабочую память, определяющую текущее состояние задачи и содержащую данные, описание цели и промежуточные результаты;
- Интерпретатор правил, который решает, когда надлежит применить каждое из них.

Рассмотрим детали этого механизма на примере языка CLIPS.

Продукционный набор правил. Правила в CLIPS-программе используются для выполнения ряда действий, которые необходимо выполнить в определенной ситуа¬ции. Разработчик ЭС определяет совместную совокупность правил для решения проблемы. Правило состо¬ит из двух частей: слева записываются предпосылки (антицедент или условия), которые обычно представляются в виде триады «объект – атрибут – значение атрибута», а справа – действия (заключе¬ния, консеквент):

(defrule <имя правила>
<условие 1>
….
<условие m>

=>
<действие 1>
….
<действие n>
)

Здесь для определения правил используется defrule.

Чтобы правило было выполнимым, необходимо (но не достаточно!), чтобы выполнялись все его условия. Если условная часть правила пуста, то для его активации необходимо наличие в списке фактов исходного факта (initial-fact). Условие выполняется, если соответствующий ему факт присутствует в списке фактов рабочей памяти. После определения, какие из правил являются выполнимыми, они помещаются в список активированных правил. Наконец, если какое-то одно правило из списка будет выполнено, действия задают изменения, которые должны быть внесены в состояние рабочей памяти.

Рабочая память.

Ее функция – хранить данные (факты) в формате векторов «объект – атрибут – значение атрибута». Эти данные используются интерпретатором, который активизирует те правила, условия которых на имеющихся данных выполняются.

Интерпретатор правил.

Процесс его работы описывается в терминах цикла «распознавание - действие»:

1. Сопоставить условия правил и элементы рабочей памяти;
2. Если окажется, что можно активизировать более одного правила, то выбрать одно из них (разрешить конфликт);
3. Применить выбранное правило. Результатом, скорее всего, будет добавление нового элемента данных в рабочую память и/или удаление каких-либо данных из рабочей памяти. Затем перейти к п.1.

Перед началом этого процесса в рабочую память вводится initial-fact. Цикл останавливается, если обнаруживается, что ни одно правило не активизируется, или если правило содержит команду прекращения работы (halt). Остается вопрос, связанный с разрешением конфликтных ситуаций. Рассмотрим его подробнее, т.к. он специфичен для каждой системы. Конечно, можно пытаться сформулировать такой набор правил, что в любой ситуации будет удовлетворяться только одно из них. Однако в ЭС обычно используются недетерминированные наборы правил, т.к. в реальной жизни ситуации позволяют использовать более одного правила.

Разрешение конфликтов.

Выбранные интерпретатором несколько правил называют конфликтующим множеством (в CLIPS – agenda – список заявок). Цель процедуры разрешения конфликтов – выбрать из списка одно правило. Производительность ЭС зависит от ее чувствительности к изменению рабочей памяти и стабильности как степени консерватизма системы.

При всем разнообразии свойства механизмов разрешения конфликтов можно разделить на три группы:

Разнообразие. Не следует применять к одним и тем же данным правило, которое уже применялось ранее. Самое простое – удалять из списка примененное ранее правило. Но если надо его повторить, то программист использует функцию refresh.

Новизна. В CLIPS элементы рабочей памяти снабжаются атрибутом времени их порождения и приоритет отдается правилам «реагирующим» на более «свежие» данные.

Специфика. Правила считаются более специфичными, принимающими во внимание больше информации, если включают большее число условий и, поэтому труднее удовлетворяются.

Семантические сети для представления знаний.

Продукционные порождающие правила являются очень удобной моделью знаний для представления для представления связей между состоянием проблемы и действиями, которые необходимо предпринять для ее решения. Иными словами, ПП очень подходят для ответа на вопрос «Что делать, если…?».

Иное дело, если необходимо представить знания о событиях, сложных объектах, связях между ними, их классификации или представить смысл выражений естественного языка человека. Здесь удобнее использовать структуры в виде ***семантической сети (СС).*** Это граф, вершины которого соответствуют понятиям или объектам, а дуги - отношениям между объектами.

Наиболее часто используются отношения «объект-множество», обозначающие, что объект принадлежит некоторому множеству. Это отношение называется **отношением классификации** и обозначается **ISa**. Обозначение произошло от английского «IS A». Связь ISA предполагает, что свойства объекта *наследуются* от множества. Обратное к ISA отношение используется для обозначения примеров, поэтому так и называется — «Example», или по-русски, «Например».

Отношение между надмножеством и подмножеством называется **AKO** — «A Kind Of» («разновидность»). Альтернативные названия — «SubsetOf» и «Подмножество». Это отношение определяет, что каждый элемент первого множества входит и во второе (выполняется ISA для каждого элемента), а также логическую связь между самими подмножествами: что первое не больше второго и свойства первого *множества* наследуются вторым.

Объект, как правило, состоит из нескольких частей, или элементов. Например, *компьютер* состоит из системного блока, монитора, клавиатуры, мыши и т. д. Важным отношением является **HasPart**, описывающее части/целые объекты Двигатель — это часть для автомобиля. Автомобиль — это объект, который включает в себя двигатель.

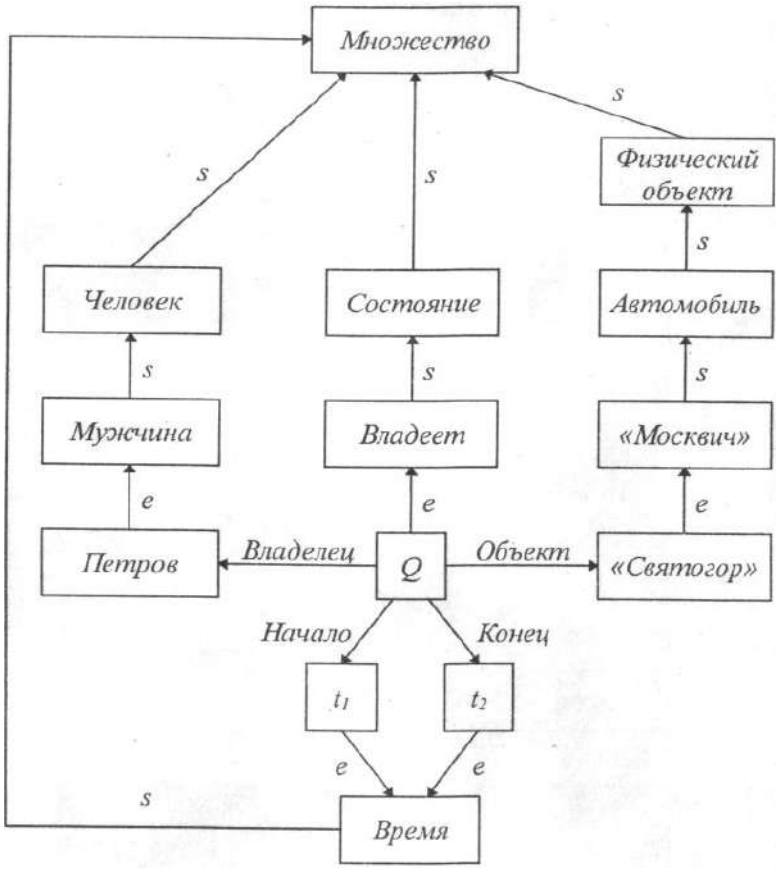
Часто в семантических сетях требуется определить отношения синонимии, а также:

- функциональные связи (определяемые обычно глаголами «производит», «влияет»…);
- количественные (больше меньше, равно…);
- пространственные (далеко от, близко от, за, под, над…);
- временные (раньше, позже, в течение…);

- атрибутивные (иметь свойство, иметь значение);
- логические (И, ИЛИ, НЕ);
- лингвистические.

Этот список может сколь угодно продолжаться: в реальном мире количество отношений огромно. Например, между понятиями может использоваться отношение «совершенно разные вещи» или: «не имеют отношения друг к другу (Солнце, кухонный чайник).

Пример. Ниже представлен фрагмент семантической сети, иллюстрирующей предложение "**Петров на протяжении периода времени с t1 по t2 владел автомобилем марки "Святогор"**:



Дуги *s, e, владелец, объект, начало, конец* на рисунке указывают на иерархическую связь понятий. Символ Q обозначает конкретную описываемую ситуацию.

СС, используемые для представления семантики ЕЯ, называют ***ассоциативными сетями*** (АС). Квиллиан не первым обратил внимание на важность обобщенного абстрактного знания для понимания ЕЯ. Но он первым предложил использовать в качестве модели памяти СС и предложил метод для извлечения информации из памяти. Этот метод напоминает организацию толковых словарей: каждое понятие в них определяется другими понятиями. Выяснилось, что предложенная модель и метод обладают важным свойством – когнитивной экономией (вычислительная машина – это конструкция их многих деталей, а РС – это тоже вычислительная машина, тогда нет смысла в явном виде

хранить эту информацию, присоединяя ее в сети к вершине PC). В современном программировании это принято называть свойством наследования

Задача. Построить сетевую модель представления знаний в предметной области «Кафе» (посещение кафе).

Описание процесса решения задачи. Для построения СС необходимо выполнить следующие шаги:

1) Определить абстрактные объекты и понятия предметной области, необходимые для решения поставленной задачи. Оформить их в виде вершин.

2) Задать свойства для выделенных вершин, оформив их в виде вершин, связанных с исходными вершинами атрибутивными отношениями.

3) Задать связи между этими вершинами, используя функциональные, пространственные, количественные, логические, временные, атрибутивные отношения, а также отношения типа «являться наследником» (АКО) и «являться частью» (ISA).

4) Добавить конкретные объекты и понятия, описывающие решаемую задачу. Оформить их в виде вершин, связанных с уже существующими отношениями типа «являться экземпляром», «есть».

5) Проверить правильность установленных отношений (вершины и отношения при правильном построении образуют предложение.

Решение.

1) Ключевые понятия данной предметной области – кафе, тот, кто посещает кафе (клиент) и те, кто его обслуживают (повара, метрдотели, официанты, для простоты ограничимся только официантами). У обслуживающего персонала и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Продукцией кафе являются блюда, которые заказывают клиенты.

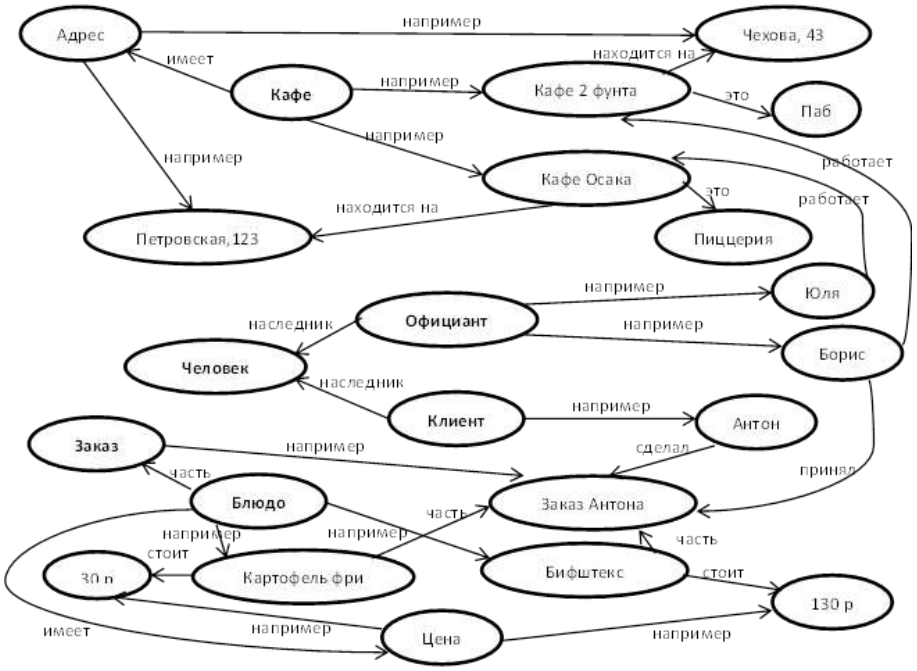
Исходя из этого, вершины графа будут следующими: «**Кафе**», «**Человек**», «**Официант**», «**Клиент**», «**Заказ**» и «**Блюдо**».

2) У этих объектов есть определенные свойства и атрибуты. Например, кафе располагаются по определенным адресам, каждое блюдо из меню имеет свою цену. Поэтому добавим вершины «**Адрес**» и «**Цена**».

3) Определим для имеющихся вершин отношения и их типы.

4) Добавим знание о конкретных фактах решаемой задачи. Пусть имеется два ресторана: «Осака» и «2 фунта», в первом работает официантка Юля, а во втором официант Борис. Антон решил пойти в кафе «2 фунта» и сделал заказ официанту на 2 блюда: **картофель фри** за 30 р., **бифштекс** за 130 р. Также известны адреса этих ресторанов и их специфика.

Исходя из этого, добавим соответствующие вершины в граф и соединим их функциональными отношениями и отношениями типа «*например* или *являться экземпляром*». Полученный в результате граф изображен на рис.:



5) Осуществим проверку установленных отношений. Например, возьмем вершину «**Блюдо**» и пройдем по установленным отношениям. Получаем следующую информацию: ***блюдо является частью заказа, примерами блюд могут служить картофель фри (по цене 30 р) и бифштекс (по цене 130 р)***.

Для получения ответа на какой-либо вопрос по этой задачи, необходимо найти соответствующий участок сети и, используя отношения, получить результат.

Например, вопрос «**Какова цена заказа Антона**?» Из запроса понятно, что необходимо найти следующие вершины: «**Цена**», «**Антон**», «**Заказ**» и «**Заказ Антона**». Часть СС, находящаяся между этими вершинами, содержит ответ, а именно, частью заказа Антона являются картофель фри и бифштекс, которые стоят 30 и 130 р. соответственно. Больше информации о заказе Петра в модели нет, поэтому делаем вывод – Петр заплатил 160 р.

Фреймовые модели представления знаний.

Естественным желанием исследователей СИИ было объединить вместе представление знаний СС и ПП. Это привело к появлению теории ***фреймов*** (**frame** — остов, скелет, костяк, каркас). Используя идеи теории фреймов в 70-х годах в MTI провели исследования, которые привели к разработке философии ООП и созданию таких языков как Smalltalk, C++, Java.

Идея фреймов в том, что представление понятий в мозге базируется на довольно расплывчатых понятиях. Человек обращает внимание на свойства, которые у него ассоциируются с объектами-прототипами, наиболее ярко представляющими свой класс (птица – воробей, четырехугольник – прямоугольник и т.п.). Границы между разными классами всегда размыты, в каждом правиле или классе встречаются исключения. При использовании фреймов знания не «размазываются» по программному коду приложения, как в ПП, но и не собираются воедино в виде метазнаний, как в СС или АС.

Марвин Минский определил фрейм как «структуру данных для представления стереотипных ситуаций». Реализованная им идея заключалась в том, чтобы сконцентрировать все данные (знания) о конкретном объекте или событии в единой структуре, а не распределять ее между множеством более мелких структур.

Фрейм имеет собственное название, а также список *слотов* и их значений (*наполнителей*). Значениями могут быть данные любого типа, а также название другого фрейма. Таким образом, фреймы образуют сеть. Кроме того, существует связь между фреймами типа АКО (a kind of), которая указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются список и значения слотов. При этом возможно множественное наследование – перенос свойств от нескольких прототипов.

Фрейм, как абстрактный образ, может быть представлен следующим образом:

(*ИМЯ ФРЕЙМА:*
(имя 1-го слота: значение 1-го слота),
(имя 2-го слота: значение 2-го слота),
……..
(имя N-го слота: значение N-го слота)).

Слоты заполнены значениями разнообразных атрибутов, ассоциирующихся с сущностью.

Табличное представление структуры фрейма со слотами выглядит следующим образом:

Имя фрейма			
Имя слота	Значение слота	Способ получения значения	Демон

Передать данные во фрейм, заполнив его слоты можно по-разному: при конструировании фрейма, через вызов функции, указанной в слоте, через присоединенную к слоту процедуру, которая называется *демоном*, из диалога с пользователем, через наследование свойств от других фреймов, из базы данных. Способ получения значения определяет, как именно устанавливается значение конкретного слота, а выбор способа зависит от свойств самих данных.

Демоном называется процедура, автоматически запускаемая при выполнении некоторого условия (события) при обращении к соответствующему слоту. Демонов может быть несколько. Наиболее похож механизм присоединенных процедур к триггерам в реляционных базах данных.

Существует несколько видов фреймов, которые позволяют описать предметную область и решаемую задачу: прототипы, экземпляры, структуры, ситуации, сценарии, роли.

Задача. Построить фреймовую модель представления знаний в предметной области «Кафе» (посещение кафе).

Описание процесса решения. Для построения фреймовой модели представления знаний необходимо выполнить следующие шаги:

1) Определить абстрактные объекты и понятия предметной области, необходимые для решения поставленной задачи. Оформить их в виде **фреймов-прототипов** (**фреймов-объектов**, **фреймов-ролей**).

2) Задать конкретные объекты предметной области. Оформить их в виде **фреймов-экземпляров** (фреймов-объектов, фреймов-ролей).

3) Определить набор возможных ситуаций. Оформить их в виде **фреймов-ситуаций** (прототипы). Если существуют прецеденты по ситуациям в предметной области, добавить **фреймы-экземпляры** и/или **фреймы-ситуации**.

4) Описать динамику развития ситуаций через набор сцен. Оформить их в виде **фреймов-сценариев**.

5) Добавить **фреймы-объекты** сценариев и сцен, которые отражают данные конкретной задачи.

Решение.

1) Ключевые понятия данной предметной области – кафе, клиент и те, кто его обслуживают (повара, метрдотели, официанты, для простоты ограничимся только официантами). У официантов и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Тогда фреймы «Кафе» и «Человек» являются **прототипами-образцами**, а фреймы «Официант» и «Клиент» - **прототипами-ролями**. Также нужно определить основные слоты фреймов – характеристики, имеющие значения для решаемой задачи. Опишем указанные фреймы в виде таблиц:

КАФЕ			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
название		из внешних источников	
адрес		из внешних источников	
часы работы		из внешних источников	
специализация		из внешних источников	
класс	средний / высший	из внешних источников	

2) **Фреймы-образцы** описывают конкретную ситуацию: какие кафе имеются в городе, как именно организовывается посещение, кто является посетителем, кто работает в выбранном кафе и т.д. Поэтому определим следующие фреймы-образцы, являющиеся наследниками фреймов-прототипов:

КАФЕ «ОСАКА» (АКО **КАФЕ**)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
название	ОСАКА	из внешних источников	
адрес	Таганрог, Петровская, 123	из внешних источников	
часы работы	10:00-23:00	из внешних источников	
специализация	пиццерия	из внешних источников	
класс	средний	из внешних источников	

3) **Фреймы-ситуации** описывают возможные ситуации. В кафе клиент попадает в несколько типичных ситуаций: заказ и оплата. Конечно, возможны и другие не типичные ситуации: клиент подавился, у клиента нет наличности для оплаты счета и т.д. Рассмотрим несколько типичных ситуаций:

ОПЛАТА			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
вид платежа		из внешних источников	IF-ADDED (изменяет слот «чаевые»)
чаевые		присоединенная процедура	
оплатил	фрейм-образец	присоединенная процедура	
заказ	фрейм-образец	из внешних источников	IF-ADDED (изменяет слот «оплатил»)

4) Ситуации возникают после наступления каких-то событий, выполнения условий и могут следовать одна за другой. Динамику предметной области можно отобразить в **фреймах-сценариях**. Их может быть множество, опишем наиболее общий и типичный сценарий посещения кафе:

ПОСЕЩЕНИЕ КАФЕ			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
посетитель	фрейм-объект	из внешних источников	
кафе	фрейм-объект	из внешних источников	IF-ADDED, IF-

ПОСЕЩЕНИЕ КАФЕ			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
			REMOVED (изменяют слот «Официант»)
официант	фрейм-объект	присоединенная процедура (определяется по выбранному кафе)	
сцена 1	вход, выбор	из внешних источников	
сцена 2	заказ	из внешних источников	IF-ADDED (изменяет слот «оплатил»)
сцена 3	еда	из внешних источников	
сцена 4	оплата	из внешних источников	
сцена 5	выход	из внешних источников	

5) Пусть в рамках нашей задачи Антон посетил кафе «2 фунта». Тогда фреймы будут заполнены следующим образом:

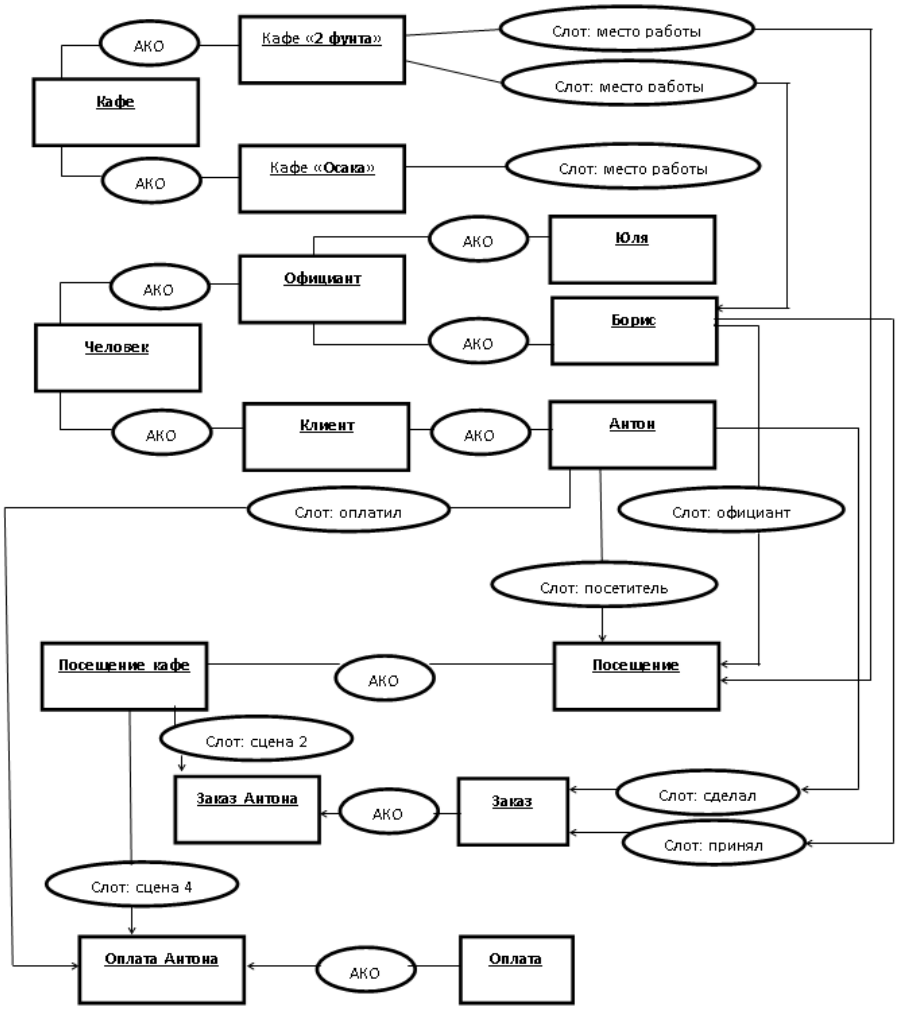
ОПЛАТА АНТОНА (АКО **ОПЛАТА**)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
вид платежа	наличные	из внешних источников	IF-ADDED (изменяет слот «чаевые»)
чаевые	30	присоединенная процедура	
оплатил	**АНТОН**	присоединенная процедура	
заказ	**ЗАКАЗ АНТОНА**	из внешних источников	IF-ADDED (изменяет слот «оплатил»)

Человек			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
сцена 1	М или Ж	из внешних источников	
возраст	от 0 до 150 лет	из внешних источников	

Ниже, на рисунке представлена граф-схема взаимосвязи фреймов в предметной области «Кафе».

Использование фреймовой модели аналогично семантической сети, только в процессе получения ответа кроме вершин учитываются и слоты.

Например, получить ответ на вопрос «Кто работает официантом в кафе “2 фунта”?» можно следующим образом: из запроса понятно, что необходимо найти фрейм «Кафе “2 фунта”» и проследить связь с фреймом «Борис», являющимся наследником фрейма «Официант». Также можно найти слот «Место работы» и, проверив его значение во фреймах наследниках фрейма «Официант» определить, что официантом в кафе “2 фунта” работает Борис.



4. Метод резолютивного вывода в интеллектуальных системах

Современным методом доказательства теорем, решения трудных логических задач и головоломок является дедуктивный **метод Робинсона-Маслова**. На нем основан язык логического программирования **Пролог**. Суть метода выражают две простые теоремы:

Теорема 1. Даны булевы формулы **F1, F2,…, Fn** и **G**. Формула **G** является *логическим следствием* формул **F1, F2,…, Fn** тогда и только тогда, когда формула (теорема) **F1&F2&…&Fn→G** *общезначима*.

Теорема 2. Даны формулы **F1, F2,…, Fn** и **G**. Формула **G** является логическим следствием формул **F1, F2,…, Fn** тогда и только тогда, когда формула **F1&F2&…&Fn&¬G** *противоречива*.

Таким образом, факт, что формула G является *следствием* последовательности других формул, сводится к доказательству *общезначимости* (Теорема 1) или *противоречивости* (Теорема 2) некоторой формулы. Это полностью аналогично выводу заключения теоремы из множества аксиом или утверждений. Многие задачи в математике и логике формулируются как задачи доказательства теорем. Обозначим общезначимую формулу через ■, противоречивую – через □.

Задача 1. Проверить общезначимость формулы:

F=(P→Q)&(Q→R)&(R→L)&(M→¬L) → (P→¬M).

От противного, пусть F=0. Тогда (P→Q)&(Q→R)& (R→L)&(M→¬L)=1, а (P→¬M)=0.

Если (P→¬M)=0, то P=1, M=1. Тогда (1→Q)&(Q→R)&(R→L)&(1→¬L) = Q(¬L)(¬Q+R)(¬R+L) = Q(¬L)R(¬R+L)=0+0=0. Противоречие! Поэтому формула общезначимая.

Задача 2а. По делу о похищении автомобиля были допрошены 4 гангстера – Андре, Боб, Стив, Том. Андре сказал, что машину похитил Боб. Боб утверждал, что виноват Том. Том сказал, что Боб лжет. Стив настаивал, что машину он не угонял. Один из гангстеров сказал правду. Кто похитил автомобиль?

Вначале выделим элементарные высказывания и сведем задачу к их проверке. Обозначим высказывания «Андре украл», «Боб украл», «Стив украл», «Том украл» через A, B, S, T. Тогда показания гангстеров имеют вид B, T, ¬T, ¬S. Из матлогики известно, что формула (T+¬T) тождественно истинна (общезначима), поэтому одно из утверждений T или ¬T обязательно истинно. Значит, Андре и Стив сказали ложь. Так как утверждение Стива (¬S) ложно, то авто украл **Стив**.

Задача 2б. На складе совершено хищение. Подозрение пало на трех человек: A, B и C. Они были доставлены для допроса. Установлено следующее:

- никто, кроме A, B, C, не был замешан в деле;
- подозреваемый A никогда не ходит на дело без, по крайней мере, одного соучастника;

- C невиновен.

Вопрос: виновен ли B ? Выяснить, является ли утвердительный ответ на вопрос *логическим следствием* установленных фактов.

Ответ. **Да, виновен.** Обозначим через A утверждение "A виновен", через B - "B виновен", через C - "C виновен".

Тогда - $F_1 = A \vee B \vee C$, F_2: $A \rightarrow B \vee C$, $F_3 = \neg C$

Вопрос виновен ли B: $F_4 = B$

На интерпретациях **A=1, B=1, C=0 и A=0, B=1, C=0** имеем $F_1=F_2=F_3=1$.

Метод резолюций

Идея метода резолюций основана на следующей теореме.

Теорема. Формула **G** является логическим следствием формул $F_1, F_2,..., F_k$ тогда и только тогда, когда множество формул $L = \{F_1, F_2,..., F_k, \neg G\}$ невыполнимо.

Множество формул $\{F_1, F_2,..., F_k\}$ называется *выполнимым*, если существует интерпретация **j** такая, что $j(F_1) = j(F_2) = ... = j(F_k) = 1$.

Текст теоремы может быть сформулирован следующим образом: если формулы $F_1,..., F_k$ истинны, то истинна и формула **G**.

Правило резолюций для высказываний

Задача логического вывода сводится к задаче проверки выполнимости множества формул $\{F_1, F_2,..., F_k, \neg G\}$. Чтобы установить невыполнимость такого множества формул, применяется определённое правило. В этом правиле используются следующие понятия:

- ***литерал*** – атомарная формула или её отрицание;
- ***дизъюнкт*** – дизъюнкция одного или нескольких литералов;
- ***пустой дизъюнкт*** – специальный дизъюнкт, не содержащий литералов. Для его обозначения используется специальный символ **■.** Пустой дизъюнкт ложен при любой интерпретации;
- ***противоположные литералы*** – литералы **X** и **¬X**.

Тогда *правило резолюции* заключается в следующем:

<u>из дизъюнктов (**X** v **F**) и (**¬X** v **G**) выводим дизъюнкт (**F** v **G**).</u>

Другими словами, дизъюнкт (**F** v **G**) является логическим следствием дизъюнктов (**X** v **F**) и (**¬X** v **G**).

Резолюция – это приём, используемый при достоверном логическом выводе. Этот приём заключается в нахождении двух дизъюнктов, один из которых содержит литеру, а другой – её отрицание. На основании их сравнения формируется новый дизъюнкт, называемый резольвентой. Отметим, что именно порождение новых дизъюнктов, являясь основой метода резолюций.

Уточним понятие резолютивного вывода. Пусть **S** – множество дизъюнктов. **Выводом** из **S** называется последовательность дизъюнктов $D_1, D_2,..., D_n$ такая, что каждый дизъюнкт этой последовательности принадлежит **S** или следует из предыдущих по правилу резолюции. Дизъюнкт **D** выводим из **S**, если существует вывод из **S**, последним дизъюнктом которого является **D**.

Алгоритм резолютивного вывода

Формально алгоритм резолютивного вывода состоит в доказательстве того, что формула **G** является логическим следствием множества формул $\mathbf{F_1},..., \mathbf{F_k}$ и включает следующую последовательность шагов.

1. Составляется множество формул $\{\mathbf{F_1},..., \mathbf{F_k}, \neg\mathbf{G}\}$.
2. Каждая из этих формул приводится к конъюнктивной нормальной форме (КНФ). В КНФ зачёркиваются знаки конъюнкции. Получается множество дизъюнктов **S**.

3. Осуществляется поиск вывода пустого дизъюнкта ■ из **S**. Если пустой дизъюнкт выводим из **S**, то формула **G** является логическим следствием формул $\mathbf{F_1},..., \mathbf{F_k}$. Если из **S** нельзя вывести ■, то **G** не является логическим следствием формул $\mathbf{F_1},..., \mathbf{F_k}$.

Задача. Пусть даны два утверждения:

1) $\mathbf{X_1 \to X_2}$,

2) $\mathbf{X_2 \to X_3}$.

Докажем, что утверждение $\mathbf{X_1 \to X_3}$ является логическим следствием из этих утверждений. Для этого составляем множество формул с отрицанием доказываемого высказывания:

$\{(\mathbf{X_1 \to X_2}), (\mathbf{X_2 \to X_3}), \neg(\mathbf{X_1 \to X_3})\}$.

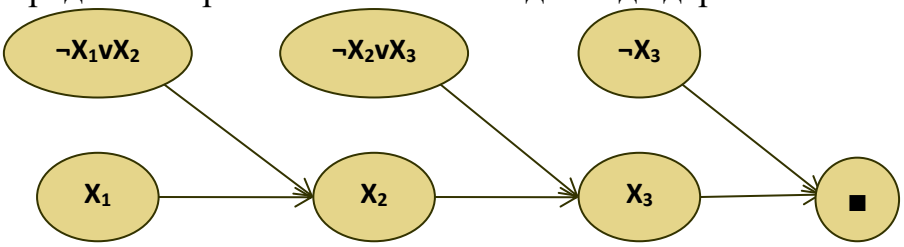
Устраняем импликации и приводим все формулы к КНФ:

$\{(\neg\mathbf{X_1} \vee \mathbf{X_2}), (\neg\mathbf{X_2} \vee \mathbf{X_3}), \neg(\neg\mathbf{X_1} \vee \mathbf{X_3})\} = \{(\neg\mathbf{X_1} \vee \mathbf{X_2}), (\neg\mathbf{X_2} \vee \mathbf{X_3}), (\mathbf{X_1} \& \neg\mathbf{X_3})\}$.

Зачёркивая конъюнкции, получим множество из четырёх дизъюнктов:

$\mathbf{S} = \{(\neg\mathbf{X_1} \vee \mathbf{X_2}), (\neg\mathbf{X_2} \vee \mathbf{X_3}), \mathbf{X_1}, \neg\mathbf{X_3}\}$.

Представим резолютивный вывод в виде дерева поиска:



Получен пустой дизъюнкт. Следовательно, утверждение $\mathbf{X_1 \to X_3}$ является логическим следствием утверждений $\mathbf{X_1 \to X_2}$ и $\mathbf{X_2 \to X_3}$

Задача. Даны следующие три высказывания:

1) если налоги в бюджет не собраны, то либо секвестрируется бюджет, либо правительство уходит в отставку;

2) если секвестируется бюджет, то падает уровень жизни;

3) налоги в бюджет не собраны.

Введем символические обозначения элементарных высказываний:

h – «налоги в бюджет не собраны»;

p – «бюджет секвестируется»;

q – «правительство уходит в отставку»;

r – «уровень жизни падает».

Справедливо или нет следующее заключение «*Либо падает уровень жизни, либо уровень жизни не падает и правительство уходит в отставку*»?

Для ответа на вопрос составьте множество формул с отрицанием доказываемого заключения. Примените алгоритм резолютивного вывода, представив его в виде вывода по дереву поиска.

Решение. $\mathbf{F_1 = h \rightarrow (p+q),}$

$\mathbf{F_2 = p \rightarrow r,}$

$\mathbf{F_3 = h,}$

$\mathbf{\neg G = \neg(r+(\neg r \& q)).}$

КНФ: $\mathbf{(\neg h+p+q), (\neg p+r), h, \neg r\&(r+\neg q).}$

$\mathbf{S = \{(\neg h+p+q), (\neg p+r), h, \neg r, (r+\neg q)\}.}$ *Вывод – пустой дизъюнкт.*

Ответ. ***Да, справедливо.***

Задача. Студент, изучающий МОиБО, высказывает следующие утверждения о трех своих знакомых девушках Свете, Даше, Маше:

1) Я люблю, по крайней мере, одну из этих трех девушек;

2) Если я люблю Свету, а не Дашу, то я также люблю Машу;

3) Я либо люблю Дашу и Машу, либо не люблю ни одну из них;

4) Если я люблю Дашу, то я также люблю Свету.

Справедливо или нет следующее заключение «Студент любит Дашу»?

Рекомендация. Для ответа на вопрос составьте множество формул с отрицанием доказываемого заключения. Примените алгоритм резолютивного вывода, представив его в виде вывода по дереву поиска.

$\mathbf{(C+M+D)\&(C\&\neg D \rightarrow M)\&(D\&M+\neg D\&\neg M)\&(D \rightarrow C\&\neg D).}$

КНФ: S = {C+M+D, ¬C+M+D, M+ ¬D, ¬M, D, ¬D+C, ¬D].}

Ответ. Да, студент любит Дашу.

Задача. Внимание трёх девушек Анны, Елены и Насти привлек проезжающий мимо автомобиль. Анна сказала: «Эта машина изготовлена в США, марка её – «Форд». Елена возразила: «По-моему, эта машина из Германии, её марка – «Мерседес». Настя добавила: «Марка машины – «Ауди», изготовлена в Германии». Оказалось, что каждая из трёх девушек права только в одном из своих высказываний.

Докажем, что марка автомобиля – «Форд», а изготовлен он в Германии. Для этого введем следующие обозначения высказываний:

R – автомобиль изготовлен в США;

G – автомобиль изготовлен в Германии;

F – марка автомобиля «Форд»;

M – марка автомобиля «Мерседес»;

A – марка автомобиля «Ауди».

Тогда высказывания трёх девушек с учётом того, что одно из них истинно, а одно ложно, записываются в виде следующих формул:

1) $\mathbf{R\&\neg F}$ v $\mathbf{\neg R\&F}$ – Анна,

2) $\mathbf{G\&\neg M}$ v $\mathbf{\neg G\&M}$ – Елена,

3) $\mathbf{A\&\neg G}$ v $\mathbf{\neg A\&G}$ – Настя.

Предположение, что проезжающая машина изготовлена в США или Германии и имеет одну из трёх марок, соответствует следующим формулам:

4) $\mathbf{R\&\neg G}$ v $\mathbf{\neg R\&G}$,

5) **F&¬M&¬A** v **¬F&M&¬A** v **¬F&¬M&A**.

Следует ли утверждение **F&G** из рассматриваемых пяти утверждений? Для ответа составляем формулы с отрицанием доказываемого высказывания:

S = {**(R&¬F** v **¬R&F)**, **(G&¬M** v **¬G&M)**, **(A&¬G** v **¬A&G)**, **(R&¬G** v **¬R&G)**, **(F&¬M&¬A** v **¬F&M&¬A** v **¬F&¬M&A)**, **¬(F&G)**}.

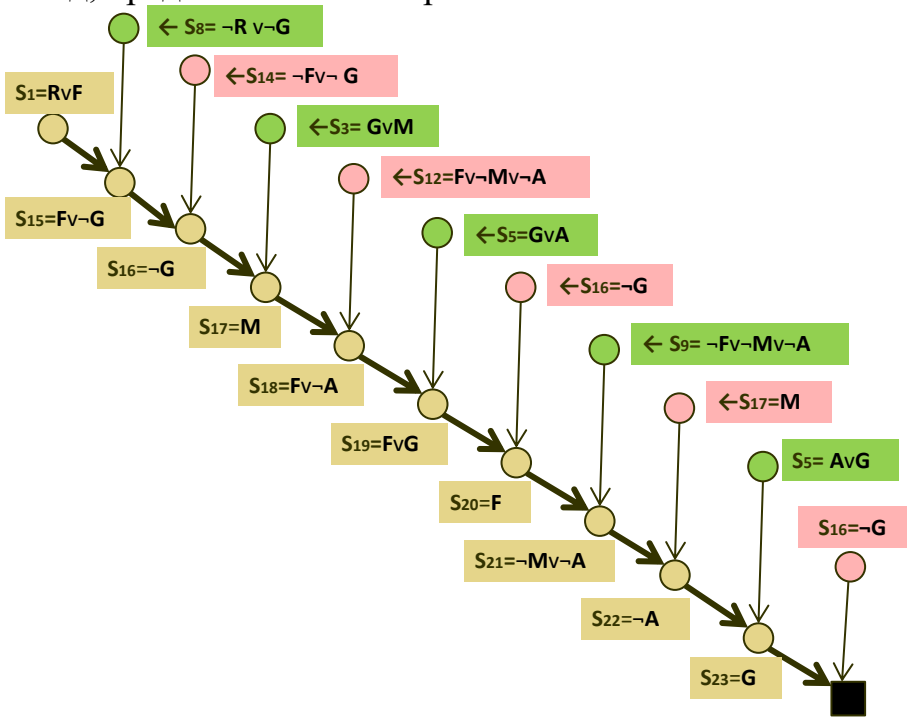
Приводим все формулы к КНФ:

{**(R**v**F)**&**(¬R**v**¬F)**, **(G**v**M)**&**(¬G**v**¬M)**, **(A**v**G)**&**(¬A**v**¬G)**, **(R**v**G)**&**(¬R**v**¬G)**, **(F**v**M**v**A)**&**(F**v**¬M**v**¬A)**&**(¬F**v**M**v**¬A)**&**(¬F**v**M**v**¬A)**&**(¬F**v**¬M**v**¬A)**, **(¬F**v**¬G)**}.

Зачёркивая конъюнкции, получаем множество S из 14 дизъюнктов:

S_1=Rv**F**, **S_2**= **¬R**v**¬F**, **S_3=G**v**M**,
S_4=¬Gv**¬M**, **S_5=A**v**G**, **S_6=¬A**v**¬G**,
S_7=Rv**G**, **S_8=¬R**v**¬G**, **S_9=F**v**M**v**A**,
S_{10}=Fv**¬M**v**¬A**, **S_{11}**= **¬F**v**M**v**¬A**, **S_{12}=¬F**v**M**v**¬A**,
S_{13}=¬Fv**¬M**v**¬A**, **S_{14}=¬F**v**¬G**.

С помощью метода *линейной резолюции* (он реализован в Прологе), попробуем найти ■, представив процесс резолютивного вывода в виде дерева поиска. В качестве исходного дизъюнкта возьмем дизъюнкт **S_1**. Линейный вывод имеет вид, представленный на рис:



Отметим также, что известное правило логического вывода «*modus ponens*» (если **X** – истина и из **X→G**, то **G** – истина) получается по правилу резолюции: из дизъюнктов (**X**) и (**¬X** v **G**) выводим дизъюнкт **G**. Кроме того, метод резолюций является обобщением метода доказательства от противного

Правила резолюции для логики предикатов. Метод резолюций применим также к логике предикатов 1-го порядка. Отличие от логики высказываний: в исчислении предикатов используется понятие *переменной*, а правило резолюций необходимо дополнить возможностью делать *подстановку* переменной.

Подстановкой называется множество равенств между переменными и термами. Подстановка *унифицирует* некоторое множество атомарных формул, если в результате подстановки все атомарные формулы множества становятся одинаковыми.

Если множество унифицируемо, то существует, как правило, не один унификатор этого множества, а несколько. Среди всех унификаторов данного множества с помощью специальных методов выделяют *наиболее общий унификатор* – это подстановка, которая делает элементы множества атомарных формул одинаковыми. В этой терминологии правило резолюций для предикатов 1-го порядка можно записать так: из дизъюнктов $(\mathbf{P(t_1,..., t_n)} \vee \mathbf{F})$ и $(\neg\mathbf{P(u_1,..., u_n)} \vee \mathbf{G})$ выводим дизъюнкт $(\mathbf{s(F)} \vee \mathbf{s(G)})$, где **s** – наиболее общий унификатор множества $\{\mathbf{P(t_1,..., t_n), P(u_1,..., u_n)}\}$ или *резольвента*.

В методе резолюций для логики предикатов, в отличие от логики высказываний, используются ещё два правила *склейки*:

1. Из дизъюнктов $(\mathbf{P(t_1,..., t_n)} \vee \mathbf{P(u_1,..., u_n)} \vee \mathbf{F})$ выводим дизъюнкт-склейка $(\mathbf{s(P(t_1,..., t_n))} \vee \mathbf{s(F)})$, где **s** – наиболее общий унификатор множества $\{\mathbf{P(t_1,..., t_n), P(u_1,..., u_n)}\}$.

2. Из дизъюнктов $(\neg\mathbf{P(t_1,..., t_n)} \vee \neg\mathbf{P(u_1,..., u_n)} \vee \mathbf{F})$ выводим дизъюнкт-склейка $(\mathbf{s(\neg P(t_1,..., t_n))} \vee \mathbf{s(F)})$, где **s** – наиболее общий унификатор множества $\{\mathbf{P(t_1,..., t_n), P(u_1,..., u_n)}\}$.

Задача. Даны два утверждения:

1) Некоторые пациенты любят своих докторов;

2) Ни один пациент не любит знахаря.

Справедливо или нет следующее заключение: «Никакой доктор не является знахарем»?

Рекомендация. Для ответа на вопрос, используя исчисление предикатов, составьте множество формул. Примените метод резолюций, представив его в виде вывода по дереву поиска.

Введем формулы для описания простых утверждений:

P(x) – x пациент,

D(y) – y доктор,

Z(y) – y знахарь.

Тогда посылки и заключение можно представить формулами:

$\mathbf{\exists x, y(P(x)\&D(y)\rightarrow L(x, y));}$

$\mathbf{\forall x(P(x)\rightarrow\forall y(Z(y)\rightarrow\neg L(x, y)));}$

$\mathbf{\forall y(D(y)\rightarrow\neg Z(y)).}$

Приведем первую и вторую формулы, а также отрицание третьей к сколемовской нормальной форме, получим:

$\mathbf{\exists x, y(\neg P(x)+ \neg D(y)+ \neg L(x, y));}$ $\mathbf{\forall x(\neg P(x)+ \neg Z(y)+\neg L(x, y));}$ $\mathbf{\exists y(D(y)\& \neg Z(y)).}$

Множество дизъюнктов

$\mathbf{S = \{(\neg P(x)+ \neg D(y)+ \neg L(x, y)), (\neg P(x)+ \neg Z(y)+\neg L(x, y)), H(a), D(y), Z(y).}$

Пустой дизъюнкт из множества **S** не выводится по дереву.

Ответ. Нет, несправедливо.

5-6. Методы правдоподобного вывода в интеллектуальных системах

Индуктивный вывод.

Индуктивное обучение предполагает вывод по обучающим примерам. Одним из алгоритмов индуктивного обучения является **ID3**. Рассмотрим пример его применения для задачи оценки риска по кредитной истории клиента, текущему долгу, гарантиям и доходу:

№	Риск	Кредитная история	Долг	Гарантии	Доход
1	Высокий	Плохая	Высокий	Нет	0 - 15
2	Высокий	Неизвестна	Высокий	Нет	15 – 35
3	Средний	Неизвестна	Низкий	Нет	15 – 35
4	Высокий	Неизвестна	Низкий	Нет	0 - 15
5	Низкий	Неизвестна	Низкий	Нет	Свыше 35
6	Низкий	Неизвестна	Высокий	Адекватные	Свыше 35
7	Высокий	Плохая	Низкий	Нет	0 - 15
8	Средний	Плохая	Низкий	Адекватные	Свыше 35
9	Низкий	Хорошая	Низкий	Нет	Свыше 35
10	Низкий	Хорошая	Высокий	Адекватные	Свыше 35
11	Высокий	Хорошая	Высокий	Нет	0 - 15
12	Средний	Хорошая	Высокий	Нет	15 – 35
13	Низкий	Хорошая	Высокий	Нет	Свыше 35
14	Высокий	Плохая	Высокий	Нет	15 – 35

Алгоритм и его модификации способны выполнять обобщение за счет поиска закономерностей в обучающих данных, строить по аналогии пояснения к обучающим примерам и т.п.

На наборе обучающих примеров можно построить несколько деревьев решений, позволяющих корректно классифицировать все примеры. Необходимо выбрать такое дерево, которое с наибольшей вероятностью позволит классифицировать неизвестные экземпляры. В ID3, предпочтение отдается простейшему дереву (принцип «бритвы Оккама»: глупо прилагать больше усилий, чем нужно для достижения цели). Дерево строится от корня сверху вниз. Вершина дерева - некоторое свойство, например доход.

Алгоритм рекурсивно строит поддерево для каждого значения (доход: 0-15, 15-35 или свыше 35). Процедура построения дерева длится до тех пор, пока все примеры не будут отнесены к одному из классов (конечные узлы дерева). В ID3 реализован критерий выбора каждого корневого узла дерева. Рассмотрим процесс построения дерева согласно таблице.

Вначале определим по формуле К.Шеннона информативность дерева с точки зрения оценки риска (высокий, средний, низкий), предполагая что все приведенные в таблице 14 примеров равновероятны (i=1,…14):

U(риск) = - $\sum_i$ p(mi)log$_2$p(m$_i$)= -6/14*log$_2$(6/14) -3/14*log$_2$(3/14) -5/14*log$_2$ (5/14) = 1,531 бит

Например, если корнем выбрать «***доход***», то все примеры будут разделены на 3 подмножества:

C$_1$={1,4,7,11}, **C$_2$**={2,3,12,14}, **C$_3$**={5,6,8,9,10,13}.

Тогда информация, необходимая для завершения построения дерева, составляет:

E[доход]=4/14*U(C1)+4/14*U(C2)+6/14*U(C3)==4/14*0+4/14*1+6/14*0,65 = **0,564 бит**,

а выбор в качестве корневой вершины дохода дает

Выигрыш[*доход*]=1,531- 0,564 = 0,967 бит.

Аналогично можно показать, что **Выигрыш [кредит. история]=0,266** бит, **Выигрыш [долг]**=0,581 бит, **Выигрыш [гарантии]**=0,576 бит. Поскольку доход обеспечивает наибольший информационный выигрыш, то именно он выбирается корневой вершиной алгоритма ID3. Это свойство удаляется из списка и алгоритм рекурсивно продолжает работу по выбору очередного корня поддерева. Тесты подтвердили хорошую работоспособность алгоритма не только на рассмотренном примере, но при решении сложной задачи классификации эндшпилей при игре в шахматы, в которой участвовали белые король и ладья и черные король и конь и где задачей было научиться распознавать позиции, приводящие к поражению черных за три хода. Кроме того, алгоритм ID3 и его модификации способны выполнять обобщение за счет поиска закономерностей в обучающих данных, строить по аналогии пояснения к обучающим примерам.

Альтернативными методами поиска являются эволюционные вычисления, логический вывод, нечеткая логика, вероятностные методы (байесовский классификатор).

Эвристический поиск. Алгоритм «первый лучший».

В качестве примера рассмотрим эвристический алгоритм «первый лучший» для головоломки «Восьмерка». Головоломка заключается в том, чтобы перемещая фишки по полю (3x3) перейти от исходной позиции к целевой за минимального числа перестановок. Эвристические процедуры поиска на графе стремятся к тому, чтобы минимизировать некоторую комбинацию стоимости пути к цели и стоимости поиска. Оценочная функция может быть разной:

$f_1 = \sum_{j=1}^{8} d_j$, где d_j - число ходов j-й фишки от текущей до целевой позиции;

$f_2 = \sum_{j=1}^{8} c_j$, указывает на число фишек, стоящих не на месте.

Ход в игре - перемещение фишки в соседнюю пустую клетку. Если пустая клетка находится в углу, то возможны 2 варианта хода, если на ребре, то – 3 варианта, если в центре, то – 4 варианта.

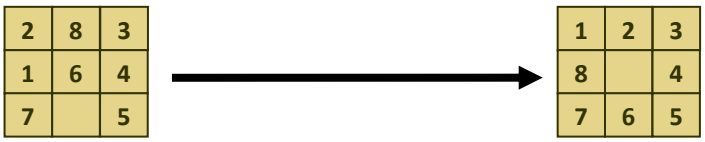
Выбор хода осуществляется по минимальному значению оценочной функции. Пусть применяется стратегия поиска в глубину, а оценочной функцией является f_1. Тогда алгоритм «первый лучший» состоит в следующем:

1. Вычисляется функция f_1 для начальной расстановки фишек (корень дерева поиска).

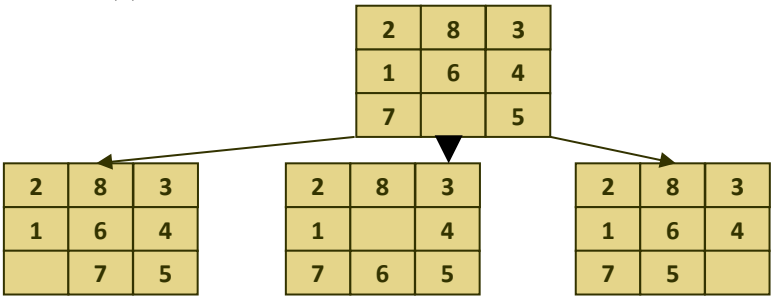
2. В зависимости от местоположения пустой клетки рассматривается несколько вариантов хода и для каждой позиции вычисляется функция f_1 (строим ветви дерева).

3. Выбираем ход (ветвь дерева) с **минимальным** значением f_1 (поиск в глубину дерева решений). Процесс повторяется (пп.1-2 алгоритма) до тех пор, пока f_1 не станет равным 0 (достигли целевой позиции).

Применим алгоритм для следующих исходной и целевой расстановки фишек:



Положение пустой клетки в исходной расстановке предусматривает 3 варианта хода:



f_1=6 f_1=4 f_1=6

Выбираем ход (ветвь дерева) с минимальным значением (f_1=4) и строим дерево решений в глубину. Повторяем процесс.

f_1=5 f_1=5 f_1=3 f_1=5

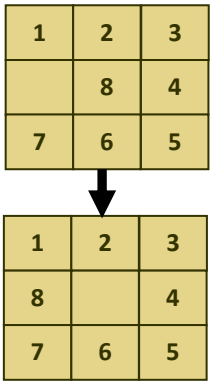
Выбираем ход с минимальным значением f_1=3 и продолжаем строить дерево решений в глубину.

f_1=4 f_1=2 f_1=4

Выбираем ход с минимальным значением (f_1=2) и строим дерево решений в глубину. Повторяем процесс.

f_1=1 f_1=3

Выбираем ход с минимальным значением (f_1=1), строим дерево решений в глубину и повторяем процесс.



Положение пустой клетки предусматривает 3 варианта хода. Однако первый ход приводит к результату $f_1 = 0$. Поиск завершается, целевое состояние достигнуто.

Сокращение перебора в задаче можно оценить количественно с помощью *критерия целенаправленности*:

P = L / T,

где **L** – длина пути от корня дерева до целевого состояния, **T** – число вершин дерева, построенных в процессе перебора (исключая корневую). Целенаправленность перебора в данной задаче равна **P=5/13=0,385**.

Вероятностный (байесовский) вывод.

В реальных условиях знания, которыми располагает человек, всегда в какой-то степени неполны, приближенны, ненадежны. Тем не менее, людям на основе таких знаний все же удается делать достаточно обоснованные выводы и принимать разумные решения. Следовательно, чтобы интеллектуальные системы были действительно полезны, они должны быть способны учитывать неполную определенность знаний и успешно действовать в таких условиях. Неопределенность (НЕ-факторы) может иметь различную природу.

Наиболее распространенный тип недостаточной определенности знаний обусловлен объективными причинами:

- действием случайных и неучтенных обстоятельств,
- неточностью измерительных приборов,
- ограниченными способностями органов чувств человека,
- отсутствием возможности получения необходимых свидетельств.

В таких случаях люди в оценках и рассуждениях прибегают к использованию вероятностей, допусков и шансов.

Идея байесовского подхода заключается в переходе от априорных знаний (или точнее незнаний) к апостериорным с учетом наблюдаемых явлений.

Особенности байесовского подхода:

•Все величины и параметры считаются случайными?

•Байесовские методы работают даже при объеме выборки 0! В этом случае апостериорное распределение равно априорному,

•В качестве оценок неизвестных параметров выступают апостериорные распределения, т.е. решить задачу оценивания некоторой величины, значит найти ее апостериорное распределение,

- Основным инструментом является формула Байеса.

Недостатки байесовского подхода:

- Принятие решения при использовании байесовских методов в нетривиальных случаях требует колоссальных вычислительных затрат,
- Фишером была показана оптимальность метода максимального правдоподобия, а следовательно — бессмысленность попыток придумать что-то лучшее.

В настоящее время наблюдается возрождение байесовских методов, которые оказались в состоянии решить многие серьезные проблемы статистики и машинного обучения. В отличие от нечеткой логики, байесовский подход теоретически обоснован и математически корректен.

Пример 1 (из области медицины). Пусть некий тест на какую-нибудь болезнь имеет вероятность успеха 95% (т.е. 5% — вероятность как позитивной, так и негативной ошибки). Всего болезнь имеется у 1% респондентов. Пусть некий человек получил позитивный результат теста (тест говорит, что он болен). С какой вероятностью он действительно болен? –

p = 0,95*0,01/(0,95*0,01+0,05*0,99) = 0,16.

Пример 2.

Проделка Фидо

Предположим, что вы оставили вашу собаку по кличке Фидо охранять ваш дом от грабителей, которые могут ворваться и украсть 10-фунтовый кусок мяса, размораживающийся на столе. Когда вы вернулись, все замки были в порядке, так что вы уверены, что никаких грабителей здесь не было. Однако мясо пропало. Само собой разумеется, что главный подозреваемый — Фидо.

На основе прошлого опыта, двух визитов к собачьему психиатру и хитрого взгляда вы оцениваете вероятность того, что это сделал Фидо, как 0,95. Однако, прежде чем обвинить Фидо, вы решаете получить еще одну улику. Вы готовите его обычный обед и предлагаете ему. К вашему удивлению, он съедает его до последней крошки. Едва ли этого можно ожидать от вора, который только что съел 10 фунтов мяса. Вы оцениваете вероятность того, что Фидо может сделать это, если он действительно съел мясо лишь в 0,02. Хотя обычно у него хороший аппетит и он съедает свой обед с вероятностью 0,99. Как вы должны пересмотреть ваши первоначальные подозрения, учитывая охотно съеденный обед? Очевидно, может оказаться полезной теорема Байеса. Учитывая только что съеденный обед, вероятность того, что Фидо виновен, можно выразить следующим образом:

$$P(\text{Виновен}\,|E) = \frac{P(E|\text{Виновен}) \times P(\text{Виновен})}{P(E|\text{Виновен}) \times P(\text{Виновен}) + P(E|\text{Невиновен}) \times P(\text{Невиновен})}.$$

Мы знаем, что

$P(\text{Виновен}) = 0{,}95;$
$P(\text{Невиновен}) = 0{,}05;$
$P(E|\text{Виновен}) = 0{,}02;$
$P(E|\text{Невиновен}) = 0{,}99.$

Поэтому

$$P(\text{Виновен}\,|E) = \frac{(0{,}02)(0{,}95)}{(0{,}02)(0{,}95) + (0{,}99)\,(0{,}05)} =$$

$$= \frac{0{,}0190}{0{,}0190+0{,}0495} =$$

$$= \frac{0{,}0190}{0{,}0685} =$$

$$= 0{,}28.$$

До эксперимента с обедом обстоятельства складывались не в пользу Фидо. Однако при помощи теоремы Байеса мы смогли учесть результаты эксперимента с обедом и заключить, что Фидо, скорее всего, невиновен. Всякий любитель собак может на этом примере увидеть полезность теоремы Байеса.

Теорема Байеса:

$$P(H:E) = \frac{P(E:H)P(H)}{P(E)}$$

Учитывая, что P(E) = P(E:H)P(H) + P(E:неH)P(неH) и P(неH) = 1- P(H), получаем формулу, позволяющую уточнять вероятность истинности проверяемой гипотезы H с учетом полученного свидетельства E:

$$P(H:E) = \frac{P(E:H)P(H)}{P(E:H)P(H) + P(E:неH)(1-P(H))}.$$

$$P(H_i:E) = \frac{P(E:H_i)\times P(H_i)}{\sum_{k=1}^{m} P(E:H_k)\times P(H_k)} \qquad (2)$$

$$P(H_i: E_1, E_2, \ldots, E_n) = \frac{P(E_1: H_i) \times P(E_2: H_i) \times \ldots \times P(E_n: H_i) \times P(H_i)}{\sum_{k=1}^{m} P(E_1: H_k) \times P(E_2: H_k) \times \ldots \times P(E_n: H_k) \times P(H_k)}$$

Пример. Предположим, что эксперт или ЛПР на основе трех экспериментов E1, E2 и E3 уточняет три состояния природы F1, F2, F3, не доверяя их априорным вероятностям. По результатам экспериментов эксперт также определяет условные вероятности каждого из трех фактов для каждого состояния q(Ej:Fi).

В таблице указаны все исходные значения вероятностей:

	F_1	F_2	F_3
$q(F_i)$	0,4	0,35	0,25
$q(E_1:F_i)$	0,3	0,8	0,5
$q(E_2:F_i)$	0,9	0	0,7
$q(E_3:F_i)$	0,6	0,7	0,9

Пусть вначале наблюдается факт E_3. Тогда согласно (12) имеем:

q (F_1:E_3) = 0,6×0,4/(0,6×0,4+0,7×0,35+0,9×0,25) = 0,34;

q (F_2:E_3) = 0,7×0,35/(0,6×0,4+0,7×0,35+0,9×0,25) = 0,34;

q (F_3:E_3) = 0,9×0,25/(0,6×0,4+0,7×0,35+0,9×0,25) = 0,32.

Состояния природы после проведения уточняющего эксперимента стали практически равновероятными.

Предположим теперь, что наблюдается факт E_1. По формуле (14) получим:

q (F_1:E_3,E_1) = 0,3×0.6×0.4/(0.3×0.6×0.4+0.8×0.7×0.35+0.5×0.9×0.25) = 0,19;

q (F_2:E_3,E_1) = 0,8×0,7×0,35/(0,3×0,6×0,4+0,8×0,7×0,35+0,5×0,9×0,25) = **0,52**;

q (F_3:E_3,E_1) = 0,5×0,9×0,25/(0,3×0,6×0,4+0,8×0,7×0,35+0,5×0,9×0,25) = 0,29.

Состояние F_2 после проведения уточняющего эксперимента E_1 стало наиболее вероятным.

Наконец, после проведения нового эксперимента и наблюдения по его результатам факта E_2, апостериорные вероятности, вычисленные по формуле (14), равны:

q(F_1:E_3,E_1,E_2) = 0,9×0,3×0,6×0,4/(0,9×0,3×0,6×0,4+0+0,7×0,5×0,9×0,25) =0,45;

q(F_2:E_3,E_1,E_2) = 0×0,8×0,7×0,35/(0,9×0,3×0,6×0,4+0+0,7×0,5×0,9×0,25) = 0;

q(F_3:E_3,E_1,E_2) =0,7×0,5×0,9×0,25/(0,9×0,3×0,6×0,4+0+0,7×0,5×0,9×0,25) =**0,55**.

При сравнении полученных апостериорных вероятностей с исходными априорными вероятностями удалось существенно уточнить вероятности состояний природы.

Процесс уточнения вероятности P(H) можно повторять снова и снова с привлечением все новых и новых свидетельств, каждый раз обращаясь к одной и той же формуле. В конечном счете, если свидетельств окажется достаточно, можно получить окончательный вывод об истинности (если окажется, что P(H) близка к 1) или ложности (если окажется, что P(H) близка к 0) гипотезы H.

Шансы и вероятности связаны между собой следующей формулой:

$$O(H) = \frac{P(H)}{1 - P(H)}$$

Нечеткий вывод.

Нечетким множеством, по Заде, называется множество, определенное на произвольном непустом множестве X как множество пар вида:

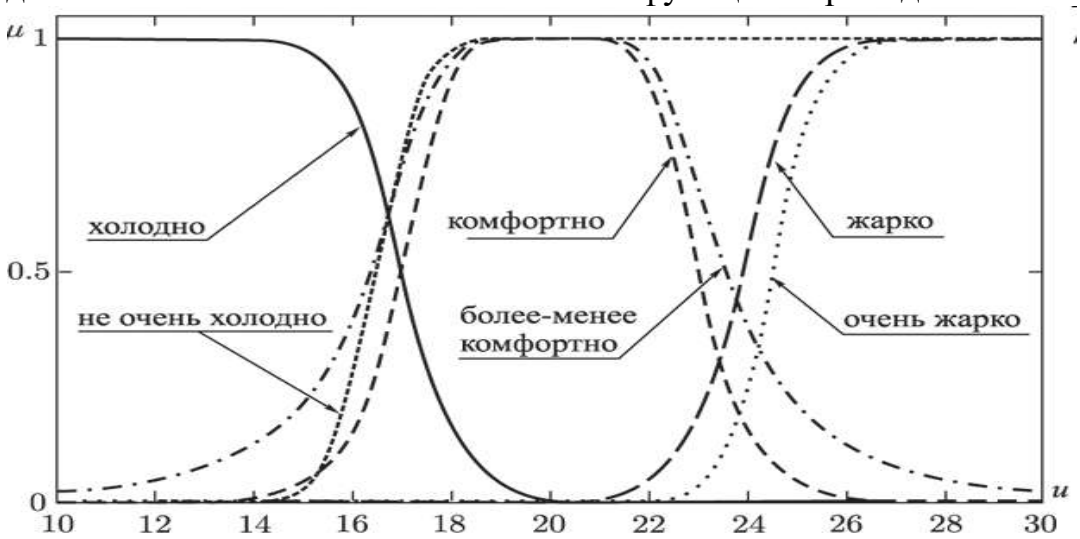
$$\tilde{A}=\{\mu_{\tilde{A}}(x)/x\}, \text{ где } x\in X,\ \mu_{\tilde{A}}(x)\in[0,1].$$

Пример. Пусть на множестве всех автомобилей задано нечеткое подмножество «быстрых авто»: Ã={(Porche/0.9), (BMW/0.5), (Lada/0.1)}. Тогда истинность высказывания (Porche/0.9)&¬(Porche/0.9) равна не 0, а 0.1! (смысл есть: это мера принадлежности данного авто к среднескоростным). Смысл есть и в простом высказывании (Porche/0.9), т.е. есть некоторая уверенность, что это не быстрое авто, например, он медленнее, чем авто F1.

Операции над нечеткими множествами. При графическом определении функций принадлежности объединенного множества необходимо в каждой точке множества выбрать максимальное значение из двух (точку того графика, который выше) и объединить все полученные точки в график, который и будет отображением новой функции принадлежности. Пересечение аналогично объединению, только выбирается минимальное значение в каждой точке. При построении дополнения необходимо зеркально отобразить график от оси, параллельной оси абсцисс и проходящей через точку 0,5 оси ординат.

Лингвистическая переменная - это переменная, значениями которой являются не числа, а слова или предложения естественного (или формального) языка.

Каждому значению лингвистической переменной соответствует определенное нечеткое множество со своей функцией принадлежности.



В целом схема процесса проектирования нечеткой системы в упрощенном виде представлена на рис. 27.

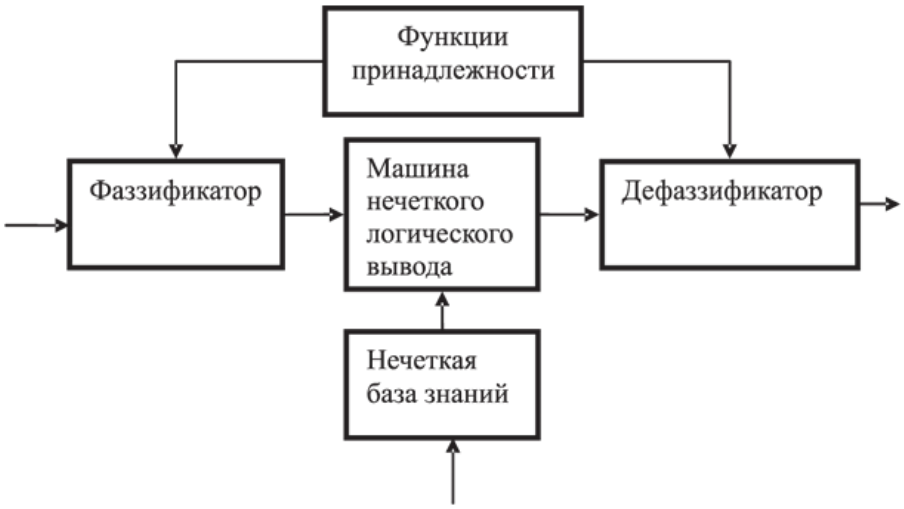


Рис. 27. Схема процесса нечеткого вывода в упрощенном виде.

Алгоритмы нечеткого вывода различаются главным образом видом используемых правил, логических операций и разновидностью дефазификации. Разработаны модели нечеткого вывода Мамдани, Сугено, Ларсена, Цукамото.

Рассмотрим пример.

Реактор, описывается тремя параметрами: температура, давление и расход рабочего вещества. Все показатели измеримы, и множество возможных значений известно. Из опыта работы с системой известны некоторые правила, связывающие значения этих параметров. Предположим, что сломался датчик, измеряющий значение одного из параметров системы, но знать его показания необходимо хотя бы приблизительно. Тогда встает задача об отыскании этого неизвестного значения (пусть это будет Давление) при известных показателях двух других параметров (Температуры и Расхода) и связи этих величин в виде следующих правил:

- если Температура низкая и Расход малый, то Давление низкое;
- если Температура средняя, то Давление среднее;
- если Температура высокая или Расход большой, то Давление высокое.

В нашем случае Температура, Давление и Расход — лингвистические переменные. Опишем каждую из них.

Температура. Универсум (множество возможных значений) – отрезок [0, 150]. Начальное множество термов {Высокая, Средняя, Низкая. Функции принадлежности термов имеют следующий вид (рис. 28):

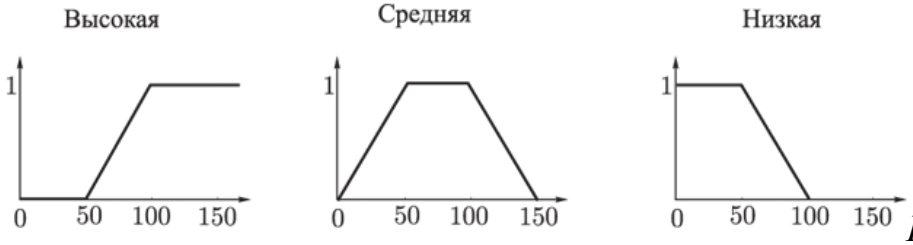


Рис. 28.

Давление. Универсум — отрезок [0, 100]. Начальное множество термов {Высокое, Среднее, Низкое}. Функции принадлежности термов имеют следующий вид:

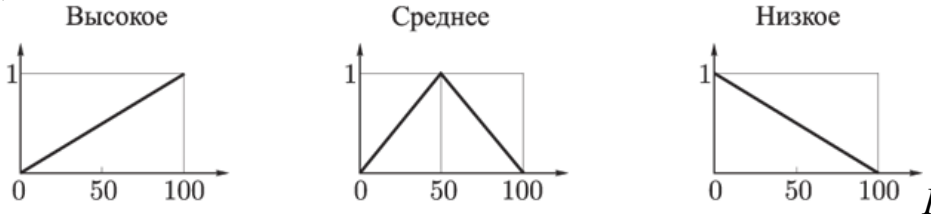


Рис. 29.

Расход. Универсум — отрезок [0, 8]. Начальное множество термов {Большой, Средний, Малый}. Функции принадлежности термов имеют следующий вид:

Рис. 30.

Пусть известны значения Температура=85 и Расход=3,5. Произведем расчет значения Давления.

Последовательно рассмотрим этапы нечеткого вывода.

Сначала по заданным значениям входных параметров найдем степени уверенности простейших утверждений вида (фаззификация), т.е. перейдем от заданных четких значений к степеням уверенности. Получаем следующие степени уверенности:

- Температура Высокая — 0,7;
- Температура Средняя — 1;
- Температура Низкая — 0,3;
- Расход Большой — 0;
- Расход Средний — 0,75;
- Расход Малый — 0,25.

Затем вычислим степени уверенности посылок правил:

- Температура низкая и Расход малый: min(Темп. Низкая, Расход Малый)= min(0.3, 0.25)=0.25;
- Температура Средняя: 1;
- Температура Высокая или Расход Большой: max(Темп. Высокая, Расход Большой)= max(0.7,0)=0,7.

Следует отметить также тот факт, что с помощью преобразований нечетких множеств любое правило, содержащее в левой части как конъюнкции, так и дизъюнкции, можно привести к системе правил, в левой части каждого будут либо только конъюнкции, либо только дизъюнкции. Таким образом, не уменьшая общности, можно рассматривать правила, содержащие в левой части либо только конъюнкции, либо только дизъюнкции.

Каждое из правил представляет нечеткую импликацию. Степень уверенности посылки мы вычислили, а степень уверенности заключения задается функцией принадлежности соответствующего терма. Поэтому,

используя один из способов построения нечеткой импликации, мы получим новую нечеткую переменную, соответствующую степени уверенности в значении выходных данных при применении к заданным входным соответствующего правила. Используя определение нечеткой импликации как минимума левой и правой частей (определение Мамдани), имеем результат, представленный на рис. 31.

Рис. 31.

Теперь необходимо объединить результаты применения всех правил (аккумуляция). Один из основных способов аккумуляции — построение максимума полученных функций принадлежности. Получим результат, представленный на рис. 32.

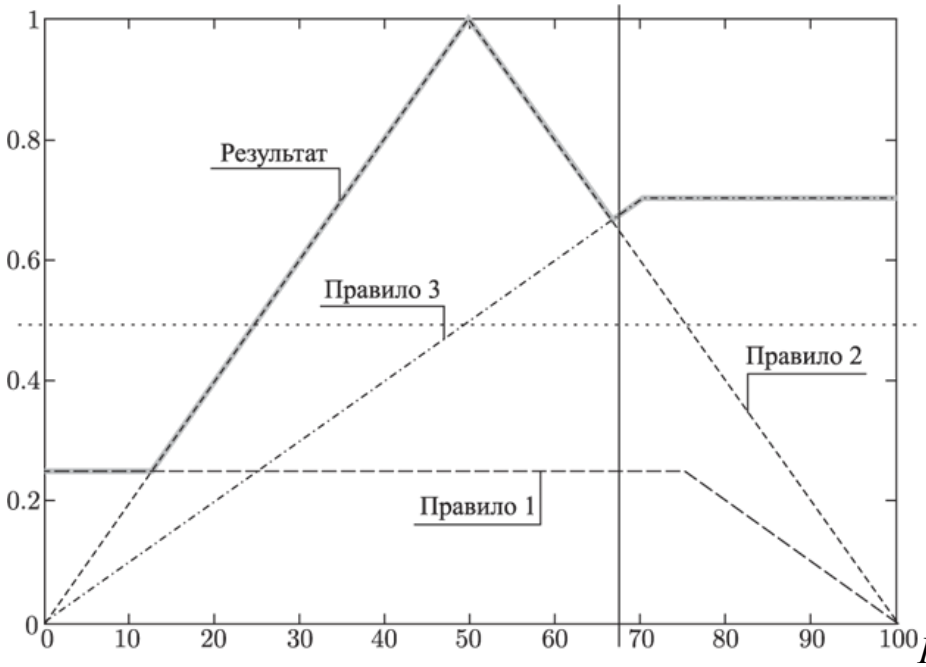


Рис. 32.

Полученную функцию принадлежности уже можно считать результатом. Это новый терм выходной переменной Давление. Его функция принадлежности говорит о степени уверенности в значении давления при заданных значениях входных параметров и использовании правил, определяющих соотношение входных и выходных переменных. Но обычно все-таки необходимо какое-то конкретное числовое значение. Для его получения используется этап дефаззификации, т.е. получения конкретного значения из универсума по заданной на нем функции принадлежности.

В нашем случае достаточно метода первого максимума. Применяя его к полученной функции принадлежности, получаем, что значение Давления — 50. Таким образом, если мы знаем, что температура равна 85, а расход рабочего вещества — 3,5, то давление в реакторе равно примерно 50.

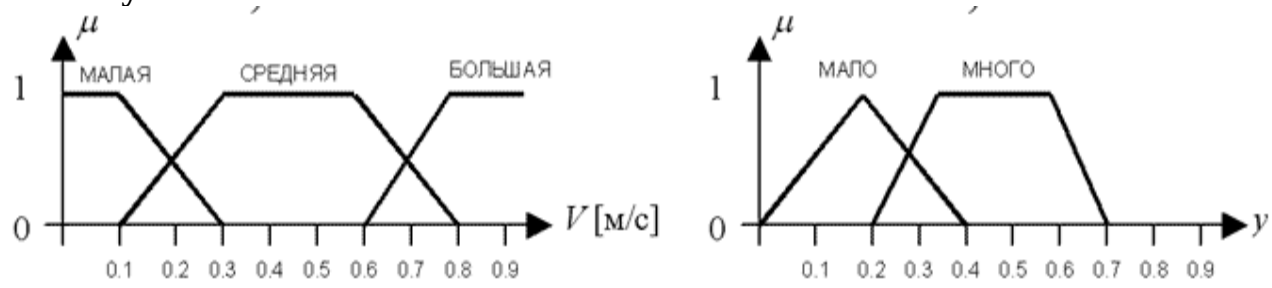
Задача «Управление скоростью»

Пусть задана следующая база правил по управлению скоростью:

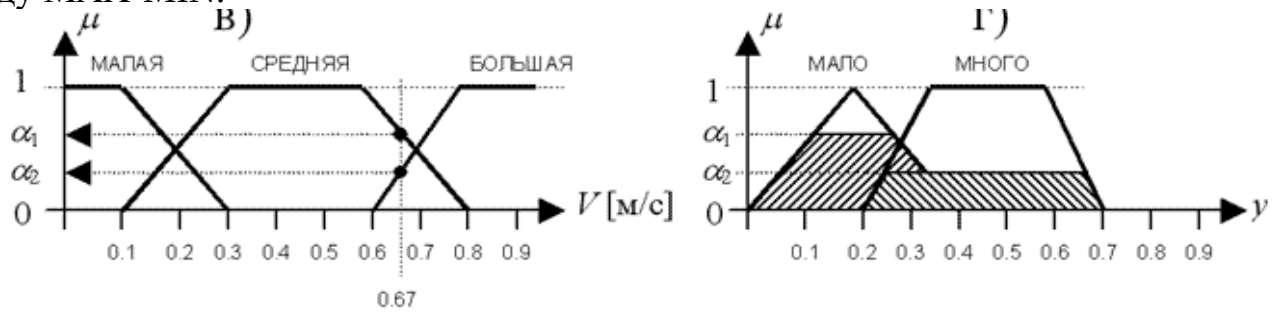
L1: ЕСЛИ **V=«СРЕДНЯЯ»**, ТО **y=«МАЛО»**,

L2: ЕСЛИ **V=«БОЛЬШАЯ»**, ТО **y=«МНОГО»**.

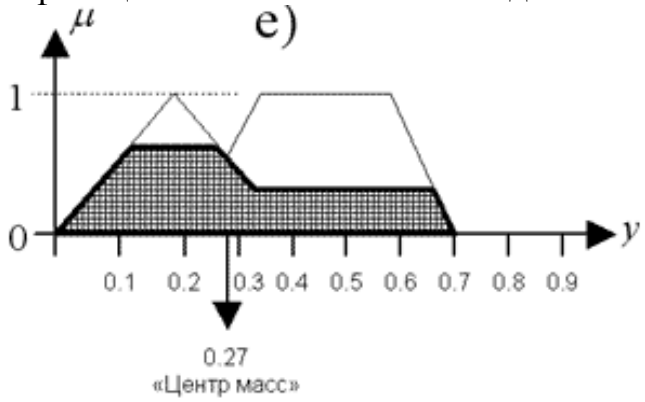
Функции принадлежности по входной переменной **V** и выходной переменной **y**:



Процесс фаззификации и процесс формирования выходных множеств по методу MAX-MIN:



После этого осуществляется операция дефаззификации. Чаще всего дефаззификация заключается в нахождении "центра масс" полученной фигуры.



Задача «Водитель-ДТП»

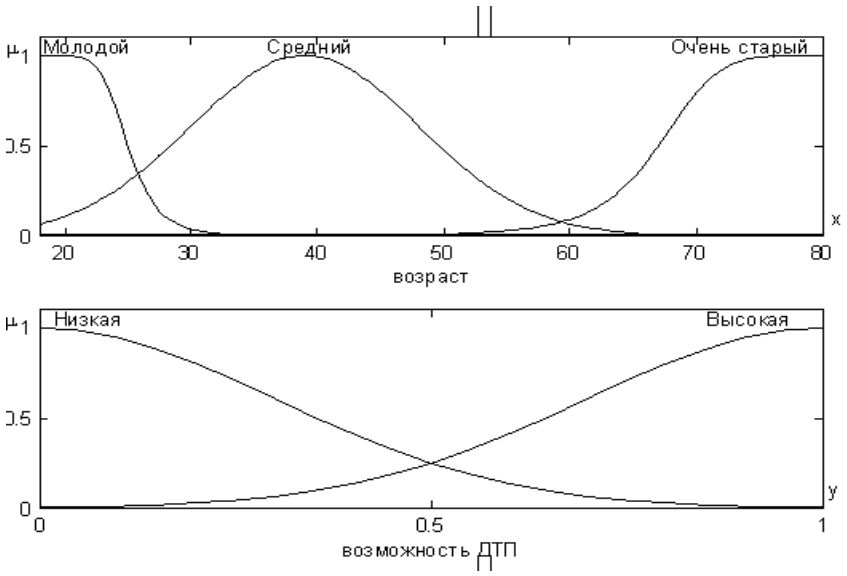
Нечеткая база знаний описывает зависимость между возрастом водителя (***x***) и возможностью дорожно-транспортного происшествия (***y***):

ЕСЛИ ***x*** = *Молодой,* ТО ***y*** = *Высокая;*

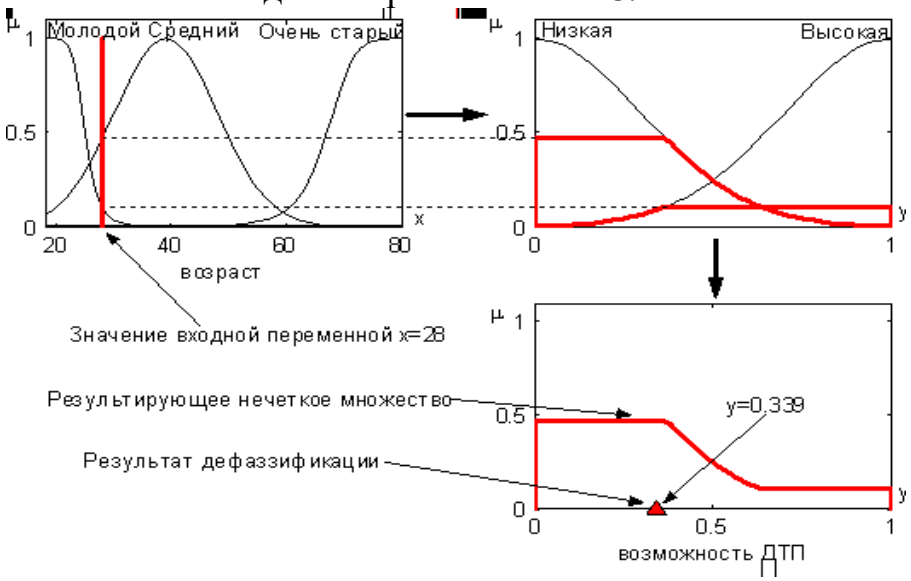
ЕСЛИ ***x*** = *Средний,* ТО ***y*** = *Низкая;*

ЕСЛИ ***x*** = *Очень старый,* ТО ***y*** = *Высокая.*

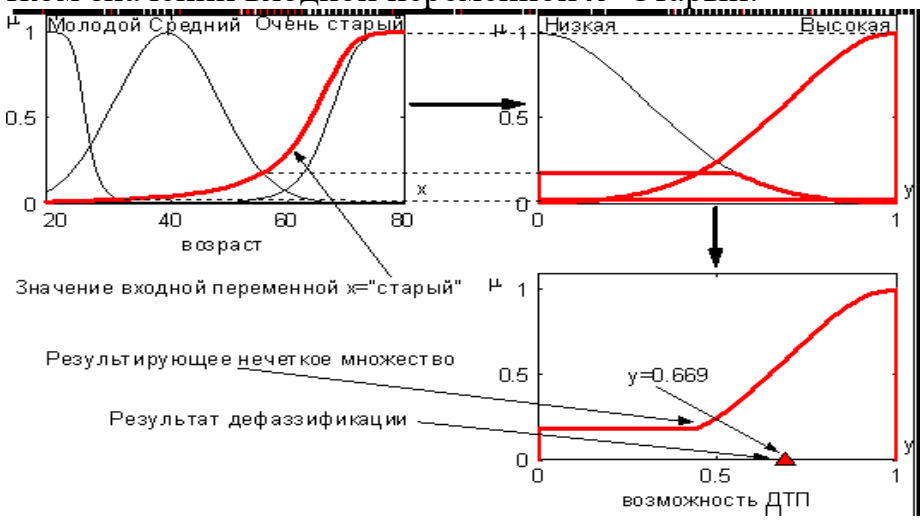
Пусть функции принадлежностей термов имеют следующий вид:



По имеющейся БЗ выполним нечеткий логический вывод по Мамдани при четком значении входной переменной ***x***=28.



По имеющейся БЗ выполним нечеткий логический вывод по Мамдани при нечетком значении входной переменной ***x***=Старый:



Нечеткий вывод по Сугено

Известна нечеткая база знаний:

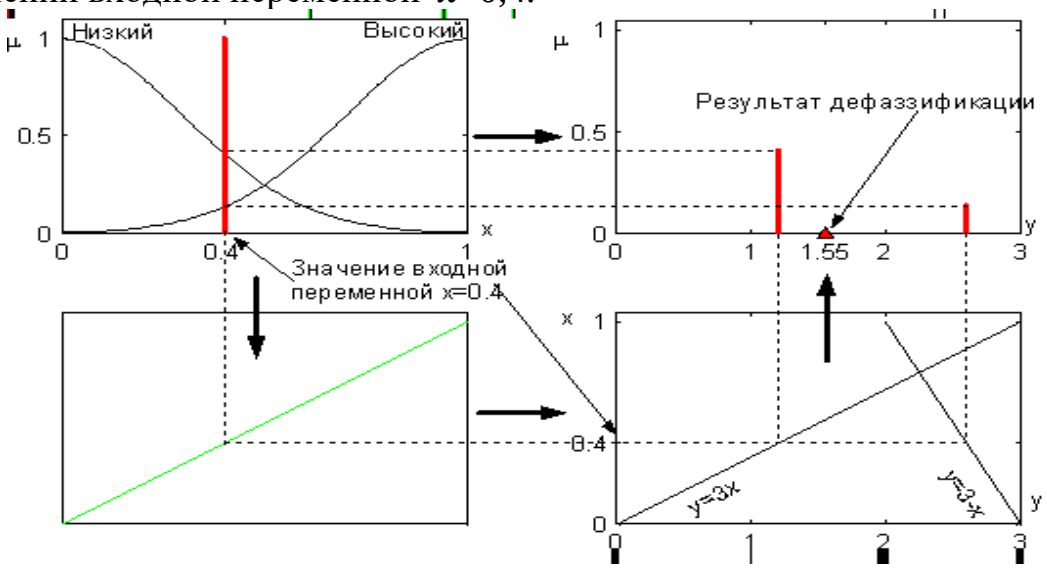
ЕСЛИ x = Низкий, ТО $y = 3x$;

ЕСЛИ x = Высокий, ТО $y = 3-x$.

Пусть функции принадлежностей термов имеют следующий вид:

$\mu_{низкий}(x)=\exp(-x^2/0{,}18)$, $\mu_{высокий}(x)=\exp(-(x-1)^2/0{,}18)$, $x \in [0, 1]$.

Необходимо выполнить нечеткий логический вывод по Сугено при значении входной переменной x=0,4.



При больших объемах выборки экспериментальных данных модели типа Сугэно обеспечивает, как правило, большую точность. Однако при этом возникают трудности с содержательной интерпретацией параметров нечеткой модели и с объяснением логического вывода.

С моделью типа Мамдани таких трудностей не возникает, ее параметры и после обучения легко интерпретируются содержательно. Процедура нечеткого логического вывода в модели типа Мамдани интуитивно понятна заказчикам нечетких моделей: технологам, экономистам, врачам, биологам.

С моделью типа Мамдани таких трудностей не возникает, ее параметры и после обучения легко интерпретируются содержательно. Процедура нечеткого логического вывода в модели типа Мамдани интуитивно понятна заказчикам нечетких моделей: технологам, экономистам, врачам, биологам.

Эволюционный вывод.

Рассмотрим пример эволюционного вывода по алгоритму генетического программирования.

Постановка задачи. Выбор математического выражения, наилучшим образом описывающего вычислительную модель для принятия решений. Решить задачу символьной регрессии означает получить математическое выражение (регрессию), которое можно анализировать, упрощать, использовать для моделирования, оптимизации и т.п.

Исходные данные. Экспериментальные данные или экспертные знания об объекте, отражающие неизвестную зависимость определенного свойства объекта **Y** от другого свойства или параметра **X** со случайной погрешностью, распределенной, как правило, по нормальному закону.

Найти: Регрессионную функцию, которая отображает зависимость **Y** от **X** с минимальной погрешностью, определив численные значения ее коэффициентов, т.е. идентифицировать объект. Например, найти функцию

Y = f(X) = ax+b

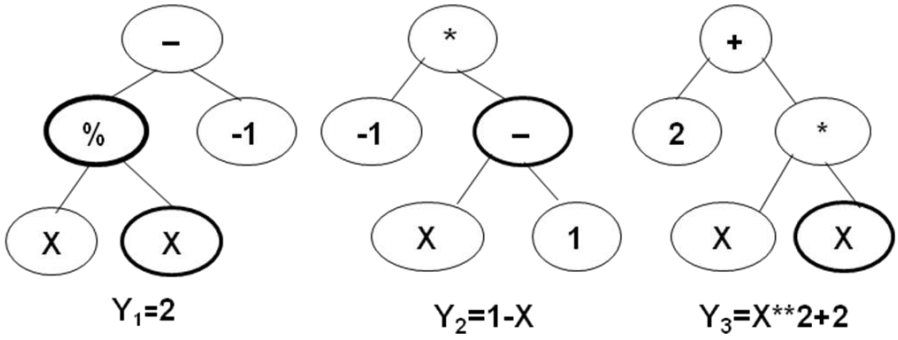
Пусть регрессионной функцией является квадратный полином вида

Y=X2–X+2**,

где **X** принимает значения в диапазоне **[-1, 1]**.

Необходимо с помощью ГП синтезировать компьютерную программу, реализующую данную регрессию.

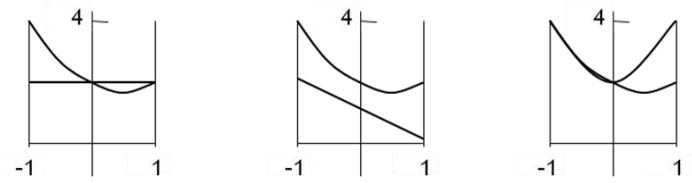
Для программного вычисления регрессии будем использовать множество математических операций: сложение (+), вычитание (–), умножение (*) и деление (%); - а также множество переменных и множество числовых констант из некоторого диапазона, например, от –2 до +2. Пусть случайно сгенерированы три программы:



Оценку программы, автоматически генерируемой с помощью генетических операторов, будем осуществлять по критерию близости к целевому значению функции получаемых программой результатов. Чем меньше ***среднеквадратичная ошибка***, тем лучше программа.

Тогда ошибка регрессии для каждой из трех хромосом совпадает с областями между кривыми на интервале [-1, 1]:

Y=x2–x+2**



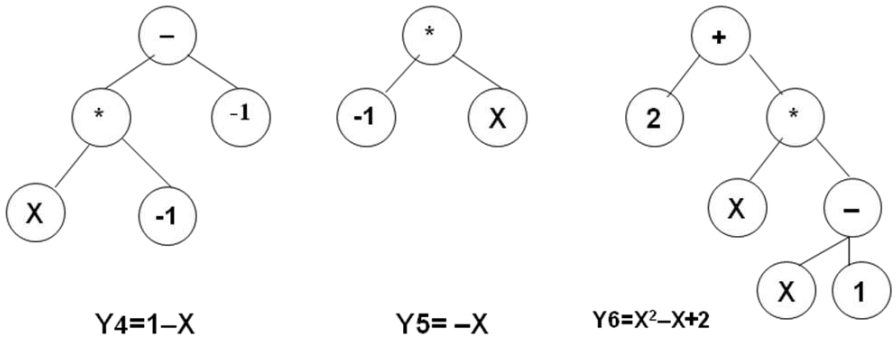
Абсолютная ошибка для Y_1 =**2** равна1.0; для Y_2 =**1**-x равна 1.33; для Y_3 =x**2+2 равна 1.0

Для формирования новой популяции хромосом используем операторы мутации и кроссинговера. Пусть **мутация** случайно заменяет в дереве, соответствующем $\mathbf{Y_1}$, вершину «**%**» на вершину «*****», а правую вершину «**X**» на вершину «**-1**». Получим регрессию $\mathbf{Y_4}$=1-x.

Кроссинговер между $\mathbf{Y_2}$ и $\mathbf{Y_3}$ дает два потомка:

$\mathbf{Y_5}$= - x и $\mathbf{Y_6}$=x**2 – x + 2

После выполнения операторов мутации и кроссинговера новые хромосомы для $\mathbf{Y_4}$, $\mathbf{Y_5}$ и $\mathbf{Y_6}$ будут иметь следующий вид:



Дерево ***справа*** эквивалентно исходной регрессии, т.е. является алгебраически правильным решением задачи автоматического синтеза компьютерной программы, реализующей вычисление функции квадратного полинома вида **Y=X**2 - X+2** в диапазоне значений **X** [-1,+1].

7-8. Методы распознавания образов

Теория распознавания образов базируется на ***гипотезе компактности***: образам соответствуют компактные множества («сгустки» точек), объединенные в классы, в пространстве признаков

Логическое распознавание

В реальных задачах, например, геологического и экономического прогнозирования, медицинской и технической диагностики имеет место следующая ситуация:

- число прецедентов (образов с известной классификацией) невелико,
- информации об их статистической природе недостаточно для обоснованного применения вероятностных моделей,
- сами прецеденты содержит разнородную или нечисловую информацию.

Однако известны логические связи между объектами и признаками.

Изображающие числа и базис.

Таблицу, которая представляет все возможные комбинации значений истинности набора переменных A, B, C,… называют ***базисом*** и обозначают ***b*[A, B, C,…]**.

Если значение «истина» обозначить 1, а значение «ложь» - 0, то для трех элементов **A, B, C** базис имеет вид:

	0 1 2 3	**4 5 6 7**
#A =	0 1 0 1	0 1 0 1
#B =	0 0 1 1	0 0 1 1
#C =	0 0 0 0	1 1 1 1

Строки базиса называются ***изображающими числами*** соответствующих элементов.

Восстановление булевой функции по изображающему числу

Чтобы по данному изображающему числу восстановить БФ в СДНФ, нужно суммировать элементарные произведения, составленные из всех элементов базиса или их отрицаний, изображающие числа которых имеют 1 в тех же разрядах, что и изображающее число БФ. Например, пусть дано изображающее число **#X** = 1001 0110 и базис ***b*[A, B, C]**:

	0 1 2 3	**4 5 6 7**
#X =	1 0 0 1	0 1 1 0
#A =	0 1 0 1	0 1 0 1
#B =	0 0 1 1	0 0 1 1
#C =	0 0 0 0	1 1 1 1

Оно имеет единицы в разрядах **0**, **3**, **5**, **6**.

Поэтому в СДНФ:

#X=10010110= **#(¬A¬B¬C+AB¬C+A¬BC+¬ABC)**.

Установление зависимости БФ. В процессе распознавания необходимо устанавливать имеются или нет логические связи между объектами и признаками. Для этого надо установить зависимы или нет БФ?

Пусть даны БФ $\mathbf{f_1(A,B,C,\ldots),\ldots, f_n(A,B,C,\ldots)}$. Надо вначале вычислить их изображающие числа $\mathbf{\#f_1,\ldots,\#f_n}$. Затем проверить, образуют ли столбцы набора

этих чисел $\mathbf{2^n}$ чисел 0,1,…, 2^{n-1}. Если все $\mathbf{2^n}$ чисел присутствуют, то ***функции независимы***, в противном случае – ***зависимы*** (разряды двоичных чисел, представленных столбцами изображающих чисел, возрастают вдоль столбца снизу вверх).

Пример 1. Установить зависимы или нет БФ $\mathbf{f_1=AB+\neg A\neg B}$ и $\mathbf{f_2=\neg B}$.

Запишем их изображающие числа в последовательные строки:

$$\underline{3\ 2\ 0\ 1}$$
$$\#\mathbf{f_1} = \#(\mathbf{AB+\neg A\neg B}) = 1\ 0\ 0\ 1$$
$$\#\mathbf{f_2} = \#(\neg\mathbf{B}) = 1\ 1\ 0\ 0$$

Колонки набора представляют все $2^2=4$ комбинации значений истинности, соответствующие числам 0, 1, 2, 3. Следовательно, функции $\mathbf{f_1}$ и $\mathbf{f_2}$ ***независимы***.

Пример 2. Установить зависимы или нет БФ $\mathbf{f_1=AB+\neg A\neg B}$, $\mathbf{f_2=\neg B}$ и $\mathbf{f_3=A\neg B+\neg AB}$. Запишем их изображающие числа:

$$\underline{3\ 6\ 4\ 1}$$
$$\#\mathbf{f_1} = \#(\mathbf{AB+\neg A\neg B}) = 1\ 0\ 0\ 1$$
$$\#\mathbf{f_2} = \#(\neg\mathbf{B}) = 1\ 1\ 0\ 0$$
$$\#\mathbf{f_3} = \#(\mathbf{A\neg B+\neg AB}) = 0\ 1\ 1\ 0$$

Эти функции ***зависимы***, т.к. в колонках набора содержатся числа 1, 3, 4, 6, а числа 0, 2, 5, 7 отсутствуют.

Если БФ зависимы, то как это установить? Возникает вопрос: если функции зависимы, то как найти явный вид неизвестной зависимости $\mathbf{F(f_1,\ldots,f_n)}$? – Для этого в базисе ***b***[**A,B,C,**…] выписываем изображающие числа $\#\mathbf{f_1}$,…,$\#\mathbf{fn}$ и определяем, какие числа отсутствуют в колонках. В изображающем числе $\#\mathbf{F(f_1,\ldots,fn)}$ в базисе ***b***[$\mathbf{f^1}$**,…,fn**] в разрядах, которые имеют номера отсутствующих чисел, поставим 0, а в остальных 1.

Пример. Требуется установить явный вид логической зависимости функций $\mathbf{f_1=\neg AC+\neg B\neg C}$, $\mathbf{f_2=\neg A\neg C+\neg BC}$ и $\mathbf{f_3=\neg B}$.

Запишем их изображающие числа в последовательные строки в базисе ***b*** [**A,B,C**]:

$$\underline{\mathbf{7\ 5\ 2\ 0\ \ 7\ 6\ 1\ 0}}$$
$$\#\mathbf{f_1} = \#(\mathbf{\neg AC+\neg B\neg C}) = 1\ 1\ 0\ 0\ \ 1\ 0\ 1\ 0$$
$$\#\mathbf{f_2} = \#(\mathbf{\neg A\neg C+\neg BC}) = 1\ 0\ 1\ 0\ \ 1\ 1\ 0\ 0$$
$$\#\mathbf{f_3} = \#(\neg\mathbf{B}) = 1\ 1\ 0\ 0\ \ 1\ 1\ 0\ 0$$

В колонках набора имеются только числа 0,1,2,5,6 и 7, а числа **3** и **4** отсутствуют.

Это означает, что по отношению к базису ***b***[**f1,f2,f3**] изображающее число неизвестной связи **F(f1,f2,f3)** = 1 имеет вид:

$\#\mathbf{F(f1,f2,f3)} = 1110\ 0111 = \#\mathbf{[(\neg f1+\neg f2)\neg f3+(f1+f2)f3}]$,

т.е. функции **f1**, **f2** и **f3** связаны соотношением $\mathbf{(\neg f1+\neg f2)\neg f3+(f1+f2)f3} = 1$.

В справедливости полученной зависимости легко убедиться, если подставить в нее выражения для **f1, f2** и **f3**:

$\mathbf{(\#\neg f1 + \#\neg f2)\ (\#\neg f3\ ) + (\#\ f1 + \#\ f2)\ (\#\ f3\ ) =}$

=(00110101+01010011) 00110011+(11001010+10101100) 11001100 =

=(01110111)(00110011)+(11101110)(11001100) = 00110011+11001100 = 1111 1111.

Оказалось, что решение многих задач логического распознавания, может быть сведено к нахождению решений булевых уравнений с одним или несколькими неизвестными.

Решение булевых уравнений с одним неизвестным

Задача. Пусть в процессе решения задачи логического распознавания на основе анализа трех объектов были установлены следующие логические зависимости между тремя признаками объектов **A, B, C**:

X(A+B)=ABC, (*)

где **X** – некоторая булева функция, которую необходимо найти и которая зависит от **A,B,C**. Причем функция **X** должна быть такой, чтобы при её подстановке в исходное уравнение, оно превращалось в тавтологию.

Решение. Чтобы решить задачу найдем изображающие числа для элементов, входящих в булево уравнение.

C	**B**	**A**	**#ABC**	**#(A+B)**	**#X**
0	0	0	**0**	**0**	×
0	0	1	**0**	**1**	**0**
0	1	0	**0**	**1**	**0**
0	1	1	**0**	**1**	**0**
1	0	0	**0**	**0**	×
1	0	1	**0**	**1**	**0**
1	1	0	**0**	**1**	**0**
1	1	1	**1**	**1**	**1**

Изображающее **X** число должно быть таким, чтобы (#**X**)*01110111= 0000 0001.

Отсюда #**X**=×000 ×001 и уравнение (*) имеет 4 решения, соответствующих изображающим числам:

(0000 0001), (1000 0001), (0000 1001) и (1000 1001), т.е.

$\mathbf{X_1=ABC}$, $\mathbf{X_2=ABC+\neg A\neg B\neg C}$, $\mathbf{X_3=C(AB+\neg A\neg B)}$, $\mathbf{X_4=\neg A\neg B+ABC}$.

Аналогично решаются ***системы булевых уравнений***.

Постановка задачи об НЛО

Некоторые жители двух близлежащих поселков утверждали, что они ночью наблюдали ***НЛО***. Чтобы проверить это среди них был проведен опрос. После систематизации ответов были получены следующие утверждения жителей поселка ***I***:

1) в небе появились сгустки слабо ионизированной светящейся пыли (**A**);

2) среди атмосферных облаков (**B**) появилось одиночное светлое дискообразное тело (**X**) радиусом около 100 метров и не было других тел (**⌐Y**);

3) далеко в небе показалась группа движущихся сигарообразных тел (**Y**), движение которых сопровождалось разрядами электричества (**C**).

Утверждения жителей поселка ***II*** сводились к следующему:

4) не было ни сгустков слабо ионизированной светящейся пыли (⌐**A**), ни одиночного светлого дискообразного тела (⌐**X**), ни группы других тел (⌐**Y**);

5) не видели ни облаков (⌐**B**), ни одиночного светлого дискообразного тела (⌐**X**);

6) наблюдались разряды электричества (**C**);

7) среди облаков (**B**) наблюдались сгустки светящейся пыли (**A**).

Необходимо на основании этих данных определить, следует ли серьезно отнестись к заявлениям очевидцев о наблюдении ***НЛО*** или это может быть объяснено атмосферными явлениями?

Решение системы булевых уравнений

Используя введенные обозначения, составим булево уравнение с 2 неизвестными (**X** и **Y**):

A + BX⌐Y + CY = ⌐A ⌐X ⌐Y + ⌐B ⌐X + C + AB (**)

Решение вопроса о существовании тел **X** и **Y** сводится к определению возможности разрешить уравнение (**) относительно **X** и **Y** и выразить их как функции от **A, B, C**.

Если решение **X(A,B,C), Y(A,B,C)** существует и, кроме того,

X ≠ ⌐A⌐B⌐C, Y ≠ ⌐A⌐B⌐C (***)

то нет оснований говорить о реальном появлении тел **X** и **Y**. Если же решения, отличного от (***), не существует, то к заявлениям «очевидцев» следует отнестись с большой серьезностью.

Решим уравнение (**). По определению операции эквивалентности (a=b ≡ ab+¬a¬b) уравнение записывается в виде:

(A+BX⌐Y+CY)&(⌐A⌐X⌐Y+⌐B⌐X+C+AB)+
+⌐(A+BX⌐Y+CY)&⌐(⌐A⌐X⌐Y+⌐B⌐X+C+AB) = 1

После упрощения получаем следующее уравнение:

A⌐B⌐X + AC + AB + BCX⌐Y + CY + ⌐AB⌐CY + ⌐A⌐CXY + ⌐A⌐B⌐CX = 1

Базис ***b***[**A,B,C,X,Y**] для уравнения:

#A= 1 1 1 × × 0 0 0
#B= 0 × 1 1 × 1 × 0
#C= × 1 × 1 1 0 0 0
#X= 0 × × 1 × × 1 1
#Y= × × × 0 1 1 1 0

Это уравнение имеет **3072** различных пар решений. Например

#X =1000 0010 = #(⌐A⌐B⌐C + ⌐ABC),
#Y = 0010 100 = #(¬AB¬C + ¬A¬BC).

Анализ всех решений показал - нет оснований считать, что **X** и **Y** действительно существуют.

Постановка задачи о выработке научно-технической политики фирмы.

Эксперты определили пять основных целей фирмы:

1) Повышение эффективности производства (**A1**);

2) Развитие производственных технологий (**A2**);

3) Достижение обоснованного уровня з/платы работников (**A3**);

4) Рост научно-технического потенциала фирмы (**A4**);

5) Сохранение и оздоровление окружающей среды (**A5**).

Все эти цели не являются независимыми. Связи между целями эксперты выразили с помощью 4 логических уравнений:

A5→A2A4 ≡ ¬A5 + A2A4 = 1 (1)

т.е. сохранение окружающей среды невозможно без развития производственных технологий и роста н/т потенциала фирмы.

¬A1¬A2¬A4→ ¬A3 ≡ A1 + A2 + A4 +¬A3 = 1 (2)

т.е. достижение обоснованного уровня з/п работников невозможно без повышения эффективности производства, развития производственных технологий и роста н/т потенциала фирмы.

A1 → A2 ≡ ¬A1 + A2 = 1 (3)

т.е. повышение эффективности производства невозможно без развития производственных технологий.

A2 → A4 ≡ ¬A2 + A4 = 1 (4)

т.е. развитие производственных технологий невозможно без роста научно-технического потенциала фирмы.

Перемножив уравнения (1-4) и упростив выражение получим:

A2A4 + ¬A1A4¬A5 + ¬A1¬A2¬A3¬A5 = 1

Отсюда получаем базис:

#A1 = × 0 0 1 1 1 1 0 0 0 0 0 0 0 0
#A2 = 1 × 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0
#A3 = × × 0 1 1 0 0 1 1 0 0 1 0 1 0 0 0
#A4 = 1 1 × 1 1 1 1 1 1 1 1 1 1 1 1 0
#A5 = × 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0

Из 32 возможных комбинаций исходов целей допустимыми (совместными по связям) являются только 11. Например, **A1A2¬A3A4¬A5** означает совместимость целей, связанных с повышением эффективности производства (**A1**), с развитием производственных технологий (**A2**) и ростом научно-технического потенциала фирмы (**A4**).

Структурные методы распознавания

Зачастую на практике информация о распознаваемых объектах содержится в записях сигналов. Традиционно для определения признаков используют разложения сигналов в ряды по ортогональным функциям (ряды Фурье, полиномы Эрмита, Чебышева и т.д.).

Возможно использование в качестве характерных признаков точек минимума, максимума и др. (например, электрокардиограммы):

Методы распознавания, использующие признаки структурных элементов позволяют избежать не только трудоемкой процедуры разложения в ряды, но и потерь информации. Структурные методы основаны на **теории формальных языков Н.Хомского**.

Идея, как формальных языков, так и структурных методов распознавания состоит в построении сложного объекта в виде иерархической ***регулярной*** структуры более простых образов. При этом используется известная схема: задаются исходные объекты и порождаются новые по некоторым правилам.

Основные понятия формальных грамматик

Грамматика включает ***терминальный*** и ***нетерминальный словари***, ***начальный символ*** и ***набор правил подстановки***.

Терминальный словарь – это набор исходных элементов, из которых строят цепочки, порождаемые грамматикой.

Нетерминальный словарь – это набор символов, которыми обозначаются классы исходных элементов, а также специальные вспомогательные символы.

Начальный символ – это выделенный нетерминальный символ, обозначающий класс объектов, для описания которых предназначена грамматика.

Правила подстановки – это выражения вида «***x→y***» (заменить ***x*** на ***y***), где ***x*** и ***y*** – цепочки с любыми терминальными или нетерминальными символами. Если имеется последовательность цепочек ***x0, x1,…, xn***, в которой каждая последующая выводима из предыдущей, то она называется ***выводом xn*** из ***x0*** в грамматике ***Г***.

Грамматика – это не алгоритм, поскольку порядок применения правил подстановки произволен!

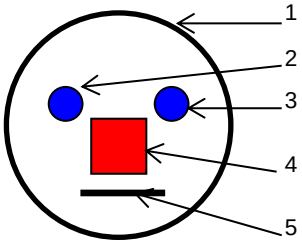
Совокупность всех терминальных цепочек, выводимых из начального символа, называется ***языком, порожденным грамматикой Г***, и обозначается ***L(Г)***. Две разные грамматики могут порождать один и тот же язык.

Как реализовать процесс структурного распознавания?

1. Построить адекватное описание объектов распознавания
2. Выбрать грамматики
3. Реализовать процесс распознавания посредством процедур синтаксического анализа
4. Использовать обучение для вывода грамматик
5. Применить в рамках структурного подхода другие методы распознавания.

Пример структурного распознавания

Пусть дана структура объекта распознавания, состоящего из **5** ***терминальных элементов***:



Необходимо, используя ***предикаты***, характеризующие взаимное расположение элементов (*слева/справа, выше/ниже, внутри*) и ***алгоритм вывода грамматик***, построить два различных структурных описания объекта.

Терминальными элементами объекта будем считать:

1 – окружность; ***2*** и ***3*** – ***точки***; ***4*** – ***квадрат***; ***5 -отрезок прямой***.

Зададим следующие ***предикаты***:

СЛЕВА (X,Y) – принимает значение истина, если X находится *слева* от объекта Y;

ВЫШЕ (X,Y) – принимает значение истина, если X находится *выше* Y;

ВНУТРИ (X,Y) – принимает значение истина, если X находится *внутри* Y.

Алгоритм вывода грамматики:

1. Определить все случаи, когда терминальные элементы удовлетворяют одному из заданных предикатов. Каждый истинный предикат образует новый элемент.
2. Построить новые элементы из исходных и вновь полученных.
3. Повторять процедуру до тех пор, пока возможности построения новых элементов не будут исчерпаны.
4. Удалить все элементы, не являющиеся компонентами исходного входного объекта.

Реализация алгоритма вывода грамматики для примера.

Построим новые элементы, предикаты которых истинны:

Цикл1. **6** - ВЫШЕ(4,5); **7** - ВНУТРИ(2,1); **8** - ВНУТРИ(3,1);

9 - ВНУТРИ(4,1); **10** - ВНУТРИ(5,1); **11** - СЛЕВА(2,3);

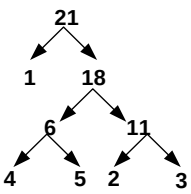
12 - СЛЕВА(2,3) ВНУТРИ(2,1) ВНУТРИ(3,1);

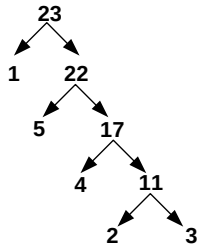
13 - ВЫШЕ(4,5) ВНУТРИ(4,1) ВНУТРИ(5,1);

Цикл2. **14** - ВНУТРИ(6,1); **15** - ВНУТРИ(11,1); **16** - ВЫШЕ(11,4);

17 - ВЫШЕ(11,4); **18** - ВЫШЕ(11,6); **19** - ВНУТРИ(16,1); **20** - ВНУТРИ(17,1); **21*** - ВНУТРИ(18,1); **22** - ВЫШЕ(17,5); **23***- ВНУТРИ(22,1)

Элементы, отмеченные звездочкой (**21** и **23**), соответствуют двум разным структурам описания исходного объекта. Представим их в виде деревьев:





Получены 2 разных структурных описания исходного образа.

9. Методы интеллектуального анализа данных и прогнозирования

Машинное обучение – это область ИИ, которая разрабатывает алгоритмы для выявления компьютером закономерностей в данных. Процесс машинного обучения представляет собой два последовательных этапа.

На первом этапе алгоритм машинного обучения применяется к набору данных для выявления в нем закономерностей. Сами закономерности могут быть представлены разными способами. Сейчас наиболее популярными из них являются *деревья решений, регрессионные модели и нейронные сети*. Проще говоря, все алгоритмы машинного обучения создают модели из данных, но каждый из них разработан для создания моделей, использующих определенный тип представления.

На втором этапе, когда модель создана, она применяется для анализа.

Большинство алгоритмов машинного обучения можно отнести либо к *обучению с учителем*, либо к *обучению без учителя*.

Цель обучения с учителем состоит в том, чтобы научить алгоритм сопоставлять разные значения разных атрибутов объекта со значением заданного атрибута этого же объекта, известного как целевой атрибут. Например, когда обучение с учителем применяется для спам-фильтра, алгоритм пытается изучить функцию, которая сопоставляет атрибуты, описывающие электронную почту, со значением (спам/не спам) целевого атрибута; функция, которую изучает алгоритм, является моделью спам-фильтра. Искомая алгоритмом закономерность является функцией, которая сопоставляет значения входных атрибутов со значением целевого атрибута, а модель, которую возвращает алгоритм, является компьютерной программой, выполняющей эту функцию. По сути, обучение с учителем осуществляется путем поиска одной из множества функций, которая наилучшим образом отображает связь между входными и выходными данными. Однако для любого набора данных разумной сложности существует так много комбинаций входных данных и их возможных сопоставлений с выходными данными, что алгоритм не может испробовать их все.

При обучении без учителя целевой атрибут отсутствует. Следовательно, алгоритмы обучения без учителя не требуют времени и усилий на маркировку целевым атрибутом объектов в наборе данных. Однако отсутствие целевого атрибута означает и то, что обучение становится более сложным: вместо конкретной задачи поиска соответствующего отображения между входным и выходным значениями, перед алгоритмом ставится более общая задача поиска закономерностей в данных. Самым распространенным типом обучения без учителя является *кластерный анализ*, когда алгоритм ищет кластеры объектов, схожих друг с другом. Часто эти алгоритмы кластеризации начинают со случайной группы кластеров, а затем итеративно обновляют кластеры (перебрасывая объекты из одного кластера в другой) таким образом, чтобы увеличить подобие внутри каждого кластера и разницу между ними.

Задача кластеризации – выяснить, как измерить подобие. Если все атрибуты в наборе данных являются числовыми и имеют одинаковые диапазоны, то, вероятно, имеет смысл просто рассчитать евклидово расстояние между ними. Объекты, которые находятся близко друг к другу в евклидовом пространстве, рассматриваются как подобные. Однако некоторых наборах данных разные числовые атрибуты имеют разные диапазоны, в результате чего разброс значений в одном атрибуте может быть не таким значительным, как в другом. В таких случаях атрибуты должны быть нормализованы путем присвоения им одинакового диапазона. Еще одним усложняющим фактором при расчете сходства является то, что подобие объектов можно определять по-разному. Порой одни атрибуты являются более важными, чем другие, поэтому имеет смысл при расчетах задавать весовой параметр некоторым атрибутам, что бывает необходимо и тогда, когда набор данных содержит нечисловые значения.

Обучение прогнозированию

Прогнозирование – это задача оценки значения целевого атрибута конкретного объекта на основе значений других его атрибутов. Проблему прогнозирования решают алгоритмы машинного обучения с учителем, которые генерируют модели прогнозирования.

Рассмотрим наиболее известные подходы к решению задачи прогнозирования: корреляционный и регрессионный анализ, нейросети и деревья решений.

Корреляционный анализ. Корреляция описывает силу взаимосвязи между двумя атрибутами. Коэффициент корреляции Пирсона ***r*** измеряет силу линейных зависимостей между двумя числовыми атрибутами и находится в диапазоне значений от –1 до +1. ***r*** = 0 – два атрибута независимы друг от друга; ***r*** = +1 – атрибуты имеют идеальную положительную корреляцию, любое изменение одного из них сопровождается эквивалентным изменением другого в том же направлении; ***r*** = –1 – атрибуты имеют идеальную отрицательную корреляцию, каждое изменение в одном из них сопровождается противоположным изменением в другом. Значение ***r***≈±0,7 указывает на сильную линейную зависимость между атрибутами, $r \approx \pm 0{,}5$ – на умеренную линейную зависимость, $r \approx \pm 0{,}3$ – на слабую зависимость, а $r \approx 0$ – на отсутствие зависимости между атрибутами.

Рассмотрим пример. В таблице представлен фрагмент набора данных для исследования сахарного диабета. Требуется определить, имеет ли какой-нибудь из атрибутов сильную корреляцию с целевым атрибутом, описывающим вероятность развития диабета у человека.

Таблица 2. Набор данных для исследования диабета

Объект	Рост (м)	Вес (кг)	Размер обуви	Продолжительность тренировок в неделю (мин.)	Вероятность развития диабета (%)
1	1,70	70	5	130	0,05
2	1,77	88	9	80	0,11
3	1,85	112	11	0	0,18

Есть и другие атрибуты, которые могут быть включены в набор, например возраст человека, и что среди представленных в таблице есть лишние (например, размер обуви, который явно не коррелирует с развитием диабета. Выбор атрибутов для набора данных – ключевая задача науки о данных. В примере мы намеренно используем такой набор, какой есть.

Исходя из наших знаний о физиологии людей, мы ожидаем, что между некоторыми атрибутами будут взаимосвязи. Например, обычно чем выше человек, тем больше размер его обуви. Или чем больше кто-то тренируется, тем меньше в нем будет избыточного веса. Также ожидаем, что не обнаружим связи между размером обуви и временем тренировок. В качестве иллюстрации на рис. представлены три диаграммы рассеяния, отражающие как эти интуитивные ожидания отражаются в данных.

Рис. 9. Диаграммы рассеяния размера обуви и роста, веса и физических упражнений, размера обуви и физических упражнений

На верхней диаграмме рассеяния наблюдается четкая закономерность, идущая из нижнего левого угла в верхний правый, указывающий на то, что по мере того, как люди становятся выше, размер их обуви тоже увеличивается. Если мы вычислим коэффициент корреляции Пирсона между размером обуви и ростом, то $r = 0{,}898$, т.е. мы имеем сильную положительную корреляцию между этой парой атрибутов. Средняя диаграмма рассеяния показывает, что атрибутами веса и физических упражнений имеется противоположное направление от левого верхнего угла до нижнего правого, что указывает на отрицательную корреляцию – чем больше люди тренируются, тем меньше их вес. Коэффициент корреляции Пирсона для этой пары признаков равен $r = -0{,}710$ (сильная отрицательная корреляция). На нижнем графике рассеяния отображается корреляция времени тренировок и размера обуви. Видно, что данные распределены случайным

образом и коэффициент корреляции Пирсона для этой пары атрибутов ***r*** = –0,272 (корреляция отсутствует).

Применение статистического коэффициента корреляции Пирсона к анализу данных не ограничивается только парами атрибутов. Эту проблему можно обойти, применяя функции для групп атрибутов, используя производные атрибуты. В нашем примере это может быть атрибут ИМТ (индекс массы тела) – это соотношение массы тела и роста человека. Он может помочь определить людей в группе населения, которые подвержены риску ожирения.

Например, на рис. 10 представлены три диаграммы рассеяния, каждая из которых показывает отношения между целевым атрибутом диабета и одним из следующих признаков: ростом, весом и ИМТ.

Между атрибутом роста и вероятностью диабета не наблюдается закономерности, отсутствует реальная корреляция (***r*** = –0,277). Средняя диаграмма указывает на положительную корреляцию между людьми с бóльшей массой тела и вероятностью диабета (***r*** = 0,655). Нижняя диаграмма напоминает среднюю диаграмму, однако корреляция между ИМТ и диабетом сильнее (***r*** = 0,877), потому что вероятность развития диабета зависит от взаимосвязи роста и веса, а атрибут ИМТ моделирует именно эту взаимосвязь. Вот почему врачи интересуются ИМТ людей, это дает им больше информации о вероятности развития диабета, чем рост или вес человека по отдельности. Не вес сам по себе способствует заболеванию, а его избыточность.

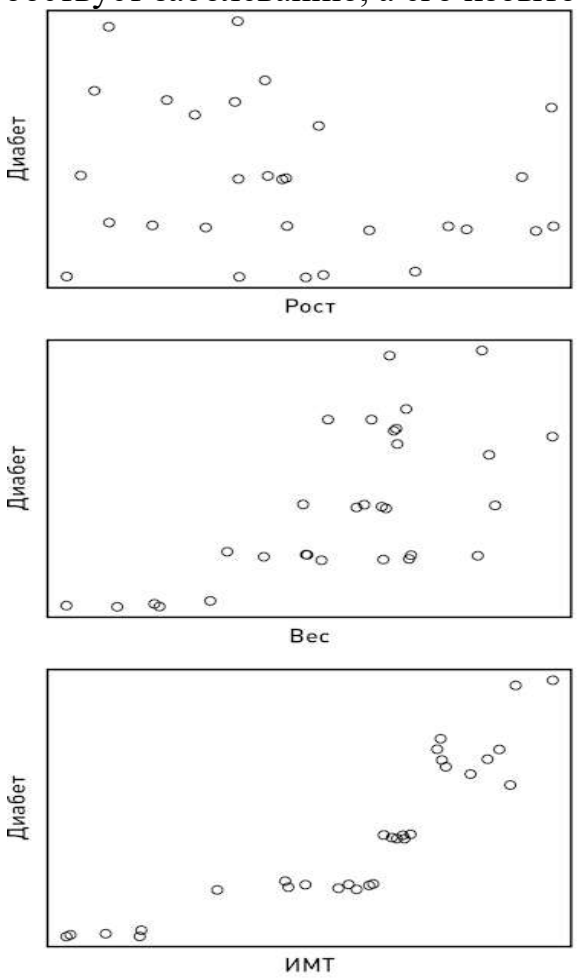


Рис. 10. Диаграммы рассеяния вероятности диабета в зависимости от роста, веса и ИМТ

Когда вы знаете значимые атрибуты для представления данных, вы можете создавать модели точно и быстро! (См. Родзин С.И. «Биоинспирированная гиперэвристика для отбора значимых признаков в задачах классификации больших данных» // Вестник компьютерных и информационных технологий. 2021. Т.18. № 5. С. 35-44. DOI: 10.14489/vkit.2021.05.pp.035-044.). Если корреляции не существует, то входные атрибуты не имеют значения при прогнозировании. входные данные и всегда прогнозировать центральную тенденцию этой цели в наборе данных. И наоборот, при сильной корреляции вероятно, что алгоритм обучения сможет сгенерировать точную модель прогнозирования.

Регрессионный анализ. Когда набор данных состоит из числовых атрибутов, часто используются модели прогнозирования, основанные на регрессии. Регрессионный анализ оценивает ожидаемое (или среднее) значение числового целевого атрибута, когда все входные атрибуты фиксированы. Первый шаг в регрессионном анализе – выдвижение гипотезы о структуре отношений между входными атрибутами и целевым. Затем определяется математическая модель предполагаемой взаимосвязи. Она называется *функцией регрессии*. Функция регрессии может иметь несколько параметров, и целью регрессионного анализа является поиск правильных настроек для этих параметров.

Поиск функции целесообразно начинать с самого простого типа - с линейной зависимости. Простейшим применением линейной регрессии является моделирование взаимосвязи между двумя атрибутами: входным атрибутом $\boldsymbol{X}$ и целевым атрибутом $\boldsymbol{Y}$:

$\boldsymbol{Y} = \omega_0 + \omega_1 \boldsymbol{X}$.

Это уравнение линейной функции, где ω_0 и ω_1 – параметры функции регрессии. Параметр ω_0 – это точка пересечения прямой с осью ординат, когда $\boldsymbol{X}$ равен нулю. Параметр ω_1 определяет угол наклона прямой. Эти параметры изначально неизвестны. Установка этих параметров эквивалентна поиску функции, которая наилучшим образом соответствует данным.

Стратегия установки этих параметров состоит в том, чтобы начать со случайных значений, а затем итеративно обновлять параметры, уменьшая общее отклонение функции в наборе данных. Общее отклонение рассчитывается в три этапа:

1. Функция применяется к набору данных и для каждого объекта в наборе оценивает значение целевого атрибута.

2. Отклонение функции для каждого объекта вычисляется путем вычитания полученного значения целевого атрибута из его фактического значения.

3. Отклонение функции для каждого объекта возводится в квадрат и эти значения суммируются.

Возведение в квадрат придает отклонению положительное значение. Этот параметр известен как сумма квадратов отклонений, а стратегия подбора линейной функции путем поиска параметров называется методом наименьших квадратов SSE:

$$\mathrm{SSE} = \sum_{i=i}^{n} (target_i - prediction_i)^2,$$

где набор данных содержит n объектов, $target_i$ – это значение целевого атрибута для объекта i в наборе данных, а $prediction_i$ – оценка функции цели для того же объекта.

Пример. Создадим линейную регрессионную модель прогнозирования, которая оценивает вероятность развития диабета у человека с учетом его ИМТ. Для этого применим метод SSE, чтобы найти подходящую прямую для набора данных. Рис. 11 a иллюстрирует эту прямую. На рис. 11 b пунктиром показано отклонение для каждого объекта в этой прямой.

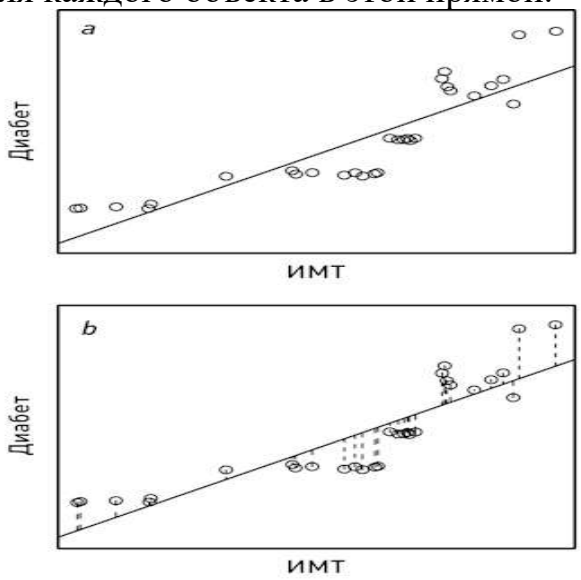


Рис. 11. *a* — линия регрессии, наиболее подходящая для модели Diabetes = −7,38431 + 0,55593 × ИМТ; *b*— пунктирные вертикальные линии иллюстрируют остаточную погрешность для каждого объекта

По методу SSE линией наилучшего соответствия будет прямая:

Диабет = −7,38431 + 0,55593 × ИМТ.

Чтобы предсказать вероятность развития диабета у человека, мы просто вводим его значение ИМТ в модель. Например, когда ИМТ=20, модель возвращает прогноз 3,73% для атрибута «Диабет». Если ввести среднее значение ИМТ в наборе данных (например, ИМТ=24,0932), модель оценивает атрибут диабета как 4,29%, что является средним значением для всего набора данных.

Объекты, которые имеют экстремальные значения (выбросы), могут оказать непропорционально большое влияние на прогноз. Поэтому перед использованием метода SSE важно проверить наличие выбросов в наборе данных.

Модели линейной регрессии могут быть расширены, чтобы принимать несколько входных значений. Например, в нашем примере можно расширить модель, включив в нее атрибуты веса и времени, затраченного на физические упражнения. Функция регрессии будет:

Диабет = ω_0 + ω_1 ИМТ + ω_2 Упражнения + ω_3 Вес.

В целом между корреляцией и регрессией наблюдается сходство.

Нейросети и глубокое обучение. По своей сути нейрон – это функция линейной регрессии с несколькими входами. Единственное существенное

различие состоит в том, что в нейроне выходной сигнал определяется другой функцией, которая называется *функцией активации* (бывает нелинейной).

В качестве функций активации наиболее часто применяются *логистическая* функция и функция *tanh* (рис. 12).

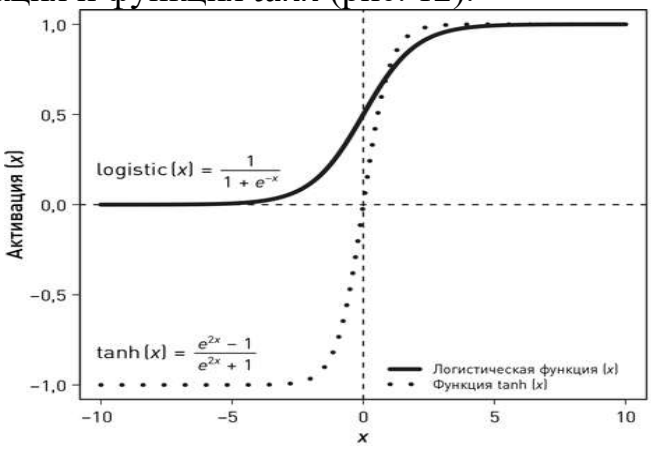


Рис. 12. Отображение логистической функции и функции tanh, применяемое к входящему *x*

Каждый нейрон в нейросети выполняет очень простой набор операций:

1. Умножает каждый вход на его вес;
2. Суммирует результаты умножения;
3. Проводит этот результат через функцию активации.

Операции 1 и 2 являются просто вычислением функции регрессии, а операция 3 использует функцию активации.

Пример. Дана нейросеть, которая принимает значение процентного содержания жира (Жир%) в организме человека и его максимальное потребление кислорода (VO_2) в качестве входных данных и вычисляет индивидуальный уровень физической подготовки.

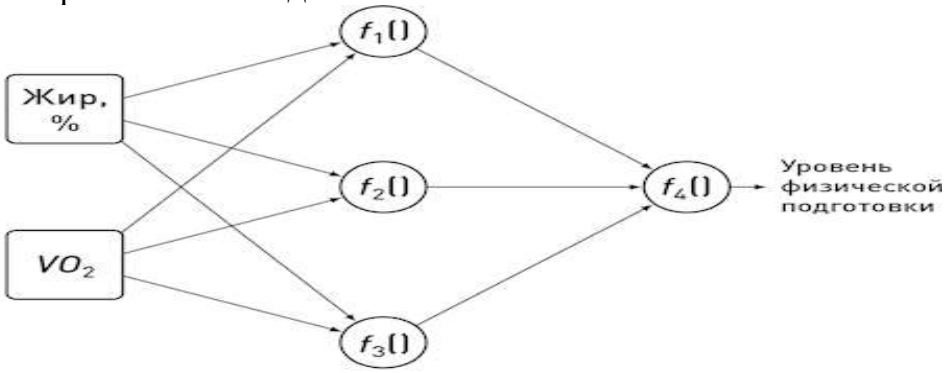


Рис. 14. Нейронная сеть, которая прогнозирует уровень физической подготовки человека

Все нейроны в среднем слое сети вычисляют функцию на основе Жир% и VO_2: f1(), f2() и f3(). Каждая из этих функций моделирует взаимодействие между входами иначе, чем две другие. Эти функции по существу представляют собой новые атрибуты, которые получены сетью из необработанных входных данных. Они схожи с производным атрибутом ИМТ, описанным ранее, который был рассчитан как функция веса и роста. Однако чаще производный атрибут, рассчитанный нейроном, не будет нести никакого смысла для человека. Просто эти атрибуты фиксируют взаимодействия между другими атрибутами, которые сеть сочла полезными. Последний узел в сети f4 вычисляет другую функцию –

скомпонованную из f1(), f2() и f3(), – на выходе которой получается прогноз уровня физической подготовки, возвращаемый сетью. Эта функция не может быть значимой для человека, кроме того факта, что она определяет взаимодействие, которое, как обнаружила сеть, имеет высокую корреляцию с целевым атрибутом.

Обучение нейросети включает в себя поиск правильных весов для ее связей. Как обучается отдельный нейрон рассчитывать вес связи? Предположим, что имеется обучающий набор данных, который имеет для каждого объекта и входные значения, и целевой атрибут. Пусть входящим связям нейрона уже назначены веса. Если мы возьмем объект из набора данных и представим нейрону значения входных атрибутов этого объекта, он выдаст прогноз для цели. Вычитая этот прогноз из значения целевого атрибута в наборе данных, мы сможем измерить отклонение нейрона для этого объекта. Используя ряд простых вычислений, можно вывести правило для обновления весов входящих связей нейрона:

1. Если отклонение равно 0, не меняйте веса на входах.

2. Если отклонение положительное, требуется увеличить прогнозное значение, поэтому нужно прибавить веса всех связей с положительным входом и понизить веса связей с отрицательным.

3. Если отклонение отрицательное, требуется уменьшить прогнозное значение, поэтому нужно понизить веса всех связей с положительным входом и прибавить веса связей с отрицательным.

Сложность обучения нейросети в том, что обновление веса требует оценки ошибки в нейроне. Вычислить ошибку для нейрона в выходном слое сети просто, а для нейронов в скрытых слоях намного сложнее. Стандартный способ обучения многослойной нейросети с учителем – алгоритм обратного распространения ошибки. Он предполагает наличие набора обучающих данных. Описание этого алгоритма (без формул) выглядит так:

1. Рассчитайте ошибку для каждого из нейронов в выходном слое и обновите согласно правилу выше веса входящих связей этих нейронов.

2. Поделитесь ошибкой, рассчитанной для нейрона, с каждым из нейронов в предыдущем слое, который связан с ним, пропорционально весу связей между двумя нейронами.

3. Для каждого нейрона на предыдущем уровне вычислите общую ошибку сети, за которую он ответственен, суммируя с теми ошибками, которые были переданы обратно в него, и используйте результат этого суммирования, чтобы обновить веса на связях, входящих в этот нейрон.

4. Пройдите таким же образом остальные слои в сети, повторяя шаги 2 и 3 до тех пор, пока веса между входными нейронами и первым слоем скрытых нейронов не будут обновлены.

В алгоритме вес, обновляемый для каждого нейрона, высчитывается так, чтобы уменьшить, но не устранить полностью ошибку нейрона в обучающем экземпляре. Причина этого заключается в том, что цель обучения сети – дать ей возможность сделать выводы, которых нет в данных обучения, а не просто запомнить эти данные. Таким образом, каждое обновление весов делает сеть

более подходящей к набору данных. В некоторых версиях алгоритма веса обновляются только после того, как сеть был обучена на нескольких объектах, чтобы избежать переобучения сети.

Одним из наиболее удивительных технических достижений последних 10 лет стало появление нейросетей глубокого обучения. Это нейросети, имеющие несколько скрытых слоев.

Выходной слой также может иметь несколько нейронов. Это полезно, если целью является номинальный или порядковый тип данных, имеющий разные уровни. Сеть настраивают так, чтобы для каждого уровня существовал один выходной нейрон, и обучают ее так, чтобы для каждого входа только один из выходных нейронов выводил высокую активацию (означающую прогнозируемый целевой уровень).

Разработаны также сети с обратными связями. Например, рекуррентные нейросети (РНС). Это дает сети память. Они подходят для обработки последовательных данных, таких как естественный язык. Другой популярной архитектурой глубоких нейронных сетей являются сверточные нейронные сети (СНС). СНС были первоначально разработаны для использования с данными изображений. СНС должна обнаруживать на изображении некоторый визуальный признак независимо от того, в какой части изображения он встречается. Например, распознавать форму глаз.

Сила глубоких нейронных сетей в том, что они могут автоматически изучать полезные атрибуты, обучаться признакам. Нейрон в первом слое сети может изучать некоторую функцию, например зависимость ИМТ от веса и роста. Однако выход этого нейрона подается в нейроны второго слоя, изучающие функции от выходных данных и т.д. По мере роста глубины сети, она может изучать все более и более сложные зависимости. Это делает модели глубокого обучения точными при выполнении задач с многомерным вводом, таких, как обработка изображений и текста.

Чем глубже нейросеть, тем более сложные наборы данных она способна изучать. Однако алгоритм обратного распространения ошибки не очень хорошо работает с глубокими сетями. Только в последние годы были разработаны новые типы нейронов и модификации алгоритма обратного распространения, которые помогают решить эту проблему. Также было обнаружено, что требуется осторожная инициализация весов сети. Усложняли обучение глубоких сетей также требуемые большие вычислительные ресурсы и большое число обучающих данных.

Деревья решений.

Линейная регрессия и нейронные сети лучше всего работают с числовыми входными данными. Если входные атрибуты в наборе данных в основном номинальные или порядковые, лучше использовать другие алгоритмы и модели машинного обучения, такие как деревья решений.

Дерево решений кодирует условный оператор ЕСЛИ-ТО-ИНАЧЕ в древовидной структуре. Все пути в структуре дерева решений от корня до листа определяются правилом классификации, состоящим из последовательных тестов. Цель обучения дерева решений состоит том, чтобы найти такие правила

классификации, которые делят обучающий набор данных на группы объектов, имеющих одинаковое значение целевого атрибута.

Прародителем большинства современных алгоритмов машинного обучения деревьев решений является алгоритм **ID3**. Он строит деревья решений рекурсивным способом в глубину, добавляя один узел, начиная с корневого узла.

ID3 начинается с выбора атрибута для проверки в корневом узле. Затем ID3 наращивает каждую ветвь, до тех пор, пока все объекты каждой ветви не будут иметь одинаковое значение целевого атрибута. Тогда в дерево добавляется конечный узел и помечается значением целевого атрибута, общего для всех объектов ветви.

ID3 выбирает оптимальный атрибут для тестирования в каждом узле дерева, чтобы минимизировать количество тестов. Для этого используется информационная энтропия К.Шеннона. ID3 выбирает для тестирования в узле атрибут, который приводит к наименьшему значению энтропии после разделения набора данных с использованием этого атрибута.

Пример. В табл. 3 приведен фрагмент списка электронных писем, в котором каждое описывается рядом атрибутов и тем, является оно спамом или нет.

Таблица 3. Набор данных электронных писем

Вложение	Подозрительные слова	Неизвестный отправитель	Спам
Нет	Нет	Да	Нет
Нет	Нет	Да	Нет
Нет	Да	Нет	Нет
Нет	Нет	Нет	Нет
Нет	Нет	Нет	Нет

Номинальные атрибуты «Вложение», «Подозрительные слова» и «Неизвестный отправитель» - входные атрибуты. Атрибут «Спам» – цель.

Атрибут «Неизвестный отправитель» разбивает набор данных на две группы (одна из них содержит большинство объектов, где Спам = Истина, а другая, где Спам = Ложь). «Неизвестный отправитель» помещается в корневой узел.

Все объекты в правой ветви имеют одинаковое целевое значение, а в левой – разное. Разделение объектов в левой ветви с использованием атрибута «Подозрительные слова» приводит к образованию двух наборов, где для одного Спам = Ложь, для другого Спам = Истина (рис. 18).

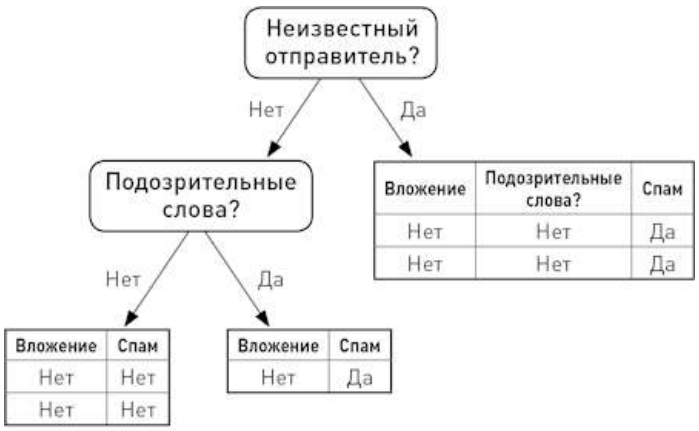


Рис. 18. Добавление второго узла в дерево

Среди алгоритмов машинного обучения нет одного наилучшего алгоритма. NFL-Теорема утверждает, что не существует ни одного алгоритма МО, в среднем превосходящего все другие алгоритмы по всем возможным наборам данных. Поэтому этап моделирования обычно включает в себя построение нескольких моделей с использованием разных алгоритмов и их последующее сравнение, чтобы определить, какой из алгоритмов генерирует наилучшую модель.

Золотое правило оценки моделей заключается в том, что их никогда не следует тестировать на тех же данных, на которых они были обучены. Стандартный процесс для обеспечения этого правила таков: данные разбиваются на три части – *обучающий набор, оценочный набор и тестовый набор*. Пропорции, обычно составляют 50:20:30 или 40:20:40. Обучающий набор используется для обучения начальной группы моделей. Оценочный набор – для сравнения эффективности этих моделей на новых данных. После выявления алгоритма, который сгенерировал лучшую модель, используется тестовый набор.

Машинное обучение на языке Python

Процесс поиска подходящего алгоритма машинного обучения носит нелинейный характер. Python, будучи интерпретируемым высокоуровневым языком программирования, как будто специально придуман для опробования разных вариантов. К тому же, он работает быстро, хотя и медленне *C,* и содержит огромное число простых в использовании библиотек.

Работая на *Python*, очень просто перепоручить численные расчеты низкоуровневым расширениям, реализованным на *C*. Именно так устроены библиотеки *NumPy* и *SciPy* (http://scipy.org/Download). *NumPy* отвечает за многомерные массивы – основную структуру данных для большинства современных алгоритмов. А *SciPy* на базе этих массивов реализует быстрые численные алгоритмы. Наконец, matplotlib (http://matplotlib.org/) – удобная библиотека для построения качественных графиков на Python.

В *Windows, Mac и Linux*, существуют готовые инсталляторы *NumPy, SciPy и matplotlib*.

Наборы данных для программирования задач машинного обучения можно использовать из репозитория наборов данных **UCI** (Калифорнийский университ в Ирвайне). Там свыше 200 наборов. Адрес репоэитория http://archive.ics.uci.edu/ml/ .

Примеры программ на *Python* для решения разнообразных задач классификации, кластеризации, регрессии, прогнозирования, машинного зрения можно найти в книгах:

- ***Коэльо Л., Ричарт В***. Построение систем машинного обучения на языке *Python*. – М.: ДМК Пресс, 2016. – 302 с.

- ***Сегаран Т.*** Программируем коллективный разум. – СПб: Символ-Плюс, 2008. – 368 с.

Профессор Стэнфордского университета *Эндрю Нг (Andrew Ng*) ведет большоii открытыii онлайновый курс по машинному обучению на сайте Coursera (http://www.coursera.org). Курс бесплатный, но требует много времени.

Отличный способ больше узнать о машинном обученнн – посоревноваться с кем-то! Сайт ***Kaggle*** (http://www.kaggle.com)- место, где проводятся такие конкурсы. Здесь вы найдете конкурсы с различной структурой и, зачастую, с денежными призами. Как правило, победу приносит не изобретение нового алгоритма, а продуманная предварительная обработка, нормировка и сочетание различных методов.

10. Решение стандартных задач анализа данных

Одним из важнейших навыков специалиста по данным является способность сформулировать проблему как стандартную задачу анализа данных. Большинство проектов в этой области можно отнести к одному из четырех основных классов задач:

- кластеризация (или сегментация);
- обнаружение аномалий (или выбросов);
- поиск ассоциативных правил;
- прогнозирование (классификация и регрессия).

Существует множество алгоритмов машинного обучения, и каждый предназначен для конкретной задачи анализа данных. Например, алгоритмы, генерирующие модели дерева решений, в первую очередь предназначены для решения задач прогнозирования.

Поскольку задача влияет как на структуру набора данных, так и на выбор алгоритмов машинного обучения, определиться с ее типом необходимо на раннем этапе жизненного цикла проекта, в идеале – на этапе понимания бизнес-целей CRISP-DM. Рассмотрим стандартные типы задач.

Кластеризация

Задача поддержки маркетинговых кампаний и продаж. Как сформулировать проблему как задачу кластеризации? Диапазон атрибутов, которые можно использовать для описания клиентов в процессе кластеризации, огромен, но есть наиболее типичные: демографическая информация (возраст, пол и т.д.), место жительства (почтовый индекс, адрес и т.д.), транзакционная информация (продукты или услуги, которые приобретал клиент, доход от него, жаловался ли на услугу и проч.). *Большая проблема – определить, какие атрибуты должны быть включены, а какие исключены, чтобы добиться наилучших результатов.* Принятие решения о выборе атрибутов основано на итерациях экспериментов, их анализе специалистом.

Наиболее известным алгоритмом машинного обучения для кластеризации является *алгоритм k-средних*. Буква ***k*** в названии указывает количество кластеров, которые алгоритм ищет в данных. Значение ***k*** задается заранее и часто устанавливается экспериментальным путем, методом проб и ошибок. Алгоритм k-средних предполагает, что все атрибуты, описывающие клиентов в наборе данных, являются числовыми. Если набор данных содержит нечисловые атрибуты, то они должны быть соотнесены с числовыми значениями, иначе потребуется другой алгоритм. Данный алгоритм рассматривает каждого клиента как точку в облаке точек (или в диаграмме рассеяния), где позиция клиента определяется значениями атрибутов в его профиле. Цель алгоритма – найти положение центра каждого кластера в облаке точек.

Алгоритм работает в 2 этапа: сначала каждый объект назначают ближайшему к нему кластерному центру, а затем обновляют этот центр таким образом, чтобы он оказался в середине назначенных ему объектов:

1. Процесс начинается с выбора ***k*** объектов в качестве начальных кластерных центров. Первый центр устанавливается путем выбора случайного объекта в наборе данных.
2. Последующие центры кластеров – путем выбора объектов с вероятностью, пропорциональной квадрату расстояния от ближайшего существующего кластерного центра.
3. Как только все ***k*** кластерных центров инициализированы, происходит первая итерация назначения объектов ближайшему центру.
4. После этого центры перемещаются так, чтобы совпасть с центром назначенных им объектов. Перемещение кластерных центров сместит их ближе к одним объектам и отодвинет от других, в том числе и от объектов, им назначенных.
5. Затем объекты переназначаются снова ближайшему обновленному кластерному центру. Некоторые объекты останутся назначенными одному и тому же центру, другие могут быть переназначены новому.
6. Этот процесс назначения объектов и обновления центра продолжается до тех пор, пока при очередной итерации никакие объекты не будут переназначены новому кластерному центру.

Алгоритм ***k***-средних недетерминирован - разные начальные позиции кластерных центров, вероятно, будут давать и разные кластеры. Поэтому алгоритм обычно запускается несколько раз, а затем результаты этих прогонов сравниваются, чтобы увидеть, какие кластеры выглядят наиболее адекватными с учетом предметной области и ее понимания специалистом по данным.

Каждый кластерный центр определяет отдельный профиль клиента с описанием значений атрибутов назначенных ему объектов. В алгоритме ***k***-средних нет обязательного условия, что все кластеры должны быть одного размера. Размеры кластеров могут дать полезную информацию для управления маркетингом. Например, процесс кластеризации может выявить небольшие целевые кластеры клиентов, которые отсутствуют в текущих маркетинговых кампаниях. Другая стратегия - сосредоточиться на кластерах с клиентами, приносящими наибольший доход. Стратегии могут быть разными, но при любой из них понимание сегментов клиентской базы является предпосылкой успеха маркетинга.

Кластеризации может применяться к большинству типов данных. Например, для анализа учебных курсов с целью выявления групп студентов, которые предпочитают разные методы обучения; для идентификации групп похожих документов в корпусе текстов; в биоинформатике для анализа последовательностей генов в процессе, называемом микрочиповым анализом.

Обнаружение аномалий

Обнаружение аномалий (анализ выбросов) включает в себя поиск и выявление объектов, которые не соответствуют типичным данным в наборе. Эти несоответствующие объекты называют аномалиями или выбросами. Обнаружение аномалий используется сейчас при анализе финансовых транзакций с целью выявления потенциальных мошеннических действий и запуска расследований. Например, оно позволяет определить мошеннические

действия по кредитным картам путем выявления транзакций, происходящих в необычном месте или на необычно большую сумму по сравнению с другими транзакциями по этой кредитной карте.

Вначале вручную определяется ряд правил, основанных на экспертных знаниях в конкретной области, которые помогают идентифицировать аномальные события. Часто набор этих правил описывают на SQL и запускают в базах или хранилищах данных. Распространенная схема мошенничества с кредитными картами заключается в том, что вор проверяет, работает ли украденная карта, совершая по ней небольшую покупку, а затем, если транзакция проходит, как можно быстрее покупает что-нибудь дорогое, прежде чем карта будет аннулирована. Программист баз данных пишет сценарии, которые выявляют последовательности транзакций по кредитной карте, соответствующие этой закономерности, и либо автоматически блокируют карту, либо предупреждают компанию-эмитента.

Основным недостатком подхода, основанного на правилах, является то, что он может идентифицировать аномальные события только после того, как они произошли и попали в поле внимания организации. В идеале организация хочет выявлять аномалии, когда они происходят впервые. Обнаружение аномалий является противоположностью кластеризации. Цель кластеризации найти группы схожих элементов, цель обнаружения аномалий – поиск элементов, непохожих на остальную часть набора данных. Существует два метода:

- первый группирует нормальные данные вместе, а аномальные помещает в отдельные кластеры;
- второй метод заключается в измерении расстояния между объектом и центром кластера. Чем дальше объект находится от центра кластера, тем выше вероятность того, что он окажется аномальным и требует расследования.

Разработаны алгоритмы машинного обучения, известные как одноклассные классификаторы, которые предназначены для работы с несбалансированными данными при обнаружении аномалий.

Метод опорных векторов (SVM) анализирует данные как один кластер и выявляет основные характеристики и ожидаемое поведение объектов. Затем алгоритм маркирует каждый объект, чтобы указать, насколько он похож или отличен от основных характеристик и ожидаемого поведения. С помощью этой информации выявляют аномалии. Чем больше объект не похож на остальные, тем выше необходимость его исследования.

Тот факт, что аномалии редки, означает, что их легко можно упустить и трудно идентифицировать. По этой причине специалисты по данным часто комбинируют друг с другом модели для обнаружения аномалий. Например, если транзакция идентифицирована как мошенническая только одной из четырех моделей, то система принятия решений не будет определять ее как случай мошенничества и игнорирует. И наоборот, если три или четыре модели из четырех идентифицируют транзакцию как возможное мошенничество, она будет помечена для обработки аналитиком данных.

Обнаружение аномалий может применяться клиринговыми центрами при мониторинге финансовых транзакций для выявления любых действий, которые требуют дальнейшего расследования, – от потенциально мошеннических до отмывания денег. Обнаружение аномалий применяется при анализе страховых претензий для выявления нетипичных. В кибербезопасности оно используется для обнаружения возможных взломов или нетипичного поведения сотрудников в сети. В области медицины выявление аномалий в историях болезней пациентов может быть полезно для диагностики заболеваний. Наконец, с распространением датчиков и технологии интернета вещей (IoT) обнаружение аномалий будет играть важную роль при мониторинге данных и формировании предупреждений, когда происходят нештатные ситуации и требуется вмешательство.

Поиск ассоциативных правил

Стандартная стратегия продаж – перекрестные продажи, т.е. предложение клиентам дополнительных продуктов, которые они могут захотеть приобрести. Понимание связей между продуктами является основой перекрестных продаж.

Поиск ассоциативных правил (АП) – это метод анализа данных при обучении без учителя. Его суть состоит в поиске групп элементов, часто встречающихся вместе. Для такого анализа данных бизнес отслеживает корзину товаров каждого покупателя при каждом посещении магазина. При поиске АП каждая строка в наборе данных описывает содержимое корзины, оплаченной конкретным покупателем в конкретное время. Атрибуты в этом наборе данных – приобретенные товары. На основе данных алгоритм поиска ассоциативных правил ищет товары, которые встречаются в каждой корзине. В отличие от кластеризации и обнаружения аномалий, которые фокусируются на выявлении сходств или различий между объектами (или строками) в наборе данных, поиск АП фокусируется на рассмотрении связей между атрибутами (или столбцами) в наборе данных. Этот тип анализа ищет корреляции между продуктами.

Основным алгоритмом создания АП является алгоритм ***Apriori***, состоящий из двух этапов:

1. Найти все комбинации товаров в наборе транзакций, которые случаются с заданной минимальной частотой. Эти комбинации называются частыми предметными наборами.

2. Рассчитать правила, которые отражают совместное вхождение товаров в частые предметные наборы. Алгоритм ***Apriori*** вычисляет вероятность появления элемента в частом предметном наборе с учетом присутствия в нем других предметов.

Алгоритм ***Apriori*** генерирует АП, которые выражают вероятностные отношения между элементами в форме:

ЕСЛИ {предпосылка} – ТО {следствие}.

Например, правило, выведенное из частых предмет наборов, содержащих предметы А, В и С:

ЕСЛИ {хот-доги, кетчуп} – ТО {пиво}.

Один из супермаркетов в 1980-х гг. использовал алгоритм ***Apriori*** и выявил неожиданную ассоциацию клиентов, покупающих вместе подгузники и пиво:

семьи с маленькими детьми готовились к уик-энду и знали, что им нужно запастись подгузниками и купить пиво, чтобы дома было что выпить. Магазин разместил эти два товара рядом, и продажи выросли.

АП имеют два основных статистических показателя: поддержка и достоверность. Процент поддержки указывает, как часто элементы встречаются вместе. Поддержка – это отношение транзакций, которые включают в себя элементы (и предпосылки, и следствия) к общему числу транзакций. Процент достоверности АП указывает на вероятность появления предпосылки и следствия в одной и той же транзакции. Достоверность – это условная вероятность, с какой следствие наступает в случае предпосылки. Достоверность рассчитывается как отношение поддержки к количеству транзакций, в которые входит предпосылка. Например, показатель достоверности 75 % для АП, касающегося хот-догов, кетчупа и пива, указывает на то, что в 75 % случаев, когда покупатель покупал хот-доги и кетчуп, он также покупал и пиво. Поддержка 5 % для того же примера будет показывать, что 5 % всех корзин в наборе данных содержали все три элемента правила.

Даже небольшой набор данных может содержать большое количество АП. Чтобы упростить их анализ, набор обычно ограничивают только теми правилами, которые имеют высокие значения поддержки и достоверности. Правила, которые являются тривиальными или их невозможно объяснить, также не принимаются во внимание. Тривиальные правила представляют собой ассоциации, которые очевидны и известны каждому, кто разбирается в данной сфере. После сокращения набора правил специалист по данным может проанализировать оставшиеся и дать рекомендации для сайтов, рекламы в магазинах, рассылок, перекрестных продаж и т.д.

Поиск АП становится более эффективным, если корзины товаров связаны с демографическими данными клиента. По этой причине многие ритейлеры используют программы лояльности. Например, АП, основанные на демографии, могут применяться к новым клиентам, о привычках и предпочтениях которых у компании нет информации. Вот пример АП, учитывающего демографические данные:

ЕСЛИ пол (мужской), возраст (<35) и {хот-доги, кетчуп} – ТО {пиво}[поддержка = 2 %, доверие = 90 %].

Привычная область поиска АП – содержимое покупательских корзин. Однако поиск ассоциативных правил полезен и в ряде других областей. К примеру, в индустрии телекоммуникаций применение АП в отношении клиента помогает компаниям проектировать различные сервисы и объединять их в пакеты. В страховании АП используются, чтобы обнаруживать связи между страховыми продуктами и требованиями клиентов. В области медицины с их помощью проверяют взаимосвязь между существующими и новыми методами лечения и лекарственными средствами. А в банковских и финансовых услугах используют АП для определения соответствия продуктов и конкретных клиентов, чтобы применить их к новым клиентам.

Классификация

Стандартной бизнес-задачей в сфере управления взаимоотношениями с клиентами является оценка вероятности того, что отдельный клиент предпримет какое-либо действие. Для описания этой задачи используют термин «моделирование склонности», поскольку цель состоит в том, чтобы смоделировать склонности человека. Это могут быть реакция на маркетинг, дефолт по кредиту или отказ от услуг. Возможность идентифицировать тех, кто может покинуть сервис, особенно важна для операторов мобильной связи, которым требуются значительные инвестиции для привлечения новых клиентов. Привлечение нового клиента обходится в пять-шесть раз дороже, чем удержание постоянного. В результате многие операторы готовы биться за сохранение своих нынешних клиентов, стараясь при этом минимизировать затраты. Поэтому удержание клиентов за счет снижения тарифов для всех и замены старых телефонов на новые – неподходящий вариант. Вместо этого компании нацеливают свои предложения на тех клиентов, которые могут уйти в ближайшем будущем. Цель - идентифицировать клиента, который собирается сменить оператора, и попытаться убедить его остаться, предлагая новый телефон взамен старого или выгодный тарифный план.

Проблема выявления клиентов, которые могут уйти в ближайшем будущем, называется *прогнозированием оттока.* Задача прогнозирования состоит в том, чтобы классифицировать клиента, подпадает он под риск оттока или нет. Компании в телекоммуникационной, коммунальной, банковской, страховой и других отраслях используют этот вид анализа для прогнозирования оттока клиентов. Растущая сфера применения – прогнозирование текучести кадров или оттока персонала.

Классификация – это метод машинного обучения с учителем, в ходе которого берется набор данных с помеченными экземплярами и строится модель классификации. Помеченный набор данных называется обучающим. Он состоит из объектов, целевой результат которых уже известен. В одних случаях назначить метку оттока для записи клиента несложно. Например, клиент сам связался с компанией и недвусмысленно отменил свою подписку или контракт. В других случаях вероятность оттока может быть неявной. Определить, собирается ли клиент с таким типом контракта прекратить пользование услугами, бывает непросто. После того как факт оттока был установлен с точки зрения бизнеса, необходимо реализовать это определение в коде, чтобы назначить целевую метку клиенту в наборе данных.

Другим фактором, усложняющим прогнозирование оттока, является необходимость учета временных задержек. Например, модель может быть обучена предсказывать, что клиент уйдет в течение одного или двух месяцев, в зависимости от скорости предпринятых бизнесом мер по его удержанию. Определение итогового периода влияет на то, какие данные следует использовать в качестве входных для модели.

Почти во всех моделях склонности клиентов в качестве атрибутов используется демографическая информация: возраст, пол, род занятий и т.д. В телекоммуникационных моделях оттока клиентов также могут присутствовать атрибуты, характерные для этой отрасли. Например, средний счет клиента,

изменения сумм счетов, привычки, превышение количества минут тарифного плана, соотношение вызовов внутри сети и за ее пределами, подробности, касающиеся телефонного аппарата и проч. В Южной Корее полезным оказался атрибут, описывающий зависимость оттока клиентов от возраста телефонного аппарата. В Канаде этот атрибут стал бесполезным. Причина такой разницы заключалась в том, что в Южной Корее оператор мобильной связи предлагал большие скидки на мобильные телефоны только новым клиентам, тогда как в Канаде такие же скидки предлагались как новым, так и действующим клиентам. В результате в Южной Корее устаревание телефона приводило к оттоку клиентов, которые были заинтересованы в том, чтобы перейти к другому оператору за новыми скидками, а в Канаде такого стимула не было.

После создания маркированного набора данных начинается построение модели классификации с использованием алгоритма машинного обучения.

Модели прогнозирования могут также определять степень достоверности прогноза. Этот показатель называется вероятностью прогноза и принимает значение от нуля до единицы. Чем оно выше, тем выше вероятность того, что прогноз верен.

Регрессия

Ценовое прогнозирование – это задача оценки стоимости товара в определенный момент времени. Товаром может быть автомобиль, дом, баррель нефти, акции или медицинская процедура. Очевидно, что качественное ценовое прогнозирование будет востребовано любым, кто рассматривает возможность покупки товара.

Ценовое прогнозирование включает в себя оценку значения непрерывного атрибута. Это решается как проблема регрессии. Проблема регрессии похожа на проблему классификации – в обоих случаях предполагается построение модели, которая может предсказать будущее значение на основании набора входных атрибутов. Отличие в том, что классификация оценивает значения категориального атрибута, а регрессия – значения непрерывного. Регрессионный анализ требует набора данных, в котором указано значение целевого атрибута для каждого из объектов.

Модель линейной регрессии является базовой. Базовая структура регрессионных моделей прогнозирования цены одинакова независимо от товара – меняется только имя и количество атрибутов. Например, для прогнозирования цены на дом входные данные должны включать в себя такие атрибуты, как размер дома, количество комнат, этажность, средняя цена $м^2$ в этом районе и т.д. Чтобы предсказать цену автомобиля, атрибуты должны включать марку, возраст автомобиля, пробег, объем двигателя и т.д. При наличии соответствующих данных алгоритм регрессии определяет, какое влияние каждый из атрибутов оказывает на окончательную цену.

Регрессионный анализ может быть использован в самых разных областях, в том числе для решения таких задач, как расчет прибыли, стоимости, объема продаж, спроса, размеров, расстояний, дозировок и объемов.

Литература

1. Карпович Е.Е. Языки программирования интеллектуальных систем [Электронный ресурс]: учебник/ Карпович Е.Е. - Электрон. текстовые данные. – М.: Издательский Дом МИСиС, 2018. - 172 с. - Режим доступа: http://www.iprbookshop.ru/84436.html. - ЭБС «IPRbooks».
2. Ясницкий Л.Н. Интеллектуальные системы [Электронный ресурс]: учебник/ Ясницкий Л.Н. - Электрон. текстовые данные. – М.: Лаборатория знаний, 2016. - 222 с. - Режим доступа: http://www.iprbookshop.ru/89033.html. - ЭБС «IPRbooks».
3. Интеллектуальные системы [Электронный ресурс]: учебное пособие для СПО/ А.М. Семенов [и др.]. - Электрон. текстовые данные. - Саратов: Профобразование, 2020. - 236 с. - Режим доступа: http://www.iprbookshop.ru/91871.html. - ЭБС «IPRbooks».
4. Нестеров С.А. Интеллектуальный анализ данных средствами MS SQL Server 2008 [Электронный ресурс] / Нестеров С.А. - Электрон. текстовые данные. – М.: ИНТУИТ, 2016. - 303 с. - Режим доступа: http://www.iprbookshop.ru/62813.html. - ЭБС «IPRbooks».

Дополнительная литература

5. Брусенцев А.Г. Анализ данных и процессов. Ч.1. Методы статистического анализа данных [Электронный ресурс]: учебное пособие/ Брусенцев А.Г. - Электрон. текстовые данные. - Белгород: БелГТУ им. В.Г. Шухова, ЭБС АСВ, 2017. - 63 с. - Режим доступа: http://www.iprbookshop.ru/92237.html. - ЭБС «IPRbooks».
6. Барский А.Б. Искусственный интеллект и логические нейронные сети [Электронный ресурс]: учебное пособие/ Барский А.Б. - Электрон. текстовые данные. - Санкт-Петербург: Интермедия, 2019. - 360 с. - Режим доступа: http://www.iprbookshop.ru/95270.html. - ЭБС «IPRbooks».
7. Курейчик, В.В., Курейчик В.М., Родзин С.И. Теория эволюционных вычислений [Текст]: [монография] - М.: Физматлит, 2012. - 260 с. – Режим доступа: https://www.rfbr.ru/rffi/ru/books/o_1780980#1
8. Родзин С.И. Искусственный интеллект [Электронный ресурс]: учебное пособие/ С.И. Родзин. - Таганрог: Технологический институт ФГОУ ВО "Южный федеральный университет" в г. Таганроге, 2015. - 148 с. - Режим доступа: https://www.elibrary.ru/item.asp?id=28809262
9. Яхъяева Г.Э. Нечеткие множества и нейронные сети [Электронный ресурс]: учебное пособие/ Яхъяева Г.Э. - Электрон. текстовые данные. – М.: Саратов: ИНТУИТ, Вузовское образование, 2017. - 320 с. - Режим доступа: http://www.iprbookshop.ru/67390.html. - ЭБС «IPRbooks».

Ресурсы Интернет

10. название ресурса №1: Войтович И.Д. Интеллектуальные сенсоры [Электронный ресурс]: учебное пособие/ Войтович И.Д., Корсунский В.М. - Электрон. текстовые данные. – М.: Саратов: ИНТУИТ, Ай Пи Ар Медиа, 2020. - 1163 с. - Режим доступа: http://www.iprbookshop.ru/89436.html. - ЭБС «IPRbooks».
11. Седова Н.А. Теория нечетких множеств [Электронный ресурс]: учебное пособие/ Седова Н.А., Седов В.А. - Электрон. текстовые данные. - Саратов: Ай Пи Ар Медиа, 2019. - 421 с. - Режим доступа: http://www.iprbookshop.ru/86526.html. - ЭБС «IPRbooks.
12. название ресурса №3: Прокопенко Н.Ю. Системы поддержки принятия решений [Электронный ресурс]: учебное пособие/ Прокопенко Н.Ю. - Электрон. текстовые данные. - Нижний Новгород: НГАСУ, ЭБС АСВ, 2017. - 189 с. - Режим доступа: http://www.iprbookshop.ru/80838.html. - ЭБС «IPRbooks».

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Кафедра МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ И ПРИМЕНЕНИЯ ЭВМ

Дисциплина:

Машинное обучение и биоинспирированная оптимизация

ЛЕКЦИЯ 1

Машинное обучение как направление в искусственном интеллекте. Модели представления знаний и методы вывода знаний в интеллектуальных системах

Родзин Сергей Иванович,
профессор ИКТИБ ЮФУ,
srodzin@sfedu.ru

История ИИ. Роботы

Греческая мифология: бог **Вулкан**, **Пигмалион** и **Галатея**, **Талос**.

Математики **Архит Тарентский**, **Герон Александрийский**.

Леонардо да Винчи, **Раймонд Луллий**.

Легенда о **Големе**, о **Франкенштейне**, **Железный Дровосек**, «**Терминатор**».

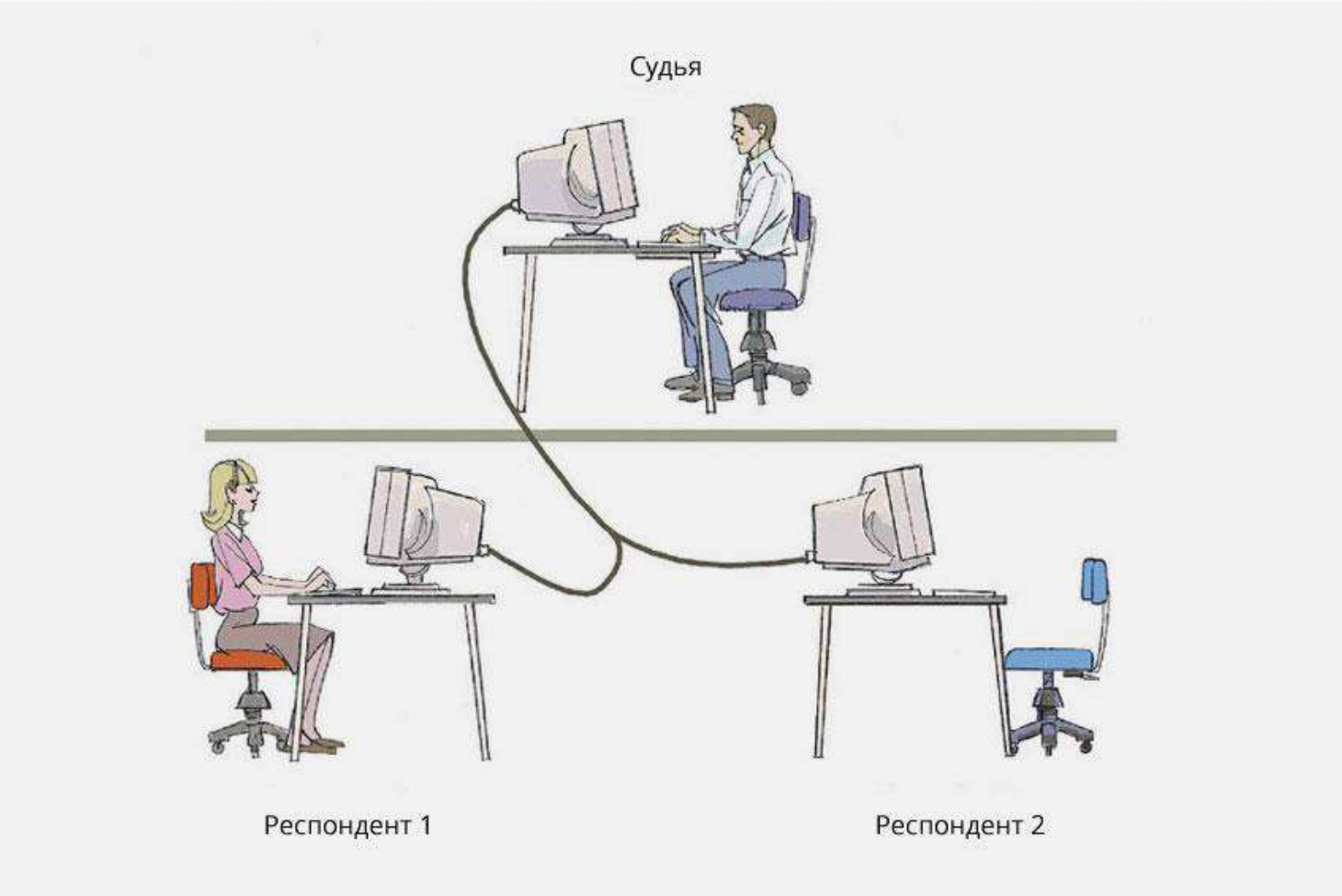
К. Чапек, пьеса «**R.U.R**» - Универсальные роботы Россума.

Алан Тьюринг *(премия Тьюринга, с 1966 г.)*

Год	Лауреат	Причина присуждения
1969	Марвин Минский	За его огромный вклад в создание, формирование, продвижение и развитие научного направления ***AI***
1971	Джон Мак-Карти	Лекция «Современное состояние исследований по **ИИ**» освещает состояние дел в области ИИ
1975	А. Ньюэлл, Г. Саймон	За фундаментальный вклад в **ИИ**, психологию механизмов человеческого восприятия и обработку списков
1994	Фейгенбаум, Редди	За первопроходческие разработки и создание крупномасштабных систем **ИИ** и демонстрацию практической важности технологий ИИ
2011	Джуда Перл	За фундаментальный вклад в ИИ путем разработки исчисления для вероятностных и причинно-следственных рассуждений
2016	Тим Бернерс-Ли	За изобретение **Всемирной паутины**, первого веб-браузера и основополагающих протоколов и алгоритмов, повлиявших на распространение Интернета
2018	Й.Бенжио, Д.Хинтон, Я.Лекун	За концептуальные и инженерные прорывы, сделавшие глубинные **нейросети** краеугольным компонентом в вычислительной технике
2020	Кэтмул, Ханрахан	За компьютерные технологии, вдохнувшие новую жизнь в анимацию, спецэффекты, виртуальную реальность и **ИИ**

Алан Тьюринг *(тест Тьюринга)*

A. Turing. Computing Machinery and Intelligence // Mind, vol. LIX, no. 236, October 1950, pp. 433—460 .



Программа «Элиза»

Терапевт: Чем я могу помочь вам?
Пациент: Я не знаю...
Терапевт: Попробуйте рассказать что-нибудь об этом.
Пациент: Нет.
Терапевт: Можно узнать причину?
Пациент: Я сегодня устал и неважно себя чувствую.
Терапевт: Почему? Есть у вас на этот счет какие-нибудь предположения.
Пациент: Да... Я стар.
Терапевт: Пожалуйста, продолжайте.
Пациент: Мне кажется, что я потерял интерес ко всему происходящему.
Терапевт: Расскажите мне подробнее о ваших ощущениях.
Пациент: У меня есть кое-какие соображения по этому поводу...
Терапевт: У вас есть соображения.
Пациент: Я думаю, что мне недостает семьи.
Терапевт: Что заставляет вас переживать из-за того, что вам недостает семьи?
Пациент: В этом есть какой-то вызов...
Терапевт: Почему это так важно для вас?
……

Рэй Курцвэйл

Шаг 1. **Сканирование мозга умершего человека методом стеклования.**

Шаг 2. **Данные со сканеров загружают в компьютер, реконструируется трехмерная нейросеть мозга. Полученную карту комбинируют с библиотекой нейровычислительных моделей.**

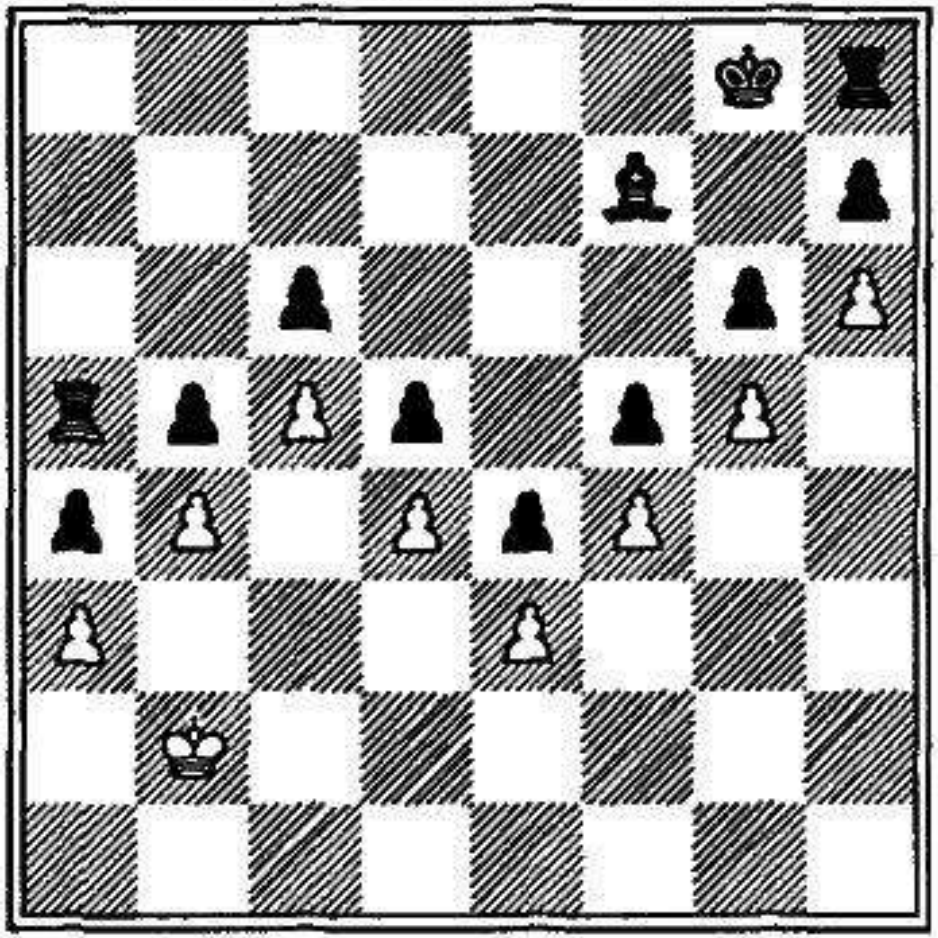
Шаг 3. **Полученная нейросеть становится цифровой копией исходного интеллекта с неповрежденной памятью и нетронутым типом личности. Разум будет существовать в виде программного обеспечения на компьютере, сможет как обитать в виртуальном пространстве, так и взаимодействовать с реальным миром при помощи роботизированных конечностей.**

Китайская комната *(Дж. Сирл)*

Теорема Пенроуза

Какой бы мощностью ни обладало устройство, имеющее архитектуру конечного автомата, человеческое мышление имеет возможности, недоступные такому устройству.

Робот, способный думать и обладающий человеческим сознанием, противоречит квантовым законам. Человеческий мозг неповторим. Теорема Гёделя о неполноте доказала, что арифметика неполна, так принцип неопределенности Гейзенберга докажет, что машины в принципе не способны думать по-человечески. Мозг – это не компьютер.



Подход «сверху-вниз»

Воспроизведение в компьютере когнитивных способностей человека без обращения к уровню отдельных нейронов.

Проблемы:

- Распознавание образов
- Программирование законов «здравого смысла»

Постулат кибернетики черного ящика:

- **Для ИИ не важно, как он устроен. Главное, чтобы на заданные воздействия он реагировал бы так же, как и мозг человека.**

По сути, это постулат Тьюринга, который считал, что **все проблемы, решаемые людьми, могут быть сведены к набору алгоритмов**.

Подход «снизу-вверх»

Построение интеллекта от нейронов к общим уровням когнитивных процессов.

Проблемы:

- у человека примерно 100 миллиардов нейронов
- нейросеть из нескольких сотен нейронов уже считается выдающейся

Постулат нейроинформатики:

- **Единственный объект, способный мыслить — это человеческий мозг. Поэтому любое «мыслящее» устройство должно каким-либо образом воспроизводить структуру человеческого мозга.**

Проект ЮФУ: Центр "Искусственный интеллект"

- **Интеллектуальные средства взаимодействия для лиц с ограниченными возможностями на основе конвергенции с технологиями автономных мобильных роботов;**
- **Каналы коммуникации и управления на основе интерфейсов “мозг-компьютер” и “мозг-мозг”,**
- **Интеллектуальный автопилот для беспилотных авиационных систем,**
- **СИИ управления процессами производства и распределения энергоресурсов,**
- **Системы интеллектуального анализа больших данных в социально-экономических системах.**

Национальная стратегия развития ИИ в России

«Комплекс технологических и программных решений, приводящих к результату, аналогичному или превосходящему результат интеллектуальной деятельности человека (в том числе способности к самообучению), и используемых для решения прикладных задач на основе больших данных с помощью следующих систем: компьютерное зрение, обработка ЕЯ, распознавание и синтез речи, рекомендательные системы и ИСППР, перспективные методы и технологии ИИ».

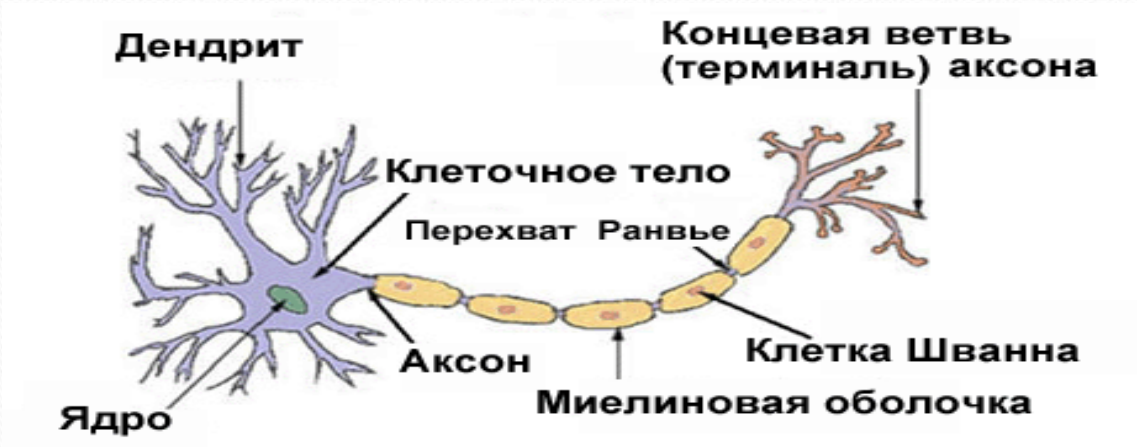
Выделены 6 движущих факторов развития ИИ:

- Алгоритмы и математические методы;
- ПО;
- Данные, работа с данными, регулирование и использование данных;
- Аппаратное обеспечение;
- Образование и кадры;
- Нормативное регулирование.

По каждому из факторов сформулированы цели

Постулат нейроинформатики: *Единственный объект, способный мыслить — это человеческий мозг. Поэтому любое «мыслящее» устройство должно каким-либо образом воспроизводить структуру человеческого мозга.*

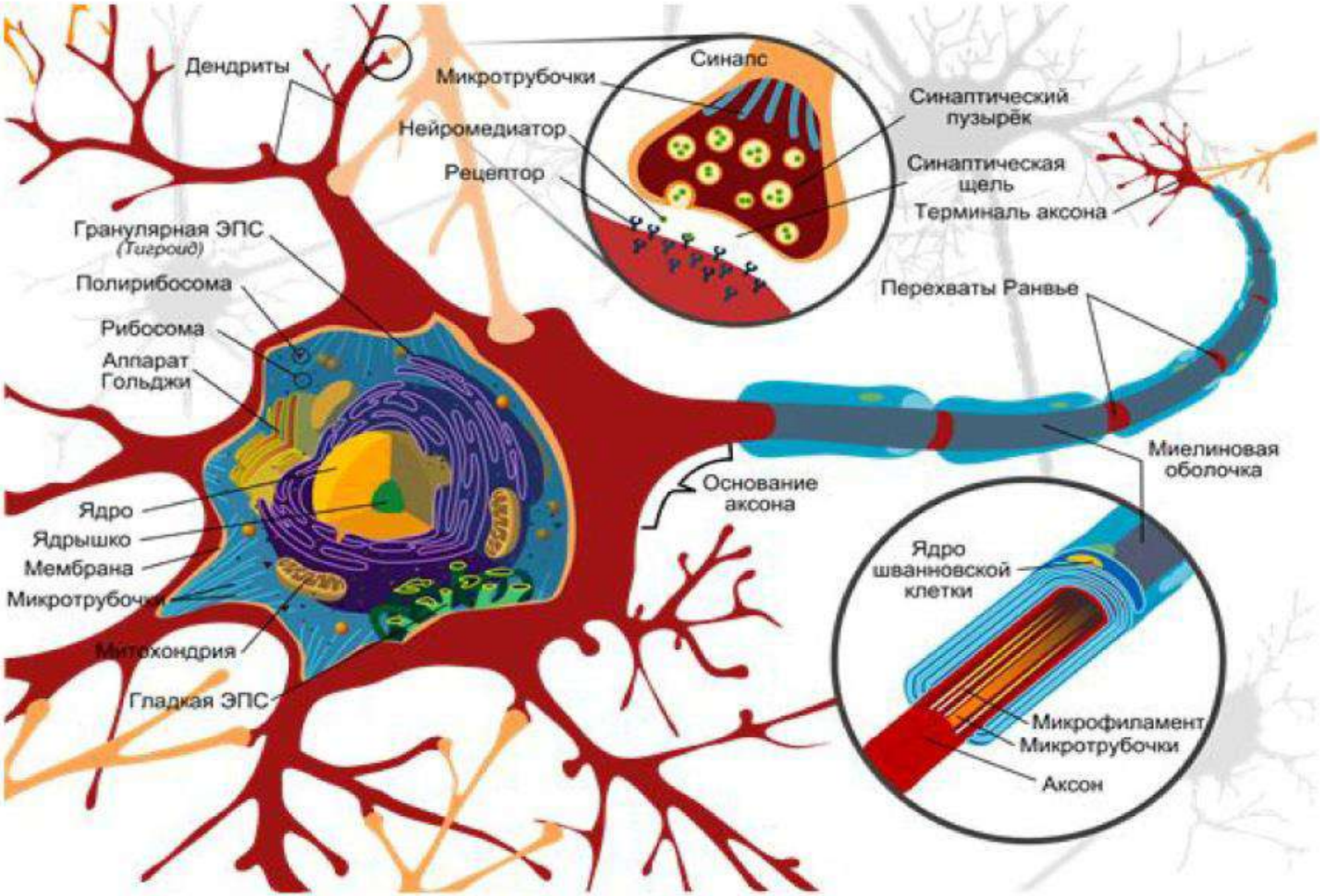
Биологический нейрон мозга:



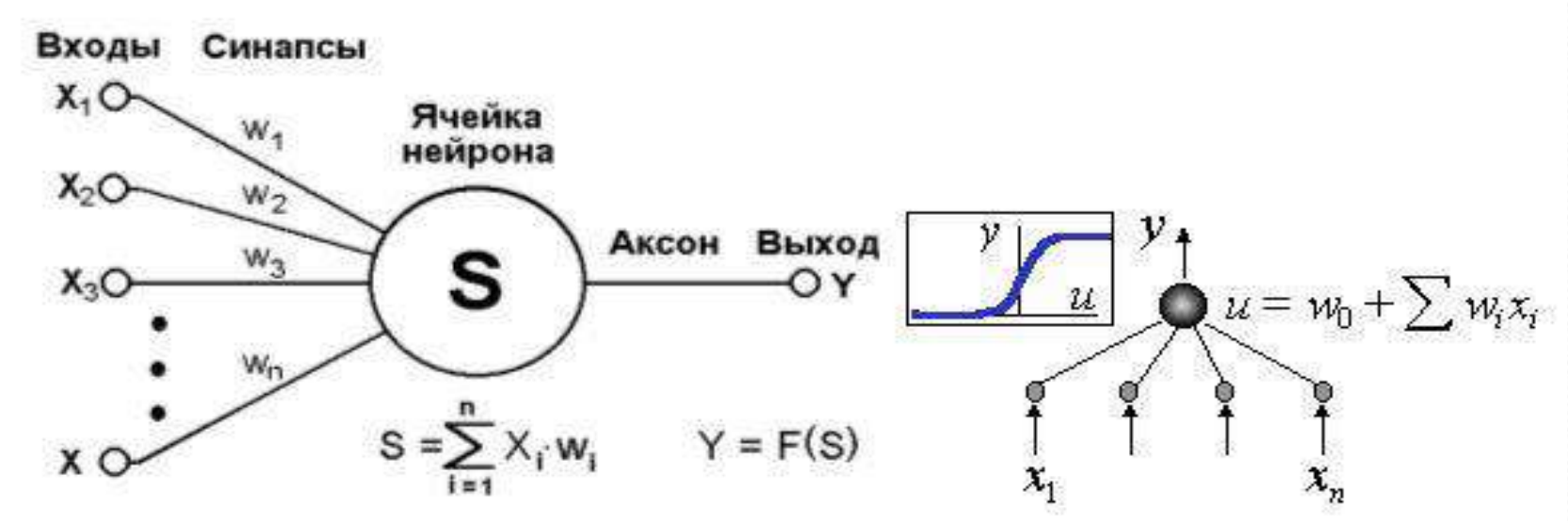
Параметры мозга как системы обработки информации:

Количество нейронов неокортекса примерно 100 млрд. Неокортекс: толщина 2-4 мм, площадь около 2200 см^2. Потребляемая мощность мозга: 1 – 25 Вт. Нейрон имеет в среднем 10000 связей. 100 триллионов синапсов. Нервный импульс из мозга движется со скоростью 274 км/час. Мозг – тихоход (время реакции нейрона несколько миллисекунд). Емкость мозга человека превышает 4 терабайта. Мозг генерирует больше электрических импульсов в течение одного дня, чем любой мобильный телефон.

В реальности всё посложнее..



Математическая модель нейрона:



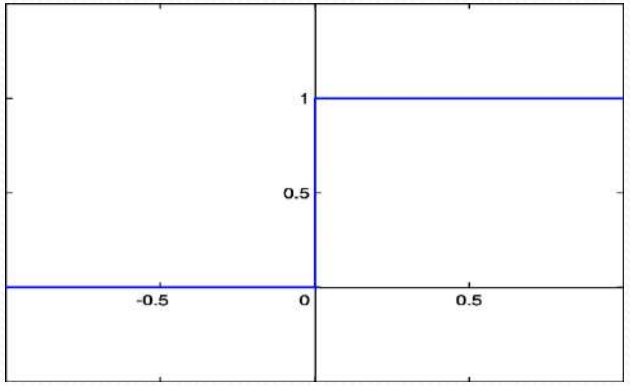
Здесь x_i - сигналы на входах нейрона, w_i - веса входов. Сигналы на входах нейрона считают заданными в интервале [0, 1]. Они могут быть либо дискретными (0 или 1), либо аналоговыми. Дополнительный вход x_0 и его вес w_0 подразумевают смещение активационной функции нейрона по горизонтальной оси.

Функция $f(u)$ называется передаточной или активационной. Эта функция определяет зависимость сигнала на выходе нейрона от взвешенной суммы сигналов на его входах. В большинстве случаев она является монотонно возрастающей и имеет область значений [-1, 1] или [0, 1]. Нейрон полностью характеризуется своей передаточной функцией. Использование различных передаточных функций позволяет вносить нелинейность в работу нейрона и в целом нейронной сети.

Передаточные (активационные) функции нейрона:

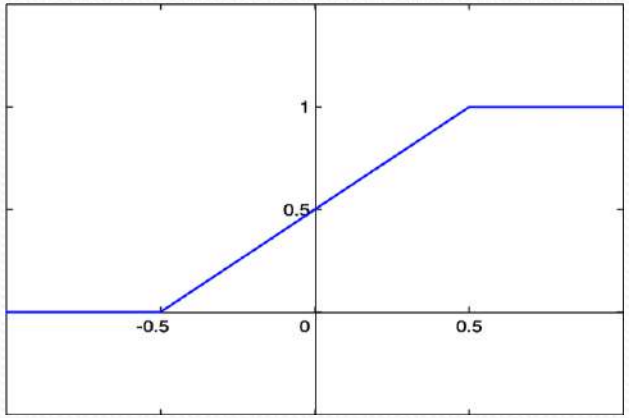
Пороговая функция

$$Y(x) = \begin{cases} 1, & если \;\; x \geq \Theta, \\ 0, & иначе. \end{cases}$$



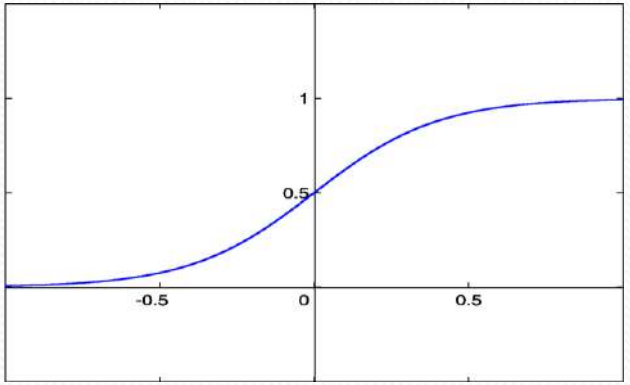
Линейная функция

$$Y(x) = \begin{cases} 0, & если \;\; x \leq -0{,}5 \\ 1, & если \;\; x \geq 0{,}5 \\ x, & иначе. \end{cases}$$

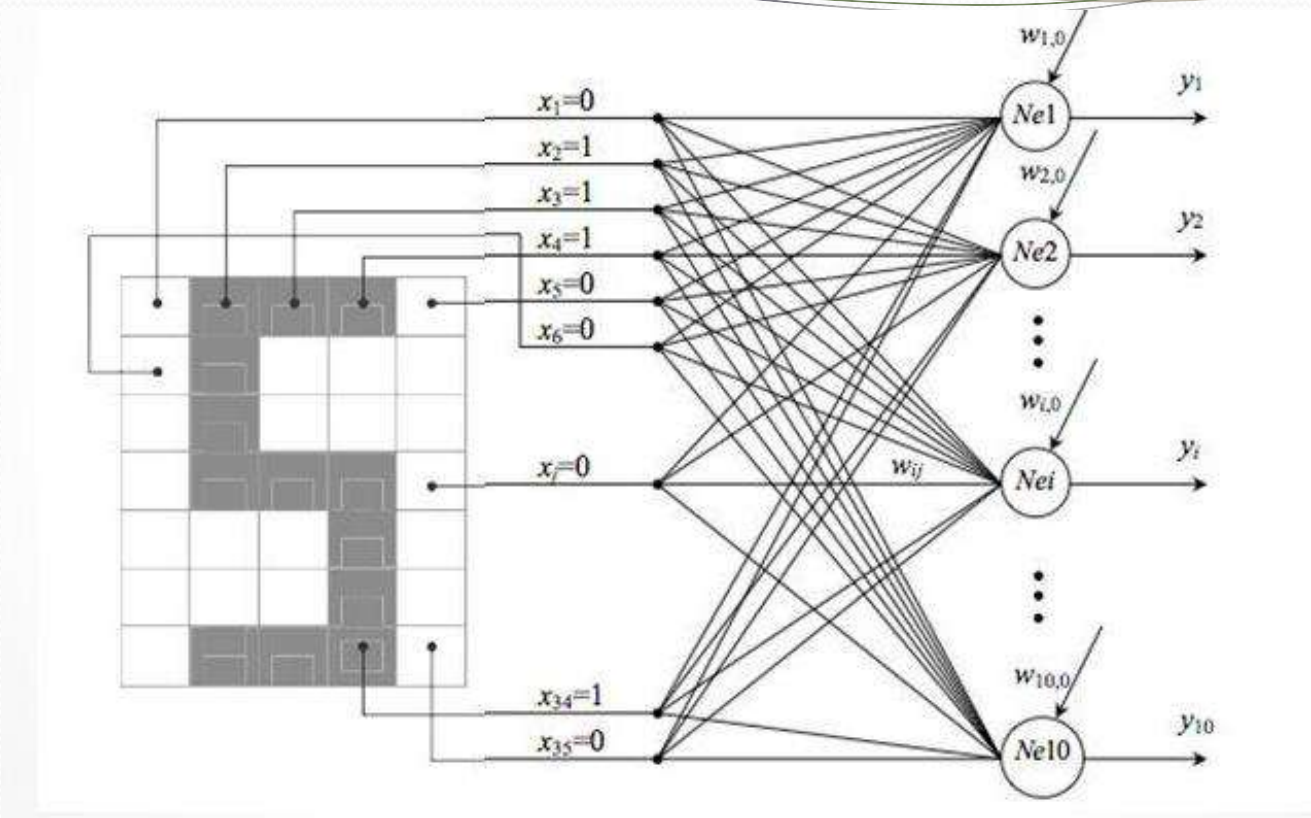


Логистическая функция

$$Y(x) = \frac{1}{1 + \exp(-ax)},$$



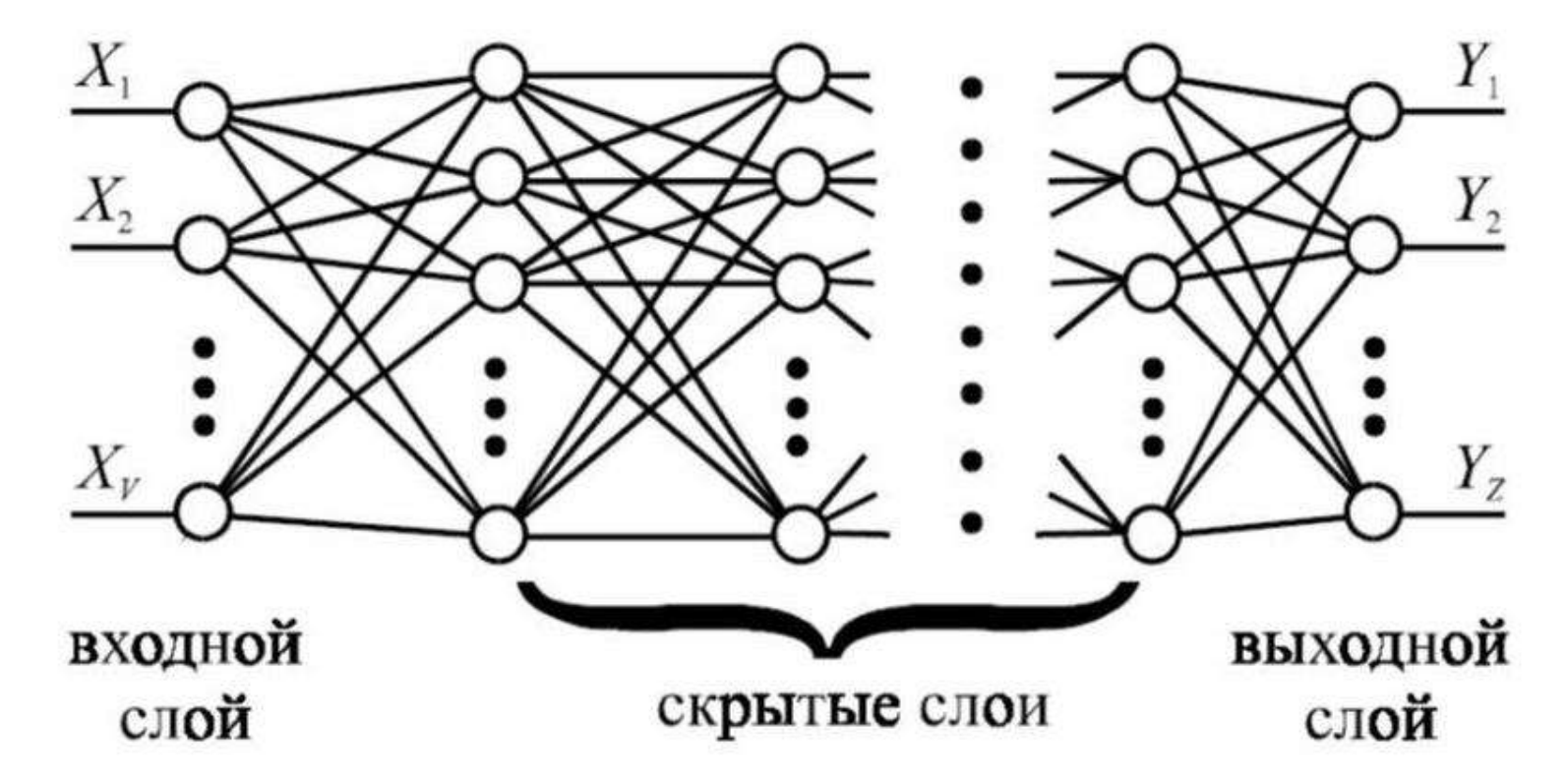
Правило Хебба



Задача распознавания цифр. На фотопластину накладывается карточка с изображением цифры. Фотоэлемент срабатывает (x_i = 1), если на него попадает фрагмент цифры, иначе – x_i = 0. Каждый из 10 нейронов выполняет взвешенное суммирование входных сигналов и пороговый активационный элемент вырабатывает выходной сигнал (0 или 1). Обучение происходит по *правилу Хебба*:

- *если выход неправильный и равен 0, то увеличить веса для* x_j=1;
- *если выход неправильный и равен 1, то уменьшить веса для* x_j=1.

Многослойная нейросеть прямого распространения



Сеть состоит из множества входных узлов, которые образуют входной слой; одного или нескольких скрытых слоев нейронов и одного выходного слоя нейронов

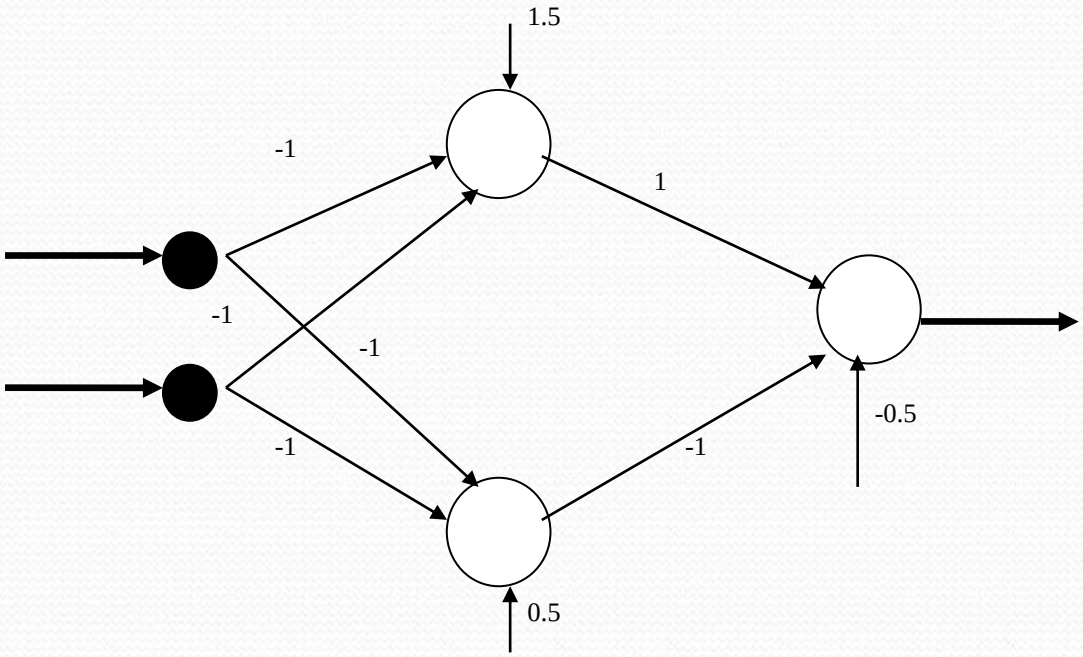
Многослойная нейронная сеть может моделировать функцию практически любой степени сложности, причем число слоев и число элементов в каждом слое определяют сложность функции.

Пример многослойной нейросети

Сеть содержит два входных элемента, два скрытых элемента и один выходной элемент. Все связи идут от входного слоя к выходному. К нейронам скрытого слоя доступа нет.

$$e_j = \sum_{i=0}^{n} x_i * w_{ij}$$

$$y(e_j) = \begin{cases} 1, & \text{если } e_j \geq 0, \\ 0, & \text{если } e_j < 0. \end{cases}$$



Пусть $x_1 = 1$, $x_2 = 1$. Для первого скрытого элемента со смещением 1.5 получаем

$e_1 = x_0 \times 1.5 + x_1 \times (-1) + x_2 \times (-1) = 1 \times 1.5 + 1 \times (-1) + 1 \times (-1) = -0.5$. На его выходе 0.

Для второго скрытого нейрона со смещением 0.5 получаем

$e_2 = 1 \times (-0.5) + x_1 \times (-1)) + x_2 \times (-1) = (1 \times 0.5 + 1 \times (-1) + (1 \times (-1) = -1.5$. На его выходе 0.

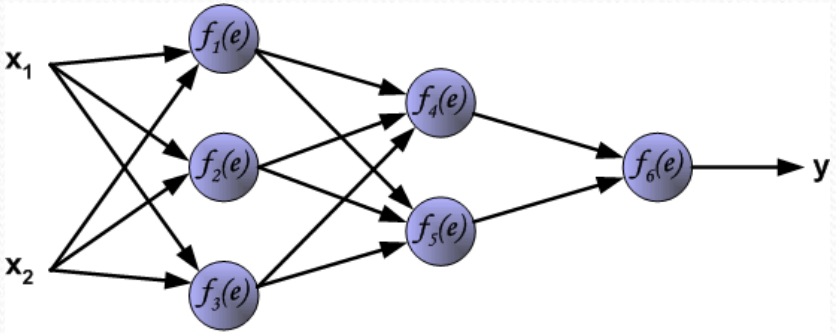
Для выходного элемента получаем

$e_3 = 1 \times (-0.5) + 0 \times (1) + 0 \times (-1) = -0.5 + 0 + 0 = -0.5$, поэтому на выходе сети будет 0.

Если процедуру повторить для трех оставшихся пар входов, то можно убедиться в том, что вывод представленной выше сети соответствует функции *XOR*.

Алгоритм *back propagation*

Сеть содержит два входных элемента, два скрытых элемента и один выходной элемент. Все связи идут от входного слоя к выходному. К нейронам скрытого слоя доступа нет.



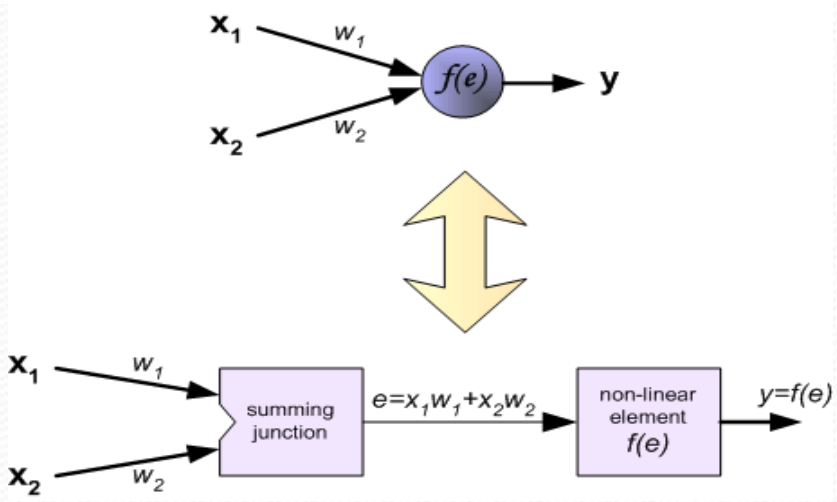
Шаг 1. Обычно начальные значения весов всех нейронов всех слоев полагаются случайными числами. Формируется обучающая выборка (ОВ).

Шаг 2. Сети из ОВ предъявляется образ ***X***, сигнал от которого распространяется через нейроны скрытого слоя до выхода:

$$\boldsymbol{y_1} = f_1(w_{(x1\ \ )1}x_1 + w_{(x2\ \ )1}x_2)$$

$$\boldsymbol{y_2} = f_2(w_{(x1\ \ )2}x_1 + w_{(x2\ \ )2}x_2)$$

$$\boldsymbol{y_3} = f_3(w_{(x1\ \ )3}x_1 + w_{(x2\ \ )3}x_2)$$



$$\boldsymbol{y_4} = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3),\ \boldsymbol{y_5} = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3),\ \boldsymbol{y}\ \ = f_6(w_{46}y_4 + w_{56}y_5).$$

Шаг 3. Выходной сигнала ***y*** сравнивается с желаемым выходным сигналом ***z***, который хранится в ОВ. Разница между этими двумя сигналами называется *ошибкой* δ (дельта) выходного слоя сети:

$$\boldsymbol{\delta = z - y}$$

Невозможно непосредственно вычислить сигнал ошибки для нейронов скрытого слоя, т.к. выходные значения этих нейронов, неизвестны.

Алгоритм *back propagation* (продолжение)

Шаг 3. Идея заключается в распространении сигнала выходной ошибки δ обратно на все нейроны вплоть до входов:

$$\boldsymbol{\delta_5} = w_{56}\delta, \qquad \boldsymbol{\delta_4} = w_{46}\delta,$$

Если ошибка пришла от нескольких нейронов – она суммируется:

$$\boldsymbol{\delta_3} = w_{34}\delta_4 + w_{35}\delta_5, \qquad \boldsymbol{\delta_2} = w_{24}\delta_4 + w_{25}\delta_5, \qquad \boldsymbol{\delta_1} = w_{14}\delta_4 + w_{15}\delta_5$$

Функционал квадратичной ошибки сети для данного входного образа имеет вид:

$$\boldsymbol{E} = \frac{1}{2}\delta^2$$

Данный функционал подлежит ***минимизации***. Классический ***градиентный*** метод оптимизации состоит в итерационном уточнении аргумента (t: = t + 1) согласно формуле:

$$\boldsymbol{w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j \frac{df_j(e)}{de} y_i}$$

В этой формуле $\frac{df_j(e)}{de}$ – производная функции активации нейрона, чьи веса корректируются. Поэтому функция должна иметь первую производную. Чаще всего в качестве функции активации используется сигмоид:

$$\boldsymbol{f(e)} = \frac{1}{1 + \exp(-e)}$$

производная от ***f(e)*** выражается через саму функцию:

$$\frac{df}{de} = f(e) * (1 - f(e))$$

Это позволяет существенно сократить вычислительную сложность метода.

Алгоритм *back propagation* (продолжение)

Шаг 4. Корректировка весов:

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1 \qquad w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$

$$w'_{(x1)2} = w_{(x1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1 \qquad w'_{(x2)2} = w_{(x2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2$$

$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1 \qquad w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1 \qquad w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2 \qquad w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$

$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1 \qquad w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2 \qquad w'_{35} = w_{14} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$

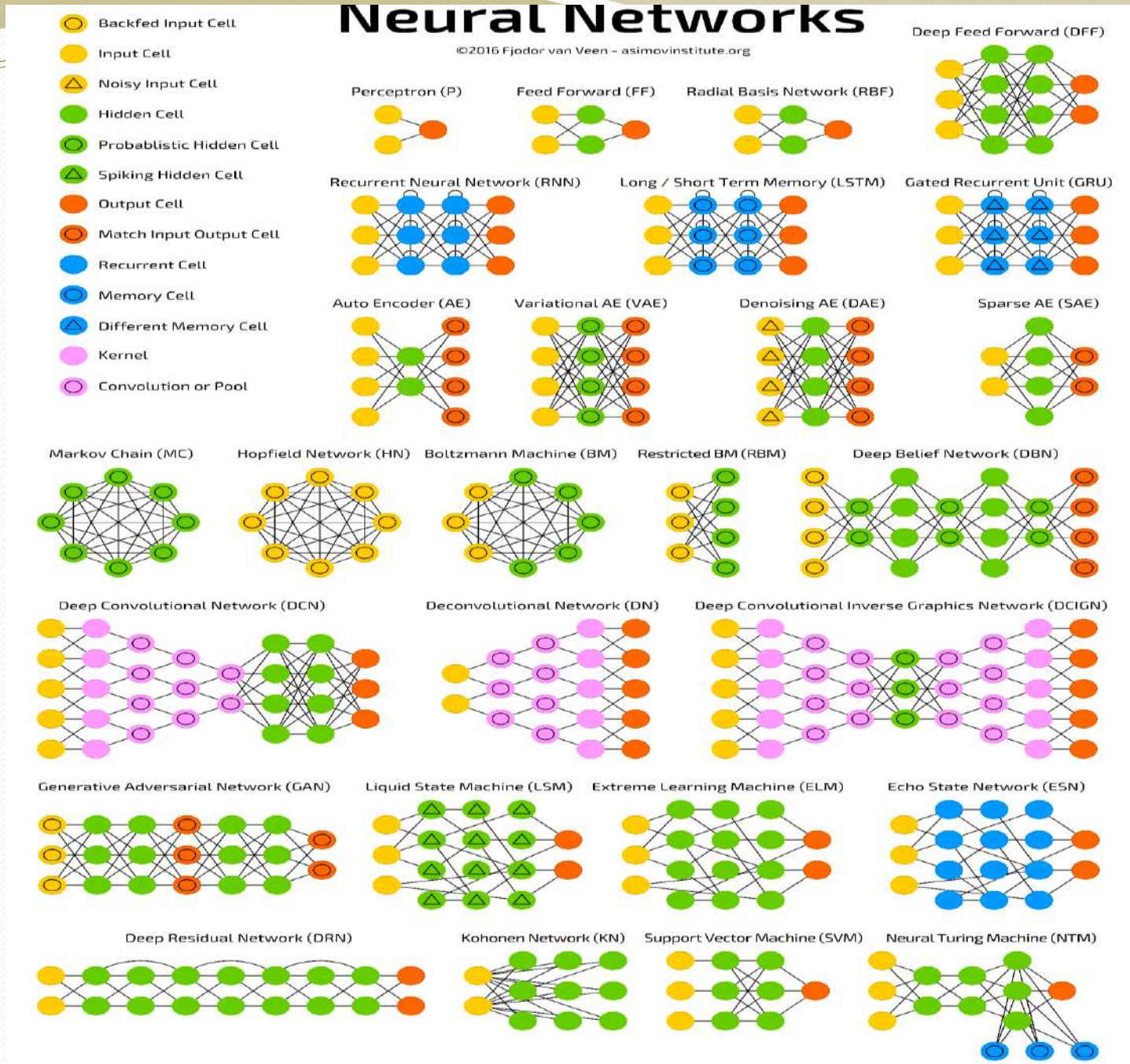
$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4 \qquad w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$

Коэффициент **η (эта)** влияет на скорость обучения сети. Есть несколько методов для выбора этого параметра. Первый способ – начать обучение с большим значением параметра **η**. Во время коррекции весовых коэффициентов, параметр постепенно уменьшают. Второй – более сложный способ обучения, начинается с малым значением параметра **η**. В процессе обучения параметр увеличивается, а затем вновь уменьшается на завершающей стадии обучения.

Проблема переобучения (*оверфиттинг* или слишком близкая подгонка).

Шаг 5. Шаги 2-4 повторяются для всех обучающих примеров. Обучение завершается по достижении малой квадратичной ошибки ***E*** или максимально допустимого числа итераций

Практика показывает, что сходимость *back propagation* медленная. Алгоритм применятся для обучения с учителем. Для обучения без учителя - алгоритмы Хебба и Кохонена. В последнее время, сформировалась еще одна парадигма – обучение с подкреплением.

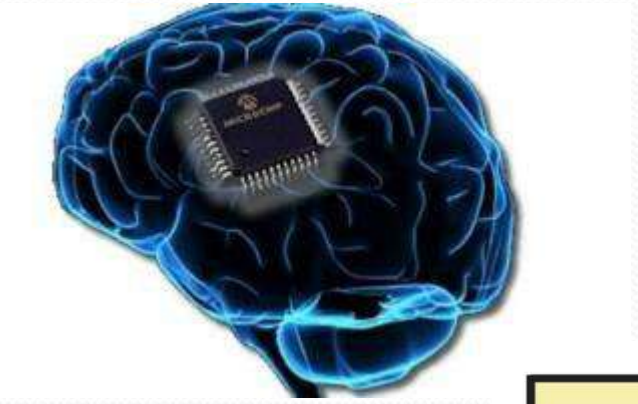

Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Отличия между биологическими нейросетями и ЭВМ

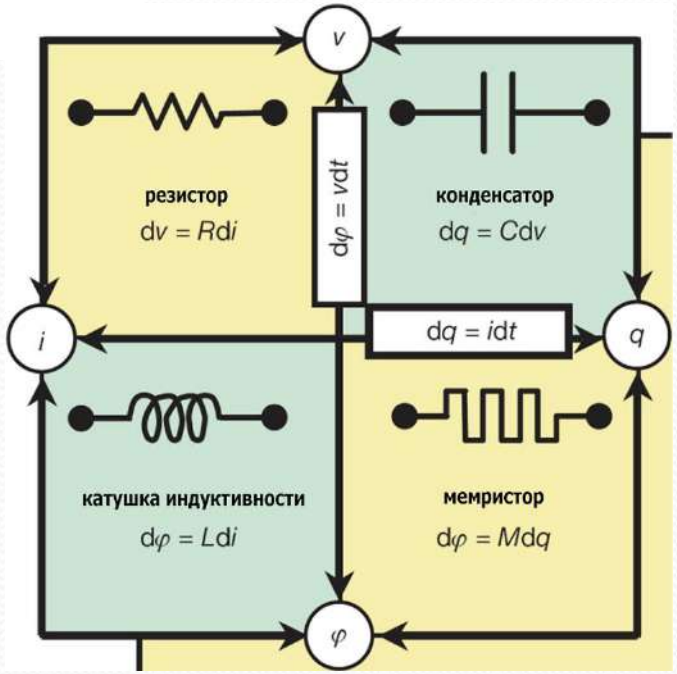
	Машина фон Неймана	Биологическая нейронная система
Процессор	Сложный	Простой
	Высокоскоростной	Низкоскоростной
	Один или несколько	Большое количество
Память	Отделена от процессора	Интегрирована в процессор
	Локализована	Распределенная
	Адресация не по содержанию	Адресация по содержанию
Вычисления	Централизованные	Распределенные
	Последовательные	Параллельные
	Хранимые программы	Самообучение
Надежность	Высокая уязвимость	Живучесть
Специализация	Численные и символьные операции	Проблемы восприятия
Среда функционирования	Строго определенная	Плохо определенная

Нейрочипы

- Нейрочипы на ПЛИС,
- Цифровые сигнальные процессоры (DSP),
- Гибридный биологический нейрочип, способный записывать сигналы головного мозга,
- Нейрочипы для мозга,



- Нейрочипы на мемристорах (у мемристоров сопротивление непостоянно и зависит от силы тока. Чем больше она была, тем ниже становится сопротивление на какое-то время. Однако оно не падает до 0 и сохраняется даже когда ток перестаёт протекать, что делает мемристоры вариантом энергонезависимой памяти),

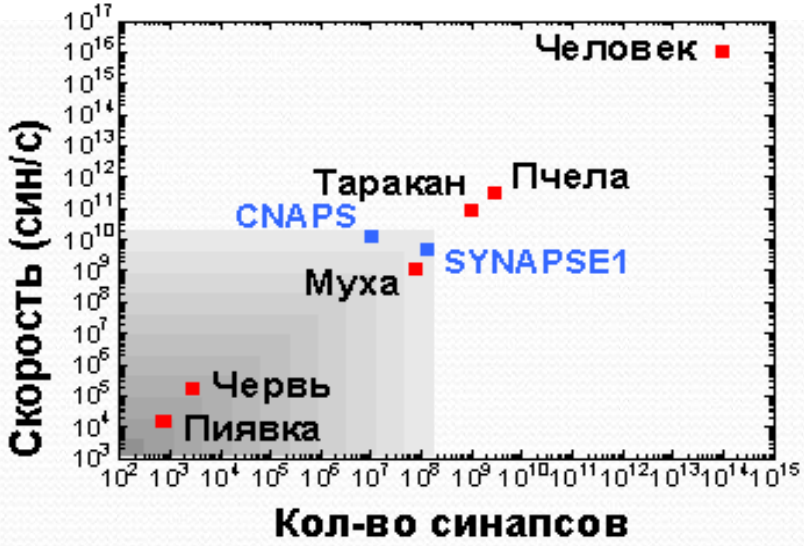


Нейрокомпьютеры

Сравнительные характеристики некоторых нейрокомпьютеров

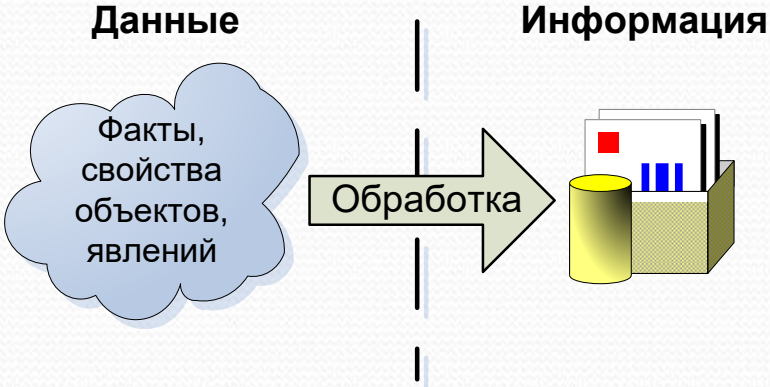
Название нейросхемы, фирма-производитель	Тип	Количество процессорных элементов	Производит-сть (умножений с суммированием в с),	$
CNAPS/PC (Adaptive Solutions)	PC-ускоритель	2 *CNAPS*-1016 процессора (128 нейронов)	$2,5*10^9$	10^4
CNAPS (Adaptive Solutions)	Нейрокомпьютер	8 *CNAPS*-1016 процессоров (512 нейронов)	10^{10}	10^5
SYNAPSE-1 (Siemens)	Нейрокомпьютер	8 MA-16 процессоров (512 нейронов)	$3*10^9$	10^5

Сравнительные возможности искусственных и природных нейросетей



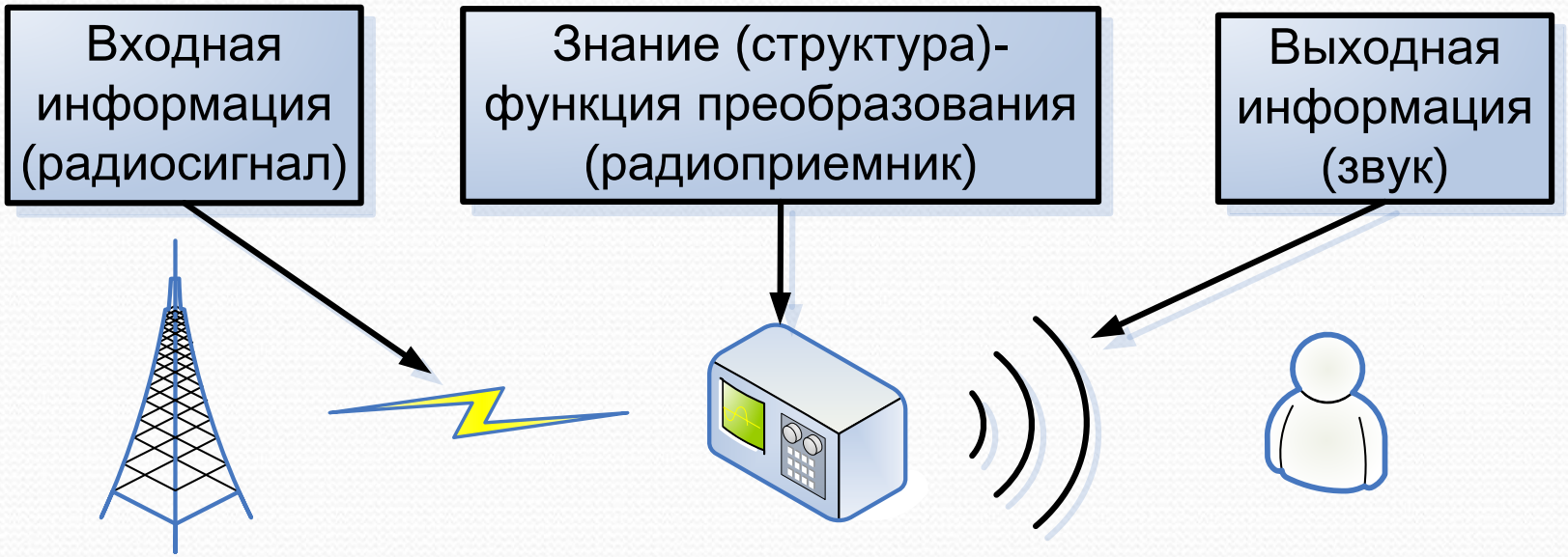
Данные→Информация→Знания

Данные - это факты или любые сигналы, характеризующие информационные свойства объекта, которые наблюдаются, но не используются, а только хранятся.



Информация – это мера сложности обработанных системой данных, которые используются для генерации знаний, уменьшая их неполноту.

Знание - это обработанная информация.



Данные

Это необработанная информация (факты, сообщения, сигналы и т.п.). Данные объективны, а методы их обработки субъективны.

Много ли в мире накоплено цифровых данных? - В 2015-16 объем цифровой информации, созданной человечеством, составил **4 зетабайт** ($4*10^{21}$) данных. К 2020 ожидается увеличение этого объема до 40 зетабайт. При этом 90% накопленной информации было создано в течение последних 5 лет. Это и называется информационным взрывом!

Основными точками роста сегмента разработки ПО на ближайшие годы станут

- «Облачные" технологии,
- Системы автоматизации бизнеса,
- Технологии обработки больших массивов данных (Big Data),
- Приложения для мобильных устройств.

Характеристики Big Data (3V): объем, скорость прироста, многообразие типов:

- В 2003 самой большой считалась база данных объёмом 25 терабайт,
- К 2014 компания Google хранила на своих серверах до 10-15 экзабайт (10^{18}) данных,
- К 2025 году генетики будут располагать данными о геномах до 2 миллиардов человек, и для хранения подобного объёма данных потребуется 40 экзабайт.

Что такое информация ?

Информация – это обработанные данные. Информация, в отличие от данных, имеет смысл. Как измеряется информация? **Мера Хартли**: $I = k\log_a N$, где k – коэфф. масштабирования, a – основание рассматриваемой системы, N – число равновероятных состояний системы.

Пример. Имеются 50 монет, из которых одна фальшивая. Определим сколько взвешиваний нужно произвести, чтобы определить ее. Если положить на весы равное количество монет, то получим 3 возможности: а) левая чашка ниже; б) правая чашка ниже, в) чашки в равновесии. Таким образом, каждое взвешивание дает количество информации $I=\log_2 3$. Следовательно, для определения фальшивой монеты нужно сделать не менее k взвешиваний, где k удовлетворяет условию $\log_2 3^k \geq \log_2 50$. Отсюда, $k \geq 4$ (надо сделать не менее 4 взвешиваний).

Мера К. Шеннона: $I = -\sum_{i=1}^{N} p_i \log_2 p_i$, где p_i - вероятность перехода системы в i-ое состояние, причем $\sum_{i=1}^{N} p_i = 1$. Если все состояния равновероятны (т.е. $p_i=1/N$), то $I=\log_2 N$, т.е. мера Шеннона совпадает с мерой Хартли. Сообщение о наступлении события с меньшей вероятностью несёт в себе больше информации, чем сообщение о наступлении события с большей вероятностью. В термодинамике известна аналогичная формула Больцмана, которая известна как *энтропия* или мера хаоса, беспорядка в системе.

Информация также характеризуется скоростью приема, полнотой, доступностью, своевременностью, точностью, достоверностью и т.д.

при распространении информации проявляется такое ее свойство, которое не присуще материальным объектам – **отсутствие закона сохранения**.

Как классифицируются знания?

- По степени общности: **абстрактные** и **конкретные** знания (абстрактные применимы в разных ПО, конкретные – в одной);

- По способу представления: **фактические** и **концептуальные** (фактические задаются совокупностью фактов, концептуальные – обобщающим понятием);

- По форме представления: **образные** и **знаковые** (образные реализуют отношение целое-часть; знаковые – отношение часть-целое);

- По изменчивости во времени: **статические** и **динамические**;

- По способу существования: **хранимые** и **воспроизводимые**;

- По языку описания: **содержательные** (на языке ПО) и **формальные** (на математическом языке);

- По степени истинности: **теоретические** и **эвристические**;

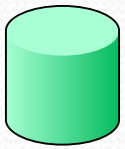
- По квалификация источника: **обыденные** и **экспертные**;

- По степени соответствия реальности: **достоверные** и **правдоподобные**;

- По способу размещения в пространстве: **локальные** и **распределенные**;

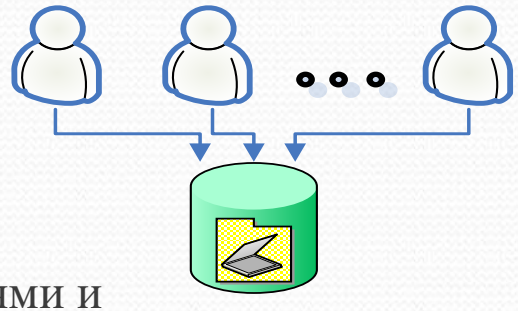
- По способу программирования: **процедурные** (информация о свойствах и фактах предметной области) и **декларативные** (информации о способах решения задач в проблемной области, а также различные инструкции, методики и т.п.).

Онтология и База знаний



Под ***базой знаний*** понимается некоторая совокупность информации, образующей целостное описание предметной области с доступной степенью детализации.

Под ***онтологией*** понимается база знаний специального типа, которые могут читаться, пониматься, отчуждаться от разработчика и/или физически разделяться между пользователями.



Целостность предполагает наличие связей между знаниями и выводимость одних знаний из других.

База знаний (БЗ) должна удовлетворять требованиям ***полноты*** и ***непротиворечивости***.

Полнота — это отношение, выводимое из модели знаний с помощью каких-то определенных операций и полученное с помощью средств БЗ.

Непротиворечивость — это свойство, которое означает:

- совпадение значений одних и тех же знаний и данных, расположенных в разных частях распределенной БЗ или БД;
- синтаксическое и семантическое совпадение значений данных и знаний с их описаниями.

Проект Cyc

Cyc – энциклопедическая онтология, позволяющей решать сложные задачи из области ИИ на основе логического вывода и привлечения здравого смысла. Типичным примером знаний в базе являются «Всякое дерево является растением» и «Растения смертны». Если спросить «Умирают ли деревья?», машина вывода может дать правильный ответ. БЗ на языке CycL, который основан на исчислении предикатов и имеет схожий с ЛИСПом синтаксис.

Проект Cyc (Ленат) - ответ на японский проект ЭВМ «пятого поколения».

Идея проекта. Ленат представил в трех основных пунктах.

- Разработать язык и закачать в хранилище миллионы терминов, фактов и правил, составляющих интуитивное знание ("здравый смысл"). Создать "насос знаний".
- Разработать систему общения с компьютером на естественном языке.
- Обеспечить новому сверхмозгу возможность саморазвития (2000- е гг.).

К настоящему времени частично реализован только первый пункт.

Сегодняшний день Cyc. Сейчас размер базы знаний Cyc по разным источникам составляет от 1 до 2 млн статей. Разработчиками созданы собственные продукты:

- *Cyc Knowledge Base* (KB). Формальное представление фундаментальных человеческих знаний, на языке CycL. База знаний состоит из терминов, образующих словарь CycL и высказываний, сделанных на этом языке. Высказывания включают простые утверждения и правила. Cyc KB содержит десятки тысяч терминов.
- *Cyc Inference Engine*. Уникальная система выводов знаний в Cyc, представляющая собой механизм для реализации дедуктивных умозаключений.
- *Cyc Secure KB*. Представляет собой собрание знаний об уязвимых местах и способах хакерских атак (по слухам, активно используется Пентагоном и спецслужбами США).

На сайте *opencyc.sourceforge.net* появилась свободно распространяемая версия OpenCyc. Появление в свободном доступе OpenCyc открывает новые возможности для исследований.

Свойства знаний

1). **Внутренняя интерпретируемость знаний.**

Данные в памяти лишены имен и могут идентифицироваться только программой, извлекающей их из памяти. Что скрывается за тем или иным двоичным кодом машинного слова, системе неизвестно. При переходе к знаниям каждая информационная единица должна иметь уникальное имя, по которому ИС находит ее, а также отвечает на запросы, в которых это имя упомянуто. Если, например, в память ЭВМ нужно записать сведения о сотрудниках учреждения, то без внутренней интерпретации в память была бы занесена совокупность из четырех машинных слов, соответствующих строкам этой таблицы:

Фамилия	**Год рождения**	**Специальность**	**Стаж работы**
Попов	1975	Экономист	15
Сидоров	1981	Радиотехник	10
Иванов	1962	Юрист	30
Петров	1987	Программист	5

Информация о том как закодированы сведения у системы отсутствует. Ее знает лишь программист. Система не может ответить на вопрос "Что известно о Попове?" или "Есть ли среди специалистов юрист?"

При переходе к знаниям в память ЭВМ вводится информация о некоторой *протоструктуре* информации. Например, о том в каких разрядах хранятся те или иные сведения. По ним можно осуществлять поиск нужной информации. Каждая строка таблицы будет экземпляром протоструктуры.

Свойства знаний

2). **Структурированность** – возможность включения некоторых знаний в состав других (рекурсивная вложимость) и, наоборот, выделение из знаний отдельных информационных единиц.

Иными словами, информационные единицы должны обладать гибкой структурой. Для них должен выполняться "принцип матрешки", т.е. рекурсивная вложимость одних информационных единиц в другие. Каждая информационная единица может быть включена в состав любой другой, и из каждой информационной единицы можно выделить некоторые составляющие ее информационные единицы. Другими словами, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа ***род-вид, целое-часть, элемент-класс***.

Свойства знаний

3). **Связность.**

В системе между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Эти связи могут характеризовать отношения между информационными единицами. Семантика отношений может носить декларативный или процедурный характер.

Отношения "одновременно", "причина - следствие" или "быть рядом" характеризуют декларативные знания. Если между двумя информационными единицами установлено отношение "аргумент - функция", то оно характеризует процедурное знание, связанное с вычислением определенных функций.

Между информационными единицами могут устанавливаться, например, связи, определяющие порядок выбора информационных единиц из памяти или указывающие на то, что две информационные единицы несовместимы друг с другом в одном описании.

Перечисленные три особенности знаний позволяют ввести общую модель представления знаний - семантическую сеть с вершинами, в которых находятся информационные единицы с именами. Дуги – семантические связи между ними.

Свойства знаний

4). **Семантическая метрика.**

На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее ситуационную близость информационных единиц, т.е. силу ассоциативной связи между информационными единицами.

Это отношение обычно называют отношением релевантности или смысловой близости. Такое отношение дает возможность выделять в информационной базе некоторые типовые ситуации (например, "покупка", "регулирование движения на перекрестке"). Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным .

5). **Активность.** С момента появления ЭВМ и разделения используемых в ней информационных единиц на данные и команды создалась ситуация, при которой данные пассивны, а команды активны. Все процессы в ЭВМ инициируются командами, а данные используются этими командами лишь в случае необходимости.

Для ИС эта ситуация не приемлема. Как и у человека, в ИС актуализации тех или иных действий способствуют знания в системе. Выполнение программ в ИС должно инициироваться текущим состоянием информационной базы. Появление в базе фактов, событий или установление связей между информационными единицами и понятиями предметной области может стать источником активности системы.

Понятийные знания

Знание делят на фрагменты, которые называются ***предметными областями***.

Знание характеризуется с помощью **понятий** и **образов**.

Под ***понятием*** (лат. concept) понимается класс сущностей (denotat), объединяемых на основе общности признаков.

Сущность — это объект произвольной природы, принадлежащий реальному или виртуальному миру.

Класс сущностей — это предмет, свойство (атрибут), состояние, процесс, событие, оценка события, модификатор, квантификатор, модальность. *Например:*

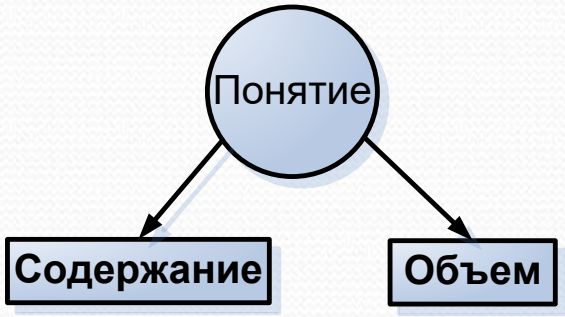
Предмет	**Программа**
Свойство	**Надежность**
Состояние	**Неисправность**
Процесс	**Тестирование**
Событие	**Запуск программы**
Оценка события	**удовлетворительно/неудовлетворительно**
Модификатор	**быстродействие**
Квантификатор	**часто**
Модальность	**возможно**

Содержание и объем понятия

Множество существенных признаков, характеризующих понятие, называется его содержанием (***интенсионалом***).

Содержание понятия задается *описанием свойств*

принадлежащих ему сущностей, например, понятие A={A1,A2,…,Ак}, где Ai — описание свойств понятия, его существенных признаков.



Пример: Стул :={Приспособление для фиксации сидячей позы, Плоская поверхность, Ножки, Спинка}.

Содержание понятия должно отвечать двум взаимно противоположным требованиям: ***полноты*** и ***неизбыточности***.

Содержание понятия зависит от границ предметной области. Для минимизации содержания понятия желательно явно указать предметную область.

Объем **- класс сущностей, объединяемых в понятие (*экстенсионал*).**

Сущности, входящие в объем, задаются ***перечислением***. Объем понятия, в отличие от содержания, может изменяться путем задания ограничений (например, число стульев в комнате измеряется единицами, хотя в мире миллиарды стульев).

Треугольник Фреге

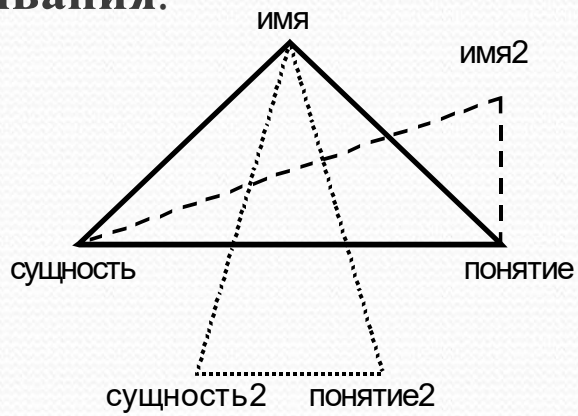
Говорят о двух основных формах представления любого понятия:

- **знаковой**
- **образной**

Носителем знаковой формы представления знаний является естественный язык (ЕЯ). Для представления понятия в ЕЯ используется слово, называемое ***именем*** понятия.

Имя — это единица языка, *семантически* отражающая сущность какого-то объекта или явления, а *синтаксически* — конкретный субъект или объект высказывания.

В ***предикатной функции*** (выражение с неопределенными переменными, которое при выборе конкретных значений переменных преобразуется в истинное или ложное высказывание) имя представляется переменной или константой. Отношение, связывающее сущность (denotat) с отражающим ее понятием (concept) и его именем, графически выражается **треугольником Фреге.**



а)Треугольник ***«сущность-имя2-понятие»*** – это случай ***синонимии***
б)Треугольник ***«имя-сущность2-понятие2»*** иллюстрирует случай ***омонимии***

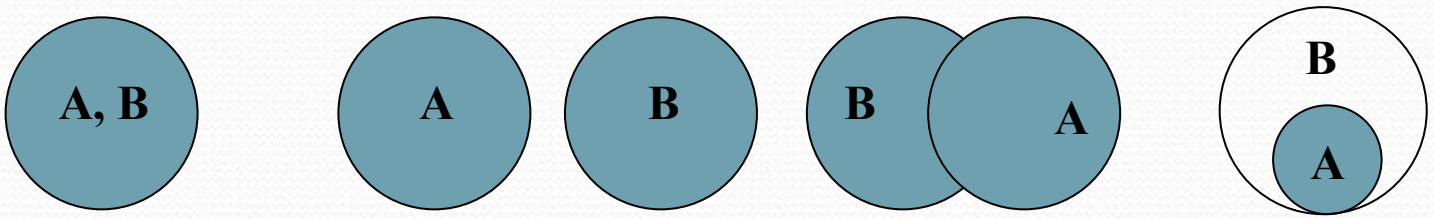
Отношения между понятиями

Каждой сущности в реальном мире соответствует *понятие, представление* о нем и *имя* (треугольник Фреге). Попытки представления системы понятий в предметной области привели к возникновению БЗ специального типа – **онтологий: <X, R, F>**, где **X** – множество понятий, **R** – множество отношений на X, **F** – множество функций для интерпретации понятий).

Понятия определяются через *содержание* и *объем*. Чем *шире* набор свойств, составляющих содержание понятия, тем *уже* класс объектов с этими свойствами. Наоборот, чем *уже* содержание понятия, тем *шире* его объем.

Понятия **A** и **B**, выраженные через их содержание, могут находиться в следующих отношениях:

Равнозначность Несравнимость Толерантность Подчиненность

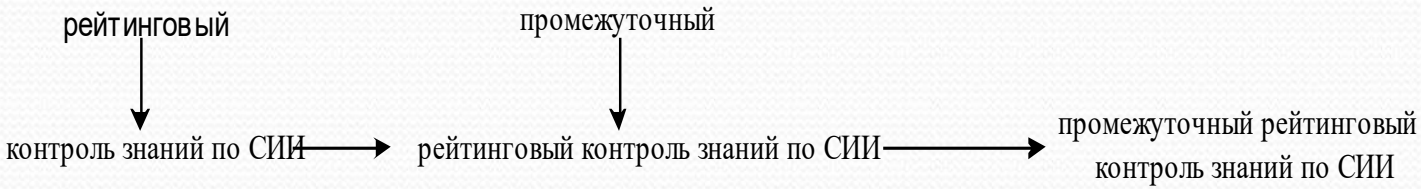


Как оценить сходство/различие двух понятий ?

Родо-видовая классификация понятий

Родо-видовая связь понятий соответствует отношению «общее-частное»: одно понятие является исходным (родовым), а расширяющий родовое понятие видовой признак называется видовым отличием. В силу родства некоторых понятий для оценки их семантической близости обычно используют ранг **ρ** (степень родства). Для родового понятия ранг ρ=0, а для образования любого видового понятия применяется рекурсивная формула вида: $A_{\rho+1,B} = A_{\rho} \bigcup B$

где Aρ – исходное понятие, В – видовое отличие. Данная формула отражает наследование признаков. Например:



В этой схеме понятие «**контроль знаний по СИИ**» является минимальным по содержанию, а «**промежуточный рейтинг-контроль знаний**» – максимальным.

Объем видового понятия Aρ+1,B определяется по формуле: $A_{\rho+1,B} = A_{\rho} \bigcap B$

Объем видового понятия **уже**, чем объем родового понятия «контроль знаний по СИИ», в то время как его содержание **шире**. В объем видового понятия обычно включаются элементы и их классы. Связь «элемент-класс» является частным случаем родо-видовой связи.

В английской терминологии отношение «элемент-класс» обозначают **is a.** Например, Petrov **is a** student. Если Белов, Орлов и Петров учатся в ТТИ ЮФУ, то их можно **обобщить** в класс студенты. **Конкретизацией** понятия «студенты» является подкласс «успевающие студенты», например Орлов и Петров. Обобщение индуктивно, в отличие от конкретизации, которая дедуктивна.

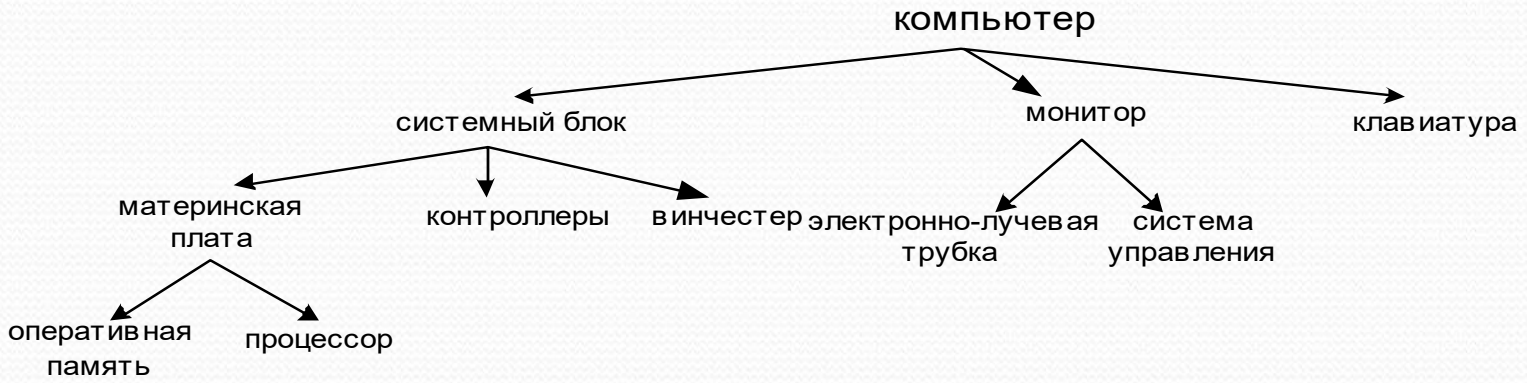
Элементы, входящие в объем понятия, наследуют различные видовые отличия, по признаку **основания деления**.

Партитивная и собирательная связь понятий

Партитивная связь понятий – это связь между целым и его частями. В английской терминологии отношение «целое - часть» обозначают как *part of* (Monitor is *part of* a PC), откуда и следует название партитивная связь.

Партитивная связь характеризует разделение целого понятия на отдельные его части. Число уровней такого деления на практике определяется степенью детализации. ***В чем различие между партитивным и родо-видовым понятием***?

Различие заключается в следующем: понятия-части не являются однородными; свойства понятия-целого не распространяются на свойства понятия-части. Ниже в качестве примера приводится состав персонального компьютера:



Собирательная связь объединяет понятия таких объектов, которые существуют самостоятельно, но могут быть объединены в целое (агрегат), причем свойства понятия-агрегата могут быть эмерджентными (вновь появившимися), т.е. не присущими отдельным его частям.

Собирательная связь может быть применена как к однородным объектам (**человек - народ**), так и к неоднородным объектам (**человеко-машинная система**).

Классификация интеллектуальных задач

Анализ данных - процесс выявление закономерностей и смысла в данных, в причинно-следственных связях путем многовариантного анализа данных.

Диагностика - процесс соотнесения объекта с некоторым классом объектов и/или обнаружения неисправности в системе, заболевания живых организмов, природных аномалий и т.д.

Мониторинг - наблюдение и накопление по ряду показателей данных и фактов в предметной области, сигнализация о выходе параметров за допустимые пределы.

Обучение - особым образом организованное общение между теми, кто обладает знаниями и определённым опытом, и теми, кто их приобретает, усваивает.

Планирование – поиск планов действий, относящихся к объектам, способным выполнять некоторые функции.

Поддержка принятия решения – совокупность процедур, обеспечивающая ЛПР необходимой информацией для оптимального выбора решений среди альтернатив.

Прогнозирование – предсказание событий или явлений на основании анализа имеющихся данных.

Проектирование – подготовка спецификаций на создание объектов с заранее определёнными свойствами.

Управление – процессы использование знаний для достижения определенных целей системы в соответствии с заданными спецификациями.

Архитектура экспертных систем

- Эксперт
- Инженер по знаниям
- Пользователь

Интерфейс инженера по знаниям

База данных

База знаний

СУБЗ

Система лог. вывода

Подсистема объяснений

Интерфейс пользователя

Модели знаний

Наиболее общей моделью представления знаний является **алгебраическая система Мальцева:**

< A, C, F, P>,

где **A** – множество переменных, **C** – множество констант,

F – множество функций, **P** – множество предикатов (отношений).

Σ=<F, P> называется *сигнатурой* или *языком системы*.

Согласно Мальцеву, его система состоит из двух частей:

<A, C, F> - алгебра и **<A, C, P> - реляционная система.**

Алгебре в ИИ соответствует **процедурная модель**.

Реляционной системе - **декларативная модель**.

Круг реально используемых в СИИ моделей представления знаний очень широкий. В него входят как классические (символьные) модели, подражающие мышлению и структуре памяти человека, так и модели ИНС, эволюционного программирования, немонотонных логик.

К классическим моделям относятся:

-Продукционные системы;

- Фреймы;

- Семантические сети;

- Логики Аристотеля, Буля, Заде;

- Формальные системы.

Продукционные правила

ПРОДУКЦИОННАЯ МОДЕЛЬ представления знаний - это модель, основанная на правилах (продукциях), позволяющая представить знания в виде предложений типа:

IF <условие> THEN <действие>

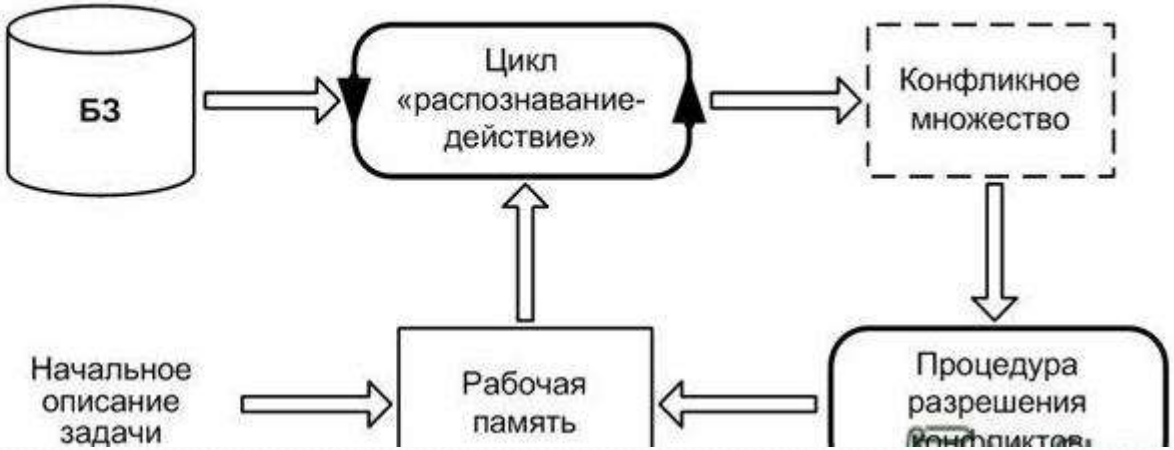
Условная часть (посылка) **Утвердительная часть (заключение)**

Левая часть правила определяет некоторое условие, которое может быть истинным или ложным.
Если условие истинное, то выполняются <действия>, иначе никаких действий по данному правилу не производится.
Выполнение действия представляет собой создание нового факта (временного) в БЗ.
Условия в правилах могут быть различной сложности.

УСЛОВИЯ делятся на:
простые условия - состоят из одного факта;
сложные условия – могут включать
- Различные типы и виду переменных
- Логические операторы
- Числовые операторы
- Числовые функции
- Символьные функции

Архитектура продукционных систем представления знаний

- БЗ продукционных правил;
- рабочая память;
- цикл управления распознавание–действие.
- Моделирование решения задачи основано на процессе сопоставления с образцом (pattern matching), в ходе которого текущее состояние решения сравнивается с имеющимися знаниями для определения дальнейших действий.



Достоинства/недостатки продукционных моделей знаний

Продукционные модели близки к логическим моделям, но **более наглядно отражают знания**, поэтому являются наиболее распространенными средствами представления знаний. Чаще всего они применяются в промышленных экспертных системах, в качестве решателей или механизмов выводов.

Достоинства продукционных моделей:

– наглядность;

– высокая модульность – отдельные логические правила могут быть добавлены в базу знаний, удалены или изменены независимо от других, модульный принцип разработки систем позволяет автоматизировать их проектирование;

– легкость внесения дополнений и изменений;

– простота логического вывода.

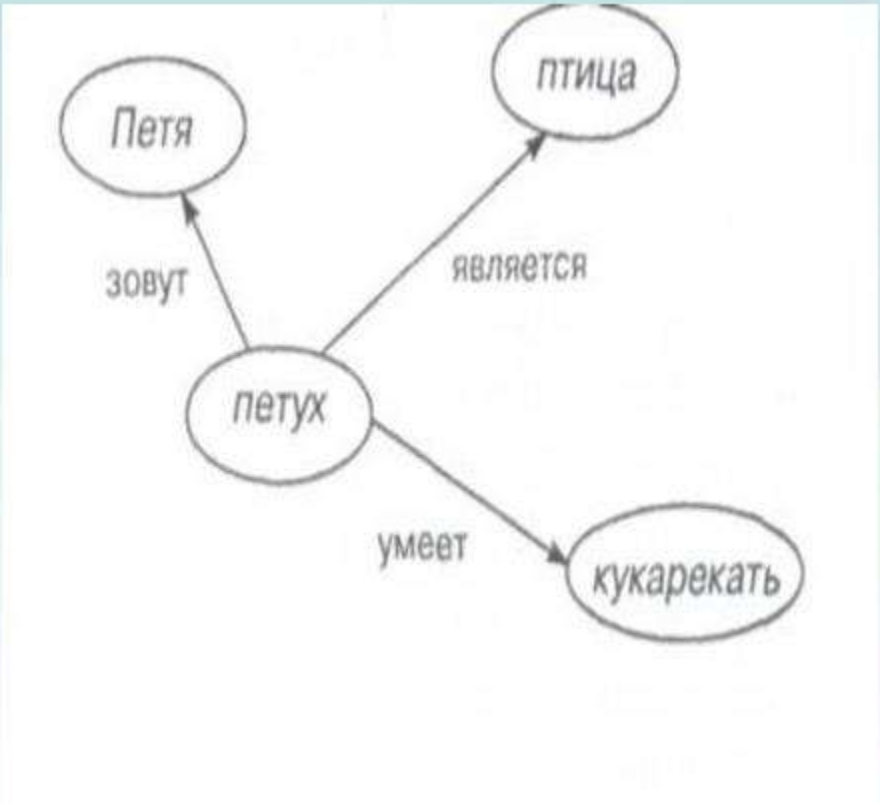
Недостатки продукционных моделей:

– при большом количестве продукционных правил в базе знаний, изменение старого правила или добавления нового приводит к непредсказуемым побочным эффектам;

– затруднительна оценка целостного образа знаний, содержащего в системе.

Семантическая сеть

Семантической сетью называется система понятий и отношений между ними, представленная в графической форме.



Вывод и поиск знаний в семантической сети

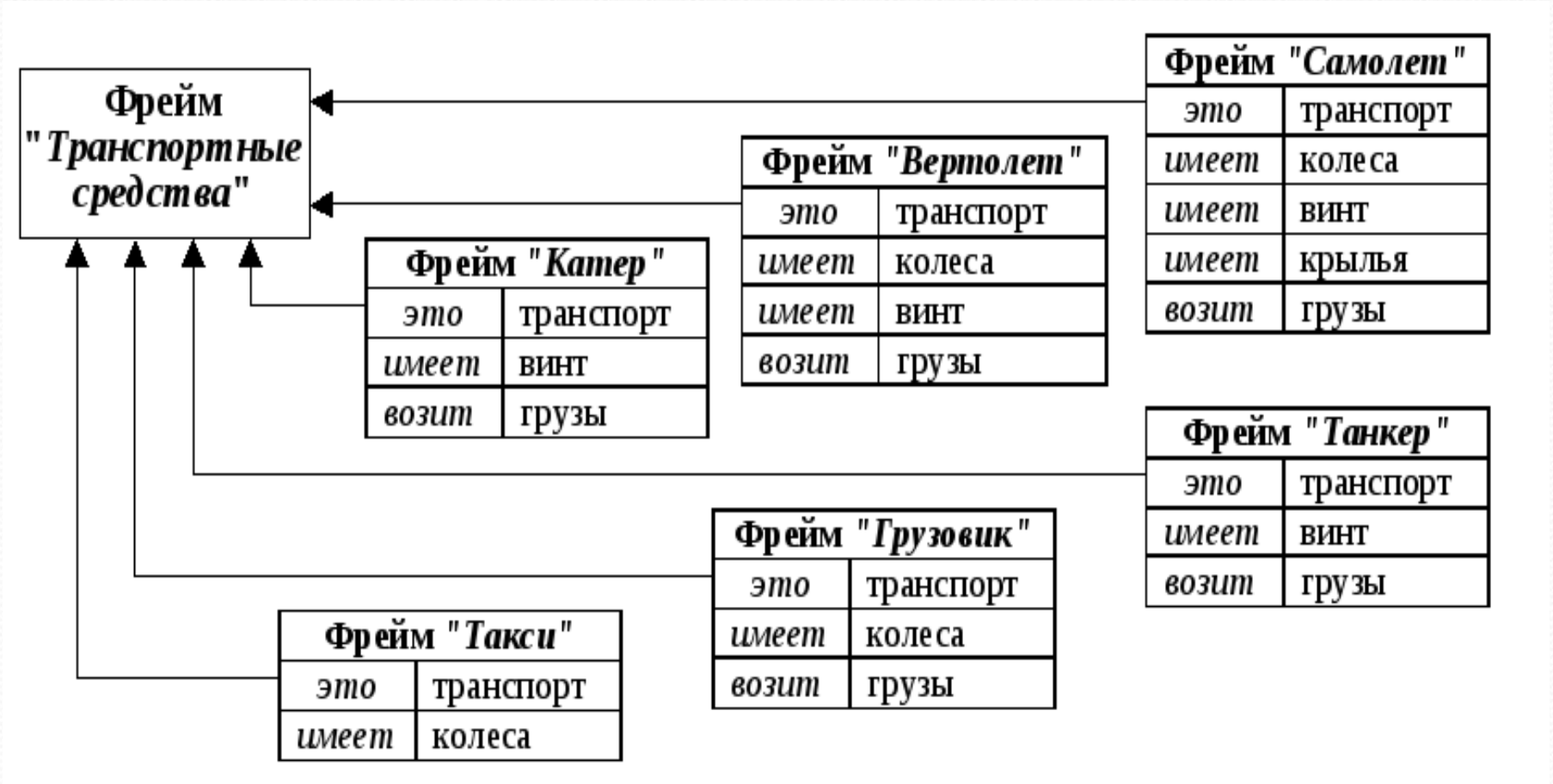
Фреймы

Теория фреймов (автор Минский):

«Когда человек сталкивается с новой ситуацией (или существенно меняет точку зрения на прежнюю задачу), он извлекает из памяти определенную структуру, называемую фреймом. Эту хранящуюся в памяти систему следует при необходимости привести в соответствие с реальностью путем изменения ее деталей».

Фрейм - это структура данных, предназначенная для представления знаний о стереотипной ситуации, причем детали фрейма с изменением текущей ситуации могут меняться.

Сеть фреймов в базе знаний «Транспортные средства»



Декларативно-процедурные модели

К ним относятся, например, **сети Петри: <T, P, I, O>**, где **T** – множество вершин переходов, **P** – множество вершин позиций, **I –** множество функций входов, **O** – множество функций выходов.
С помощью сетей Петри моделируются многие трудные проблемы параллельного программирования, систем реального времени:

С помощью сетей Петри успешно моделируются задачи синхронизации, сетевого планирования и управления, конвейерная обработка, генерируются оптимальные коды для компиляторов алгоязыка, химические соединения и т.п.

Логический вывод

Вывод в логике - процесс рассуждения, в ходе которого осуществляется переход от некоторых исходных суждений (предпосылок) к новым суждениям - заключениям.

В системах искусственного интеллекта, таких как экспертные системы, вывод проводится с использованием правил, принципов и законов логики на основе заданных фактов и правил с использованием методов и средств логического программирования.

Различают ***достоверный*** и ***правдоподобный*** механизм вывода.

Общая формулировка задачи вывода знаний:

Пусть **Th** – знания из БЗ, **f** – наблюдаемый факт или запрос к БЗ, **h** – гипотеза для объяснения **f**. Тогда задача вывода формулируется как: **Th & f ├ h**. Возможными механизмами вывода являются:

Дедукция - найти **h**, если **Th** полностью определено;

Индукция – найти и оценить правдоподобность **h**, если правила вывода в **Th** не полностью определены, но фактам **f** можно доверять;

Абдукция – найти **h**, для объяснения **f** при неполно определенном **Th**;

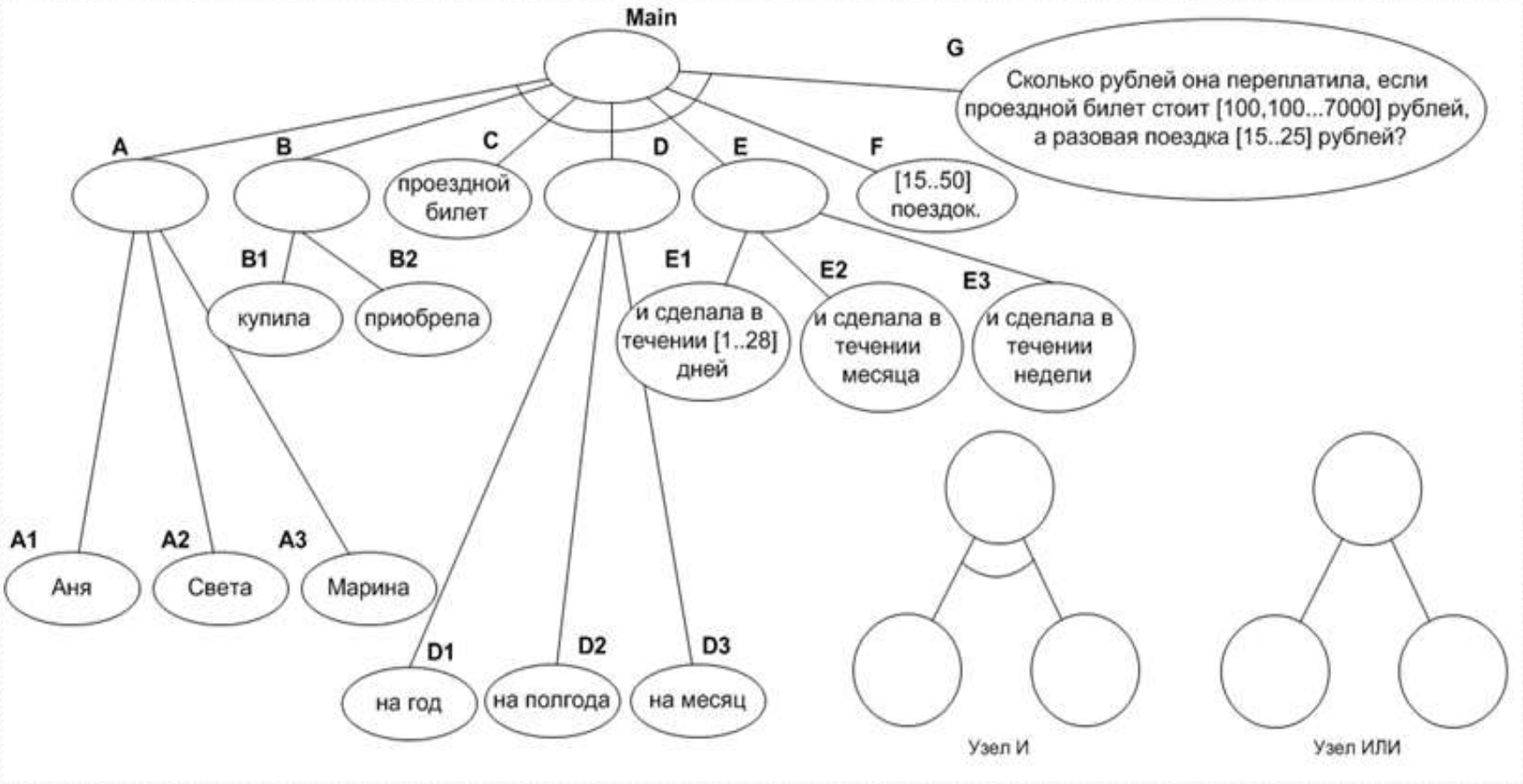
Аналогия - **Th** по отношению к **f** и **h** не является полным, но известна аналогия в **Th´** между **f´** и **h´**;

Вероятностный и нечеткий вывод - **Th** и **f** заданы вероятностно или нечетко;

Нейросетевой и эволюционный вывод – **f** и **h** неявно определяются архитектурой нейросети или правилами эволюции.

Вывод, как задача поиска по дереву решений

И/ИЛИ-дерево: задаётся множество начальных вершин графа, с которых поиск может начинаться, и множество целевых вершин, при достижении которых поиск прекращается. И/ИЛИ-граф обладает следующими свойствами: при движении по входным дугам к некоторой вершине реализуется либо конъюнкция, либо дизъюнкция.



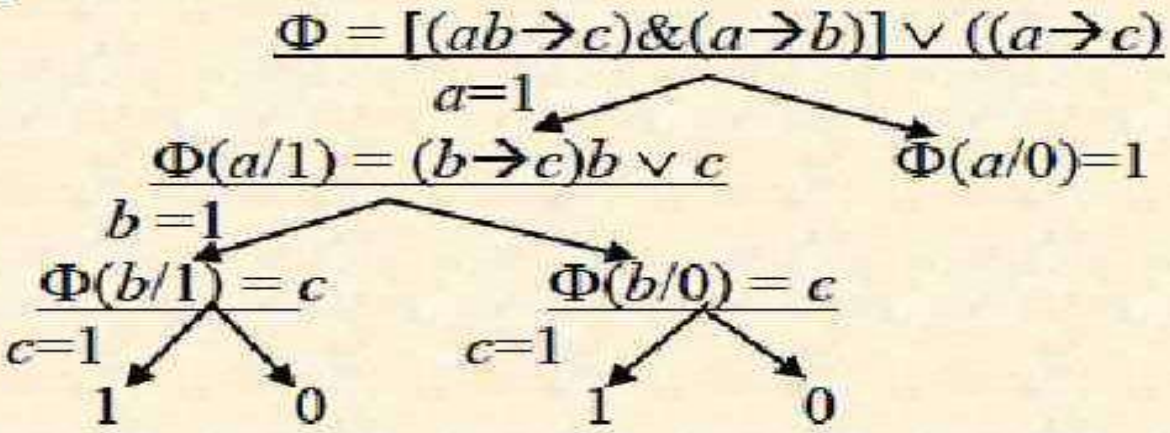
Достоверный вывод. Общезначимые высказывания

При достоверном логическом выводе используется понятие **общезначимости** (*формула общезначима, если истина при любых значениях переменных*).

Как проверить общезначимость формулы? – Метод Квайна, метод от противного, таблица истинности.

Метод Квайна заключается в следующем: последовательно подставляются значения истинности в формулу для аргументов и вычисляются значения истиности, или выполняются алгебраические преобразования формул до тех пор, пока не получим конечные значения *T* или *F*.

Алгоритм вычислений строится в виде бинарного дерева (двоичной диаграммы) – концевые вершины обозначают все возможные значения формулы.



Исчисление высказываний. Логическое следствие

Исчисление – это знаковая система, состоящая из алфавита, аксиом и процедур вывода истинных (синтаксически правильных) формул из аксиом. Если символам языка приписать конкретные значения, то это формальный язык.

Различают исчисление высказываний и исчисление предикатов.

В **исчислении высказываний** вопрос об истинности высказываний решается с помощью формул, а также логических операций конъюнкции, дизъюнкции, отрицания, импликации и др. В естественном языке высказыванием может быть повествовательное предложение, о котором можно сказать, истинно оно или ложно.

В **исчислении предикатов** наряду с формулами исчисления высказываний, используются формулы, в которые могут входить отношения (предикаты), связывающие между собой элементы исчисления высказываний, а также кванторы общности (∀) и существования (∃). Язык логики предикатов наиболее приближен к человеческому языку. Язык ИИ Пролог основан на логике предикатов 1-го порядка .

Определение 1. Формула **G** называется *логическим следствием* формул $\mathbf{F_1, F_2, \ldots, F_k}$, если из $\mathbf{F_1 = F_2 = \ldots = F_k = 1}$ следует, что **G = 1**.

Определение 2. Множество формул $\{\mathbf{F_1, F_2, \ldots, F_k}\}$ называется *выполнимым*, если существует хотя бы один набор переменных **X** такой, что $\mathbf{F_1(X) = F_2(X) = \ldots = F_k(X) = 1}$.

Достоверный вывод. Метод резолюций

Идея **метода резолюций** основана на следующей теореме..

Теорема. Формула **G** является логическим следствием формул $\mathbf{F_1}$, $\mathbf{F_2}$,..., $\mathbf{F_k}$ тогда и только тогда, когда множество формул $\mathbf{L} = \{\mathbf{F_1}, \mathbf{F_2},..., \mathbf{F_k}, \neg \mathbf{G}\}$ *невыполнимо*.

Иными словами, задача достоверного логического вывода сводится к задаче проверки выполнимости множества формул $\{\mathbf{F_1}, \mathbf{F_2},..., \mathbf{F_k}, \neg \mathbf{G}\}$.

В этом методе используются следующие понятия:

- ***литерал*** – *атомарная* формула (формула без операций) или ее отрицание;
- ***дизъюнкт*** – дизъюнкция одного или нескольких литералов;
- ***пустой дизъюнкт*** – специальный дизъюнкт, не содержащий литералов. Для его обозначения используется специальный символ ■. Пустой дизъюнкт ложен;
- ***противоположные литералы*** – литералы **X** и **¬X**.

Резолюция. Из дизъюнктов (**X** v **F**) и (**¬X** v **G**) выводим дизъюнкт (**F** v **G**). Другими словами, дизъюнкт (**F** v **G**) является логическим следствием дизъюнктов (**X** v **F**) и (**¬X** v **G**).

Резолюция – это основной приём, используемый при достоверном выводе. Этот приём заключается в нахождении двух дизъюнктов с противоположными литералами. На основании их сравнения формируется новый дизъюнкт, называемый *резольвентой*. Порождение новых дизъюнктов - основа метода резолюций, широко применяется в интеллектуальных системах.

Метод резолюций

Метод резолютивного вывода состоит в доказательстве того, что формула **G** является логическим следствием множества формул $\mathbf{F_1},..., \mathbf{F_k}$ и включает шаги:

1. Составляется множество формул $\{\mathbf{F_1},..., \mathbf{F_k}, \neg \mathbf{G}\}$.
2. Каждая из этих формул приводится к конъюнктивной нормальной форме (КНФ). В КНФ зачеркиваются знаки конъюнкции. Получается множество дизъюнктов **S**.
3. Осуществляется поиск вывода пустого дизъюнкта ■ из **S**. Если пустой дизъюнкт выводим из **S**, то формула **G** является логическим следствием формул $\mathbf{F_1},..., \mathbf{F_k}$. Если из **S** нельзя вывести ■, то **G** не является логическим следствием формул $\mathbf{F_1},..., \mathbf{F_k}$ ..

Пример 1. Пусть даны два утверждения: «*Яблоко красное и ароматное*», «*Если яблоко красное, то яблоко вкусное*». Докажем утверждение, что «*Яблоко вкусное*».

1. Введем множество формул, описывающих простые высказывания-утверждения:

$\mathbf{X_1}$ – «Яблоко красное», $\mathbf{X_2}$ – «Яблоко ароматное», $\mathbf{X_3}$ – «Яблоко вкусное».

Тогда исходные утверждения имеют вид следующих формул:
$\mathbf{X_1}$ & $\mathbf{X_2}$ - «Яблоко красное и ароматное», $\mathbf{X_1 \rightarrow X_3}$ - «Если яблоко красное, то яблоко вкусное». Утверждение, которое надо доказать, выражается формулой $\mathbf{X_3}$.

2. Формулы $\{(\mathbf{X_1 \& X_2}), (\mathbf{X_1 \rightarrow X_3}), \neg \mathbf{X_3}\}$ приводим к КНФ: $\{(\mathbf{X_1 \& X_2}), (\neg \mathbf{X_1} \vee \mathbf{X_3}), \neg \mathbf{X_3}\}$.

Зачеркивая конъюнкции, получаем следующее множество дизъюнктов:
$\mathbf{S} = \{\mathbf{X_1}, \mathbf{X_2}, (\neg \mathbf{X_1} \vee \mathbf{X_3}), \neg \mathbf{X_3}\}$.

3. Используя резолюцию, находим вывод пустого дизъюнкта ■. Яблоко вкусное.

В чем отличие правдоподобного от достоверного вывода?

Особенности правдоподобного вывода:

- Задачи, стоящие перед реальными системами, основанными на знаниях, предполагают правдоподобный вывод из-за неполных и нечетких данных;
- Механизмы пересмотра рассуждений показали себя как мощный инструмент для реализации правдоподобного вывода, что позволяет решать гораздо более широкий класс задач, в отличие от монотонного дедуктивного вывода.

Например, в **экспертных системах** допускается сочетание достоверного и правдоподобного вывода, возможность немонотонных рассуждений, когда поступившие факты могут изменить истинность ранее выведенных заключений. Методы правдоподобного вывода служат дополнением или частичной заменой эксперта

Познавательная деятельность, является сочетанием достоверного и правдоподобных методов вывода (Ч. Пирс). **Абдукция** осуществляет принятие правдоподобных гипотез (объяснение наблюдаемых фактов) $\Rightarrow$ **Дедукция** выводит следствия из гипотез $\Rightarrow$ **Индукция** производит тестирование следствий из гипотез.

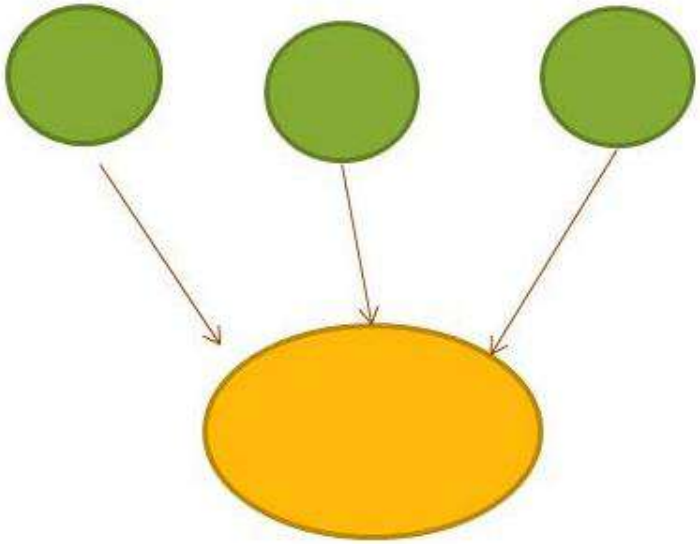
Дедуктивный вывод. *Все студенты из группы направления Программная инженерия (ПИ) юны. Эти студенты являются студентами группы ПИ.* Следовательно, *Эти студенты юны*.

Индуктивный вывод. *Эти студенты являются студентами группы ПИ*; *Эти студенты юны*. Следовательно, *Все студенты из группы ПИ юны*.

Абдуктивный вывод. *Все студенты из группы ПИ юны. Эти студенты юны.* Следовательно, *Эти студенты являются студентами группы ПИ.*

Индуктивный метод вывода

- Индуктивный метод – рассуждение, при котором, опираясь на ряд частных результатов приходят к одному общему выводу.



Методы индукции

- **Метод единственного сходства** – во всех случаях наблюдения какого-либо явления обнаруживается лишь один общий факт, все остальные - различны. Следовательно, единственный сходный фактор является причиной данного явления.
- **Метод единственного различия** – Если обстоятельства возникновения какого-то явления и обстоятельства, при которых оно не возникает, почти во всем сходны и различны лишь одним фактором, присутствующем только в первом случае, то можно сделать вывод, что этот фактор является причиной данного явления.
- **Соединенный метод сходства и различия** представляет собой комбинацию двух вышеуказанных методов.
- **Метод сопутствующих изменений.** Если определённые изменения одного явления всякий раз влекут за собой некоторые изменения в другом явлении, то отсюда вытекает вывод о причинной связи этих явлений.
- **Метод остатков.** Если сложное явление вызывается многофакторной причиной, причем некоторые из этих факторов известны как причина какой-то части данного явления, то отсюда следует вывод: причина другой части явления – остальные факторы, входящие в общую причину этого явления.

Аргументация и абдуктивный вывод

Абдукция – это процесс формирования объясняющей гипотезы. Точнее, при заданной теории и наблюдении, предложенном для объяснения, абдуктивный вывод должен определить одно или более наилучших объяснений наблюдения на основе заданной теории. Термин «абдукция» впервые ввел американский философ Пирс.

На сегодняшний день теория аргументации проявила себя как мощное средство моделирования различных форм правдоподобного вывода. Основная идея аргументационного вывода состоит в том, что утверждению можно доверять, если оно может быть аргументированно защищено от атак аргументов. Как оказалось, теория аргументации может успешно применяться и для организации абдуктивного вывода.

Абдукция предназначена для поиска объяснений или причин наблюдаемых явлений или фактов. Соответствующий вывод имеет форму (в теории вывода Пирса):

- наблюдается удивительный факт C (то есть этот факт не следует из наших знаний о мире);
- если бы A было бы истинным, то C могло бы произойти (A – объясняющая гипотеза);
- следовательно, можно предположить, что A истинно.

Заметим, что в искусственном интеллекте до последнего времени работы по абдукции и индукции велись независимо. Но, по-видимому, перспективным является рассмотрение абдукции, индукции и дедукции в рамках одной системы. При этом абдукция используется для получения гипотез, объясняющих наблюдения. Дедукция позволяет выполнять вывод с использованием этих гипотез и получать некоторые предсказания. А с помощью индукции на основе этих предсказаний можно определить общие правила и оценить, насколько они согласуются с реальностью.

Пример абдуктивного вывода (В.Н.Вагин)

Пусть **h** – объяснение факта **f** в **Th**. Предположим, что **Th&f** – выполнимая формула, но **f** – не является следствием **Th**, т.е. знания о предметной области **Th** не объясняют факта **f**.

Необходимо вывести дополнительные факты **h**, объясняющие **f** относительно **Th**, т.е. ***Th, h |= f***, причем к фактам **h** предъявляются требования **непротиворечивости** и **минимальности** (абдуктивное объяснение ***h*** для ***f*** считается *непротиворечивым*, если **Th&h** – выполнимая формула; *минимальным* - если каждое объяснение, являющееся логическим следствием ***f***, эквивалентно ***h***).

Пусть Варвар – имя слона. Тогда: $Th : \forall x(Слон(x) \Rightarrow Серый(x))$

$$f : Серый(Варвар)$$

$$h : 1.Слон(Варвар),$$

$$2.Слон(Цезарь) \,\&\, Несерый(Цезарь),$$

$$3.Слон(Варвар) \,\&\, Женщина(Старая\ леди)$$

где 1 - минимальное и непротиворечивое объяснение факта **f** в **Th**; 2 – противоречивое объяснение; 3 – избыточное объяснение.

Вывод по абдукции успешно применяется для решения задач диагностики, понимания естественного языка, распознавания образов, накопления знаний, составления расписаний, он является очень важной составляющей СИИ.

Вывод по аналогии

Одной из форм правдоподобных выводов являются выводы по аналогии.

Аналогия – это недедуктивное умозаключение, в котором объяснение **h** наблюдаемого факта **f** в предметной области **Th** выводится на основании его сходства с объяснением **h'** факта **f'** в предметной области **Th'**.

- это метод познания, основанный на переносе одного или ряда свойств с известного явления на неизвестное; частный случай индукции.

Данный метод не имеет большой доказательной силы. Сходство, на основании которого производится доказательство, может оказаться случайным, а при выборочном анализе признаков существенные признаки могут быть заменены на несущественные.

анатомические представления о кровообращении	по аналогии →	**«Экономическая таблица» Ф.Кэне (движение товарных и денежных потоков)**
статическое равновесие в механике	по аналогии →	**идея «экономического равновесия» (А. Курно)**
теории биологической эволюции Ч. Дарвина	по аналогии →	**идея «экономической эволюции»**

ДСМ-метод автоматического порождения гипотез (АПГ) в ИС

Создан в 80-х г. в ВИНИТИ РАН (Финн В.К.). Оригинальный отечественный метод правдоподобных рассуждений в ИС. В нем формализованы и автоматизированы индуктивные рассуждения Д.С.Милля.

ДСМ-метод имеет 6 компонент: **(1)** условия применимости, **(2)** правдоподобные ДСМ-рассуждения, **(3)** представление знаний в виде квазиаксиоматических теорий, **(4)** средства исследования корректности и семантики рассуждений, **(5)** средства распознавания и вывода знаний, **(6)** интеллектуальную ДСМ-систему.

(1) Условиями применимости ДСМ-метода является существование в базе позитивных (+)-фактов и негативных (–)-фактов; эмпирических причинно-следственных зависимостей, а также возможность формализовать сходство/различие фактов. Главный принцип АПГ – сходство/различие фактов влечет наличие/отсутствие изучаемых эффектов и их повторяемость. Это – основа качественного (нестатистического) анализа данных.

(2) ДСМ-рассуждения являются синтезом трех способов вывода: абдукции, аналогии и индукции.

ДСМ-рассуждение формализует естественный познавательный процесс: анализ данных (индукция) – предсказание (аналогия) – объяснение полученных результатов (абдукция).

ДСМ-метод автоматического порождения гипотез (АПГ) в ИС *(продолжение)*

(3) Представление знаний в ДСМ-методе осуществляется на основе теорий, состоящих из аксиом, множества (±)-фактов, порождающих гипотез и множества правил вывода достоверных и правдоподобных знаний.

(4) Процедуры вывода по индукции, аналогии и абдукции могут быть представлены в виде декларативных аксиом. Их множество должно быть непротиворечивой теорией. Это обеспечивает логическую корректность ДСМ-метода.

Общая схема ДСМ-метода АПГ:

Th & **f** – БАЗА ЗНАНИЙ и ФАКТОВ

h = {$\mathbf{h}_1$ + $\mathbf{h}_2$} – множество ГИПОТЕЗ

Th & f $\vdash$ $\mathbf{h}_i$ – гипотеза $\mathbf{h}_i$ объясняет факт **f** в **Th**

Следовательно, все гипотезы из **h** правдоподобны

Здесь $\mathbf{h}_1$ – гипотезы о (±)-причинах, порожденных индукцией, а $\mathbf{h}_2$ – гипотезы-предсказания, полученные по аналогии. Сама же схема представляет абдукцию – принятие правдоподобных гипотез путем объяснения фактов в предметной области **Th**.

Формирование базы фактов **f** и формулирование целей ДСМ-рассуждений являются **междисциплинарной** проблемой формирования аксиоматических теорий.

ДСМ-метод автоматического порождения гипотез (АПГ) в ИС *(продолжение)*

(5) Исходное множество аксиом пополняется посредством обнаруженных при сравнении с расширяющимся множеством баз фактов ($БФ_1$,…, $БФ_m$): если гипотезы взаимно непротиворечивы, то обнаружен эмпирический закон; если же противоречий мало, то обнаружена эмпирическая тенденция. Они пополняют базу знаний ИС-ДСМ.

(6) ИС – практический инструмент ДСМ-метода. Автоматическое порождение гипотез генерирует решатель задач, реализующий ДСМ-рассуждения, используя базу фактов и базу знаний в данной предметной области **Th**.

На основе ДСМ-метода работают около десятка оригинальных отечественных интеллектуальных систем в области диагностики различных заболеваний, анализа социологических данных, прогнозирования токсичности химических соединений, анализа криминалистических данных.

Вероятностный (байесовский) вывод

Пусть правила БЗ имеют вид:

ЕСЛИ (гипотеза *H* истинна), ТО (вероятно, справедливо свидетельство *C*)

По теореме Байеса:

$$p(H|C) = \frac{p(C|H)\times p(H)}{p(C|H)\times p(H)+p(C|\neg H)\times p(\neg H)},$$

где $p(H)$ – априорная вероятность того, что гипотеза *H* истинна;

$p(C|H)$ – условная вероятность того, что гипотеза *H* истинна, будет результатом свидетельства *C*;

$p(\neg H)$ – априорная вероятность того, что гипотеза *H* ложна;

$p(H|C)$ – апостериорная вероятность гипотезы *H*, при наблюдении *C*.

Иногда для простого свидетельства C эксперт обеспечивает мн-во гипотез H1,H2,..., Hm:

$$p(H_i \mid C) = \frac{p(C \mid H_i)\times p(H_i)}{\sum_{k=1}^{M} p(C \mid H_k)\times p(H_k)}$$

Или при мн-ве свидетельств *C1, C2, ..., Cn* также имеется мн-во гипотез:

$$p(H_i \mid C_1, C_2, \ldots, C_n) = \frac{p(C_1, C_2, \ldots, C_n \mid H_i)\times p(H_i)}{\sum_{k=1}^{m} p(C_1, C_2, \ldots, C_n \mid H_k)\times p(H_k)}$$

Это ур-е требует получения условных вер-тей всех возможных комбинаций свидетельств для всех гипотез, что делает задачу невыполнимой.

Поэтому вместо неосуществимого ур-я $p(A \mid B) = \frac{p(A \cap B)}{p(B)}$

мы получаем:

$$p(H_i \mid C_1, C_2, \ldots, C_n) = \frac{p(C_1 \mid H_i)\times p(C_2 \mid H_i)\times \ldots \times p(C_n \mid H_i)\times p(H_i)}{p(C_1 \mid H_k)\times p(C_2 \mid H_k)\times \ldots \times p(C_n \mid H_k)\times p(H_k)}$$

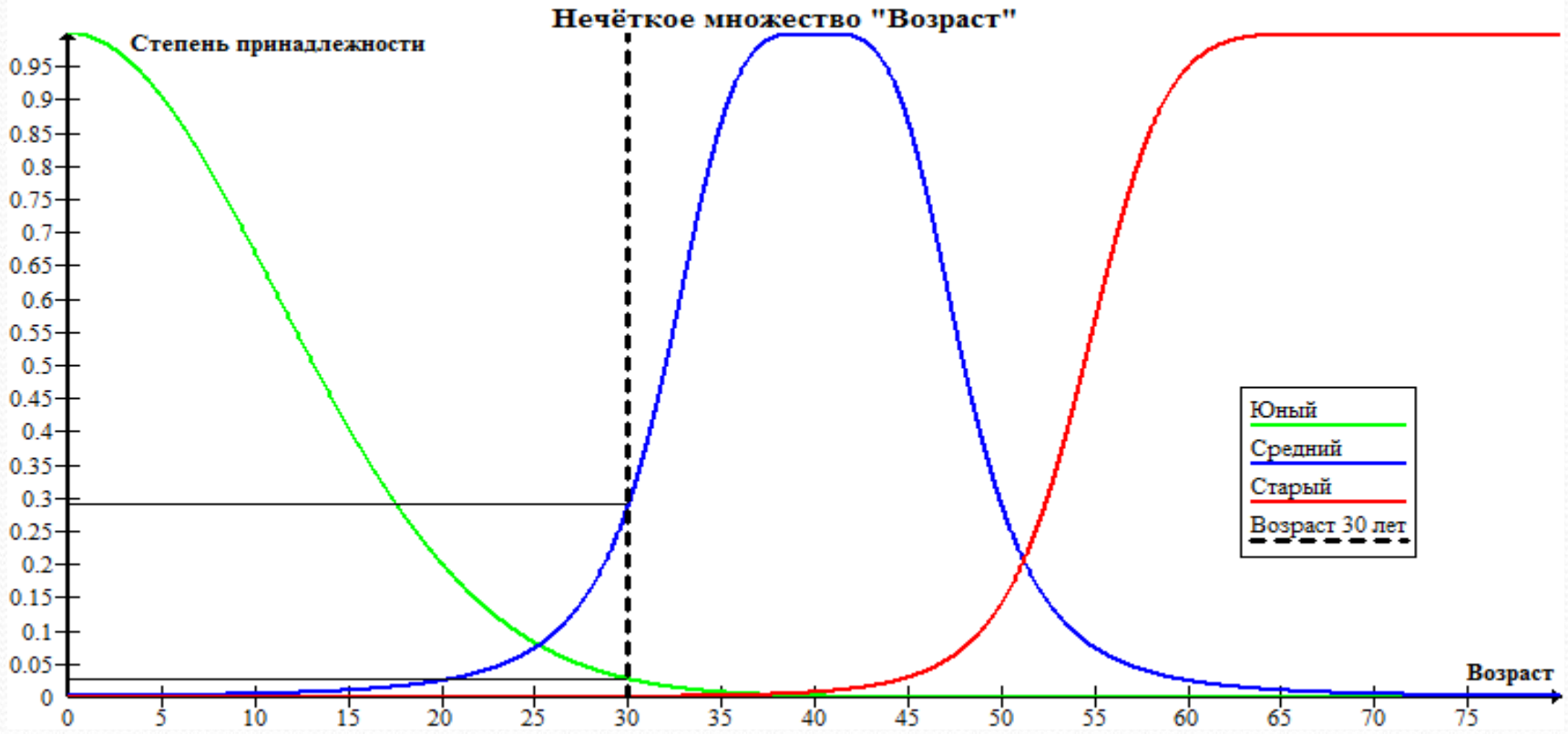
Нечеткое множество

Нечеткое множество (НМ)

$$A = \{ (x, \mu_A(x)) \}$$

определяется как совокупность упорядоченных пар, составленных из элементов x универсального множества X и соответствующих степеней принадлежности $\mu_A(x)$, или непосредственно в виде функции принадлежности

$$\mu_A(x): X \to [0,1].$$



Операции над нечеткими множествами

1. *Включение.* Пусть A и B – нечеткие множества на универсальном множестве E. Тогда A содержится в B, если $\forall x \in E \quad \mu_A(x) \le \mu_B(x)$
Обозначение: $A \subset B$

2. *Равенство.* A и B равны, если $\forall x \in E \quad \mu_A(x) = \mu_B(x)$ Обозначение: A=B

3. *Дополнение.* Пусть M = [0,1], A и B – нечеткие множества, заданные на E. A и B дополняют друг друга, если $\forall x \in E \quad \mu_A(x) = 1 - \mu_B(x)$
Обозначение: $B = \overline{A}$

4. *Пересечение* – наибольшее нечеткое подмножество, содержащее одновременно A и B ($A \cap B$): $\mu_{A \cap B}(x) = \min(\mu_A(x), \ \mu_B(x))$

5. *Объединение* – наименьшее нечеткое подмножество, включающее как A, так и B, с функцией принадлежности ($A \cup B$):
$$\mu_{A \cup B}(x) = \max(\mu_A(x), \ \mu_B(x))$$

6. *Разность* – операция с функцией принадлежности ($A - B = A \cap \overline{B}$):
$$\mu_{A-B}(x) = \mu_{A \cap \overline{B}}(x) = \min(\mu_A(x), \ 1 - \mu_B(x))$$

7. *Дизъюнктивная сумма* – логическая операция с функцией принадлежности $A \oplus B = (A - B) \cup (B - A) = (A \cap \overline{B}) \cup (\overline{A} \cap B)$
($\mu_{A-B}(x) = \max(\min(\mu_A(x), \ 1 - \mu_B(x)); \min(1 - \mu_A(x), \ \mu_B(x)))$

Нечеткий логический вывод

•Выполнение первого этапа нечеткого вывода — фаззификации — осуществляет фаззификатор.

•За процедуру непосредственно нечеткого вывода ответственна машина нечеткого логического вывода, которая производит второй этап процесса вывода на основании задаваемой нечеткой базы знаний (набора правил), а также этап композиции.

•Дефаззификатор выполняет последний этап нечеткого вывода — дефаззификацию.

Импликация – основная операция нечеткого вывода

Нечеткая импликация - имеет тот же смысл, только степень истинности может принимать любые значения в интервале [0;1]. Будем считать, что заданы универсальные множества X и Y. Под способом определения нечеткой импликации "если A, то B", где A и B – нечеткие множества на X и Y соответственно, будем понимать способ задания нечеткого отношения R на X*Y, соответствующее данному высказыванию.

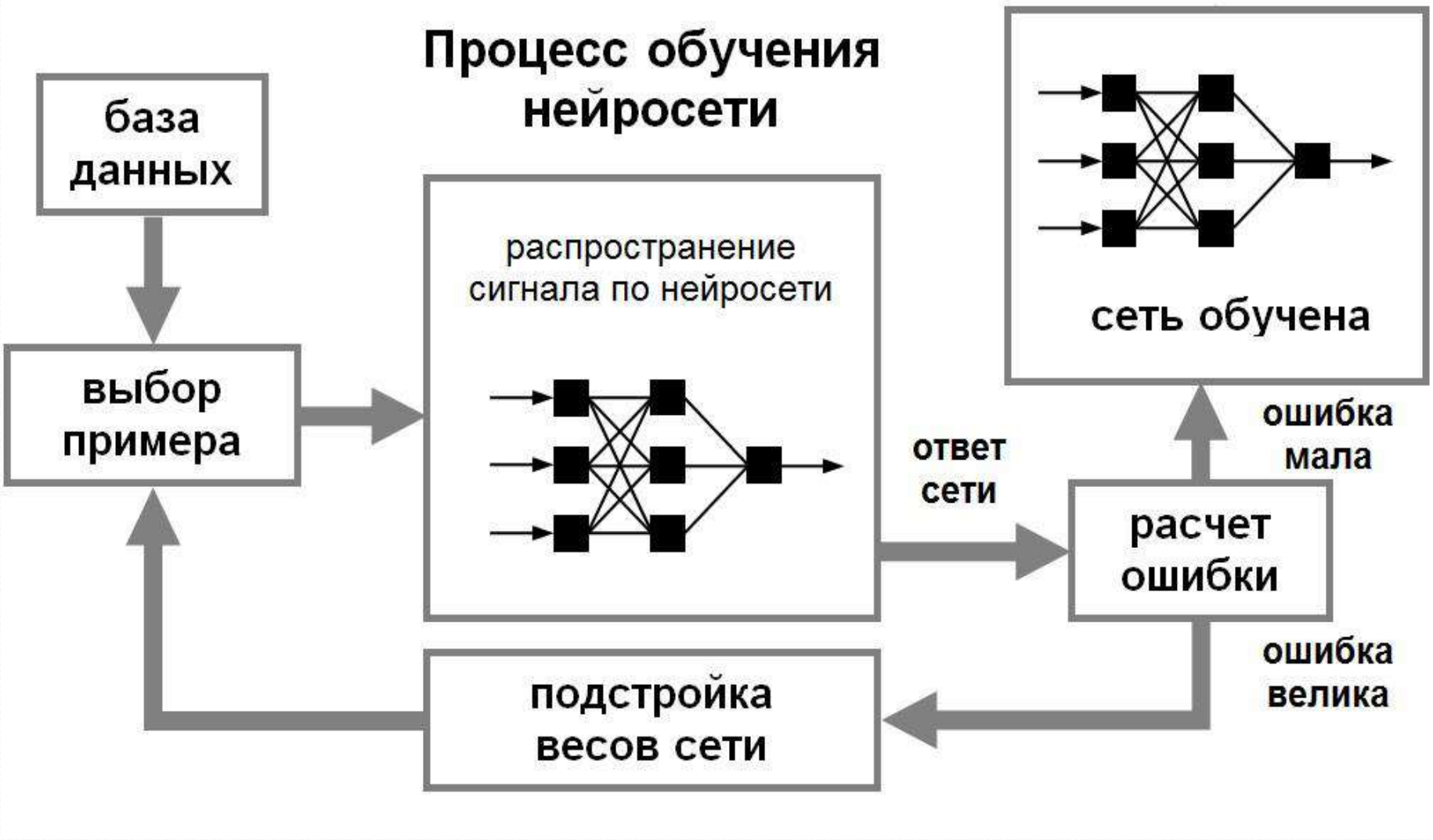
Такое отношение можно задавать по-разному, поэтому для математического представления нечеткой импликации предложено большое количество формул, ассоциируемых с фамилиями авторов, например:

Larsen: $\mu R(x,y) = \mu A(x) * \mu B(y);$

Lukasiewicz: $\mu R(x,y) = \min\{1,\ 1 - \mu A(x) + \mu B(y)\};$

Mamdani: $\mu R(x,y) = \min\{\mu A(x), \mu B(y)\}$

Процесс нейросетевого вывода/обучения



Роевые механизмы вывода

Задача РИ – изучение и описание коллективного поведения децентрализованной самоорганизующейся системы. Методы РИ рассматривается прежде всего как некие специфические механизмы **поисковой оптимизации**. Большинство алгоритмов РИ относятся к классу **метаэвристик**.

Системы РИ состоят из множества агентов, локально взаимодействующих между собой и с окружающей средой. Сами агенты просты, но все вместе, локально взаимодействуя, создают т.н. называемый РИ (в природе – колония муравьев, рой пчел, стая птиц, рыб и т.д.).

Перечень некоторых алгоритмов РИ:

- Муравьиный алгоритм (Ant colony optimization).
- Метод роя частиц (Particle swarm optimization).
- Пчелиный алгоритм (Bees algorithm).
- Оптимизация передвижением бактерий (Bacterial foraging optimization).
- Стохастический диффузионный поиск (Stochastic diffusion search).
- Алгоритм гравитационного поиска (Gravitational search algorithm).
- Алгоритм капель воды (Intelligent Water Drops algorithm).
- Светляковый алгоритм (Firefly algorithm).

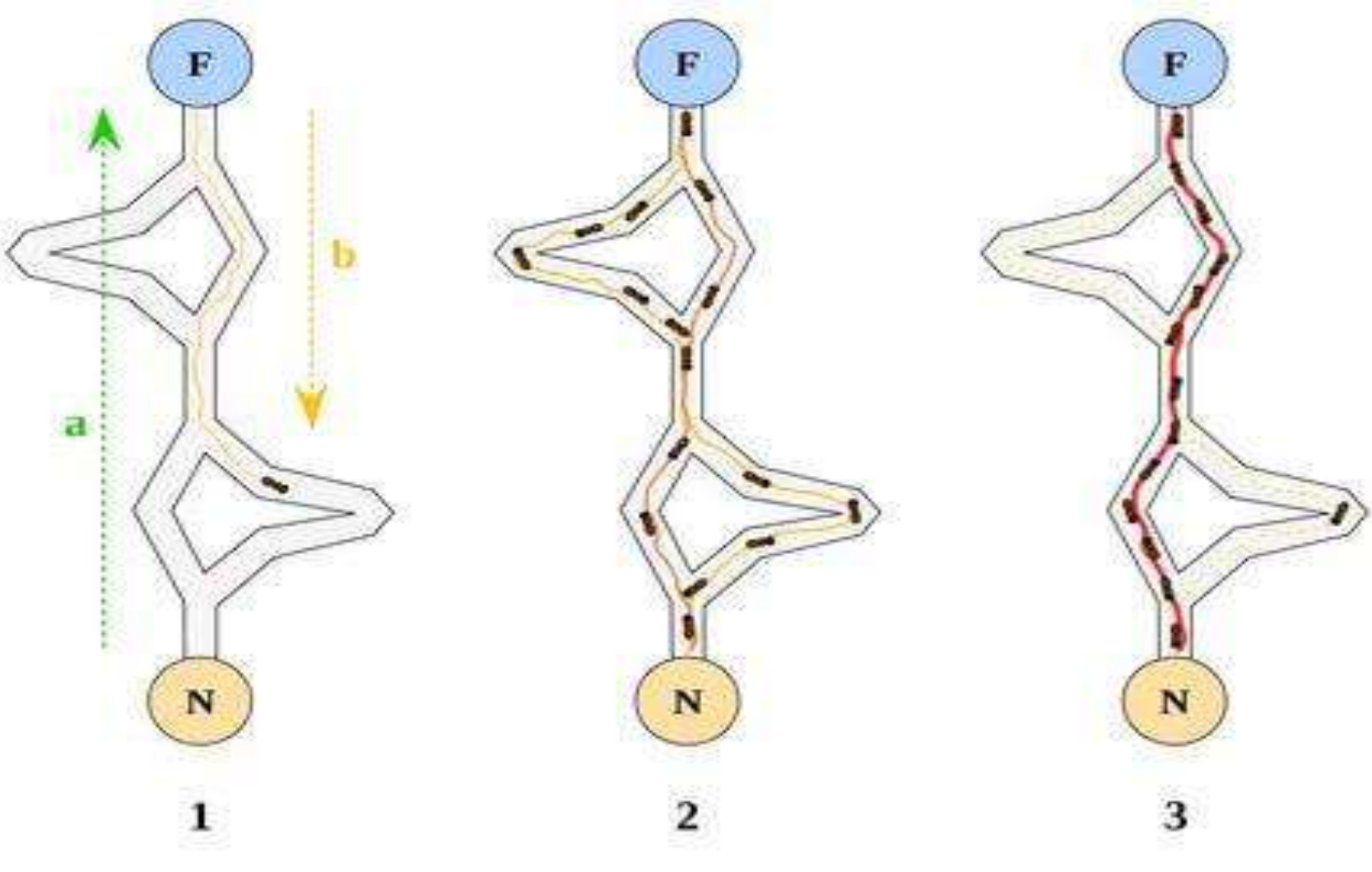
Значительная часть РА посвящена реализации моделей стайного поведения, и прежде всего – стайному **движению**.

Биоинспирированные методы вывода

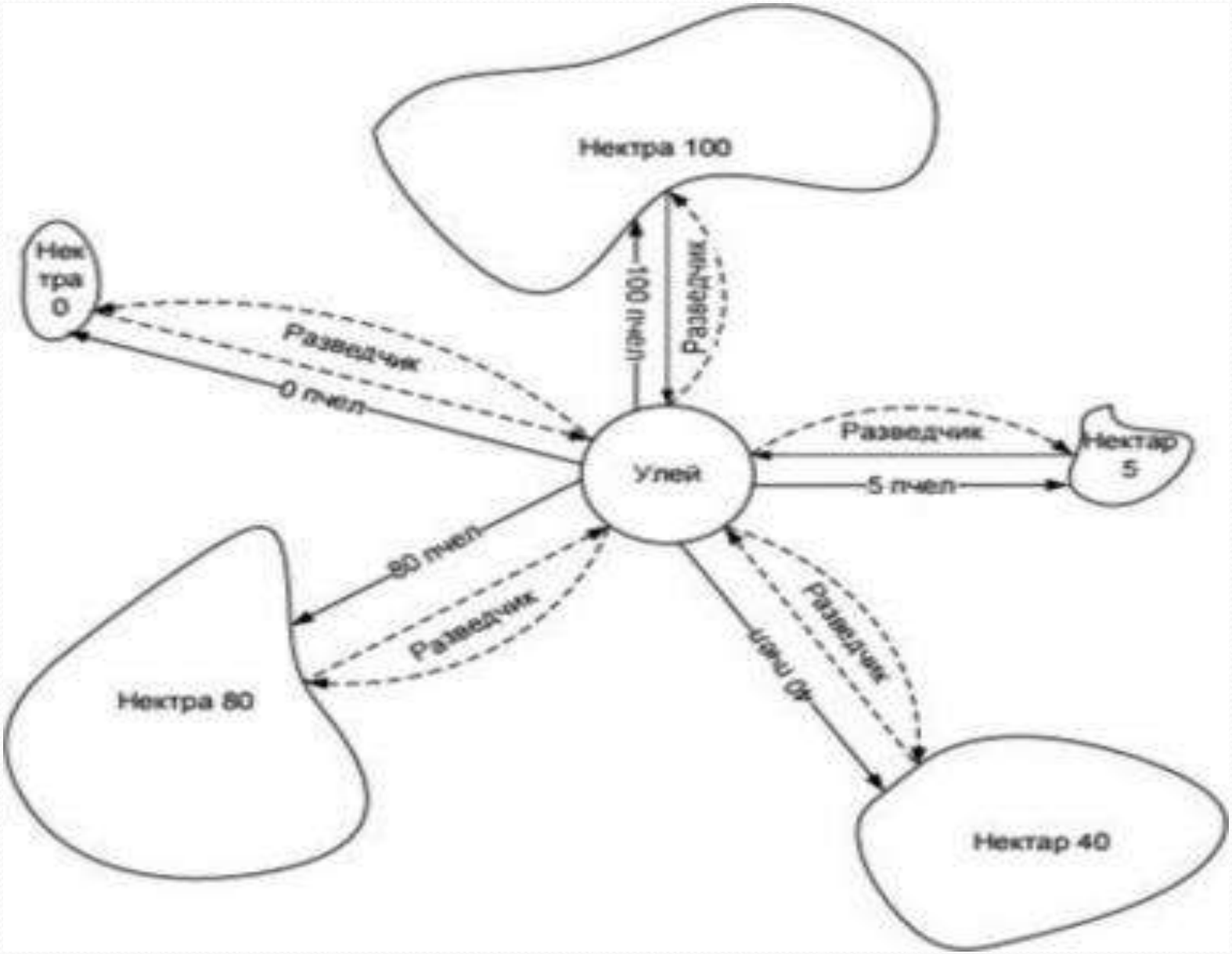
Это математические преобразования, трансформирующие входной поток посылок в выходные решения по правилам, основанным на имитации природных процессов, а также на вероятностном подходе к исследованию ситуаций и итерационном приближении к цели.

Ниже представлена схема правдоподобного вывода по правилам эволюции:

Муравьиный механизм вывода



Пчелиный механизм вывода



Благодарю за внимание !

Contact Information:

E-mail: srodzin@sfedu.ru

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Кафедра МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ И ПРИМЕНЕНИЯ ЭВМ

Дисциплина:

Машинное обучение и биоинспирированная оптимизация

ЛЕКЦИЯ 2

Машинное обучение в прикладных интеллектуальных системах: гипертекстовый поиск, многоагентные и онтологические системы, распознавание образов

Родзин Сергей Иванович,
профессор ИКТИБ ЮФУ,
srodzin@sfedu.ru

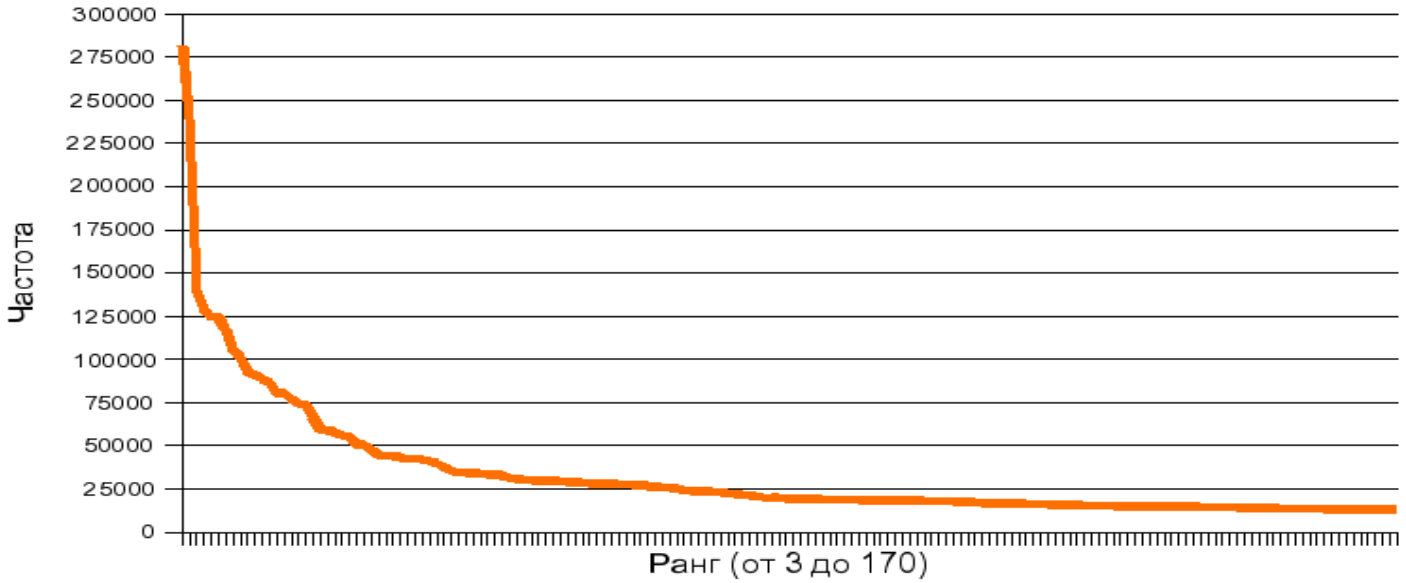
Первый закон Зипфа

Ранг - частота.

Имеется некоторый текст. Выбрав любое слово, можно подсчитать, сколько раз оно встречается в тексте. Эта – *частота* ***(y)*** вхождения слова.

Некоторые слова будут иметь одинаковую частоту. Их нетрудно сгруппировать, пронумеровать и расположить группы по мере убывания их частоты. Номер группы назовём *рангом* ***(x)*** частоты. Часто встречающиеся слова будут иметь ранг 1, следующие - 2 и т.д.

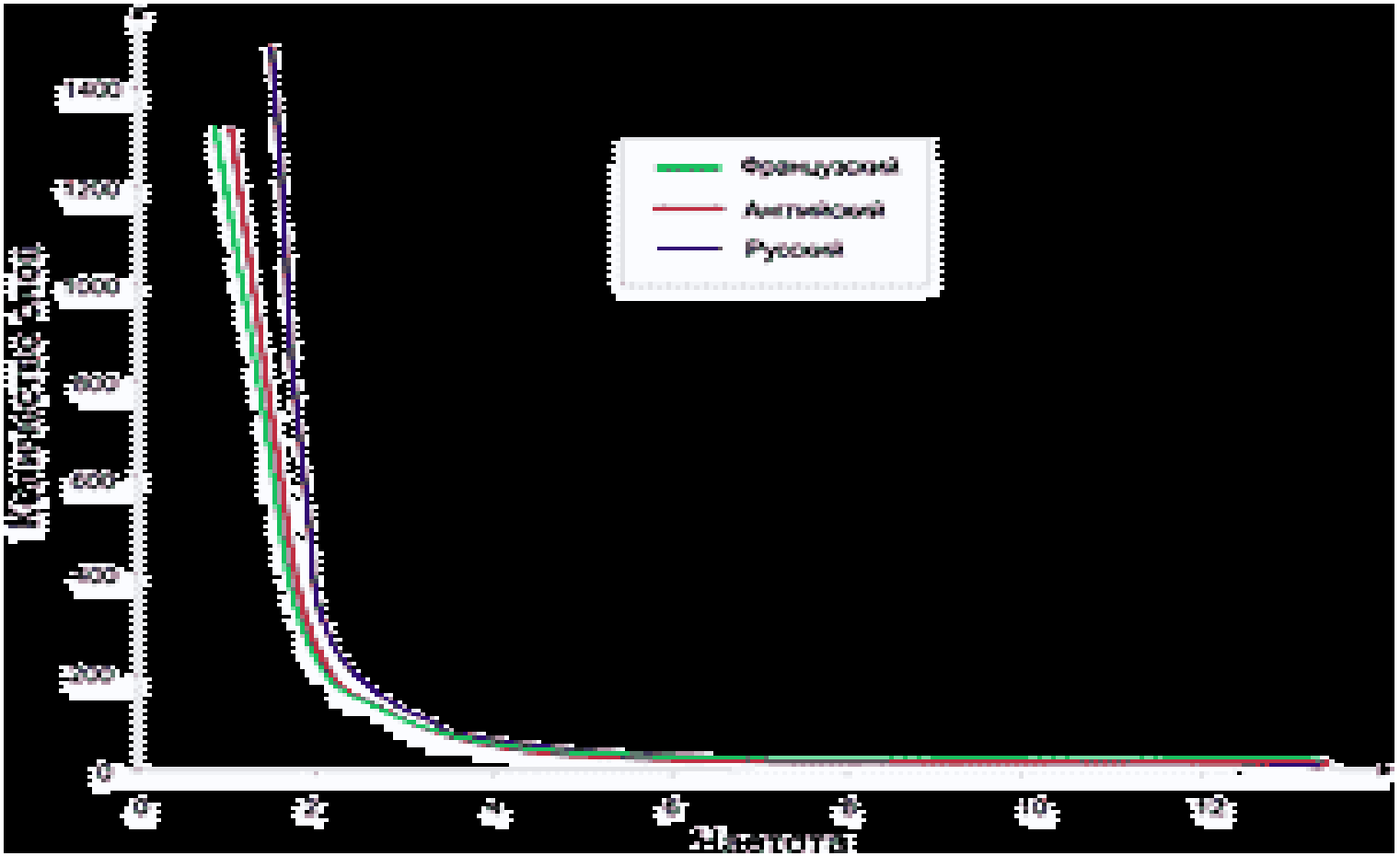
Зипф обнаружил интересную закономерность: $y = k/x$, где k - константа ($0<k<0,2$; для русского языка $k = 0,06..0,07$; для английского $k = 0,1$), зависящая от языка.



Второй закон Зипфа

Разные слова входят в текст с одинаковой частотой. Зипф установил, что *частота* ***(x)*** и *количество слов* ***(y)***, входящих в текст с этой частотой, тоже связаны между собой: *количество* слов в данной частоте = *константа*/*частота* слова, т.е.

$y = k/x$. Это также гипербола:

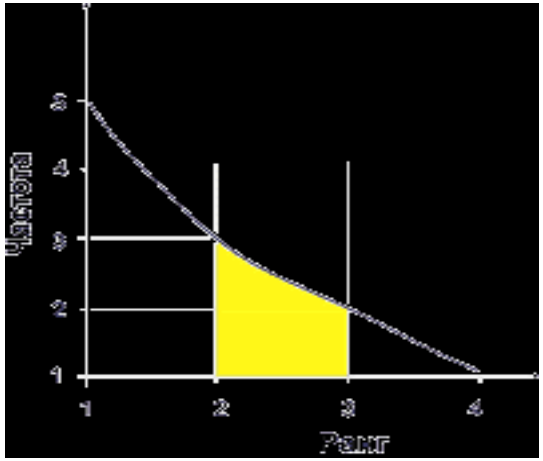


Как законы Зипфа помогают извлечь из текста ключевые слова?

Дан следующий текст: Законы Зипфа универсальны. В принципе, они применимы не только к текстам. Аналогична, например, зависимость количества городов от числа проживающих в них жителей. Характеристики популярности узлов в сети Интернет - тоже отвечают законам Зипфа. Не исключено, что в законах отражается "человеческое" происхождение объекта. Учёные давно бьются над расшифровкой манускриптов Войнича. Никто не знает, на каком языке написаны тексты и тексты ли это вообще. Однако исследование манускриптов на соответствие законам Зипфа доказало: это созданные человеком тексты. Графики для манускриптов Войнича точно повторили графики для текстов на известных языках.

Анализ по Зипфу показал - ключевыми будут слова из диапазона частот от 2 до 3:

Интернет-запрос типа:
«**закон&Зипфа+манускрипт&Войнича**»
непременно найдет этот документ. Правда, в диапазон 2..3 попадает «шум»: слова *на, не, для, например, это*. Их надо исключить, используя стоп-лист.

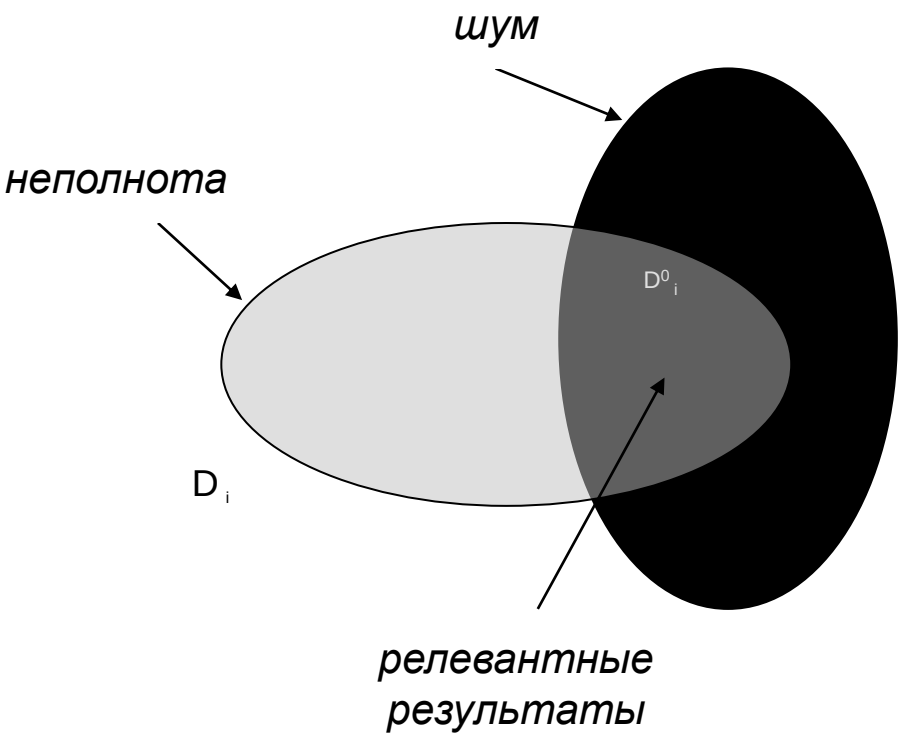


Оценка эффективности поиска: информационная полнота и шум

D^0_i – подмножество релевантных документов из D_i, по i-запросу, m – большое число . Информационная полнот ($\boldsymbol{k_П}$) и информационный шум ($\boldsymbol{k_Ш}$) на интервале [0, 1]. Идеальный вариант: $\boldsymbol{k_П}=1$, $\boldsymbol{k_Ш}=0$.

$$k_{П} = \lim_{k\to m} \frac{1}{k}\sum_{i=1}^{k} \frac{|D_i \cap D_i^0|}{|D_i|}$$

$$k_{Ш} = \lim_{k\to m} \frac{1}{k}\sum_{i=1}^{k} \frac{|D^0{}_i \setminus D_i|}{|D^0{}_i|}$$



В идеале $D^0_i = D_i$. Отсюда оценка эффективности информационного поиска:

$$E_1 = 2\lim_{k\to m} \frac{1}{k}\sum_{i=1}^{k} \frac{|D_i \cap D_i^0|}{|D_i| + |D_i^0|}$$

Мера ван Рисбергена

Вместо $k_{Ш}$ часто используют обратный эму показатель – ***коэффициент точности***: $k_T = 1 - k_{Ш}$

Тогда оценка ***эффективности информационного поиска*** равна:

$$E_1 = 2k_T k_П / (k_T + k_П)$$

При оценке эффективности инфопоиска часто используется мера ван Ризбергена:

$$E_\beta = \frac{(\beta^2 + 1) k_T k_П}{\beta^2 k_T + k_П}$$

где β – параметр, отражающий предпочтение пользователя ИПС (β = [0, ∞]).

При $\beta = 1$ точность и полнота одинаково важны. На интервале [0, 1] приоритет имеет точность, а на интервале [1, ∞] – полнота. Важные частные случаи:

- $E_{\beta=1} = E_1$;
- $E_{\beta=0} = k_T$ (значима только точность);
- $E_\beta \to \infty = k_П$ (значима только полнота).
  - для **документальных** ИПС: $k_{Пmax} = 0{,}5$; $k_{Шmax} = 1$;
  - для **фактографических** ИПС: $k_{Пmax} = 1$; $k_{Шmax} = 0$;
  - для **гипертекстовых** ИПС: $k_{Пmax} = 0{,}9 \div 1$; $k_{Шmax} = 0{,}1 \div 0{,}2$.

Поисковые агенты

- ***Кроулеры*** (просматривают заголовки страниц и возвращают машине только первую найденную ссылку),
- «***Роботы***» (проходят по ссылкам с различной глубиной),
- «***Пауки***» (сообщают о содержимом найденного документа, индексируют его и пересылают информацию в БД машины поиска).

Автоматическое реферирование и аннотирование.

Системы машинного перевода:

По Интернету до сих пор бродит шутка о переводчике ПРОМТ, который перевёл фразу «*My cat has given birth to four kittens, two yellow, one white and one black*», как «*Моя кошка родила четырёх котят, два желтых цвета, одного белого и одного афроамериканца*». Главной трудностью является семантический анализ и понимание ЕЯ-текста.

Автоматическая классификация документов.

Программные продукты, реализующие технологии обработки текстов на ЕЯ

- Смысловой анализатор текста ***Text Analyst***;
- ИПС ***ERW***;
- Пакет ***NeurOK Semantic Suite***.

Интеллектуальный агент (ИА)

1. В **информатике и программной инженерии** ИА – это программа, самостоятельно выполняющая некоторое задание;
2. В **ИИ** – это разумная сущность, воспринимающая окружающую среду и способная действовать в ней.

ИА может быть математически представлен как некоторая агентская функция:

f: P (результат восприятия) → A (действия).

В зависимости от типа функции различают следующие типы:

Агенты с простым поведением действуют только на основе текущих знаний. Их агентская функция основана на схеме продукционных правил типа «условие-действие»:

IF (условие) THEN действие.

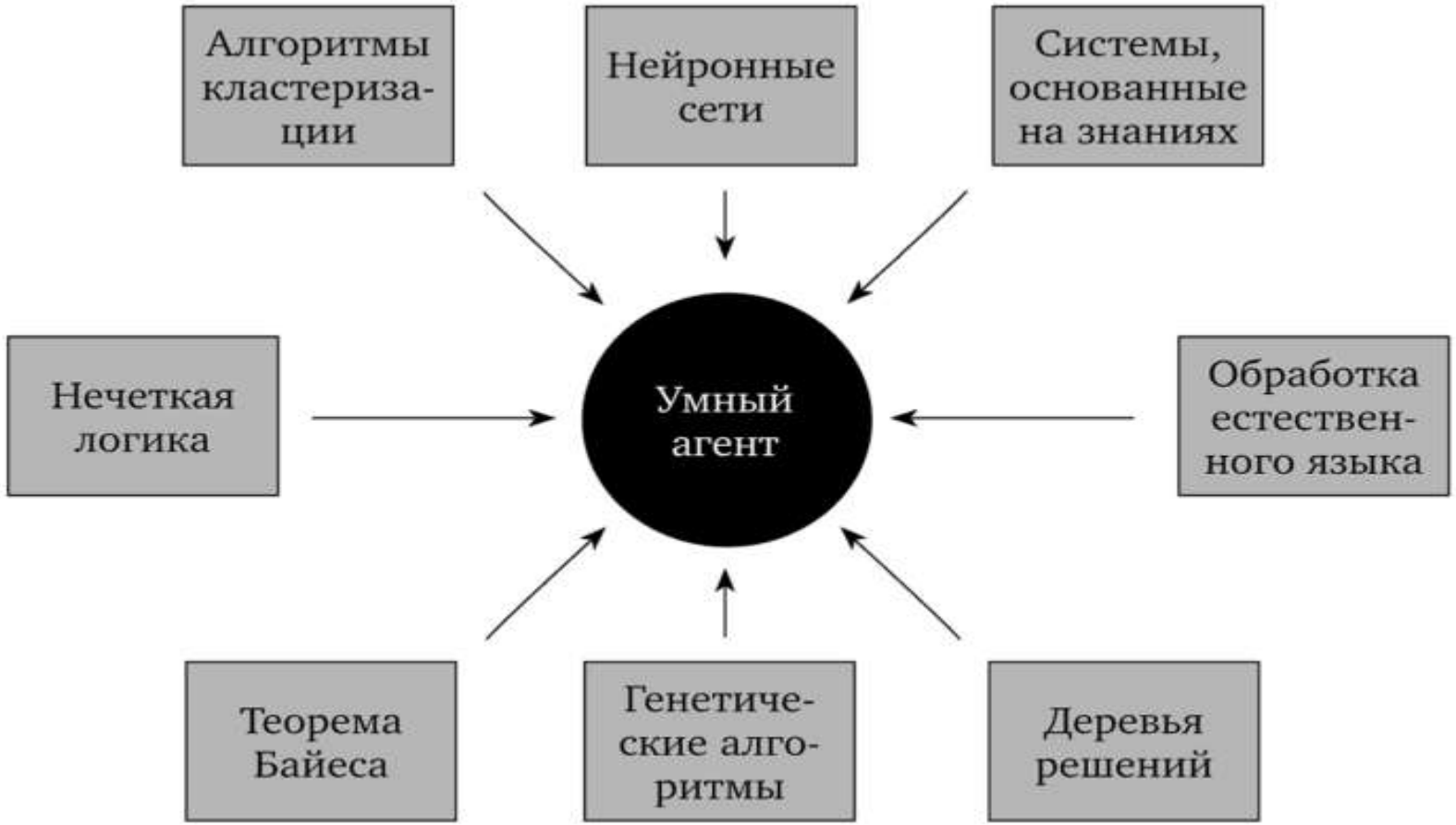
Целенаправленные агенты хранят информацию о тех ситуациях, которые для них желательны. Это дает агенту способ выбрать среди многих путей тот, что приведет к нужной цели.

Практичные агенты различают, когда цель достигнута, и когда не достигнута, а также насколько желательно для них текущее состояние с помощью «функции полезности».

Обучающиеся агенты (автономные интеллектуальные агенты) должны обладать некоторой независимостью, способность к обучению и приспособлению.

Области знания и технологии, используемые ИА

Простая компьютерная программа отличается от ИА тем, что «не утруждает» себя целевым поведением и анализом результатов. Напротив, ИА, представляющий интересы пользователя, «заинтересован» в том, чтобы задание было выполнено. Агенты всегда формируют список выполненных действий, результаты тестирования и верификации и отсылают его в управляющую систему.



Интеллектуальный агент Эдди (Eddie)

Недавно создана программа-ИА **Эдди (Eddie)**. Разработчики считают, что Эдди обладает интеллектом 4-х летнего ребенка и способен обучаться. Для обучения Эдди создали аватар, который живет в среде компьютерной игры ***SecondLife***. Специалисты полагают, что, общаясь с другими аватарами, Эдди многое узнает и поймет.

Примеры ИА

Роботы по закупкам, просматривают сетевые ресурсы, собирают информацию о товарах и услугах. *Amazon.com* является отличным примером такого робота. Веб-сайт предложит Вам список товаров, основываясь на том, что Вы покупали в прошлом.

Пользовательские или ***персональные агенты*** - это агенты, которые действуют в ваших интересах, от вашего имени. Например, проверяют почту, сортируют её, оповещают о поступлении писем; играют в компьютерной игре как ваш оппонент; собирают новости; т.п.

Управляющие и ***наблюдающие агенты,*** например, ведут наблюдение за компьютерными сетями и следят за конфигурацией каждого компьютера, подключенного к сети, или управляют ботами компьютерных игр (*Quake, Robot soccer, Hoshimi*).

Агенты, добывающие информацию, действуют в хранилищах данных, собирая информацию, и предупреждая Вас о наличии новой информации.

Многоагентная система

Многоагентные системы (МАС) являются *распределенными* интеллектуальными системами и представляют собой системы с взаимодействующими интеллектуальными агентами.

МАС создаются для решения проблем, которые сложно или невозможно решить с помощью одного агента или монолитной системы (онлайн-торговля, ликвидация чрезвычайных ситуаций, моделирование социальных структур).

В МАС агенты имеют несколько важных характеристик:

- Автономность (агенты частично независимы),
- Ни у одного из агентов нет представления о всей системе,
- Децентрализация (нет агентов, управляющих всей системой).

В МАС может проявляться самоорганизация и сложное поведение, даже если каждый агент прост (***роевой интеллект***).

Агенты могут обмениваться полученными знаниями, используя некоторый специальный язык и подчиняясь установленным протоколам. Примеры таких языков ***KQML*** и ***ACL***.

Какие темы сейчас исследуются в рамках МАС? - Самоорганизация, координация, взаимодействия агентов, знания и намерения агентов, мультиагентное обучение.

Тарасов В.Б. От многоагентных систем к интеллектуальным организациям. – М.:Эдиториал УРСС, 2002. – 352 с.

КЛАССИФИКАЦИЯ МНОГОАГЕНТНЫХ СИСТЕМ

Координация в МАС

Координация предназначена для согласования целей и поведения агентов, при которых каждый агент улучшает или не ухудшает свою функцию полезности, а система в целом улучшает качество решения общей задачи. Методы решения задачи координации базируются на теории принятия решений, теории игр и планирования.

В основе большинства известных методов координации МАС лежит **гипотеза**: все механизмы координации в МАС могут быть выражены в совместных **обязательствах** и **соглашениях** агентов. Одна из форм обязательств агента - его роль, а соглашение фиксирует условия, при которых обязательства выполняются.

Примерами координации в МАС являются:

1: Управление воздушным движением в районе аэропорта и разрешение непредвиденных конфликтов.

2: Команда агентов, которые функционируют в динамической внешней среде в присутствии шума и противодействия со стороны других команд соперников (команды роботов в соревнованиях ***RoboCup***).

Во многих приложениях агенты, так или иначе, конкурируют. Это "эгоистичные" агенты, каждый из них стремится максимизировать свою функцию полезности, игнорируя интересы других агентов. В таких МАС взаимодействие агентов выражается в форме **переговоров**.

Самоорганизация в МАС

Сложность современных приложений такова, что всякое централизованное управление просто невозможно. Как управлять системой, в которой информация поставляется миллионами сенсоров?

Самоорганизация – это динамические и адаптивные процессы, ведущие к поддержанию системы без внешнего управления.

Пример самоорганизации – роевые **муравьиные алгоритмы** на принципах:

1. Взаимодействие между особями через среду.
2. Запаздывающая положительная обратная связь (например, увеличение количества феромона, оставляемого муравьями при обнаружении источника пищи)
3. Запаздывающая отрицательная обратная связь (испарение феромона по времени)
4. Изменение поведения с увеличением количества феромона на пути к пище.

Какие основные черты самоорганизации:

1. **Автономность** – взаимодействие с внешним миром без управления извне.
2. **Адаптивность** и **робастность** по отношению к изменениям среды.
3. **Возрастание порядка** с возрастанием организации системы; появление состояний системы на метауровне – аттракторов и новых свойств системы;
4. **Динамика** – самоорганизация это процесс, а не конечное состояние.

Примерами сложного взаимодействия с чертами самоорганизации являются **аукционы** и **биржи**.

Пример прикладной МАС

Управление заказами такси. МАС планирования и оптимизации использования ресурсов такси была разработана компанией *Magenta*. Задача характеризуется следующими количественными параметрами. В день поступает более 13 тысяч заказов, таксопарк насчитывает около 2000 машин, оснащенных средствами GPS-навигации, прием заказов выполняет 130 диспетчеров. В разработанной системе для каждого заказа и каждой машины назначается свой агент. После события получения заказа и до момента принятия решения по его выполнению имеется интервал времени. В течение этого интервала поступление других заказов может изменять текущую ситуацию диспетчерами компании.

До внедрения этой системы все управление заказами выполнялось диспетчерами компании. Внедрение МАС позволило получить следующие эффекты. 98,5% заказов теперь планируются автоматически, без участия диспетчеров; число заказов, выполненных не вовремя, сократилось в 3,5 раза; на 22,5% уменьшился холостой пробег такси; каждое такси теперь выполняет по 2 дополнительных поездки в неделю при тех же затратах времени и горючего. В целом это обеспечило получение дополнительной прибыли компании около 5 миллионов долларов в год.

Другие примеры прикладных МАС

Логистика. Наиболее распространены МАС в логистике, а также в управлении процессами с несколькими разнесёнными в пространстве участниками, установлении координации в их работе. Например, известно применение МАС в качестве систем регулирования городского движения.

Социальные отношения. Известны МАС моделирования международных отношений, торговли, военных действий.

Робототехника. Современные автоматизированные бытовые системы сильно зависят от человека. Дальнейшее их усложнение чревато понижением стабильности и надёжности их работы. Интересен путь к усложнению автоматизированных систем через взаимодействие, общение составляющих их автономных участников.

Монолитные системы сталкиваются с серьёзными проблемами. Примером могут служить всё усложняющие системы управления производством, операционные системы вычислительных устройств (аналог агентов – службы-services в ***Windows*** или процессы-демоны в ***Linux***), области исследования общественных наук.

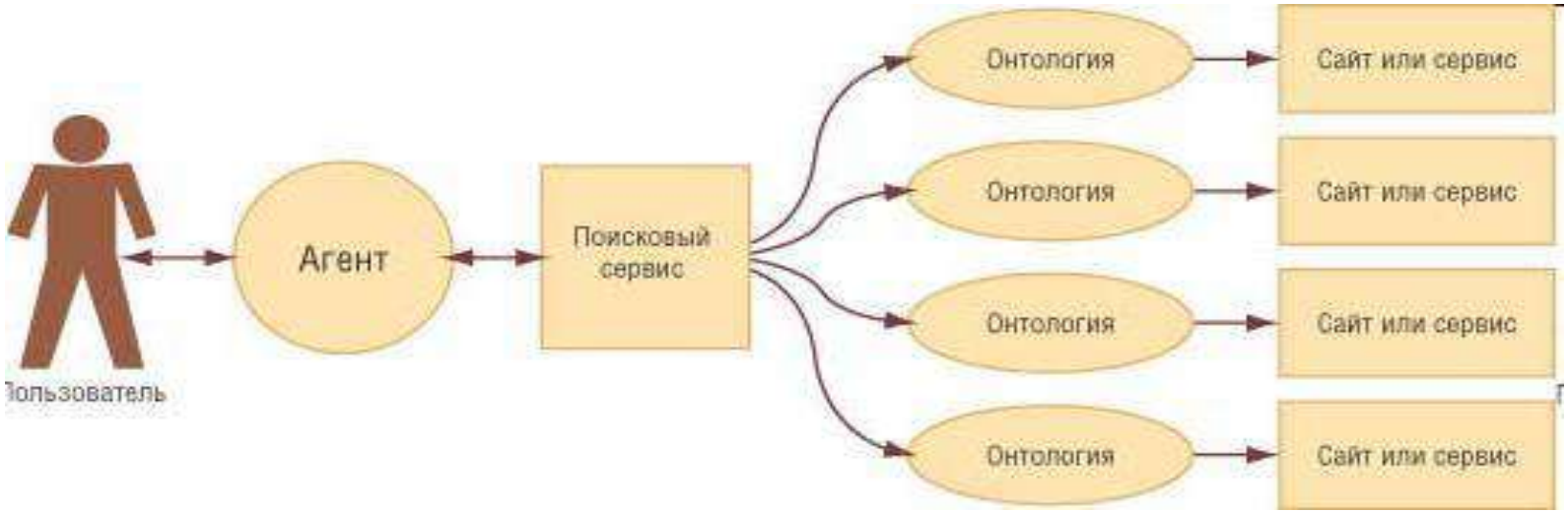
Высокопроизводительные системы начали строиться с использованием параллелизма и распределённости. Ярким примером этого являются **облачные сервисы**, использование **многоядерных процессоров в мобильных телефонах**.

Существует довольно большой набор платформ, подходящих для создания МАС. В него входят **NetLogo, StarLogo, Repast Simphony, Eclipse AMP, JADE, Jason**.

Что такое онтология?

Онтология (от греч. ontos - сущее) – это описание на формальном языке понятий ПО и семантических отношений между ними. Это знания о знаниях, т.е. метазнания.

В Интернете онтологии скоро станут обычным явлением. Сейчас в сети онтологии варьируются от больших таксономий по категориям веб-сайтов (**Yahoo**!), до каталогов продаваемых товаров и их характеристик (как на сайте Amazon.com).



В области медицины создан стандартный структурированный словарь **SNOMED** и язык (**UMLS**).

Появляются общецелевые онтологии. Например, Программа ООН по развитию онтологии **UNSPSC**, которая предоставляет терминологию товаров и услуг.

Формальное определение онтологии

O = <X, R, F>,

где **X** - множество понятий (терминов); **R** - множество отношений между понятиями; **F** - множество интерпретаций (функций) понятий.

Частные случаи онтологии:
Простой **словарь** (X ≠ Ø, R = Ø, F ≠ Ø);
Таксономия (X≠Ø, R={is-a}, F≠Ø);
Тезаурус (X ≠ Ø, R={eq-to}, F ≠ Ø);
Каталог-справочник понятий и ссылками на них.

Например, онтология издательства. Основные понятия этой предметной области: **издательство, книга** и **журнал**. Это классы онтологии:

Процесс разработки онтологии

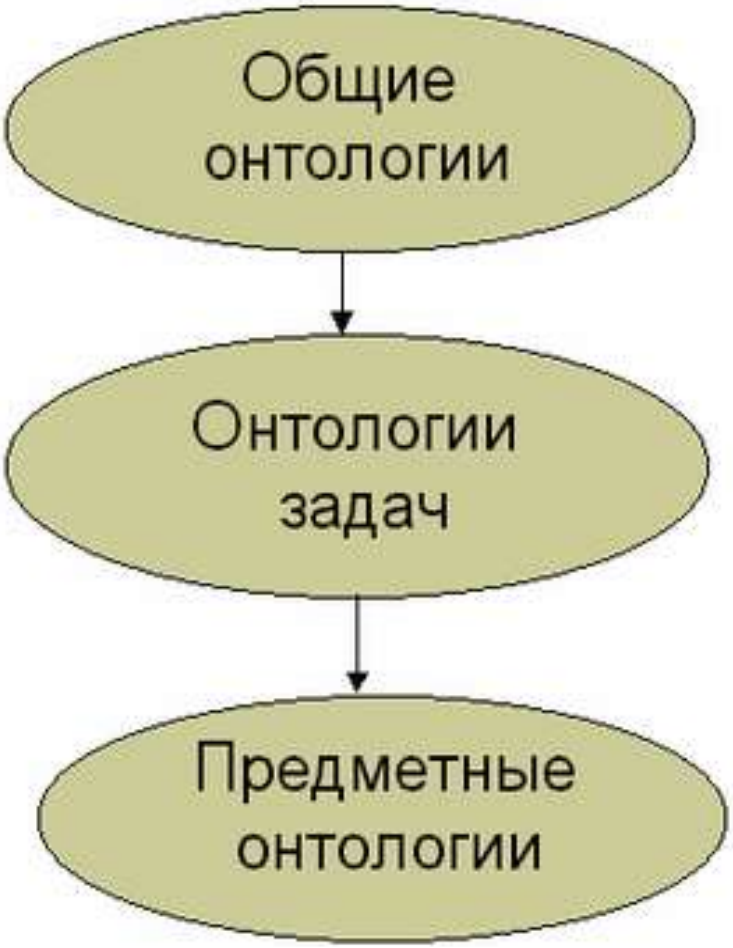
1) Составляется **глоссарий понятий** и на естественном языке создается список точных определений;

2) Строятся деревья классификации понятий (**иерархия классов**), выделяются атрибуты классов и их возможные значения. Устанавливаются основные связи между классами;

3) В зависимости от целей, для которых **разрабатывается онтология**, в нее могут добавляться экземпляры классов.

4) Эксперты по той предметной области, в которой разрабатывается онтология, создают правила логических выводов, позволяющие оперировать данными, представленными в онтологии, и извлекать из созданной онтологии новые знания.

Классификация онтологических систем по содержимому



Сущности, События, Пространство, Время...

Составление расписаний, определение целей, классификация

Множества предметов: скальпели, сканеры

Основными инструментами онтологического инжиниринга являются ***Protégé, Ontolingua, Altova Semantic Works***.

Области применения онтологий

Проект Семантической сети (**Semantic Web**);
Область информационного поиска (**Information Retrieval**).

Суть проекта Семантической сети состоит в автоматизации "интеллектуальных" задач семантической обработки тех или иных ресурсов, имеющихся в Интернет. Обработкой и обменом информации должны заниматься не люди, а специальные интеллектуальные агенты (программы, размещенные в Сети). Но для того, чтобы взаимодействовать между собой, агенты должны иметь общее формальное представление значения для любого ресурса. Именно для этой цели в Semantic Web используются онтологии.

Немало специалистов считают, что Семантический Web — это утопия. Его внедрение требует немалых усилий от разработчиков и пользователей, включая разработчиков крупных информационных порталов, поисковых систем, каталогов и небольших сайтов, а также от разработчиков интеллектуальных программ-агентов.

Что такое распознавание образов?

"Раздел ИИ, развивающий теоретические основы, методы классификации и идентификации предметов, явлений, процессов, сигналов, ситуаций и т.п., которые характеризуются конечным набором некоторых свойств и признаков" (Wikipedia.org)

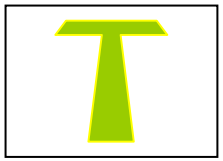
✓ Задача установления различий между исходными данными, при помощи поиска инвариантных свойств объектов, образующих образы.

✓ Процедура, позволяющая вынести решение о принадлежности данного изображения или его фрагмента к одному из n классов (**n>1)**

Психофизиологический процесс взаимодействия индивида с физическим раздражителем. Шансы распознавания зависят от прошлого опыта.

Актуальные задачи РО

Анализ спутниковых снимков



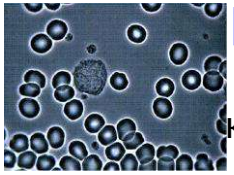
Распознавание букв

Обнаружение объектов

- Обнаружение лиц
- Обнаружение красных глаз на фото (для коррекции)
- Обнаружение антропометрических точек лица

Распознавание автомобильных номеров

Зрение роботов



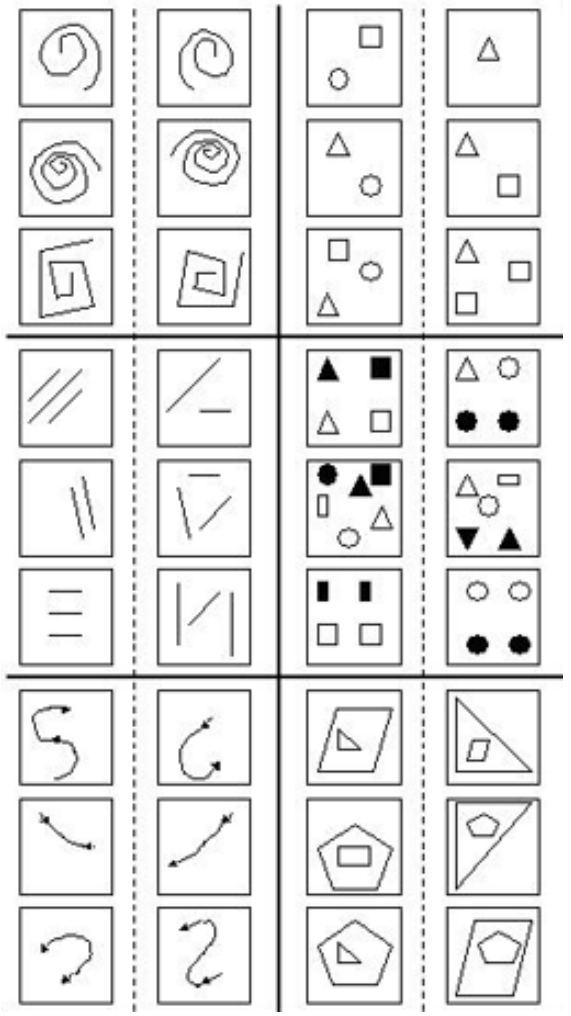
Медицина
(анализ клеток в крови)

Дефектоскопия
(автоматический поиск трещин в асфальте по ИК изображениям)

Гипотеза распознавания

Теория распознавания образов базируется на ***гипотезе компактности***: образам соответствуют компактные множества («сгустки» точек), объединенные в класс в пространстве признаков.

Образное восприятие мира — одно из самых загадочных свойств мозга. Воспринимая внешний мир, мы всегда проводим классификацию поступающей информации. Образы обладают характерными свойствами. Разные люди, одинаково и независимо друг от друга классифицируют одни и те же образы. Эта объективность образов позволяет людям понимать друг друга.



В целом проблема распознавания образов состоит из двух частей: обучения и распознавания.

Например, для указанных изображений следует отобрать признаки для различения левой триады картинок от правой.

Решение задачи требует моделирования логического мышления в полном объеме.

Приложения теории распознавания образов

Области приложения:	Примеры задач:
БИОИНФОРМАТИКА	Поиск шаблонов в ДНК
БАЗЫ ДАННЫХ	Поиск и классификация
ОБРАБОТКА ТЕКСТОВ	Тематическая классификация
АНАЛИЗ ИЗОБРАЖЕНИЙ	Распознавание букв
ПРОИЗВОДСТВО	Контроль качества продукции
ПОИСК по МУЛЬТИМЕДИА	Определение жанров,….
ПРОГНОЗИРОВАНИЕ	Погода, сейсмология, геология
БИОМЕТРИЯ	Отпечатки пальцев,….
ОБРАБОТКА РЕЧИ	Перевод аудио в текст и обратно
МЕДИЦИНА	Диагностика
ВОЕННОЕ ДЕЛО	Радиолокация объектов наблюдения
СЕЛЬСКОЕ ХОЗЯЙСТВО	Размер урожая
КОСМОС	Контроль космического пространства

Задачи распознавания образов

1. Определение полного перечня признаков, характеризующих образ. Признаки подразделяются на *детерминированные* (конкретные числа), *вероятностные*, *логические* и *структурные*;

2. Первичная классификация образов и составление *априорного алфавита классов*, т.е. выбор принципа классификации;

3. Разработка *априорного словаря признаков* (на основе задачи 1), в который вносят лишь те признаки, по которым может быть получена априорная информация;

4. Описание априорного алфавита классов на языке априорного словаря признаков;

5. Разбиение пространства признаков на области по классам алфавита с помощью *решающих функций* (трудоемкая задача по сложности практически равносильная задаче распознавания образов);

6. Выбор и применение *алгоритмов распознавания* (вычисление мер сходства/различия путем оценки расстояния от распознаваемого объекта до каждого из классов);

7. Определение *рабочих алфавита классов* и *словаря признаков*.

8. Разработка специальных *алгоритмов управления* СРО;

9. Выбор показателей *эффективности* работы СРО (вероятность правильного решения, время распознавания, величина ресурсов и т.д.).

Постановка общей задачи (проблемы) распознавания образов

Автоматизация обучения и распознавания неизвестных объектов и составляет общую задачу распознавания образов. Ее постановка состоит в следующем:

В условиях первоначального описания классов на языке признаков необходимо в пределах выделенных технических средств и ресурсов определить:

- оптимальный алфавит классов,
- оптимальный рабочий словарь признаков,

которые при наилучшем решающем правиле обеспечивают **эффективное** использование решений, принимаемых по результатам распознавания.

Методы обучения распознаванию образов

Сравнение с образцом (детерминированныеметоды):

- Применение геометрической нормализации и функций расстояния

Вероятностные (статистические) методы

- Определение вероятностного распределения для каждого класса и классификация по правилу Байеса

Логические методы

- Построение и решение системы булевых уравнений

Структурные методы

- Разбиение образа на элементы. Построение синтаксических правил в зависимости от вхождения/невхождения отдельных элементов или их последовательности в образ

Нейросети

- Выбор архитектуры сети и настройка ее весов

Классификация систем распознавания

В зависимости от *объекта распознавания* различают:

- текстовые системы распознавания (***text recognition***),
- системы распознавания речи (***speech recognition***),
- системы распознавания графических объектов (***image recognition***) и др.

По *однородности* или *неоднородности* признаков СРО разделяются на:

- ***простые*** (например, автоматы для размена монет),
- ***сложные*** (например, системы медицинской диагностики).

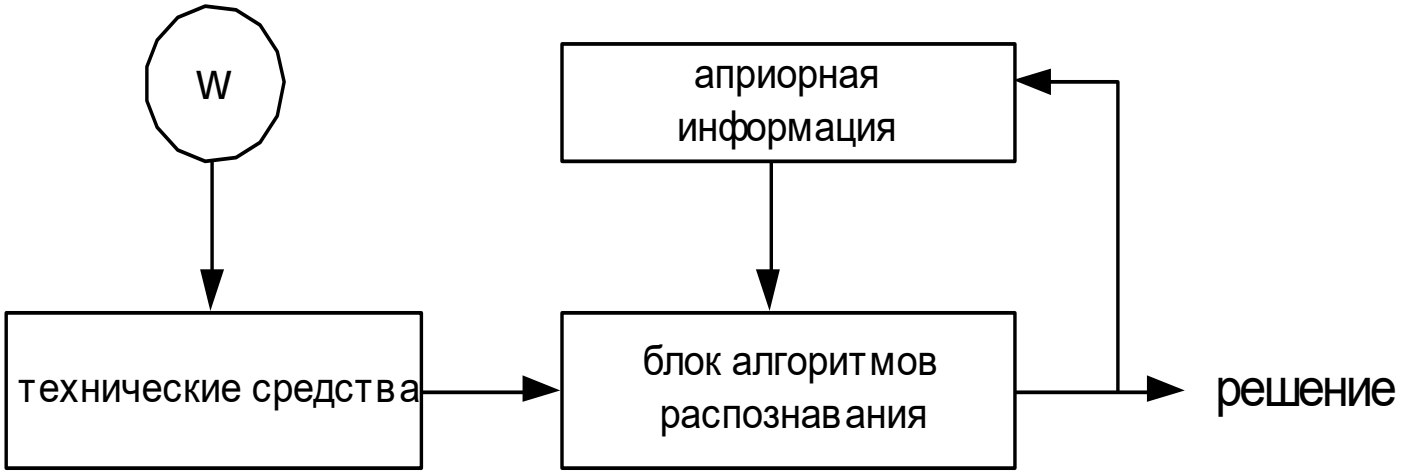
По достаточности для распознавания количества *априорной информации* различают СРО:

- ***без учителя***,
- ***системы с учителем*** и
- ***самообучающиеся системы***.

По типу априорного/рабочего словаря признаков различают СРО:

- *Детерминированные*,
- *Вероятностные*,
- *Логические*,
- *Структурные*.

Системы распознавания без учителя

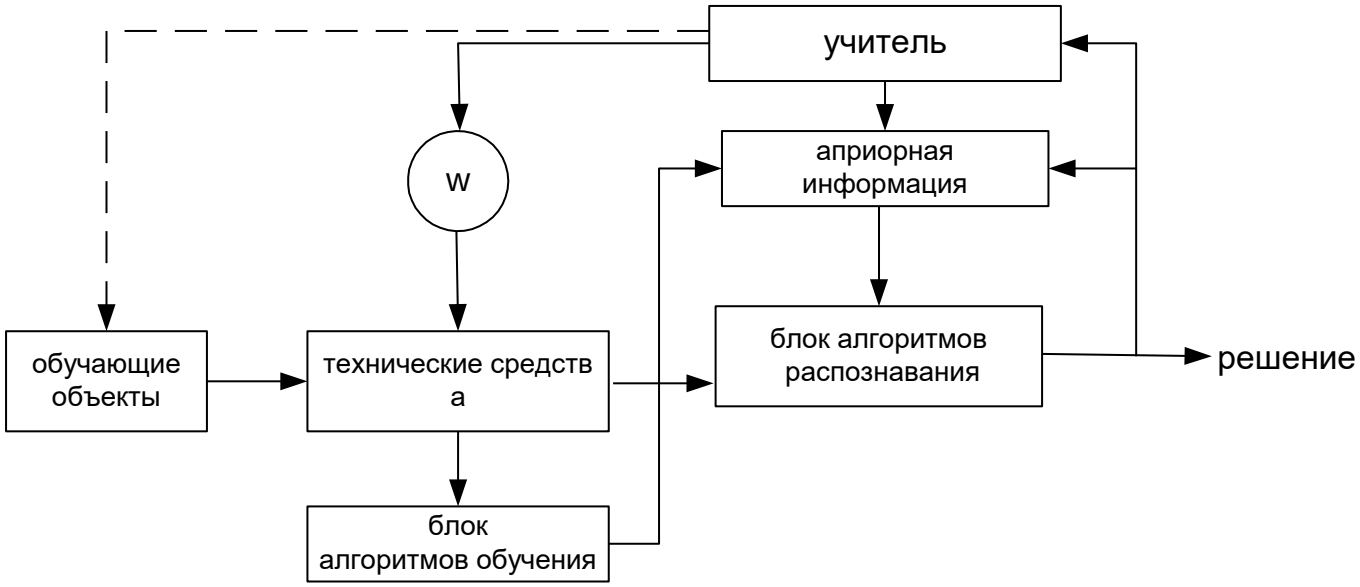


В СРО без учителя количество априорной информации достаточно, чтобы определить априорный алфавит классов, априорный словарь признаков и решающие правила. Это означает, что удалось найти общее свойство, не зависящее природы образов.

Тогда задача формулируется так: системе предъявляются объекты без указаний об их принадлежности к образам. СРО отображает множество объектов на множество их образов и, используя решающую функцию производит классификацию объектов.

Главная черта, делающая обучение без учителя привлекательным, – это его "самостоятельность". Например, процесс обучения нейросети заключается в настройке весов синапсов.

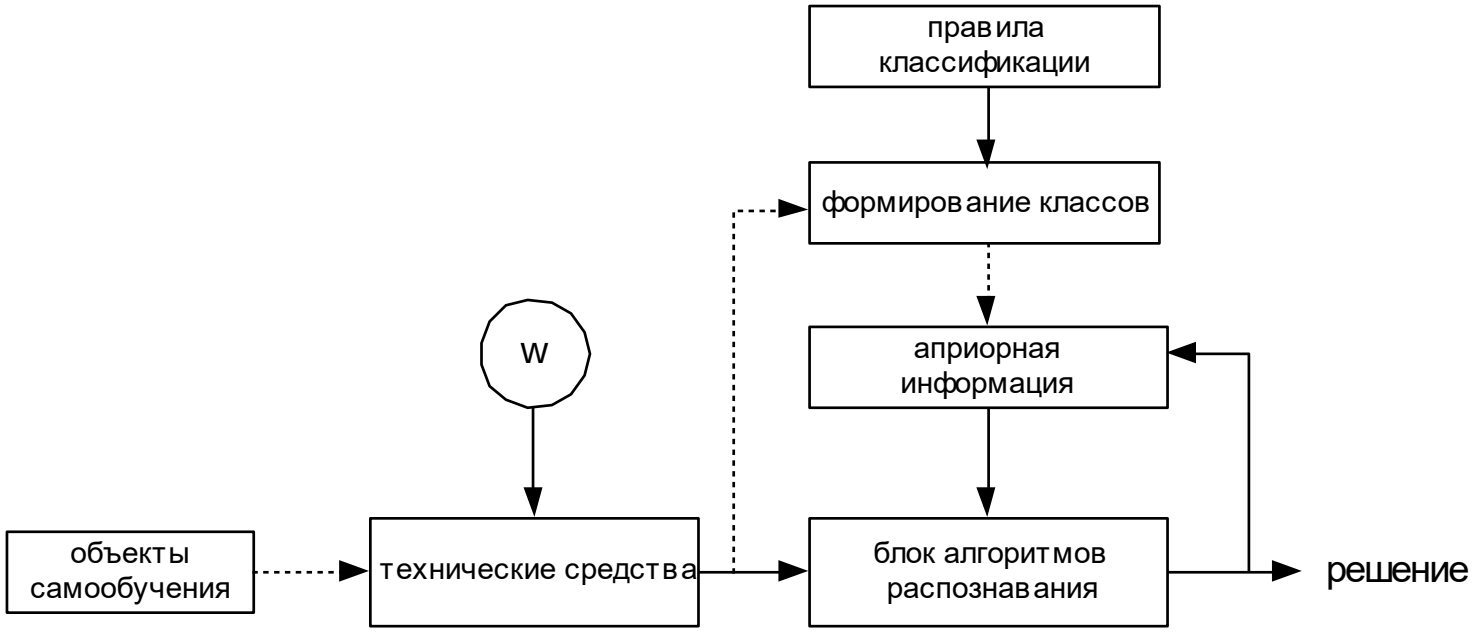
СРО с учителем



В системах с учителем априорной информации достаточно, чтобы определить априорные алфавит классов и словарь признаков, но недостаточно для описания классов на языке признаков.

В системах с учителем обучение - это процесс выработки в некоторой системе той или иной реакции на группы образов путем многократной корректировки. Такую корректировку в обучении принято называть "поощрениями" и "наказаниями". Механизм генерации этой корректировки практически полностью определяет алгоритм обучения, в котором системе сообщается информация о верности ее реакции.

Самообучающиеся СРО



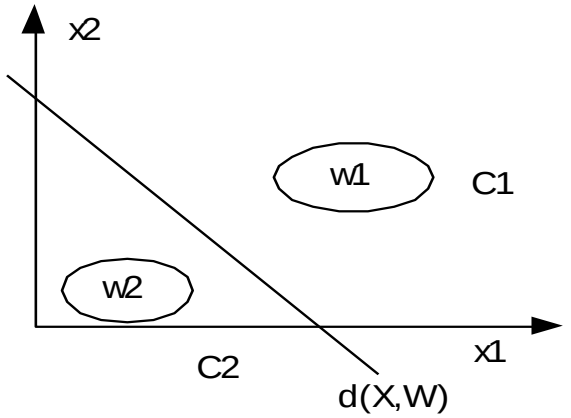
В самообучающихся системах априорной информации достаточно лишь для определения априорного словаря признаков, но не достаточно для классификации объектов, т.е. система не получает указаний о принадлежности объекта к какому-то классу.

Большинство известных алгоритмов самообучения способны выделять только абстрактные образы, т.е. компактные множества в заданных пространствах. Это повышает их ценность, т.к. часто образы заранее не определены, поэтому надо определить, какие изображения в заданном пространстве представляют собой образы. Примером такой постановки задачи являются социологические исследования, когда по набору вопросов выделяются группы людей.

Детерминированные СРО

В детерминированных СРО образы рассматриваются как точки в **n**-мерном пространстве $\mathbf{R^n}$. Задача состоит в определении принадлежности образа $\mathbf{X} \in \mathbf{R^n}$ одному из классов $\mathbf{\Omega}=\{\omega1, \omega2,\ldots\omega k\}$ путем построения в n-мерном пространстве **k** областей **Ci**, i=1..k, причем $\mathbf{X} \in \mathbf{Ci}$.

Пример разделения двумерного пространства признаков на два класса:



Правило классификации: $\mathbf{X} \in \boldsymbol{\omega i}$, если $\mathbf{X} \in \mathbf{Ci}$. Если имеется **k** классов, то необходимо построить **k** взаимно непересекающихся областей. Единственным способом решения этой задачи является нахождение ***решающей функции***. В этом случае каждая область **Ci** будет описываться с помощью системы нелинейных неравенств:

$$\begin{cases} d_1(\overline{X}) > 0, \\ d_2(\overline{X}) \le 0, \\ \ldots \\ d_m(\overline{X}) \ge 0; \end{cases}$$

где d_i *– решающая функция*

На практике чаще всего стараются применять линейные решающие функции вида: $d(X,W)=XW=x_1w_1+\ldots+x_nw_n+w_{n+1}$, где вектор $W=(w_1,w_2,\ldots, w_{n+1})$ – параметры или «веса».

Построение линейных решающих функций связано с решением систем неравенств. В общем случае система может оказаться *несовместной* (множество решений пусто), классы являются линейно неразделимыми, и решение необходимо искать для нелинейных функций.

Классификация образов по расстоянию

Человек при классификации интуитивно чувствует, к какому классу ближе расположен некоторый образ. В алгоритмах компьютерной классификации для детерминированных СРО используют ***функцию расстояния***, которая выражает степень близости распознаваемого образа к классу.

В ***кластерном анализе*** существуют несколько способов определения функции расстояния. **Критерий качества кластеризации** следующие:

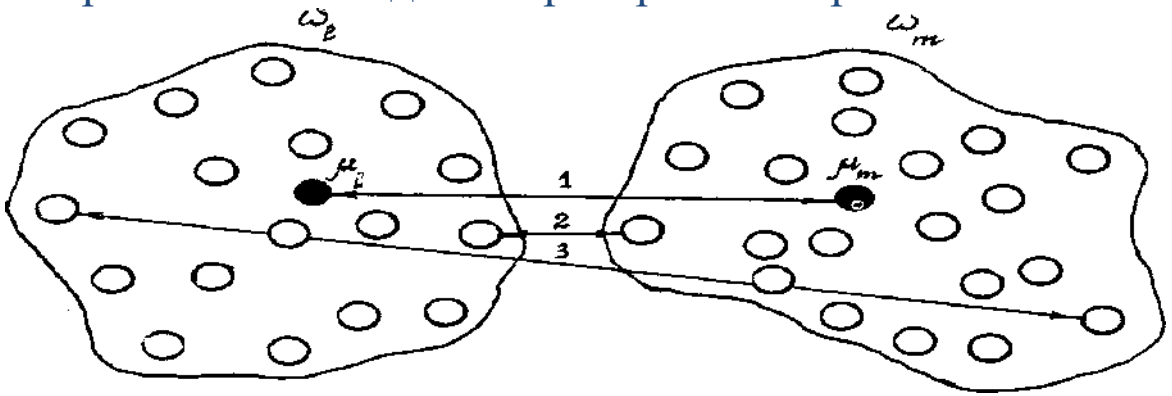
а) внутри групп объекты должны быть тесно связаны между собой;

б) объекты разных групп должны быть далеки друг от друга;

в) распределение объектов по группам желательно равномерное.

Требования **а**) и **б**) выражают гипотезу компактности классов; требование **в**) должно препятствовать объединению отдельных групп объектов.

Главным в кластерном анализе считается выбор ***метрики*** близости объектов. В конкретной задаче выбор свой, с учетом целей исследования, физической и статистической природы используемой информации и т.п., на основе специальных алгоритмов преобразования исходного пространства признаков.



Алгоритмы кластеризации

Если гипотеза о типе кластеров неверна, то это может приводить к неоптимальным или даже неправильным результатам. Поэтому в условиях априорной неопределенности применяют комплекс алгоритмов кластеризации.

Наиболее популярными алгоритмами кластеризации являются:

- ***Алгоритм k-внутригрупповых средних***;
- ***Алгоритм максиминного расстояния***;
- ***Класс алгоритмов ISODATA***.

Алгоритмы включают два этапа:

1. Задается исходное разбиение объектов на классы и определяется некоторый критерий качества автоматической классификации;
2. Объекты переносятся из класса в класс пока критерий не перестанет улучшаться.

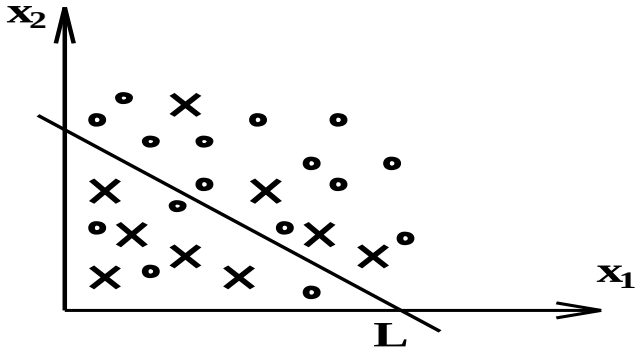
Алгоритмов много, потому что много критериев качества кластеризации. Простые критерии базируются на величине расстояния между кластерами. Они не учитывает «населенность» кластеров. Другие критерии основываются на вычислении средних расстояний между объектами внутри кластеров. Наиболее часто применяются критерии в виде отношений «населенности» кластеров к расстоянию между ними.

Функционалы качества и конкретные алгоритмы автоматической классификации достаточно полно и подробно рассмотрены в специальной литературе. У них различная трудоемкость, подчас они требуют ресурсов мощных компьютеров. Алгоритмы кластерного анализа входят в состав практически всех современных пакетов программ для статистической обработки многомерных данных.

Вероятностные СРО

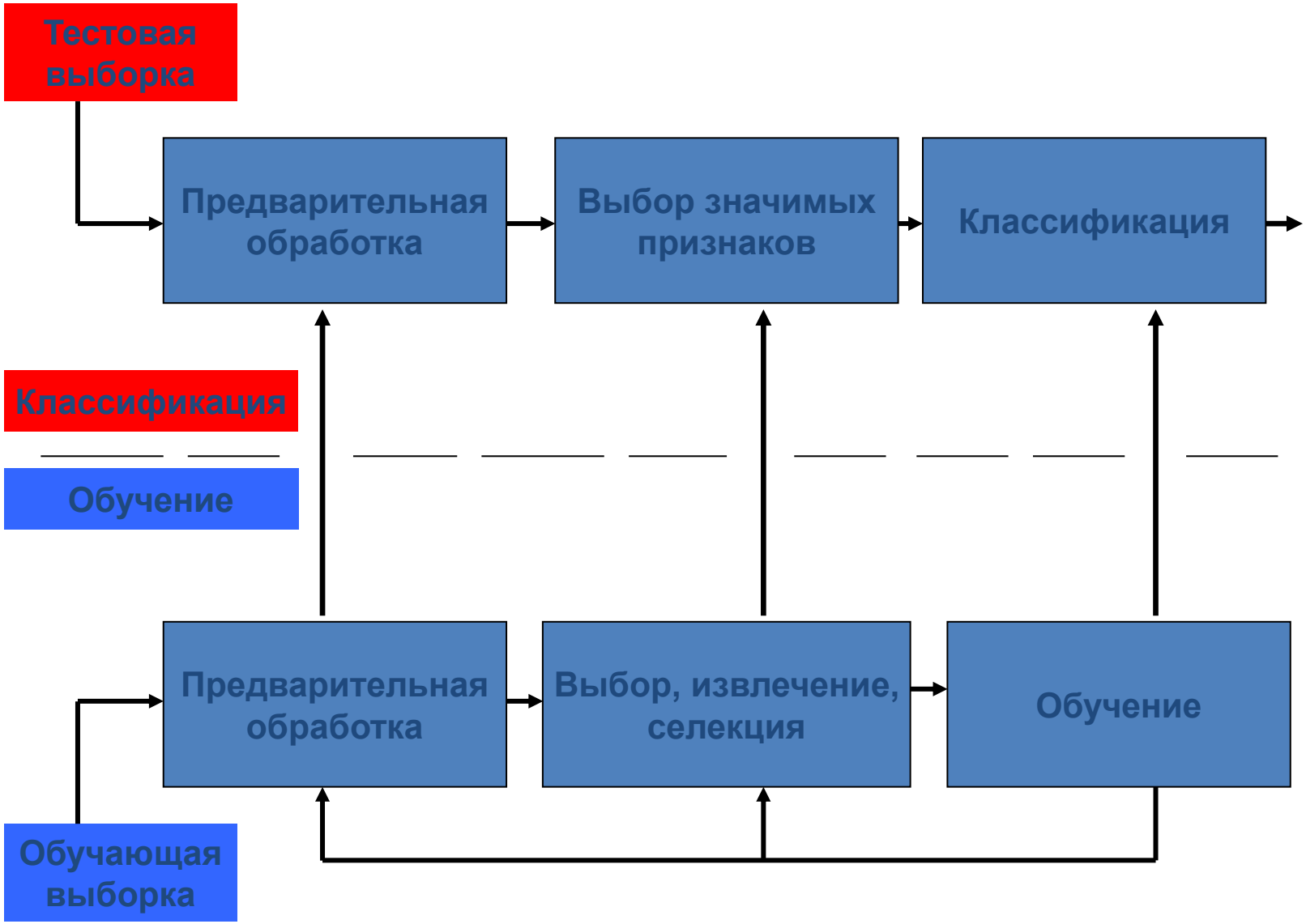
Детерминированный подход к распознаванию не позволяет понять такие важные понятия в теории распознавания образов, как обучение и *обобщение*. Обобщение заключается в умении распознавать образы, которые не встречались во время обучения. СРО, не способная к обобщению, не способна и обучаться.

Ограниченность детерминированных СРО можно преодолеть с позиции теории вероятности. В этом случае задача классификации образов заключается в минимизации вероятности ошибочного решения в заданном классе решающих функций.



На этом рисунке образы располагаются так, что нельзя построить классификатор, дающий безошибочное решение. Такая ситуация является нередкой для распознающих систем. Она может быть вызвана случайными помехами, отсутствием четких границ между классами или недостаточно полной информации об анализируемых объектах. Решающую функцию следует выбрать так, чтобы вероятность ошибочных решений была минимальной. Вероятность ошибочного решения Q можно оценить по обучающей выборке по следующей формуле: $Q = v/N$, где v - число образов в обучающей выборке из N объектов, которые классифицируются не правильно. Очевидно, что мы не можем оценить точно качество классификатора. Однако известно, что данная оценка сходится по вероятности к истинному значению при $N \to \infty$.

Модель вероятностной СРО



Как применять статистические методы распознавания?

Постановка задачи

Дана обучающая выборка (ОВ). Каждый образ в ОВ включает набор из *n* характеристических признаков и описывается *n*-мерным вектором. Необходимо построить классифицирующее правило.

Чего не хватает для полного счастья?

ОТВЕТ: функций распределения элементов ОВ в *n*-мерном пространстве. Рассмотрим варианты. Их может быть три:

Функции вероятностного распределения известны

Тип функций распределения известен. Неизвестны их параметры (математическое ожидание, среднеквадратичное отклонение)

Распределение неизвестно.

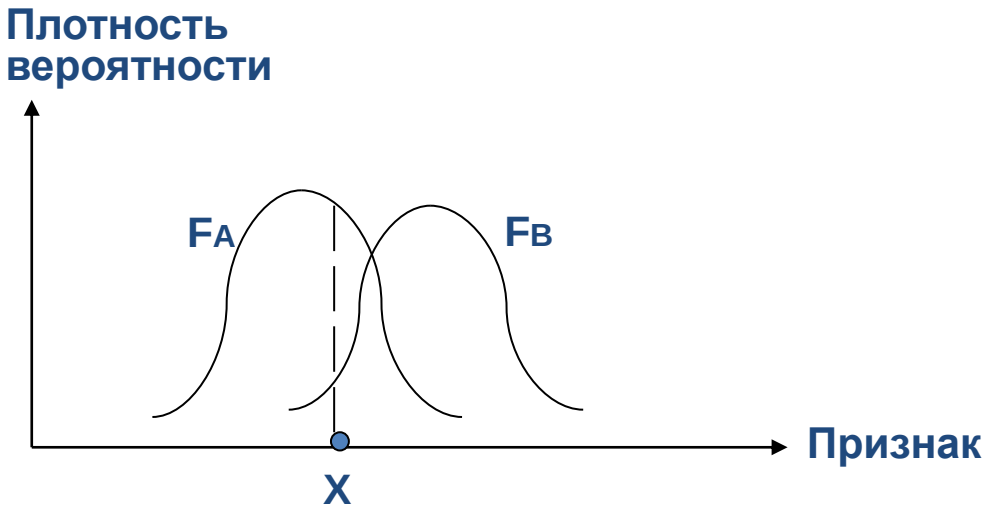
Функции вероятностного распределения известны

Необходимо просто применить правило Байеса.

Пример. Пусть даны два распределения **A** и **B**.

Значение X порождается одним их этих распределений.

Каким из них? – Интуиция подсказывает, что надо выбирать **A**, т.к. P(A/X) > P(B/X). Это же утверждает правило Байеса:



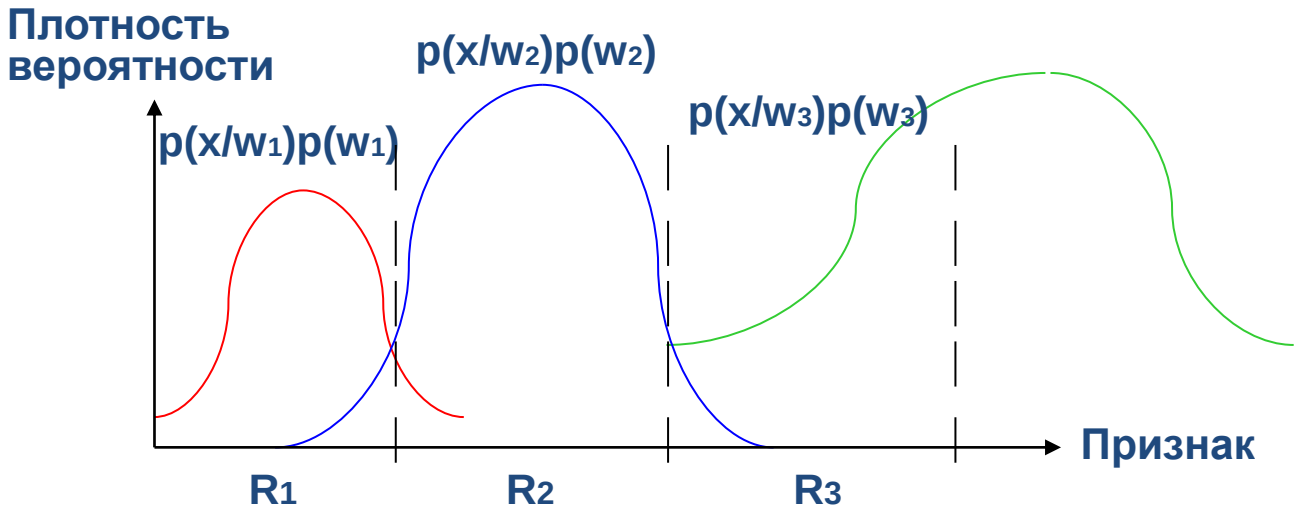
$$P(A/X) = P(X/A)*P(A)/(P(X/A)*P(A)+P(X/B)*P(B)) =$$
$$=F_A(X)*P_A/(F_A(X)*P_A+F_B(X)*P_B).$$

Надо выбирать A, если $F_A(X)*P_A > F_B(X)*P_B$.

Тип функций распределения известен, неизвестны их параметры

Необходимо вначале оценить параметры (м.о. и σ). Оценка может быть точечной или интервальной. Затем применить правило Байеса.

Пример. Пусть даны 3 интервальных распределения классов w1, w2, w3



Если, например известен нормальный закон распределения, то используя обучающую выборку можно рассчитать:

M.o.=(X1+X2+…+Xn)/n, **σ=√(1/(n-1))∑(Xi-м.о.)**2.**

Функция распределения неизвестна

Если функции распределения заранее неизвестны, то необходимо:

1. Выдвинуть **гипотезу** о распределении.

2. Построить распределение в соответствии с гипотезой.

3. По обучающей выборке одним из **известных статистических методов** **принять/отвергнуть гипотезу.**

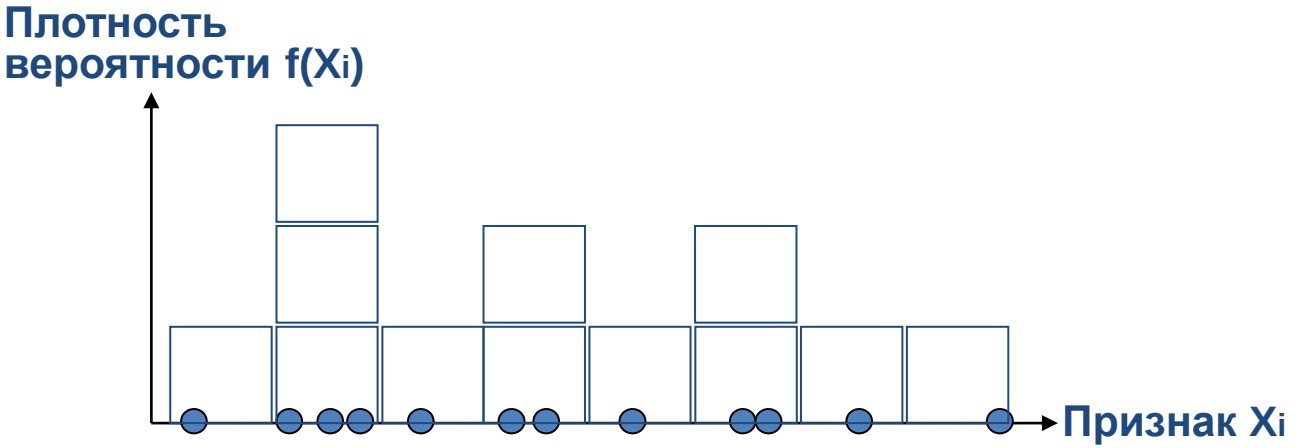
4. Если отвергли правильную гипотезу, то это **ошибка 1-го рода,** она должна быть **меньше α.**

5. Если приняли неверную гипотезу, то это **ошибка 2-го рода.** Например, если **T<t**, то принимаем гипотезу. Если **T≥t**, то отвергаем. Значение **t** определяется как функция от **α** и **n**.

6. Проверить гипотезу, например, по критерию χ^2. Например, проверяем гипотезу о ***D***-распределении для ОВ $X_1,X_2,\ldots,X_n$. Разбиваем область значений на m классов. Пусть m_j-число элементов ОВ в классе j, p_j-вероятность попадания в j-класс согласно ***D***. Обозначим через $m'_j=m_j*p_j$. Вычисляем функцию $T=\sum(m_j-m'_j)^2/m'$

Статистические методы проверки гипотезы

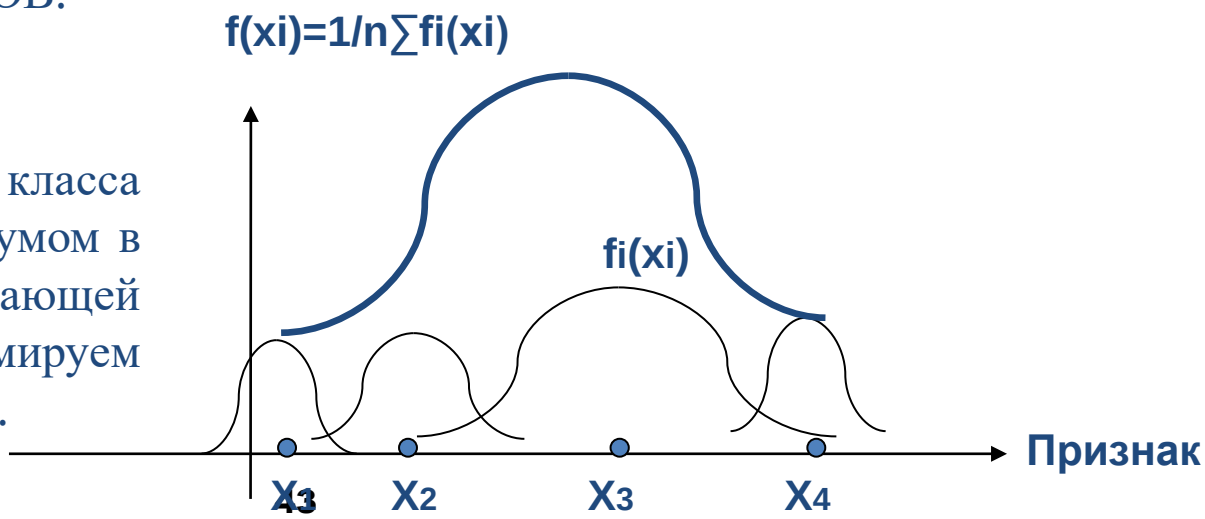
1. Метод гистограмм



Строится «в лоб» гипотетическая функция распределения путем разбиения n-мерного пространства признаков на клетки. Для каждой клетки определяется плотность распределения (в зависимости от числа точек в клетке. Недостаток метода: необходим большой объем ОВ.

2. Метод Парзена

Для каждой точки из класса строим функцию с максимумом в этой точке и быстро убывающей при удалении от нее. Суммируем эти функции для всех точек.



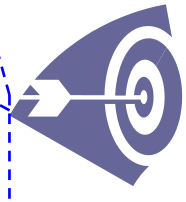
Machine Learning

Машинное обучение (англ. *Machine Learning*) — обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться. Машинное обучение рассматривает *обучение по прецедентам* или *индуктивное обучение*, основанное на выявлении закономерностей в эмпирических данных.

- Байесовский классификатор:
  - линейный дискриминант;
  - непараметрическое восстановление плотности;
  - разделение смеси вероятностных распределений;
  - метод потенциальных функций;
  - метод ближайших соседей.
- Байесовская сеть.

- Кластеризация:
  - иерархическая кластеризация.
- Нейронная сеть:
  - персептрон;
  - многослойный персептрон;
  - самоорганизующаяся сеть Кохонена;
  - гибридная сеть встречного распространения;
  - машина опорных векторов.

- Алгоритмическая композиция:
  - взвешенное голосование;
  - бустинг;
  - бэггинг;
  - метод комитетов;
  - смесь экспертов.
- Сокращение размерности:
  - селекция признаков;
  - метод главных компонент;
  - метод независимых компонент;
  - многомерное шкалирование.

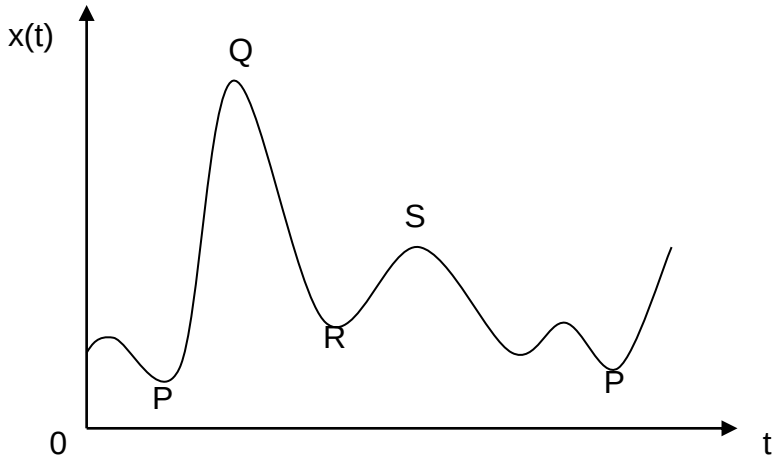


Общая характеристика структурных методов распознавания

Зачастую на практике информация о распознаваемых объектах содержится в записях сигналов. Традиционно для определения признаков используют разложения сигналов в ряды по ортогональным функциям (ряды Фурье, полиномы Эрмита, Лежандра, Чебышева и т.д.).

Возможно использование в качестве характерных признаков точек минимума, максимума и др. (например, электрокардиограммы).

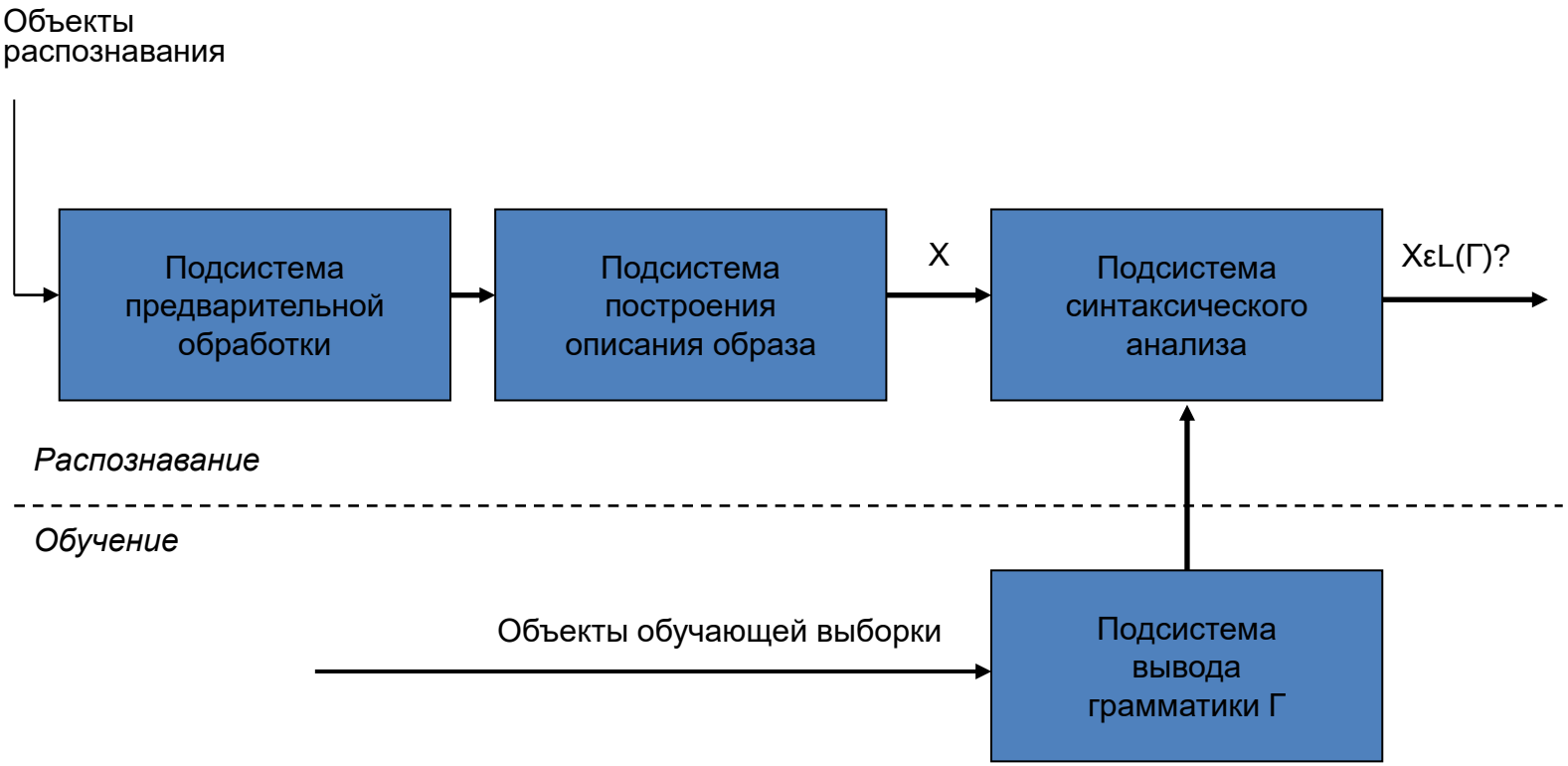
Методы распознавания, использующие признаки структурных элементов позволяют избежать не только трудоемкой процедуры разложения в ряды, но и потерь информации. Структурные методы основаны на **теории формальных языков Н.Хомского**.



Идея как формальных языков так и структурных методов распознавания состоит в построении сложного объекта в виде иерархической ***регулярной*** структуры более простых образов.

При этом используется известная из регулярных грамматик схема: задаются исходные объекты и порождаются новые по некоторым правилам индуктивного вывода (аналогия с синтаксисом ЕЯ).

Этапы структурного распознавания



На этапе *обучения* грамматику можно определить, используя априорные сведения об объектах обучающей выборки.

На этапе *распознавания* предварительная обработка заключается в кодировании, сжатии, фильтрации или восстановлении объекта. Затем объект представляется языковой структурой (цепочкой или графом): сегментация и выделение простейших элементов. Решение о правильности объекта вырабатывается синтаксическим анализатором по дереву грамматического разбора. Распознавание – это сопоставление с эталоном.

Основные понятия формальных грамматик

Грамматика включает ***терминальный*** и ***нетерминальный словари***, ***начальный символ*** и ***набор правил подстановки***.

Терминальный словарь – это набор исходных элементов, из которых строят цепочки, порождаемые грамматикой.

Нетерминальный словарь – это набор символов, которыми обозначаются классы исходных элементов, а также специальные вспомогательные символы.

Начальный символ – это выделенный нетерминальный символ, обозначающий класс объектов, для описания которых предназначена грамматика.

Правила подстановки – это выражения вида «$x \rightarrow y$» (заменить x на y), где x и y – цепочки с любыми терминальными или нетерминальными символами. Если имеется последовательность цепочек $x_0, x_1, \ldots, x_n$, в которой каждая последующая выводима из предыдущей, то она называется ***выводом*** x_n из x_0 в грамматике Γ.

Грамматика – это не алгоритм, поскольку порядок применения правил подстановки произволен!

Совокупность всех терминальных цепочек, выводимых из начального символа, называется ***языком, порожденным грамматикой*** Γ, и обозначается $L(\Gamma)$. Две разные грамматики могут порождать один и тот же язык.

Допустим, что имеем дело с 2 классами объектов Ω_1 и Ω_2, которые описываются конечным множеством признаков (словарь). Пусть существует грамматика Γ такая, что порождаемый ею язык состоит из предложений (объектов), принадлежащих исключительно классу Ω_1. Эту грамматику можно использовать для классификации объектов, т.к. неизвестный объект можно отнести к классу Ω_1, если он является грамматически правильным предложением языка $L(\Gamma)$. Иначе, объект относим к Ω_2.

Как реализовать процесс структурного распознавания?

Построить адекватное описание объектов распознавания

Выбрать грамматики

Реализовать процесс распознавания посредством процедур синтаксического анализа

Использовать обучение для вывода грамматик

Применить в рамках структурного подхода другие методы распознавания

Проблемы и перспективы развития методов распознавания

Для успешного достижения цели распознавания необходимо решить или обойти следующие проблемы:

1. Комбинаторный взрыв.
2. Время распознавания не должно сильно зависеть от размера обучающей выборки.
3. Корректное снижение размерности пространства признаков без существенной потери значимой информации.
4. Достижение высокой валидности (обоснованности) результатов.

Проблемы 1-2 имеют сходное происхождение и возникают при попытке прямого перебора вариантов кластеризации во многих методах распознавания.

Проблема 3 возникает при выявлении существенных и малозначимых признаков образа (эту проблему решает художник, когда переносит на двумерный холст изображение трехмерного пейзажа). Программа распознавания должна корректно понижать размерность до десятков признаков (это не так просто, признаки взаимосвязаны, их ценность меняется при отбрасывании других).

Проблема 4 - при разработке математической модели и программной реализации СРО (результаты часто не соответствуют представлениям эксперта, хотя соответствуют модели) такие модели обладают низкой внешней валидностью.

Сегодня многие перспективные разработки СРО ориентируются на мощные и дорогие вычислители. Эти разработки относят к категории фундаментальных работ, которые на практике пока мало кем могут быть реально использованы.

Задача логического распознавания образов

В реальных задачах, например, геологического и экономического прогнозирования, медицинской и технической диагностики имеет место следующая ситуация:

- число прецедентов (образов с известной классификацией) невелико,
- информации об их статистической природе недостаточно для обоснованного применения вероятностных моделей,
- сами прецеденты содержит разнородную или нечисловую информацию.

Однако известны логические связи между объектами и признаками.

Постановка задачи. Имеется обучающая выборка многомерных объектов, характеризующихся бинарными признаками. Найти *правила* классификации образов, каждое из которых содержит информацию не только об отдельных признаках, но и о различных их сочетаниях. Любое правило должно иметь две основные характеристики – *точность* и *полноту*.

Точность правила – это доля случаев, когда правило подтверждается, среди всех случаев его применения.

Полнота правила – это доля случаев, когда правило подтверждается, среди всех случаев, когда имеет место объясняемый исход.

Правила могут иметь какие угодно сочетания точности и полноты.

Исключение составляет лишь один случай: если точность равна нулю, то и полнота равна нулю (и наоборот).

Алгоритм «Кора»

Наиболее известным алгоритмом поиска и распознавания логических закономерностей в данных является алгоритм «Кора» (Бонгард, 1967).

Дано. Обучающая выборка (ОВ), состоящая из ***N*** образов, предварительно разделенная на 2 класса. Выделены ***M*** признаков, характеризующих каждый образ.

Найти. Путем многократного просмотра ОВ с использованием операции конъюнкции $\&_k X_{ij}$ (X_{ij} – булева переменная: равна 1, если значение ***j***-го признака превышает некоторый порог для ***i***-го образа), выделить из множества возможных комбинаций признаков непротиворечивые конъюнкции, покрывающие все множество примеров ОВ. *Непротиворечивой* считается конъюнкция, которая встречается некоторое количество раз только в одном классе и ни разу не встречается в другом. Длина конъюнкции ***k*** – наперед задана (обычно $k = 3 \div 5$).

Алгоритм. При генерации конъюнкций алгоритм действует по правилам:

1). Конъюнкции ***сортируются по продуктивности*** (по числу примеров из ОВ, для которых выделенная комбинация признаков равна 1 (не менее **Δ**).

2). Из списка ***исключаются подчиненные*** конъюнкции, ***оставляются «наилучшие»*** с точки зрения различения классов. Если есть эквивалентные, то оставляется более короткая.

3). Из списка ***исключаются конъюнкции-«предрассудки»***, не связанные с правилом классификации, но в силу ограниченности ОВ получившие хорошие оценки при обучении. Для поиска предрассудков выполняют ***bootstrep-процедуру***. Она многократно случайным образом разбивает ОВ на классы-тесты, по которым каждому признаку начисляются штрафные баллы за склонность к предрассудкам. Конъюнкции-«штрафники» из решающего правила исключаются

Общее число отобранных конъюнкций ограничиваем числом ***n.***

Прямая и обратная задачи логического распознавания

Задачи логического распознавания на практике, как правило, сводятся к решению ***системы булевых уравнений*** с одним или несколькими неизвестными.

Дано. Пусть множество образов разделено на классы $W_1, W_2, \ldots, W_m$.
Для описания этих классов используются признаки $X_1, X_2, \ldots, X_n$.
Предположим, что сведения о классах объектов и их признаках представлены в виде булевых соотношений. Пусть в результате экспериментов установлены данные о признаках X_i, присущих классам W_j. Эти данные выражены в виде булевых уравнений вида:

$$G(X1, X2, \ldots, Xn) = 1.$$

Прямая задача. Определить, какие выводы можно сделать относительно классов распознавания $W_1, W_2, \ldots, W_m$ на основе исходных сведений и выраженной в виде уравнения экспериментальной информации. То есть требуется найти неизвестную функцию $F(W_1, W_2, \ldots, W_m)$, которая удовлетворяет уравнению:

$$G(X_1, X_2, \ldots, Xn) \rightarrow F(W_1, W_2, \ldots, Wm)$$

Обратная задача. Найти неизвестную функцию $G(X1, X2, \ldots, Xn)$, из которой следует наблюдаемый факт $F(W1, W2, \ldots, Wm)$.

Методы решения прямой и обратной задач распознавания основаны на применении ***изображающих чисел БФ*** и ***сокращенного базиса***.

Изображающие числа и базис

Таблицу, которая представляет все возможные комбинации значений истинности набора переменных A, B, C,… называют ***базисом*** и обозначают ***b*[A, B, C,…]**.

Если значение «истина» обозначить 1, а значение «ложь» - 0, то для трех элементов **A, B, C** базис имеет вид:

	0 1 2 3	**4 5 6 7**
#A =	0 1 0 1	0 1 0 1
#B =	0 0 1 1	0 0 1 1
#C =	0 0 0 0	1 1 1 1

Строки базиса называются ***изображающими числами*** соответствующих элементов.

Используя базис, можно явно перечислить все значения истинности БФ для всех комбинаций переменных.

Для изображающих чисел справедливы следующие операции:

- дизъюнкция **#(A+B) = #A+#B**
- конъюнкция **#(A&B) = (#A)&(#B)**
- отрицание **#¬A = ¬(#A)**

Используя эти операции, можно найти изображающее число любой БФ.

Пример. **#(A&(B+C))**=(01010101)(00110011+00001111)=(01010101)(00111111)= 00010101.

Используя изображающие числа можно доказать любую теорему или закон алгебры логики.

Пример. Доказать, что **A&(B+C) = A&B + A&C**.

Действительно, изображающее число **#(A&(B+C))** = 00010101,

#(A&B+A&C) = (01010101)(00110011)+(01010101)(00001111) = 00010001 + 00000101 = =00010101, т.е. изображающие числа тождественны.

Решение булевых уравнений с одним неизвестным

Задача. Пусть в процессе решения задачи логического распознавания на основе анализа трех объектов были установлены следующие логические зависимости между тремя характеризующими эти объекты признаками **A, B, C**:

$$X(A+B)=ABC, \qquad (*)$$

где **X** – некоторая булева функция, которую необходимо найти и которая зависит от **A,B,C**. Причем функция **X** должна быть такой, чтобы при её подстановке в исходное уравнение, оно превращалось в тавтологию.

Решение. Чтобы решить задачу найдем изображающие числа для элементов, входящих в булево уравнение.

Изображающее **X** число должно быть таким, чтобы (#**X**)*01110111= 0000 0001.

Отсюда #**X**=×000 ×001 и уравнение (*) имеет 4 решения, соответствующих изображающим числам: (0000 0001), (1000 0001), (0000 1001) и (1000 1001), т.е.

$X_1=ABC$, $X_2=ABC+\neg A\neg B\neg C$,

$X_3=C(AB+\neg A\neg B)$, $X_4=\neg A\neg B+ABC$.

Аналогично решаются ***системы булевых уравнений***.

C	B	A	#ABC	#(A+B)	#X
0	0	0	**0**	**0**	**×**
0	0	1	**0**	**1**	**0**
0	1	0	**0**	**1**	**0**
0	1	1	**0**	**1**	**0**
1	0	0	**0**	**0**	**×**
1	0	1	**0**	**1**	**0**
1	1	0	**0**	**1**	**0**
1	1	1	**1**	**1**	**1**

Постановка задачи о выработке научно-технической политики фирмы

Эксперты определили пять основных целей фирмы:

1) Повышение эффективности производства ($\mathbf{A_1}$);
2) Развитие производственных технологий ($\mathbf{A_2}$);
3) Достижение обоснованного уровня заработной платы работников ($\mathbf{A_3}$);
4) Рост научно-технического потенциала фирмы ($\mathbf{A_4}$);
5) Сохранение и оздоровление окружающей среды ($\mathbf{A_5}$).

Все эти цели не являются независимыми. Связи между целями эксперты выразили с помощью 4 логических уравнений:

$$\mathbf{A_5 \rightarrow A_2 A_4 \equiv \neg A_5 + A_2 A_4 = 1} \qquad (1)$$

т.е. сохранение и оздоровление окружающей среды невозможно без развития производственных технологий и роста н/т потенциала фирмы.

$$\mathbf{\neg A_1 \neg A_2 \neg A_4 \rightarrow \neg A_3 \equiv A_1 + A_2 + A_4 + \neg A_3 = 1} \qquad (2)$$

т.е. достижение обоснованного уровня з/п работников невозможно без повышения эффективности производства, развития производственных технологий и роста научно-технического потенциала фирмы.

$$\mathbf{A_1 \rightarrow A_2 \equiv \neg A_1 + A_2 = 1} \qquad (3)$$

т.е. повышение эффективности производства невозможно без развития производственных технологий.

$$\mathbf{A_2 \rightarrow A_4 \equiv \neg A_2 + A_4 = 1} \qquad (4)$$

т.е. развитие производственных технологий невозможно без роста научно-технического потенциала фирмы.

Решение задачи о выработке научно-технической политики фирмы

Перемножив уравнения (1-4) и упростив выражение получим:

$$A_2A_4 + \neg A_1A_4\neg A_5 + \neg A_1\neg A_2\neg A_3\neg A_5 = 1$$

Отсюда получаем базис:

#A1 =	×	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
#A2 =	**1**	×	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
#A3 =	×	×	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
#A4 =	**1**	**1**	×	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**
#A5 =	×	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Из 32 возможных комбинаций исходов целей допустимыми (совместными по связям) являются только 11. Например, $A_1A_2\neg A_3A_4\neg A_5$. означает совместимость целей, связанных с повышением эффективности производства (A_1), с развитием производственных технологий (A_2) и ростом научно-технического потенциала фирмы (A_4).

Далее, по аналогичной методике можно приступать к разработке мероприятий, привязанных к соответствующим целям, анализу и распознаванию причинно-следственных связей и т.п.

Благодарю за внимание !

Contact Information:

E-mail: srodzin@sfedu.ru

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Кафедра МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ И ПРИМЕНЕНИЯ ЭВМ

Дисциплина:

Машинное обучение и биоинспирированная оптимизация

ЛЕКЦИЯ 3

Машинное обучение с учителем и без учителя. Обучение методами корреляционного и регрессионного анализа, нейросети и глубокое обучение, деревья решений

Родзин Сергей Иванович,
профессор ИКТИБ ЮФУ,
srodzin@sfedu.ru

Анализ Данных

Анализ данных - общий термин, используемый для описания любого процесса извлечения полезной информации из данных. Типы анализа данных включают визуализацию, сводную статистику, корреляционный анализ и моделирование с использованием машинного обучения.

Машинное обучение

Машинное обучение (Machine Learning) - область ИИ, которая фокусируется на разработке и оценке алгоритмов, способных выявлять полезные закономерности в наборах данных. Алгоритм машинного обучения принимает на вход набор данных и возвращает модель, которая кодирует закономерности, выявленные алгоритмом.

История Анализа Данных

В 17–18 вв. работы **Паскаля, Бернулли, Байеса, Лапласа** и **Гаусса** заложили основы теории вероятностей и математической статистики, начали использовать распределение вероятностей как инструментарий для анализа данных.

В 1780-1820 гг. **Плейфер** изобрел статистические графики и диаграмму для временных рядов, гистограмму и круговую диаграмму для наглядного изображения долей. Разнообразие видов графического отображения данных растет, и сегодня продолжаются разработки по визуализации многомерных данных. Например, алгоритм стохастического вложения соседей сокращает многомерные данных до двух или трех измерений, облегчая их визуализацию.

В 20 в. **Пирсон** разработал современные методы проверки гипотез, а **Фишер** – статистические методы для оценки максимального правдоподобия статистических гипотез, позволяющие делать выводы на основе относительной вероятности событий.

В 1943 г. **Мак-Каллок** и **Питтс** предложили математическую модель нейросети. В 1948 г. **Шеннон** основал теорию информации. В 1951 г. **Фикс** и **Ходжес** предложили модель дискриминантного анализа (теория распознавания образов), ставшую основой современных алгоритмов ближайших соседей.

В 1956 г. с появлением идей ИИ, машинного обучения и программной инженерии появилась возможность обучать компьютер на основе данных. Сейчас в число наиболее важных разработок входят нейросети глубокого обучения. Они произвели переворот в таких областях, как машинное зрение и обработка естественного языка.

История Анализа Данных *(продолжение)*

Начало 1970-х гг. ознаменовало приход современной технологии с реляционной моделью данных **Кодда** и последующий взрывной рост генерации данных и их хранения, который в 1990-х гг. привел к развитию хранилищ данных, а позднее – к возникновению феномена больших данных (***Big Data***).

В 1989 г. **Шапиро** ввел термин – *специалист по обнаружению знаний в базах данных* (**KDD**), навыки которого можно представить следующей схемой:.

Где сейчас используется Анализ Данных ?

Опишем три тематических кейса, которые иллюстрируют влияние науки о данных **на компании**, **на правительства**, и **на профессиональные спортивные клубы**.

Продажи и маркетинг. Компания имеют доступ к большим наборам данных о предпочтениях своих покупателей, собирая их через торговые точки, отслеживая поведение клиентов в интернет-магазине и анализируя комментарии о компании и ее продуктах в социальных сетях. Появились рекомендательные системы, которые собирают и используют данные, а затем предлагают вам варианты следующих просмотров или покупок.

Использование науки о данных государственными структурами. Наука о данных лежит в основе проектов «Цифровизация экономики России» и «Точная медицина».

Наука о данных в профессиональном спорте. Фильм 2011 г. «Человек, который изменил все» с участием Б. Питта продемонстрировал роль науки о данных в спорте. В фильме рассказана реальная история о том, как бейсбольный клуб «Окленд Атлетикс» использовал данные для улучшения отбора игроков. Ключевой вопрос - выявление информативных атрибутов. Зная их легко создавать модели, управляемые данными..

Сфера приложения науки о данных не имеет значения: важны только правильные данные и четкая формулировка проблемы. Графические процессоры сейчас адаптированы под глубокое обучение. Facebook использует глубокое обучение для распознавания лиц и анализа текста, Apple, Amazon, Microsoft, Samsung и Yandex - для распознавания речи.

Что такое набор данных и его представление ?

Набор данных (Dataset) - Совокупность данных, относящихся к набору объектов, каждый из которых описан в терминах множества *атрибутов*. В своей основной форме набор данных организован в виде матрицы ***n*** × ***m***, где ***n*** – количество объектов (строк), а ***m*** – количество атрибутов (столбцов).

➡ **Упорядоченные наборы данных**

	Дата	Количество	Сумма
2	01.01.2007	4	283,31
3	01.01.2007	1	173,32
4	01.01.2007	1	72,48
5	02.01.2007	12	405,76
6	02.01.2007	6	303,13
7	03.01.2007	6	521,16
8	03.01.2007	3	156,96

➡ **Неупорядоченные наборы данных**

1	Оператор	Машина	Дефект
2	Иванов	5	Сбой нарезки
3	Сидоров	3	Неточность выреза
4	Иванов	1	Дефект балансира
5	Петров	5	Неточность выреза

➡ **Транзакционные данные**

Одна транзакция

	Код транзакции	Товар
2	10200	Йогурт "Чудо"
3	10200	Сахарный песок
4	10201	Батон "Рязански
5	10201	Сок "Добрый"

Типы атрибутов данных

Существуют разные типы атрибутов: числовые и нечисловые (категориальные).

Для каждого из них подходят разные виды анализа.

Для представления нечисловых атрибутов используются **качественные** шкалы: *номинальная* и *порядковая*.

Для представления числовых атрибутов используются **количественные** шкалы: *интервальная* и *отношений*.

Номинальная шкала

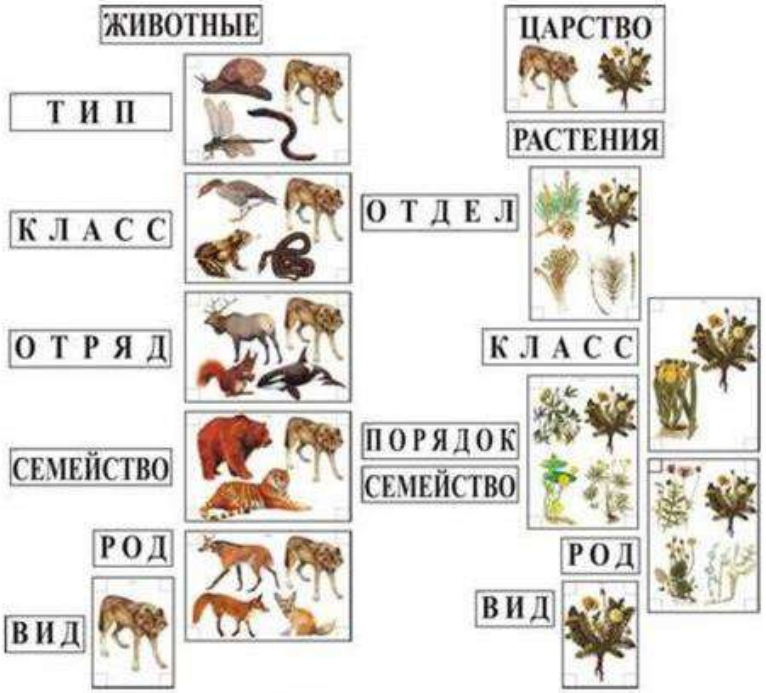
Номинальные атрибуты принимают значения, которые являются именами для категорий, классов или обстоятельств. Примеры номинальных атрибутов: **семейное положение** (холост, женат, разведен) или **тип пива** (эль, светлый эль, пильзнер, портер, стаут и т.д.).

Бинарный атрибут – это особый случай номинального атрибута, у которого набор возможных значений ограничен только двумя. Примером может служить бинарный атрибут «спам», который описывает, является электронная почта спамом (да) или не является (нет). К номинальным атрибутам не могут быть применены упорядочивание или арифметические операции.

Шкала наименований используется только с целью отличить один объект от другого.

При обработке экспериментальных данных, зафиксированных в номинальной шкале, непосредственно с самими данными можно выполнять только операцию проверки их совпадения или несовпадения.

Примеры номинальной шкалы



Характеризуется только отношением эквивалентности, отсутствует понятие больше, меньше, отсутствуют единицы измерения и нулевое значение.

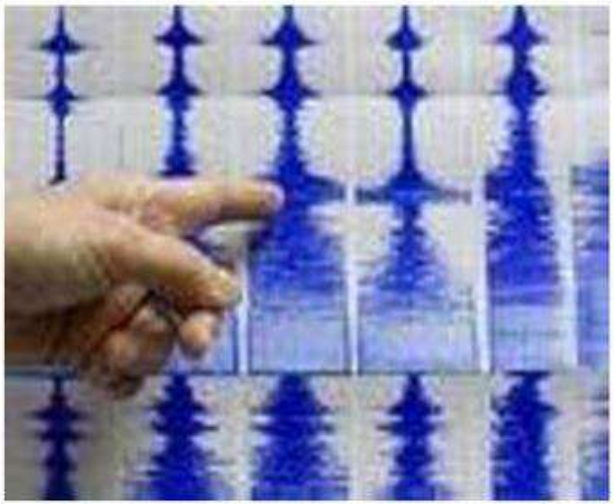
Порядковая шкала

Порядковые атрибуты аналогичны номинальным, но с той разницей, что можно ранжировать значения переменных. Например, атрибут, описывающий ответ на вопрос анкетирования, может принимать значения из области определения: «*очень не нравится*», «*не нравится*», «*нейтрально*», «*нравится*» и «*очень нравится*».

Особенность порядковых атрибутов - отсутствие понятия равного расстояния между этими значениями. Например, когнитивное расстояние между неприязнью и нейтральным отношением может быть отличным от расстояния между симпатией и сильной симпатией. В результате неуместно применять арифметические операции к порядковым атрибутам.

Граница между номинальными и порядковыми данными не всегда четкая. Для примера возьмем атрибут, который описывает погоду и может принимать значения «*солнечно*», «*дождливо*», «*пасмурно*». Один человек может сказать, что этот атрибут номинальный, значения которого не упорядочены, в то время как другой будет утверждать, что атрибут является порядковым, при этом рассматривая облачность как промежуточное значение между «*солнечно*» и «*дождливо*».

Пример-1 шкалы порядка



В данной шкале невозможно ввести единицу измерения, можно говорить лишь о том, что больше или меньше, хуже или лучше, но невозможно дать количественную оценку во сколько раз больше или меньше.

Пример-2 шкалы порядка

Шкала оценок знаний

Российские оценки	ECTS	Смысловое содержание оценки
5	A	«отлично»
4	B	«очень хорошо»
	C	«хорошо»
3	D	«удовлетворительно»
	E	«посредственно»
2	FX	«неудовлетворительно» (с правом пересдать)
*	F	«неудовлетворительно» (без права пересдать»

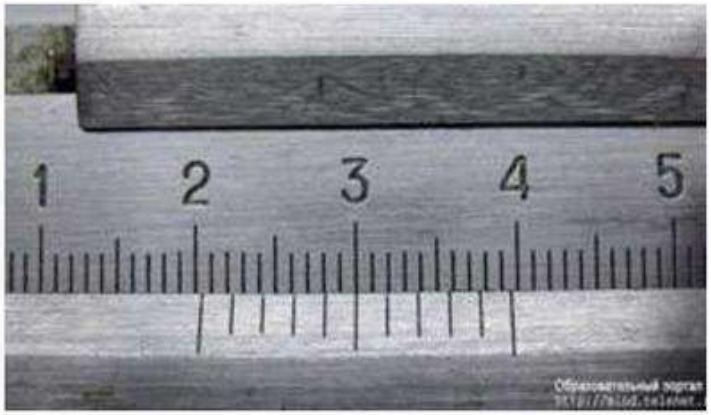
ECTS (англ. European Credit Transfer and Accumulation System) — Европейская система перевода и накопления баллов.

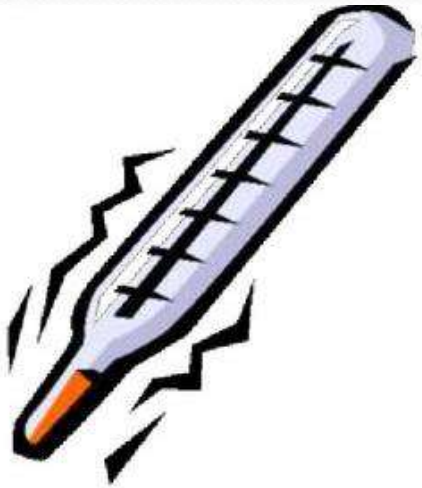
Шкала интервалов

Интервальные атрибуты измеряются по шкале с фиксированными, но произвольными единицами измерений и произвольным началом отсчета. Примерами интервальных атрибутов могут быть измерения даты и времени. К ним применяют упорядочивание и вычитание. Умножение, деление и прочие операции в этом случае не подходят.

- строится в тех случаях, когда имеется возможность не только определить количество свойства в объектах, но и установить в них равные разности значений измеряемого признака. При интервальных измерениях устанавливается единица измерения (килограмм, метр, градус). Нулевая точка в интервальных шкалах выбирается произвольно и не указывает на полное отсутствие измеряемого признака.

Примеры шкалы интервалов





Шкала состоит из одинаковых интервалов, имеет условную (принятую по соглашению) единицу измерения и произвольно выбранное начало отсчета - нуль.

Шкала отношений

Шкала отношений аналогична шкале интервалов с единственным отличием: ее нулевая точка – истинный нуль. Он указывает на то, что количество, которое могло бы быть измерено, отсутствует.

Температура – прекрасный пример для понимания разницы между шкалой интервалов и шкалой отношений. По шкале Цельсия и по шкале Фаренгейта температура измеряется интервально, поскольку значение 0 на любой из этих шкал не указывает на отсутствие тепла. Мы можем вычислить разницу между температурами на этих шкалах и сравнить различия, но мы не можем сказать, что 20 °C – это в два раза теплее, чем 10 °C.

В отличие от этого, измерение температуры в кельвинах ведется по шкале отношений, поскольку 0 K (абсолютный нуль) – это температура, при которой прекращается всякое тепловое движение. Другие распространенные примеры измерений по шкале отношений: количество денег, вес, рост и экзаменационные отметки (шкала 0–100).

Пример шкалы отношений

Характеристика шкал разного типа

Шкала	Характеристика шкалы				Примеры
	Описание объекта измерения	Ранги	Равенство интервалов измерения	Наличие о-й точки отсчета	
Номинальная	*				ИНН, пол, класс, цвет, автономера, почтовые индексы
Порядка	*	*			Твердость минералов, сила шторма, военные звания, полярная шкала
Интервальная	*	*	*		Температура, высота над уровнем моря, летоисчисление, стаж работы
Отношений	*	*	*	*	Рост, масса, число студентов на лекции, мощность (Вт, л.с.), стоимость (руб, $)

Классификация данных

Различают **структурированные** и **неструктурированные** данные.

Структурированными называются данные, которые могут храниться в таблице, где каждый объект имеет одинаковую структуру, т.е. одинаковый набор атрибутов.

Пример: демографические данные населения, где каждая строка в таблице описывает одного человека и состоит из одного и того же набора атрибутов (имя, возраст, дата рождения, адрес, пол, образование, статус занятости и т.д.).

Структурированные данные можно легко хранить, систематизировать, искать, переупорядочивать и объединять с другими структурированными данными. К ним легко применять методы анализа данных, поскольку по определению они уже находятся в формате, который подходит для интеграции в таблицу (набор данных).

Название	Издательство	Страниц	Издана	Цена
Практическое руководство по SQL	Диалектика	320	1997	130
Введение в системы баз данных	Диалектика	980	1998	96
Эффективная работа с СУБД	Питер пресс	704	1997	89

Классификация данных

В **неструктурированных** данных каждый объект в наборе может иметь собственную уникальную внутреннюю структуру. Представьте себе набор веб-страниц, у каждой из которых есть структура, отличная от других.

Неструктурированные данные встречаются гораздо чаще, чем структурированные.

Пример. Текстовые массивы (электронные письма, твиты, СМС, посты, романы и т.д.), а также коллекции звуковых, графических и видеофайлов.

Можно извлекать структурированные данные из неструктурированных, используя методы ИИ, цифровую обработку сигналов или компьютерное зрение. Однако это дорогостоящий и трудоемкий процесс.

Неструктурированные данные

Счет № 16493, Сергеев Петр Михайлович, дата рождения 1 января 1936г.; Сч. № 16593, Петрова Анна Владимировна, дата рожд. 15 марта 1955г.; № сч. 16693, Анохин Андрей Борисович, д/р 14/04/76.

Структурированные данные

№счета	Фамилия	Имя	Отчество	Дата рождения
16493	Сергеев	Петр	Михайлович	01/01/36
16593	Петрова	Анна	Владимировна	15/03/55
16693	Анохин	Андрей	Борисович	14/04/76

Классификация данных

Производные данные описывают такие данные, которые получены из других данных: средняя зарплата в компании или разница температур в комнате за период времени.

Пример.
Информативным производным атрибутом является **индекс массы тела (ИМТ)** – это соотношение массы тела и роста человека.

Он помогает нам определить людей в группе населения, которые подвержены риску ожирения. Некоторые алгоритмы способны извлекать производные атрибуты из необработанных данных.

$$\text{ИМТ} = \frac{\text{Масса тела (кг)}}{\text{Рост (м}^2\text{)}} \quad \frac{\text{кг}}{\text{м}^2}$$

Индекс массы тела	Соответствие между массой тела человека и его ростом
16 и менее	Выраженный дефицит массы тела
16 – 18,5	Недостаточная(дефицит) масса тела
18,5 – 24,99	Норма
25 – 30	Избыточная масса тела (предожирение)
30 – 35	Ожирение первой степени
35 – 40	Ожирение второй степени
40 и более	Ожирение третьей степени (морбидное)

Классификация данных

Собранные данные получают посредством прямого измерения или наблюдения. Например, основная цель опросов или экспериментов состоит в сборе конкретных данных по конкретной теме.

Выхлопные данные - побочный продукт процесса (подобно выхлопным газам). Пример. Основная цель социальных сетей – дать пользователям возможность общаться друг с другом. Однако для каждого опубликованного изображения, поста, ретвита или лайка создается ряд выхлопных данных: кто поделился, кто просмотрел, сколько людей просматривали/поставили лайк/ретвитнули и т.д.

Классификация данных

Метаданные - данные, описывающие другие данные.

Пример. Э.Сноуден опубликовал документы АНБ, касающиеся программы тотальной слежки PRISM. Он сообщил, что агентство собирало большое количество метаданных о телефонных звонках людей. АНБ фактически не записывало их содержание, но собирало данные о звонках, например когда был сделан звонок, кому, как долго длился и т. д. Исследования показали, что метаданные телефонного звонка могут раскрыть большой объем личной информации.

Данные содержатся в двух таблицах. Таблица Т1, приведена ниже, таблица Т2 на этом слайде не показана.

Т1:

ФИО	Адрес	Телефон
Иванов И.И.	Ставропольская 149	2-111-111
Петров П.П.	Ставропольская 153	2-222-222

Часть метаданных может быть записаны в двух таблицах. М1 содержит перечень таблиц, М2 – перечень столбцов

М1:

Ном_таб	Имя_таб
1	Т1
2	Т2

М2:

Ном_таб	Ном_столб	Имя_столб
1	1	ФИО
1	2	Адрес
1	3	Телефон
2	1	Назв_отдела

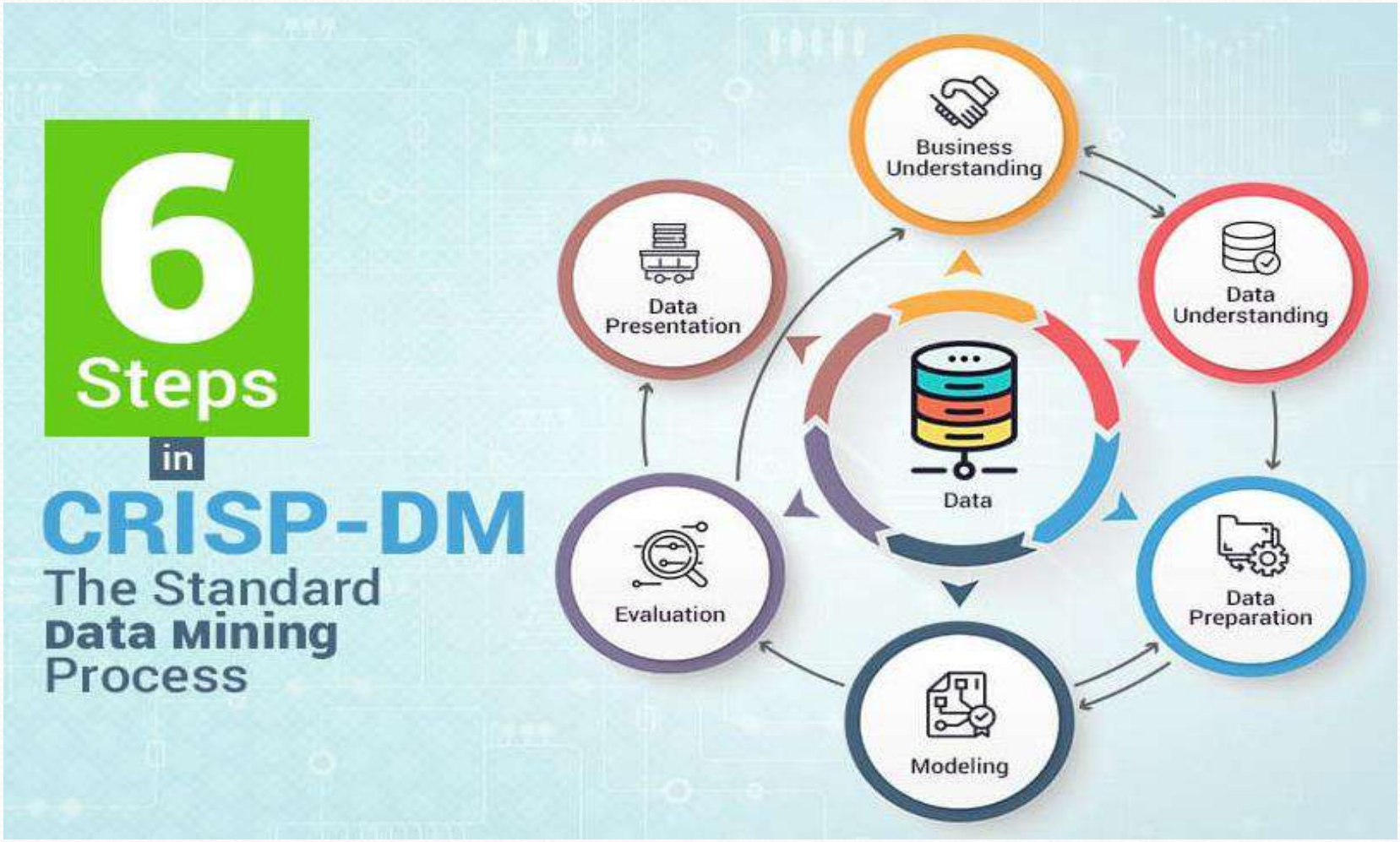
Данные-Информация-Знания-Мудрость

Цель науки о данных – использовать их, чтобы получить **знания**. Можно утверждать следующее:

- данные создаются с помощью **измерений**;
- **информация** – это данные, которые были обработаны и структурированы таким образом, что стали значимы для людей;
- **знание** – это информация, которая была понята так, что появилась возможность действовать в соответствии с ней;
- **мудрость** – это умение найти надлежащее применение знаниям.

Стандарт процесса анализа данных CRISP-DM

Жизненный цикл CRISP-DM состоит из шести этапов: понимание целей → начальное изучение данных → подготовка данных → моделирование → оценка → внедрение:



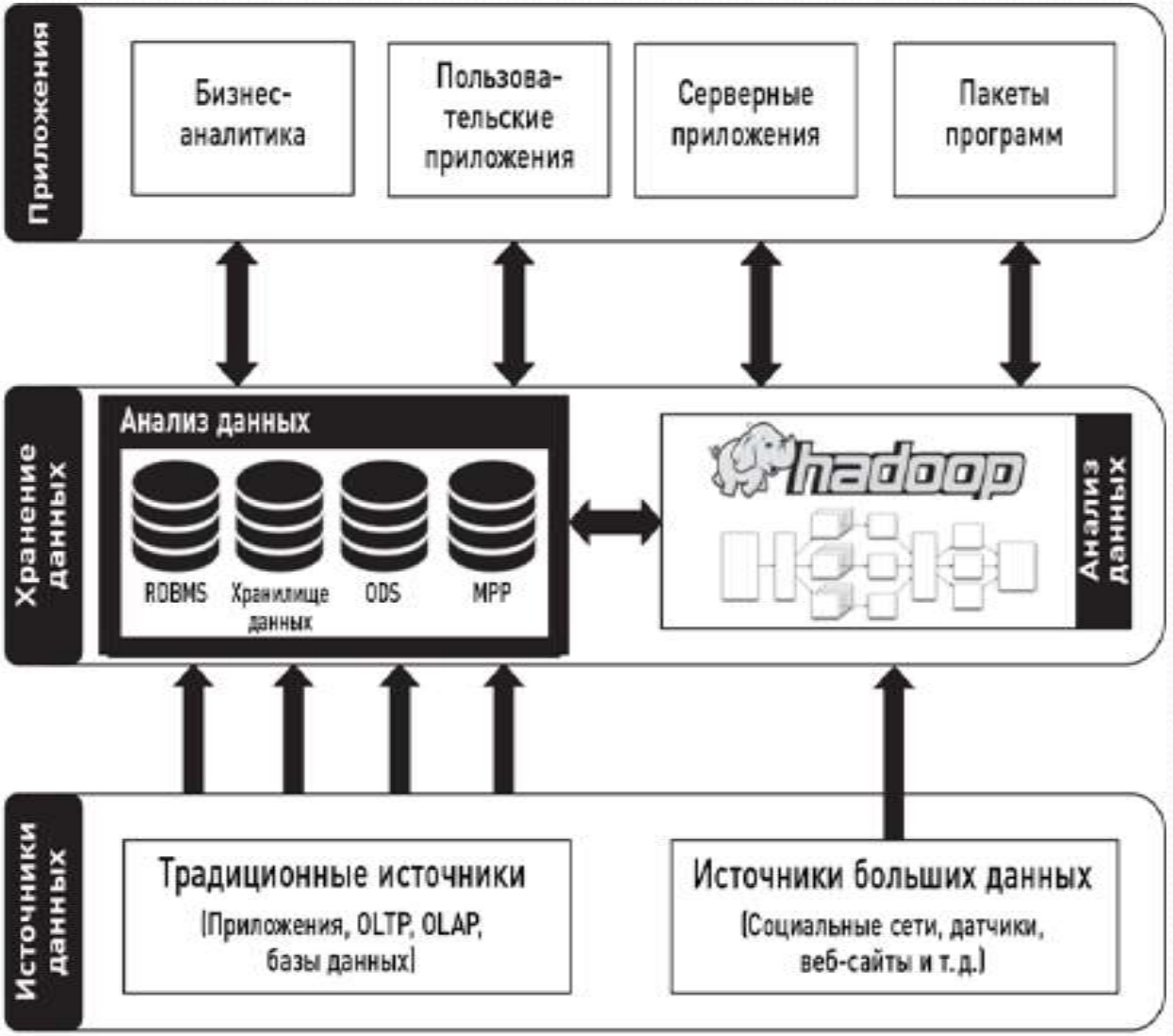
Распределение времени между задачами анализа данных

- сбор данных – **19 %**;
- очистка и организация данных – **60 %**;
- построение обучающих моделей – 3 %;
- анализ данных для выявления закономерностей – 9 %;
- уточнение алгоритмов – 4 %;
- другие задачи – 5 %.

P.S. *Как бы ни был хорош ваш анализ данных, он не найдет полезных закономерностей в неправильных данных.*

Архитектура экосистемы науки о данных

Архитектура экосистемы науки о данных содержит инструменты и узлы от поставщиков программного обеспечения, которые обрабатывают данные в разных форматах:



RDBMS — система управления реляционными базами данных,
ODS — оперативное хранилище данных,
MPP — массивно-параллельная архитектура.

Перемещение алгоритмов в данные

Традиционный подход к анализу данных включает их извлечение из БД, интеграцию, очистку, размещение, построение прогнозной модели, а затем загрузку результатов анализа в БД, чтобы их можно было использовать в виде отчетов и т.д.:

Большая часть процесса обработки данных протекает на отдельном сервере, вне баз и хранилища. При этом **значительное количество времени** может быть **затрачено** только **на перемещение данных из БД** и **загрузку в нее результатов**.

Перемещение алгоритмов в данные

Поставщики БД инвестируют в развитие масштабируемости, производительности, безопасности и функциональности своих продуктов. Современные базы данных намного более продвинуты, чем традиционные РБД. Поставщики БД (*Oracle, Microsoft, IBM, EnterpriseDB*) интегрировали в свои базы разнообразные алгоритмы машинного обучения, которые можно запускать на **SQL**. Кратко реализованная идея размещения алгоритмов машинного обучения в БД: «**Переместить алгоритмы в данные, вместо того чтобы перемещать данные в алгоритмы**»

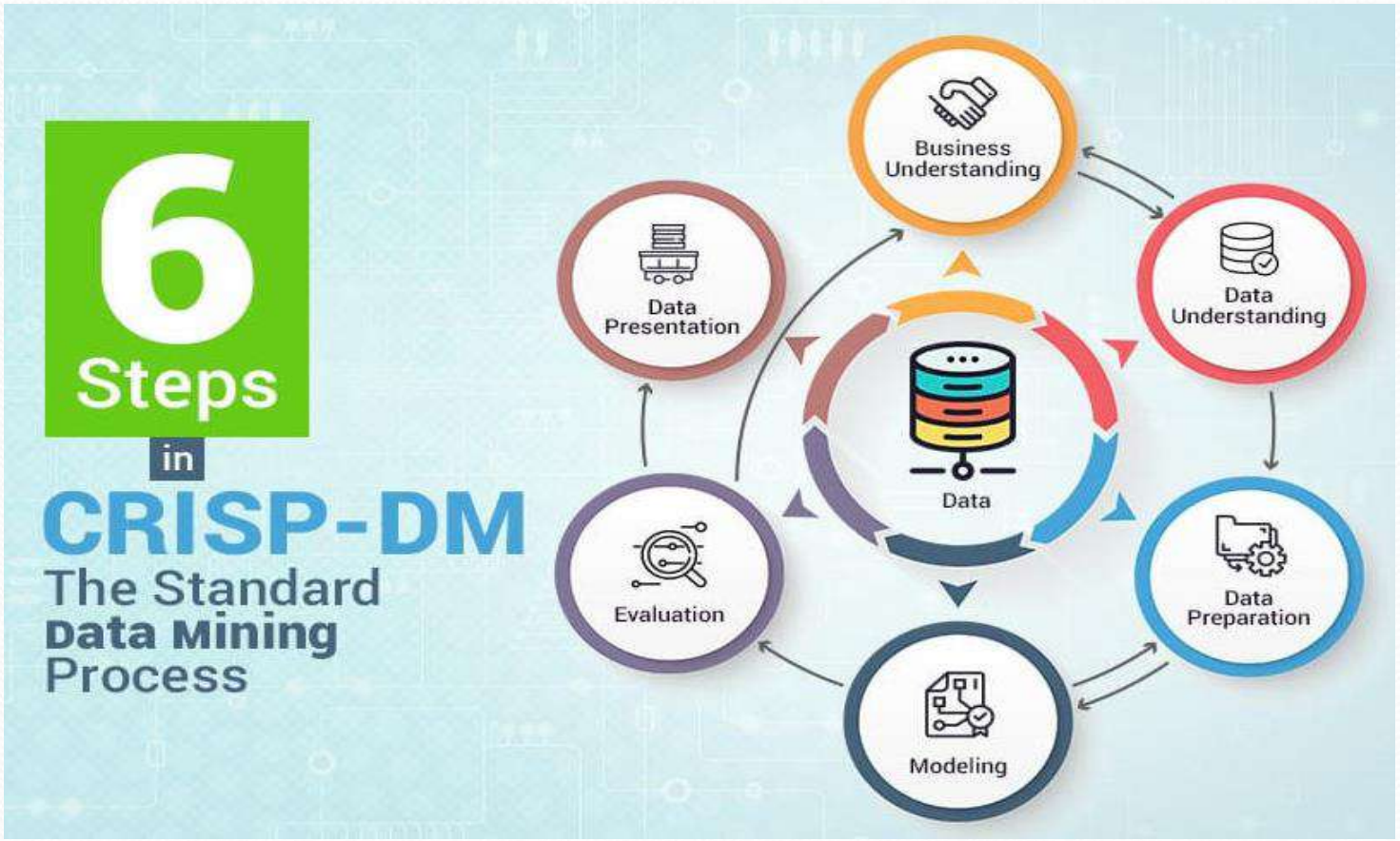
Чтобы современная база данных могла справляться с объемами до нескольких *петабайт*, требуется новая инфраструктура.

Hadoop – это платформа с открытым исходным кодом разработана *Apache Software Foundation*. Она эффективнее, чем традиционный подход. Сейчас появился широкий ассортимент продуктов для обработки и анализа данных на платформе **Hadoop**. Идея «переместить алгоритмы в данные, вместо того чтобы перемещать данные в алгоритмы», – реализована в **Hadoop**.

В Hadoop данные делятся на разделы, которые распределяются по узлам кластера. Алгоритмы обрабатывают данные в каждом из кластеров, что быстро поскольку несколько кластеров анализируются одновременно. Извлечение данных из БД не требуется. Данные анализируются там, где они хранятся. Эта же идея реализована в Google и Amazon, где ПО анализа данных (*Spark*), разворачивается на распределенных вычислителях, позволяя анализировать данные там, где они находятся.

Машинное обучение

Это область ИИ, которая разрабатывает алгоритмы для выявления компьютером закономерностей в данных. Алгоритмы и методы машинного обучения в основном применяются на **этапе моделирования в CRISP-DM** (Жизненный цикл CRISP-DM состоит из шести этапов: понимание целей → начальное изучение данных → подготовка данных → моделирование → оценка → внедрение):



Машинное обучение

Процесс машинного обучения представляет собой два последовательных этапа:

(1) Алгоритм машинного обучения применяется к набору данных для выявления в нем закономерностей. Сами закономерности могут быть представлены разными способами. Сейчас наиболее популярными из них являются **деревья решений, регрессионные модели** и **нейросети**. Эти представления известны как модели, поэтому этап жизненного цикла **CRISP-DM** называется этапом **моделирования**. Алгоритмы машинного обучения создают модели из данных, но каждый из них разработан для создания моделей, использующих определенный тип представления.

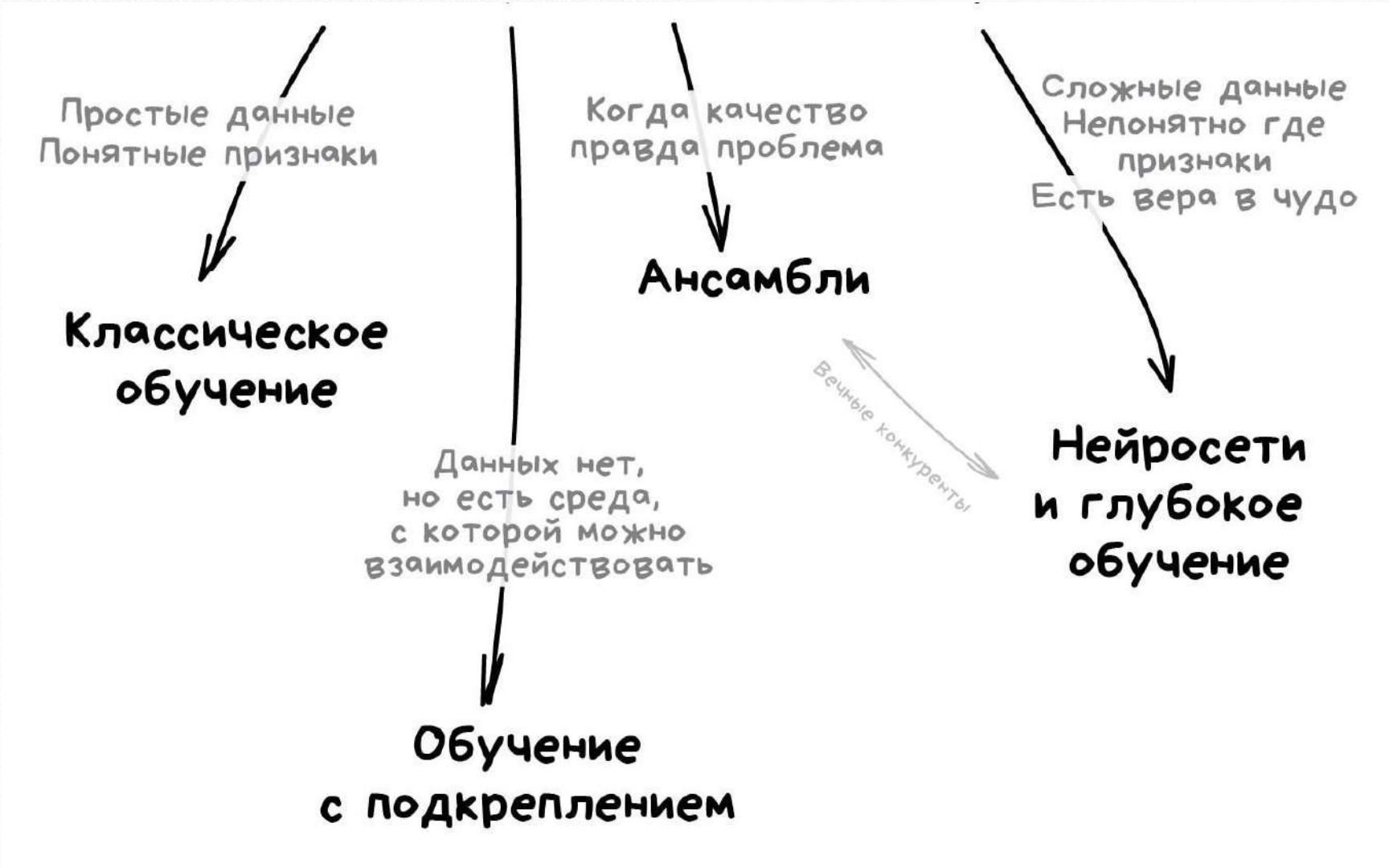
(2) Когда модель создана, она применяется для анализа данных.

Машинное обучение – междисциплинарное направление в ИИ

Общая постановка задачи машинного обучения

- Машинное обучение (machine learning) – обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться на данных.
- Имеется множество *объектов (ситуаций)* и множество возможных *ответов (откликов, реакций)*.
- *Между ответами и объектами существует* некоторая зависимость, но она неизвестна. Известна только конечная совокупность *прецедентов – пар вида «объект – ответ», – называемая* обучающей выборкой.
- ***На основе*** этих данных требуется обнаружить зависимость, то есть построить модель, способную для любого объекта выдать достаточно точный ответ. Чтобы измерить точность ответов, вводится критерий качества.

Виды машинного обучения



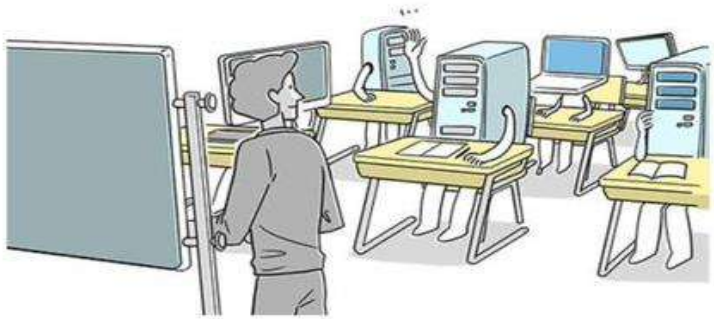
Задачи машинного обучения

Методы машинного обучения

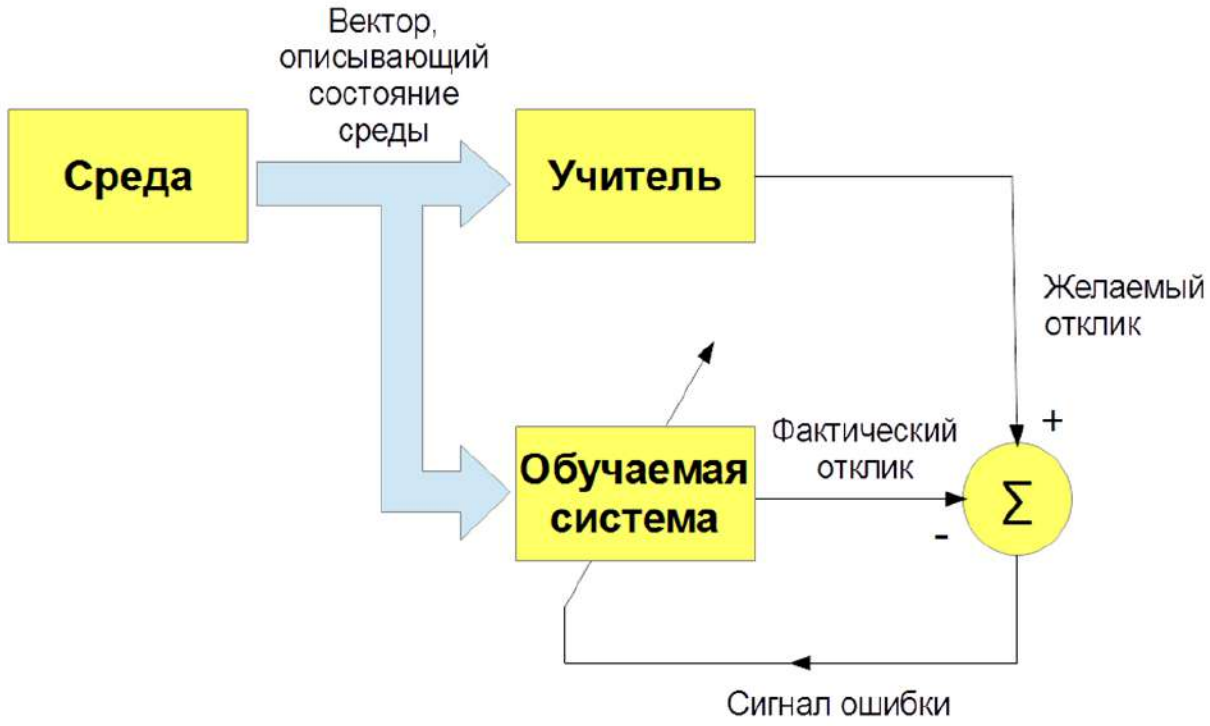
Обучение с учителем

Обучение с учителем - один из способов машинного обучения, в ходе которого испытуемая система принудительно обучается с помощью примеров «стимул-реакция». Стимул реакция - обучающая выборка.

*«При **обучении с учителем** компьютер получает образцы входных данных с отмеченными желаемыми результатами »*



Состоит в том, чтобы научить алгоритм сопоставлять разные значения разных атрибутов объекта со значением заданного целевого атрибута этого же объекта. Алгоритм находит функцию, которая сопоставляет значения входных атрибутов с целевым, а модель - это компьютерная программа, выполняющая эту функцию.



Проблемы обучения с учителем

(1) Состоит в том, чтобы научить алгоритм сопоставлять разные значения разных атрибутов объекта со значением заданного целевого атрибута этого же объекта. Алгоритм находит функцию, которая сопоставляет значения входных атрибутов с целевым, а модель - это компьютерная программа, выполняющая эту функцию.

(2) Для любого набора данных разумной сложности существует так много комбинаций входных данных и их возможных сопоставлений с выходными данными, что алгоритм не может испробовать их все.

(3) Каждый алгоритм машинного обучения предпочитает определенные типы функций во время поиска. Эти предпочтения известны как *смещение обучения алгоритма*. Реальная проблема в использовании машинного обучения состоит в том, чтобы найти алгоритм, смещение обучения которого лучше всего подходит для конкретного набора данных.

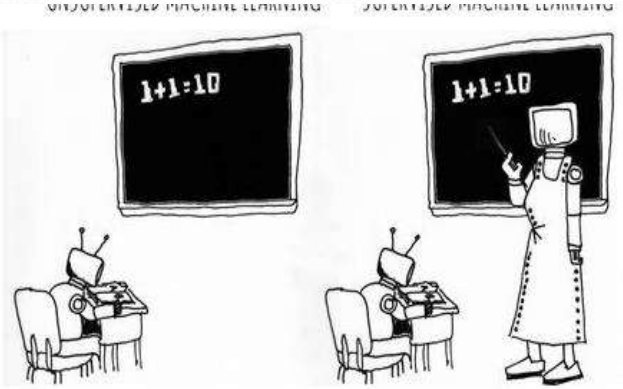
(4) Чтобы выяснить, какой из алгоритмов лучше всего работает с конкретным набором данных, требуются *эксперименты*. При обучении с учителем много времени и усилий тратится, чтобы создать набор данных с целевыми значениями атрибутов, прежде чем модель можно будет обучать.

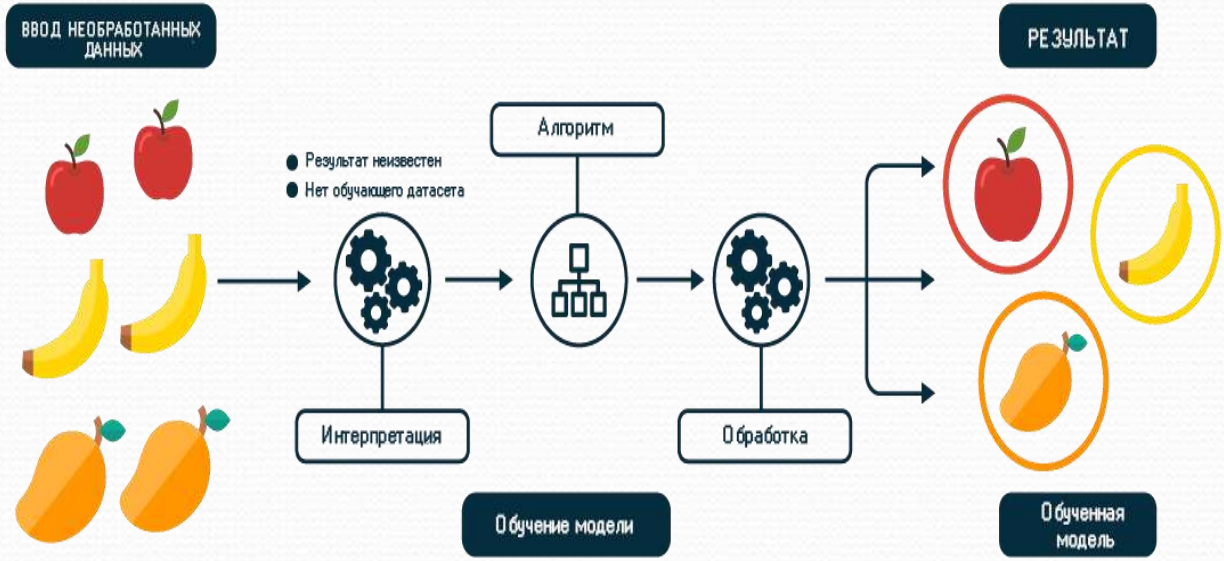
Обучение без учителя

Целевой атрибут отсутствует. Время на маркировку целевым атрибутом объектов в наборе данных не требуется. Однако обучение становится сложным - надо найти закономерности в данных. Основной метод - алгоритм ищет кластеры объектов, схожих друг с другом. Поиск начинают со случайной группы кластеров, а затем итеративно их обновляют, перебрасывая объекты из одного кластера в другой, чтобы увеличить подобие внутри кластеров.

Обучение без учителя – один из способов машинного обучения, при котором испытуемая система спонтанно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора.

*«**Обучение без учителя** часто противопоставляется обучению с учителем»*

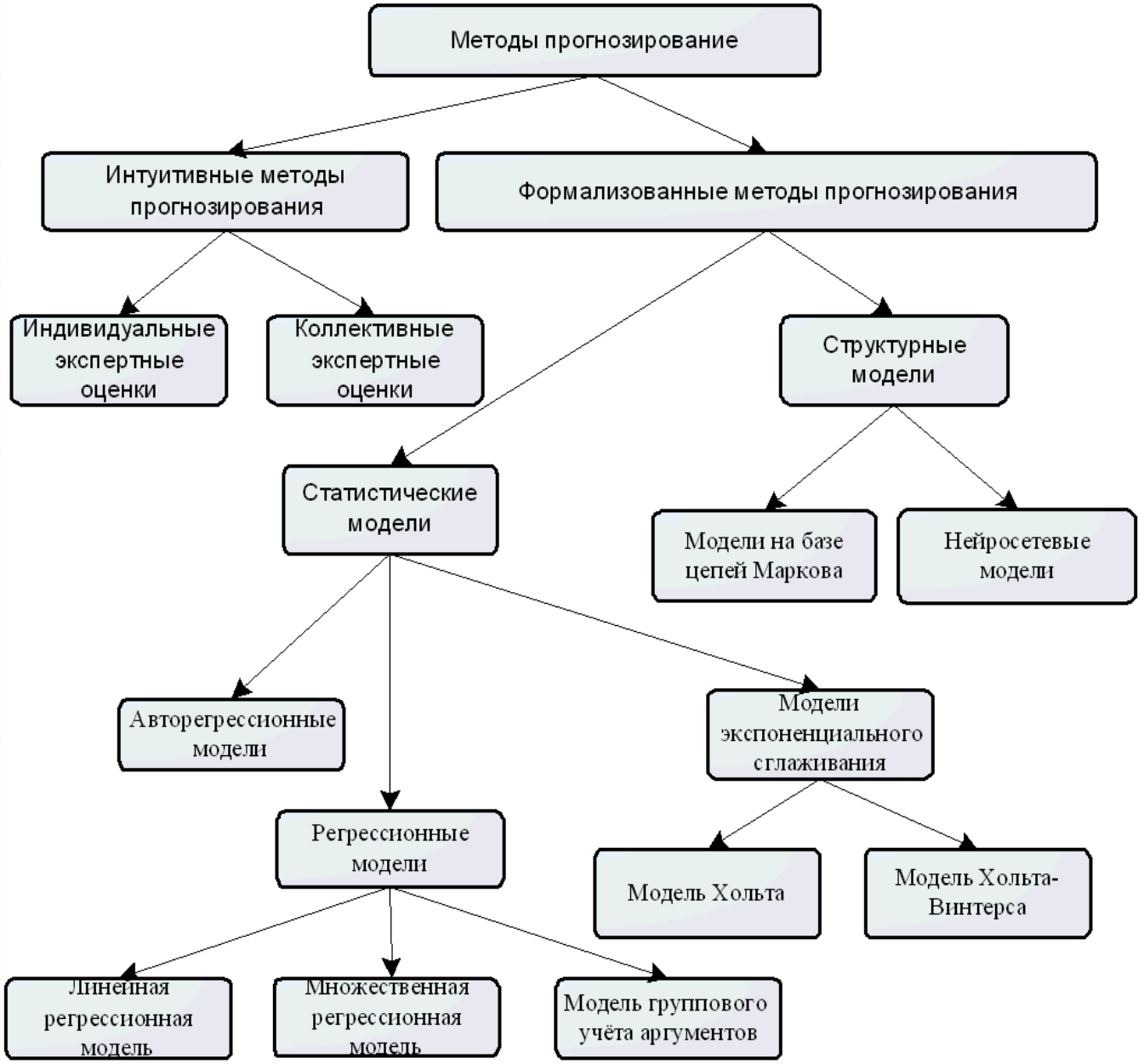




Машинное обучение прогнозированию

Прогнозирование – это задача оценки значения целевого атрибута конкретного объекта на основе значений других его атрибутов. Проблему прогнозирования решают алгоритмы машинного обучения **с учителем**, которые генерируют модели прогнозирования.

Сейчас насчитывается свыше 150 алгоритмов прогнозирования. На практике используются 15 – 20.



Прогнозирование: корреляционный анализ

Наиболее известные подходы к решению задачи прогнозирования: *корреляционный* и *регрессионный* анализ, *нейросети* и *деревья решений*.

Корреляция описывает силу взаимосвязи между двумя атрибутами. Коэффициент корреляции Пирсона ***r*** находится в диапазоне значений от –1 до +1.

Коэффициент корреляции Пирсона характеризует наличие линейной связи между признаками,

$$r_{xy} = \frac{\sum (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \cdot \sum (y_i - \bar{y})^2}}$$

где x_i — значения, принимаемые в выборке X, $\bar{X} = \frac{1}{n}\sum_{t=1}^{n} X_t$

y_i — значения, принимаемые в выборке Y;

$\bar{x}$ — средняя по X, $\bar{y}$ — средняя по Y. $\bar{Y} = \frac{1}{n}\sum_{t=1}^{n} Y_t$

- ***r*** = 0 – два атрибута независимы друг от друга;
- ***r*** = +1 – атрибуты имеют идеальную положительную корреляцию, любое изменение одного из них сопровождается эквивалентным изменением другого в том же направлении;
- ***r*** = –1 – атрибуты имеют идеальную отрицательную корреляцию, каждое изменение в одном из них сопровождается противоположным изменением в другом;
- ***r*** ≈ ±0,7 сильная линейная зависимость между атрибутами,
- ***r*** ≈ ± 0,5 – умеренная линейная зависимость,
- ***r*** ≈ ± 0,3 – слабая зависимость.

Корреляционный анализ: пример

В таблице представлен *фрагмент набора данных* для анализа сахарного диабета:

Объект	Рост (м)	Вес (кг)	Размер обуви	Продолжительность тренировок в неделю (мин.)	Вероятность развития диабета (%)
1	1,70	70	5	130	0,05
2	1,77	88	9	80	0,11
3	1,85	112	11	0	0,18

Есть и другие атрибуты, которые могут быть включены в набор, например возраст человека, и что среди представленных в таблице есть лишние (например, размер обуви, который явно не коррелирует с развитием диабета).

Выбор атрибутов для набора данных – ключевая задача науки о данных. Намеренно используем такой набор, какой есть.

Исходя из наших знаний о физиологии людей, мы ожидаем, что между некоторыми атрибутами будут взаимосвязи. Например, обычно чем выше человек, тем больше размер его обуви. Или чем больше кто-то тренируется, тем меньше в нем будет избыточного веса. Также ожидаем, что не обнаружим связи между размером обуви и временем тренировок.

Корреляционный анализ: пример 1

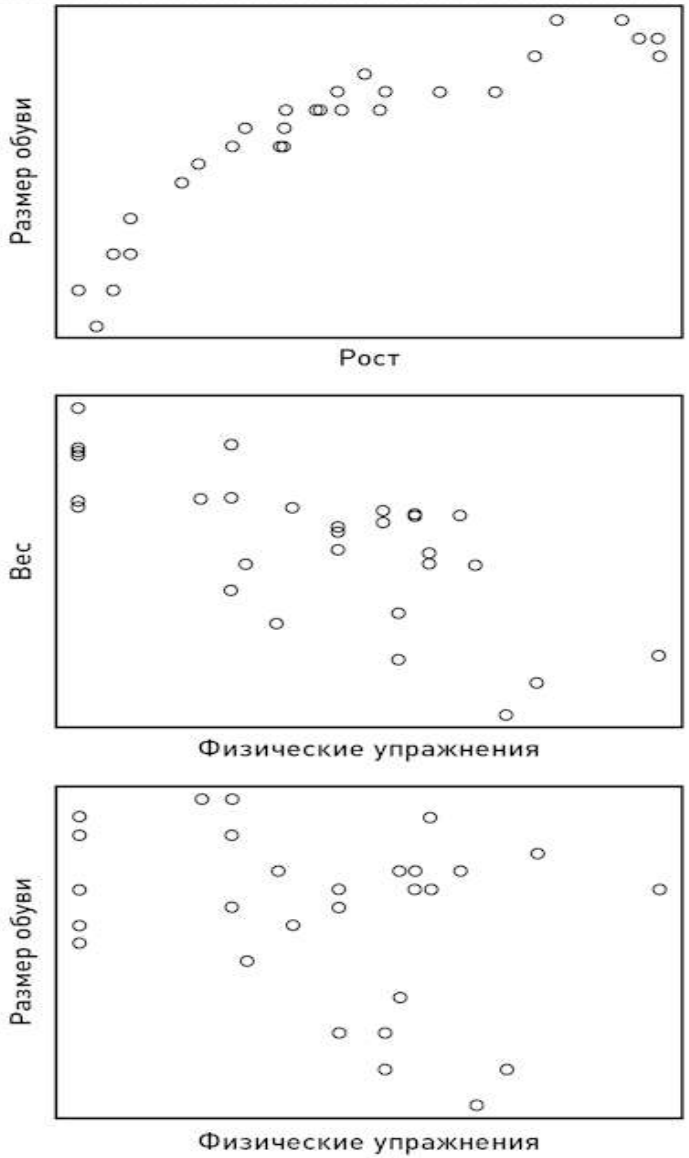
В качестве иллюстрации представлены три *диаграммы рассеяния*, отражающие данные:

Если мы вычислим коэффициент корреляции Пирсона между размером обуви и ростом, то ***r*** = 0,898 (сильная положительная корреляция).

Коэффициент корреляции Пирсона между весом и физическими упражнениями ***r*** = –0,710 (сильная отрицательная корреляция).

Коэффициент корреляции Пирсона между временем тренировок и размером обуви ***r***=–0,272 (корреляция отсутствует).

Когда вы знаете значимые атрибуты данных, то создавать модели можно точно и быстро! (см. Родзин С.И. «Биоинспирированная гиперэвристика для отбора значимых признаков в задачах классификации больших данных» // Вестник компьютерных и информационных технологий. 2021. Т.18. № 5. С. 35-44. DOI: 10.14489/vkit.2021.05.pp.035-044.)



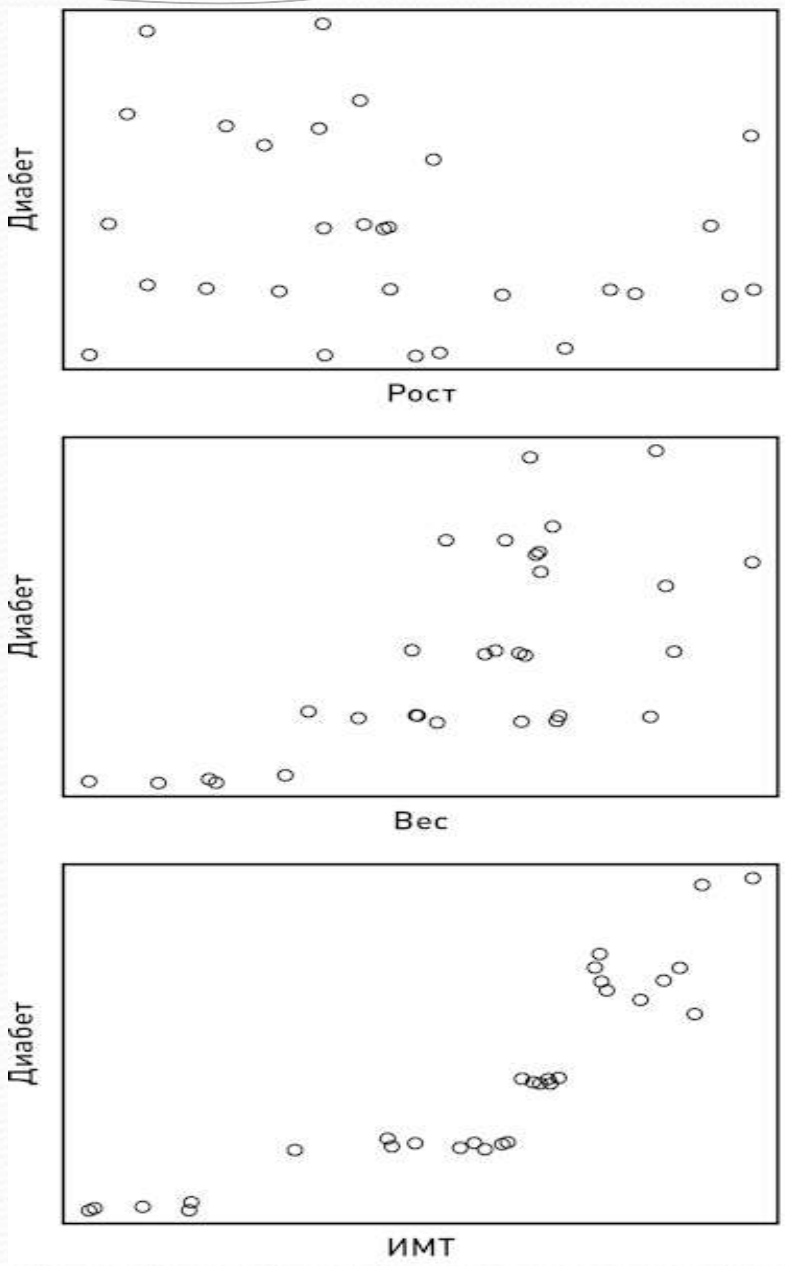
Корреляционный анализ: пример 2

Применение коэффициента корреляции Пирсона к анализу данных не ограничивается только парами атрибутов. Проблему можно обойти, применяя функции для групп атрибутов, используя производные атрибуты.

В предыдущем примере используем производный атрибут – ИМТ (индекс масса тела/рост). Он поможет определить людей, которые подвержены риску ожирения.

Например, на рис. представлены три диаграммы отношений между целевым атрибутом диабета и ростом, весом, ИМТ.

Между ростом и диабетом не наблюдается закономерности (***r*** = –0,277). Между весом и диабетом положительная корреляция (***r*** = 0,655). Между ИМТ и диабетом сильная корреляция (***r*** = 0,877) - развитие диабета зависит от взаимосвязи роста и веса, т.е. от ИМТ. Поэтому врачи интересуются ИМТ людей - это дает больше информации о вероятности развития диабета, чем рост или вес по отдельности. Не вес сам по себе способствует заболеванию, а его избыточность.

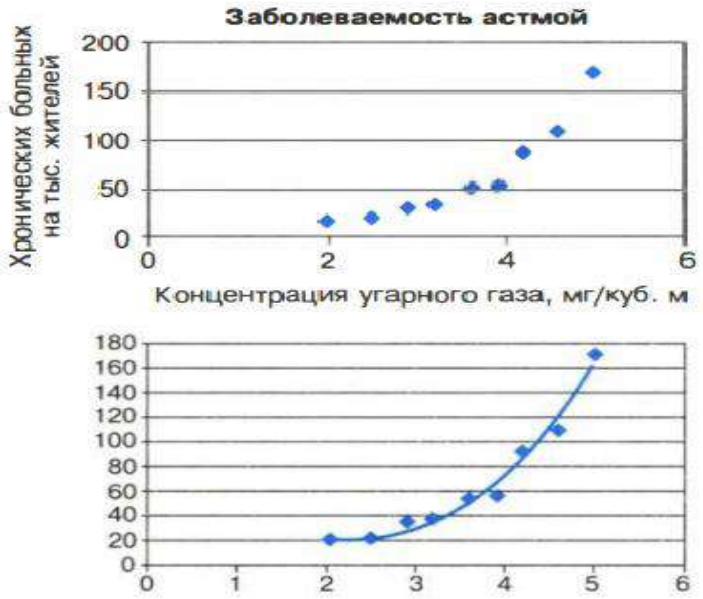


Прогнозирование: регрессионный анализ

Когда набор данных состоит из числовых атрибутов, часто используется регрессионная модель прогнозирования. Регрессионный анализ оценивает ожидаемое (среднее) значение целевого атрибута при фиксированных входных атрибутах.

(1) Выдвигается гипотеза об отношении между входными и целевым атрибутом.

(2) Определяется математическая модель предполагаемой взаимосвязи. Она называется *функцией регрессии*. Функцию регрессии можно представить как машину, которая преобразует входные данные в выходные, а параметры – в виде настроек, управляющих поведением машины. Функция регрессии может иметь несколько параметров. Целью регрессии является поиск настроек для этих параметров.



Как спрогнозировать поведение заболеваемости от концентрации и **построить** математическую модель?
Необходима **формульная зависимость** между величинами.
Вид функции построенной по этой формуле неизвестен. Получается методом **подбора** по экспериментальным данным.
График должен проходить максимально близко к точкам диаграммы.

Линейный регрессионный анализ

Поиск функции целесообразно начинать с самого простого типа - с *линейной зависимости*. Простейшим применением линейной регрессии является моделирование взаимосвязи между двумя атрибутами: входным атрибутом **X** и целевым атрибутом **Y**.

В этой задаче функция регрессии имеет следующий вид: $\mathbf{Y} = a + b\mathbf{X}$.

Это уравнение линейной функции, где a и b – параметры функции регрессии. Параметр a – это точка пересечения прямой с осью ординат, когда **X** равен нулю. Параметр b определяет угол наклона прямой. Параметры неизвестны, их надо найти.

Стратегия установки этих параметров состоит в том, чтобы начать со случайных значений, а затем итеративно обновлять параметры, уменьшая общее отклонение функции в наборе данных. Общее отклонение рассчитывается в три этапа:

1. Функция применяется к набору данных и для каждого объекта в наборе оценивает значение целевого атрибута **Y**.
2. Отклонение функции для каждого объекта вычисляется путем вычитания значения целевого атрибута из его фактического значения. Затем полученные отклонения возводятся в квадрат и суммируются.
3. Это величина известна как сумма квадратов отклонений, а стратегия подбора линейной функции путем поиска параметров – как метод наименьших квадратов (***SSE***):

$$\mathrm{SSE} = \sum_{i=i}^{n} (target_i - prediction_i)^2,$$

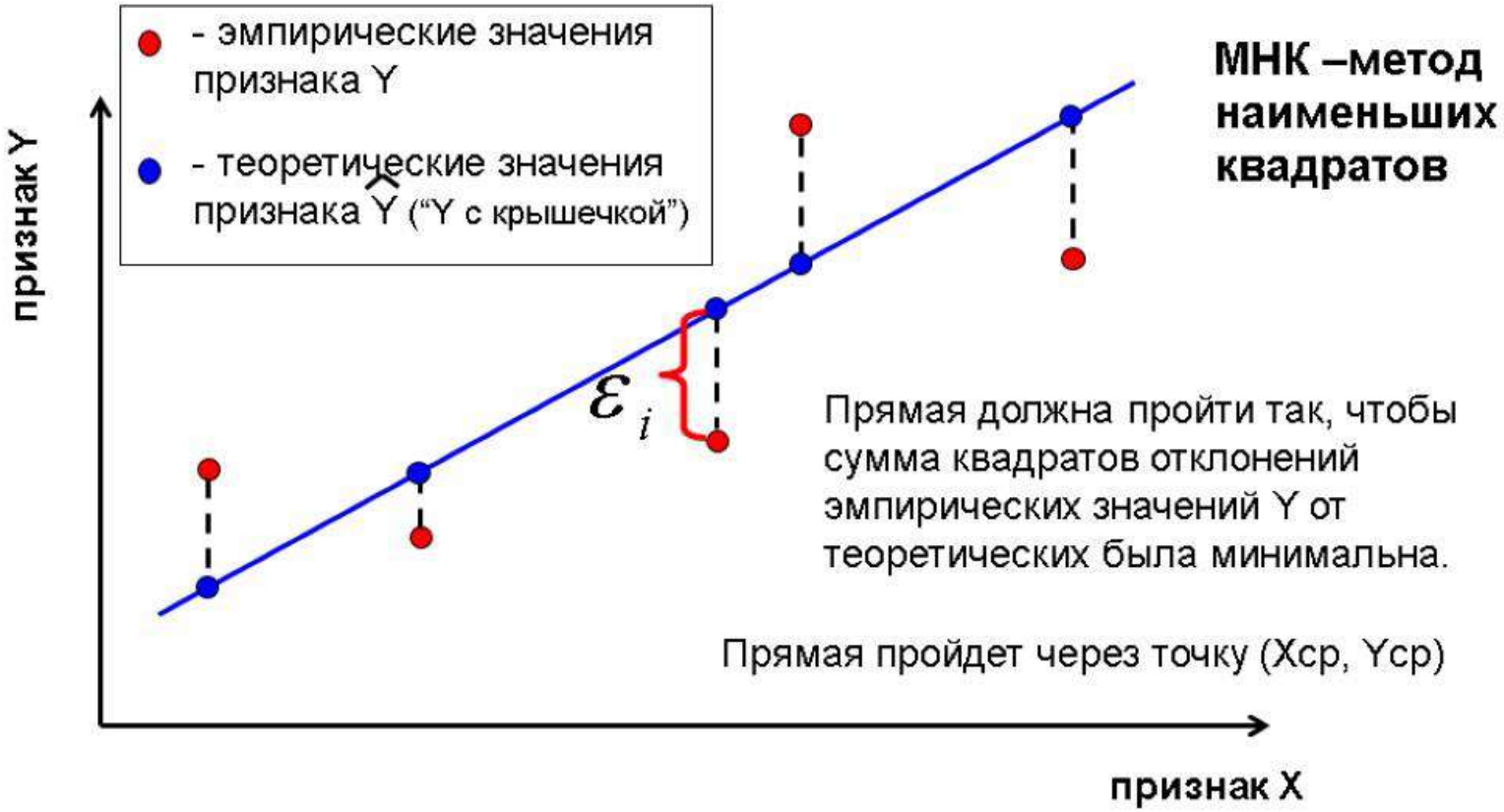
где набор данных содержит n объектов, $target_i$ – это значение целевого атрибута для объекта i в наборе данных, а $prediction_i$ – оценка функции цели для того же объекта.

Метод наименьших квадратов

Линейная регрессия

Модель – уравнение прямой – $Y = a + b*X$

Построение модели – расчет коэффициентов



Линейная регрессионная модель. Пример

Пример. Построить линейную регрессионную модель прогнозирования, которая оценивает вероятность развития диабета у человека с учетом его ИМТ.

Применим метод ***SSE***, чтобы найти подходящую прямую для набора данных. Рис. иллюстрирует эту прямую и ее расположение относительно объектов в наборе данных. Пунктиром показано отклонение для каждого объекта в этой прямой.

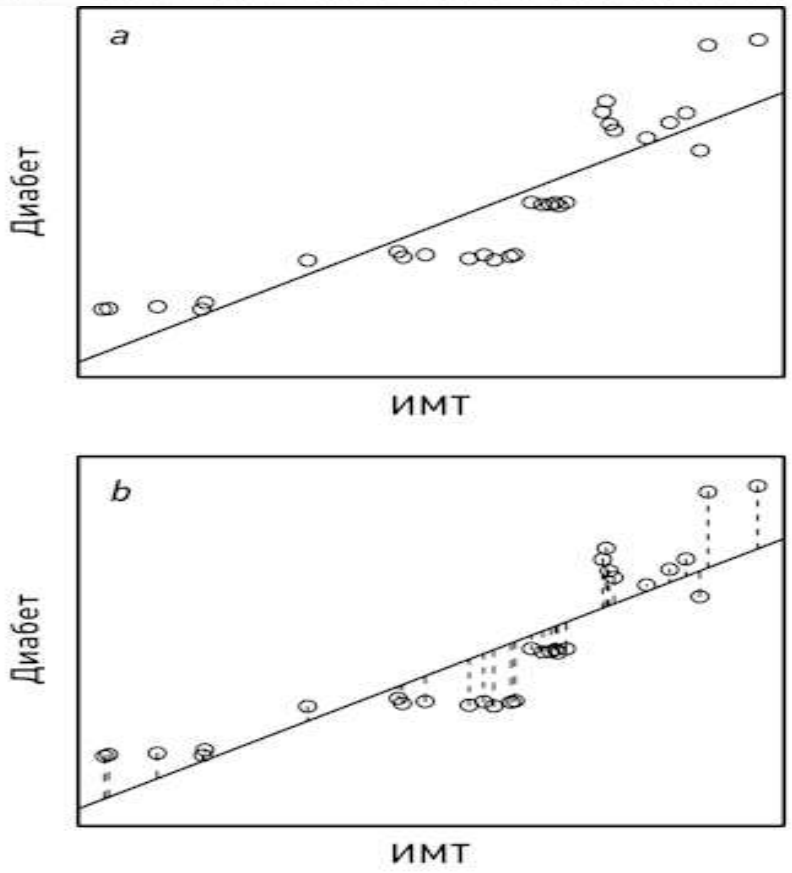
По методу ***SSE*** линией наилучшего соответствия будет прямая:

Диабет = −7,38431 + 0,55593 × **ИМТ**.

Чтобы предсказать вероятность диабета у человека, вводим его значение ИМТ в модель. Например, когда ИМТ=20, модель возвращает прогноз 3,73% для атрибута «**Диабет**». Если ввести среднее значение ИМТ в наборе данных (ИМТ=24,0932), модель оценивает атрибут диабета как 4,29%, что является средним для всего набора данных.

Перед применением ***SSE*** важно проверить наличие «выбросов» в данных.

Диабет = a + b×**ИМТ** + c×**Упр** + d×**Вес**

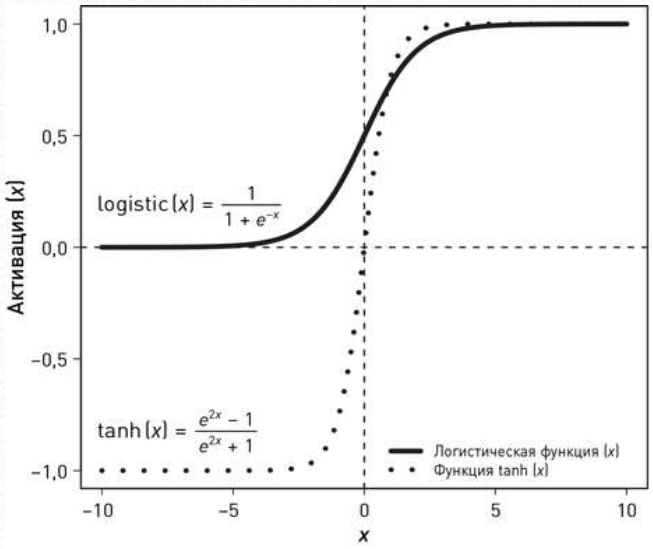


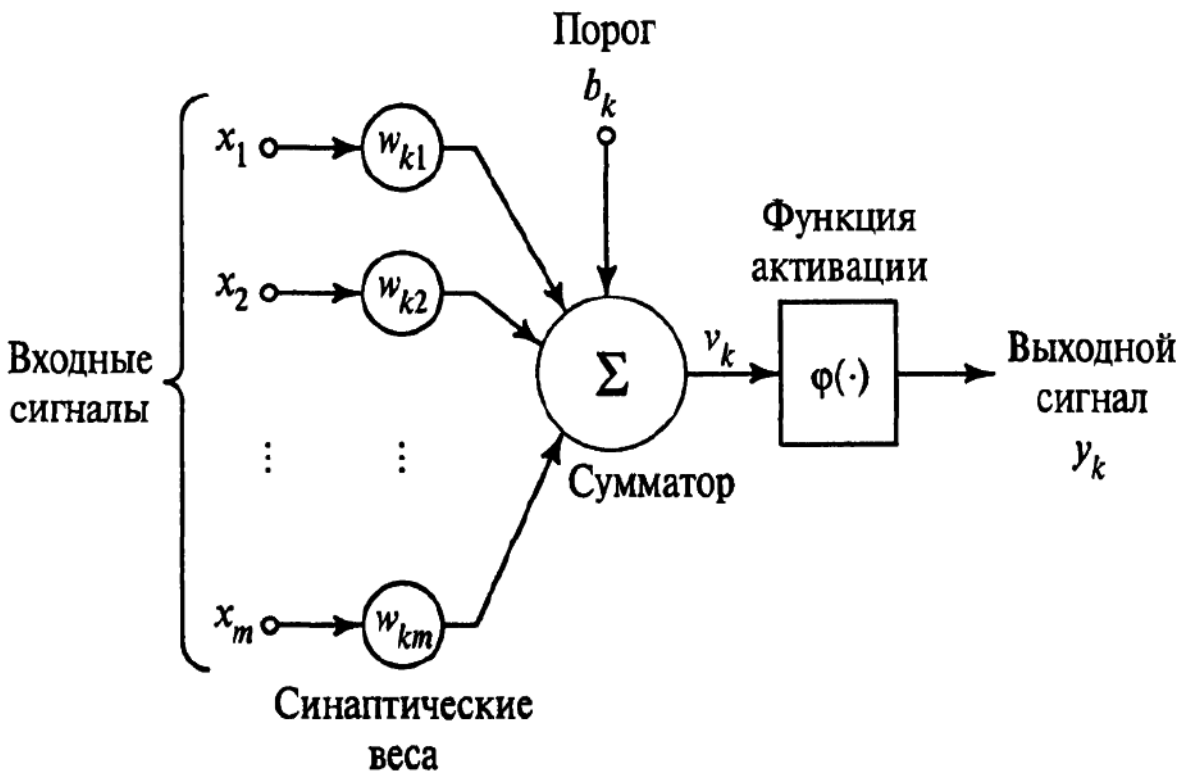
Прогнозирование: нейросети и глубокое обучение

По своей сути нейрон – это функция линейной регрессии с несколькими входами. Единственное существенное различие состоит в том, что в нейроне выходной сигнал определяется другой функцией, которая называется *функцией активации* и которая может быть нелинейной.

Нейрон выполняет очень простой набор операций:
1. Умножает каждый вход на его вес;
2. Суммирует результаты;
3. Сумму проводит через функцию активации.

Операции 1 и 2 – это просто вычисление функции регрессии, а операция 3 - функция активации.



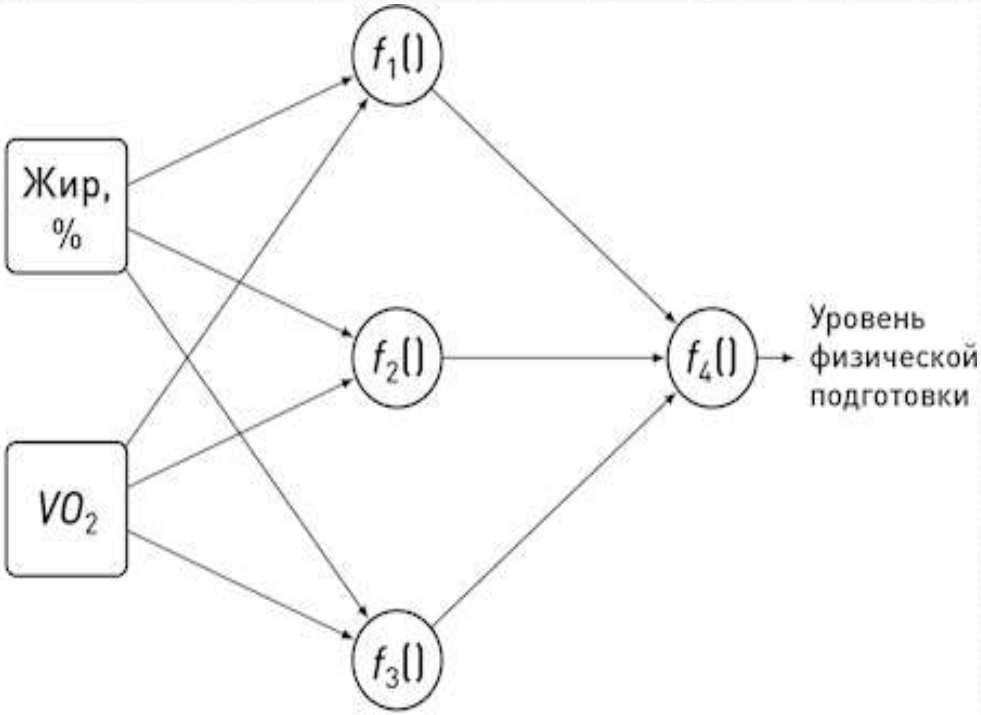


В качестве функций активации часто применяются *логистическая* функция и функция *tanh*(*x*).

Пример прогнозирования нейросетью

Дана нейросеть, которая принимает значение процентного содержания жира (*Жир*%) в организме человека и его максимальное потребление кислорода (*VO2*) в качестве входных данных и вычисляет его уровень физической подготовки:

Все нейроны в среднем слое сети на основе *Жир*% и *VO2* вычисляют функции $f_1()$, $f_2()$ и $f_3()$. Функции моделируют взаимодействие между входами по-разному. Эти функции - новые атрибуты, однако не несут никакого смысла для человека. Просто они фиксируют взаимодействия между другими атрибутами, которые сеть сочла полезными. Узел в сети f_4 вычисляет функцию от $f_1()$, $f_2()$ и $f_3()$. На выходе – прогноз уровня физической подготовки. Эта функция не значима для человека. Она лишь определяет взаимодействие, которое, как обнаружила сеть, имеет высокую корреляцию с целевым атрибутом.



Как обучается отдельный нейрон ?

Обучение нейросети включает в себя поиск правильных весов для ее связей. Как обучается отдельный нейрон рассчитывать вес связи?

Пусть имеется обучающий набор данных, который определяет для каждого объекта входные значения и целевой атрибут.

Входам нейрона назначим некоторые веса.

Возьмем объект из набора данных и представим нейрону значения входных атрибутов этого объекта.

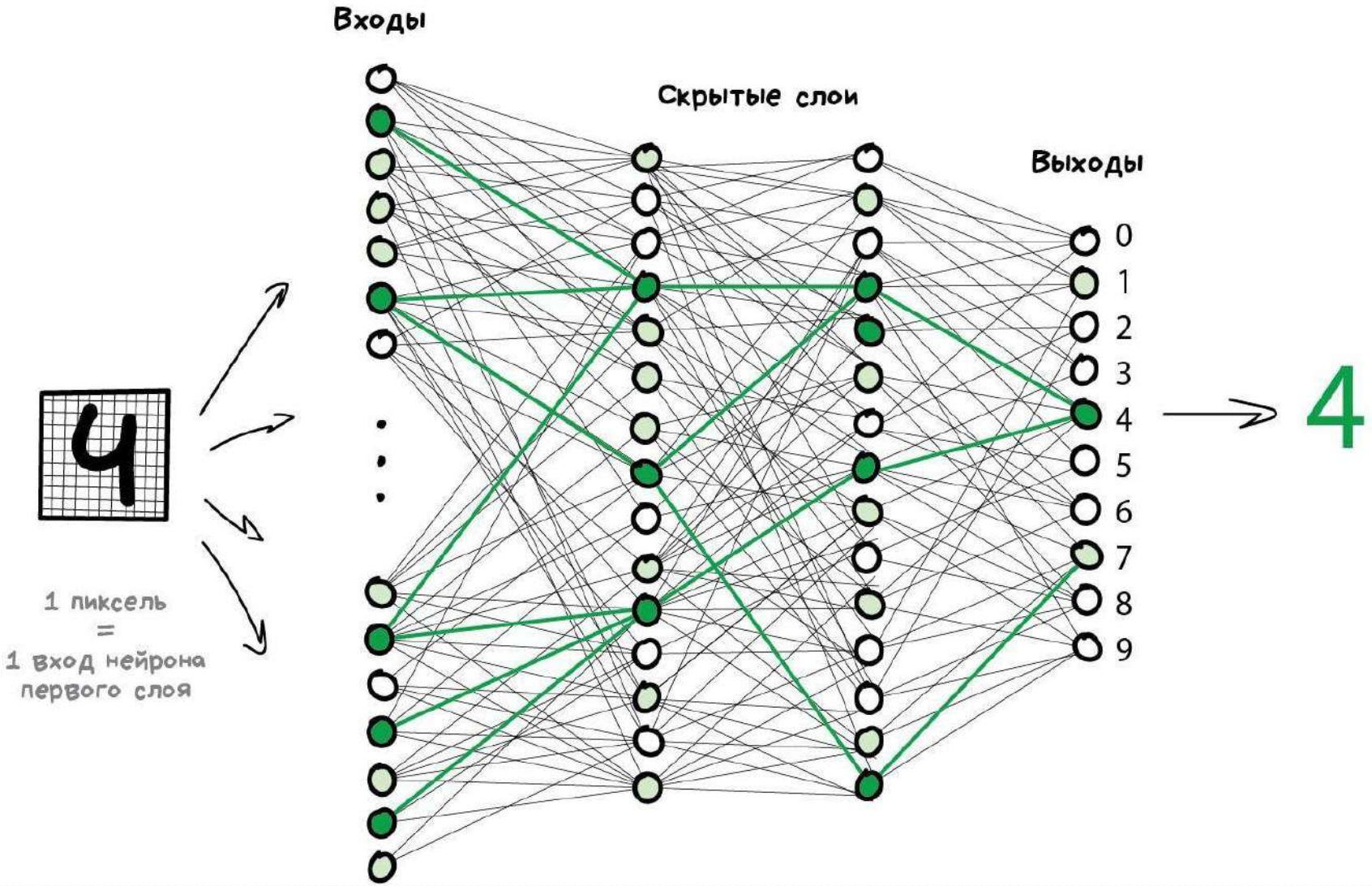
Нейрон выдаст прогноз для цели.

Вычитаем этот прогноз из значения целевого атрибута. Получим отклонение выхода нейрона для этого объекта.

Правило для обновления весов входов нейрона:

1. Если отклонение равно 0, не меняйте веса на входах.
2. Если отклонение положительное, требуется увеличить прогнозное значение, поэтому нужно прибавить веса всех связей с положительным входом и понизить веса связей с отрицательным.
3. Если отклонение отрицательное, требуется уменьшить прогнозное значение, поэтому нужно понизить веса всех связей с положительным входом и прибавить веса связей с отрицательным.

Многослойный перцептрон



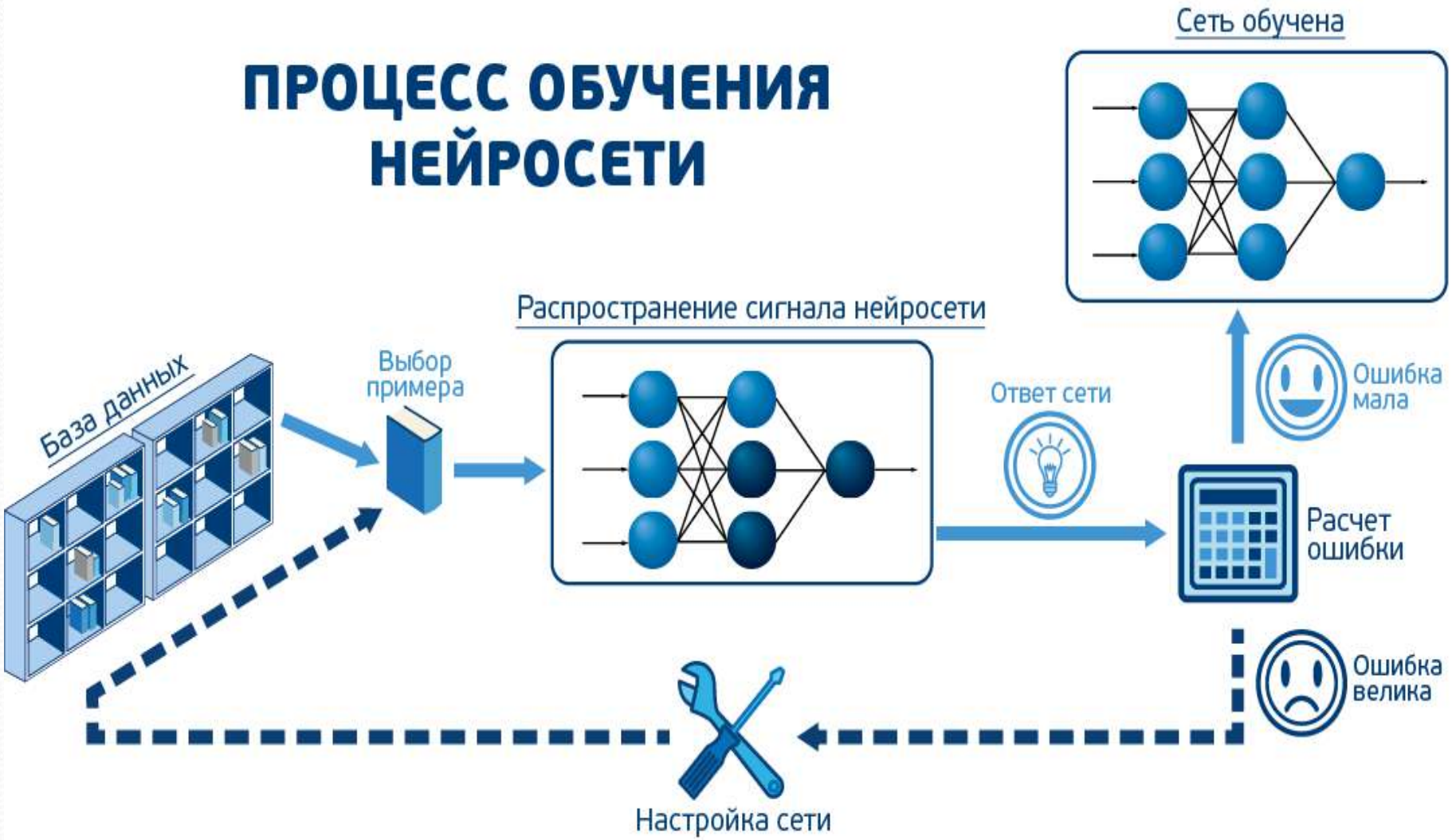
Как обучить многослойную нейросеть ?

Сложность обучения многослойной нейросети в том, что обновление веса требует оценки ошибки во всех нейронах. Вычислить ошибку для нейрона в выходном слое сети просто, а для нейронов в скрытых слоях намного сложнее. Стандартный способ обучения таких нейросетей - алгоритм обратного распространения ошибки.

1. Рассчитайте ошибку для каждого из нейронов в выходном слое и обновите согласно правилу выше веса входящих связей этих нейронов.
2. Поделитесь ошибкой, рассчитанной для нейрона, с каждым из нейронов в предыдущем слое, который связан с ним, пропорционально весу связей между двумя нейронами.
3. Для каждого нейрона на предыдущем уровне вычислите общую ошибку сети, за которую он ответственен, суммируя с теми ошибками, которые были переданы обратно в него, и используйте результат этого суммирования, чтобы обновить веса на связях, входящих в этот нейрон.
4. Пройдите остальные слои в сети, повторяя шаги 2 и 3 до тех пор, пока веса между входными нейронами и первым слоем скрытых нейронов не будут обновлены.

P.S. В алгоритме обновляемый вес нейронов высчитывается так, чтобы уменьшить, но не устранить полностью ошибку нейрона. Причина в том, что цель обучения сети – дать ей возможность сделать выводы, которых нет в данных обучения, а не просто запомнить эти данные. Обновление весов делает сеть более подходящей к набору данных. В некоторых версиях алгоритма веса обновляются только после того, как сеть был обучена на нескольких объектах, чтобы избежать переобучения сети.

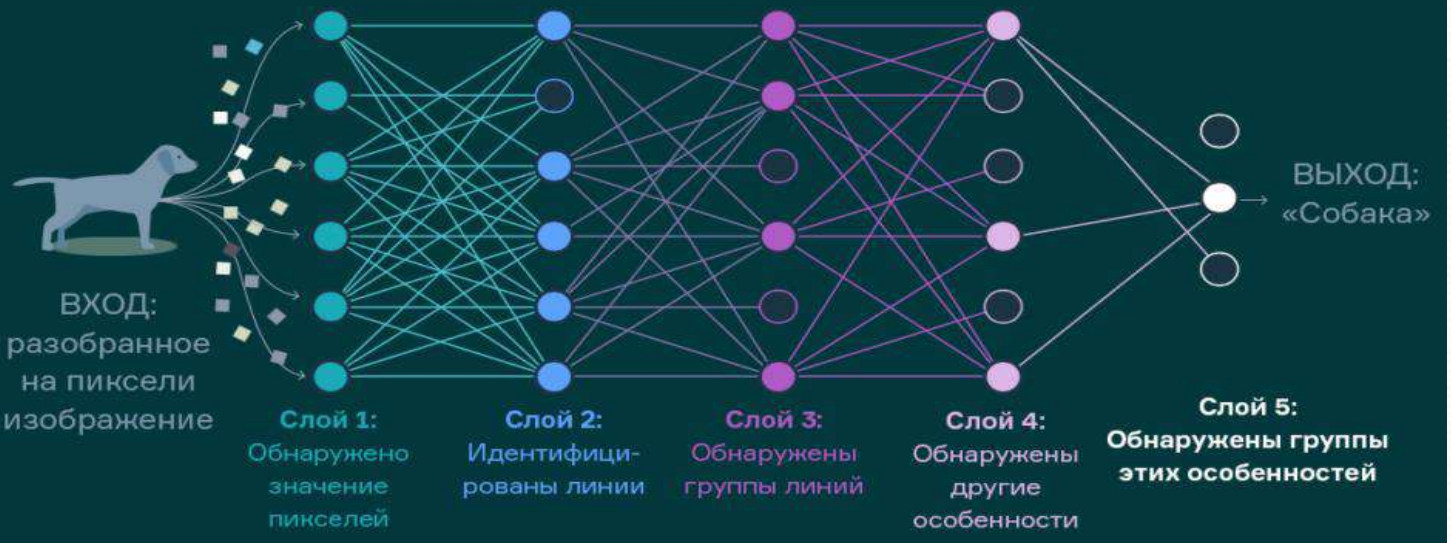
ПРОЦЕСС ОБУЧЕНИЯ НЕЙРОСЕТИ



Нейросети глубокого обучения

Одним из наиболее удивительных технических достижений последних 10 лет стало появление нейросетей глубокого обучения. Это нейросети, имеющие несколько скрытых слоев.

Нейросети передают входную информацию (изображение) через слои цифровый нейронов. Каждый слой совершает операцию «математической свертки» информации — она позволяет обнаружить различные характеристики входного сигнала. Чем больше слоев между входом и выходом, тем глубже нейросеть.



Исследования показывают, что глубокие нейросети часто работают лучше, когда на ранних слоях определяются самые примитивные особенности изображения, и чем глубже слой — тем сложнее распознаваемые им особенности. Человеческий мозг работает так же.

Прогнозирование: деревья решений

Линейная регрессия и *нейронные сети* лучше всего работают с числовыми входными данными. Если входные атрибуты в наборе данных в основном номинальные или порядковые, лучше использовать другие алгоритмы и модели машинного обучения, такие как *деревья решений*.

Дерево решений кодирует условный оператор **ЕСЛИ-ТО-ИНАЧЕ** в древовидной структуре. Все пути в структуре дерева решений от корня до листа определяются правилом классификации, состоящим из последовательных тестов. Цель обучения дерева решений - найти такие правила классификации, которые делят обучающий набор данных на группы объектов, имеющих одинаковое значение целевого атрибута.

Прогнозирование: деревья решений

Прародителем большинства современных алгоритмов машинного обучения деревьев решений является алгоритм **ID3**. Он строит деревья решений рекурсивным способом в глубину, добавляя один узел, начиная с корневого узла.

(1) ID3 начинается с выбора атрибута для проверки в корневом узле. Затем **ID3** наращивает каждую ветвь, до тех пор, пока все объекты каждой ветви не будут иметь одинаковое значение целевого атрибута. Тогда в дерево добавляется конечный узел и помечается значением целевого атрибута, общего для всех объектов ветви.

(2) ID3 выбирает оптимальный атрибут для тестирования в каждом узле дерева, чтобы минимизировать количество тестов. Для этого используется информационная энтропия К.Шеннона. $H = -\sum_{i=1}^{n} \frac{N_i}{N} \log\left(\frac{N_i}{N}\right)$

где n — число классов в наборе данных, N_i — число объектов i-го класса, N — общее число объектов в наборе данных.

ID3 выбирает для тестирования в узле атрибут, который приводит к наименьшему значению энтропии после разделения набора данных с использованием этого атрибута.

ID3 позволяет работать только с дискретной целевой переменной. Деревья решений, построенные на основе ID3, получаются квалифицирующими. Число потомков в узле неограниченно. Алгоритм не работает с пропущенными данными.

Пример прогнозирования методом ID3

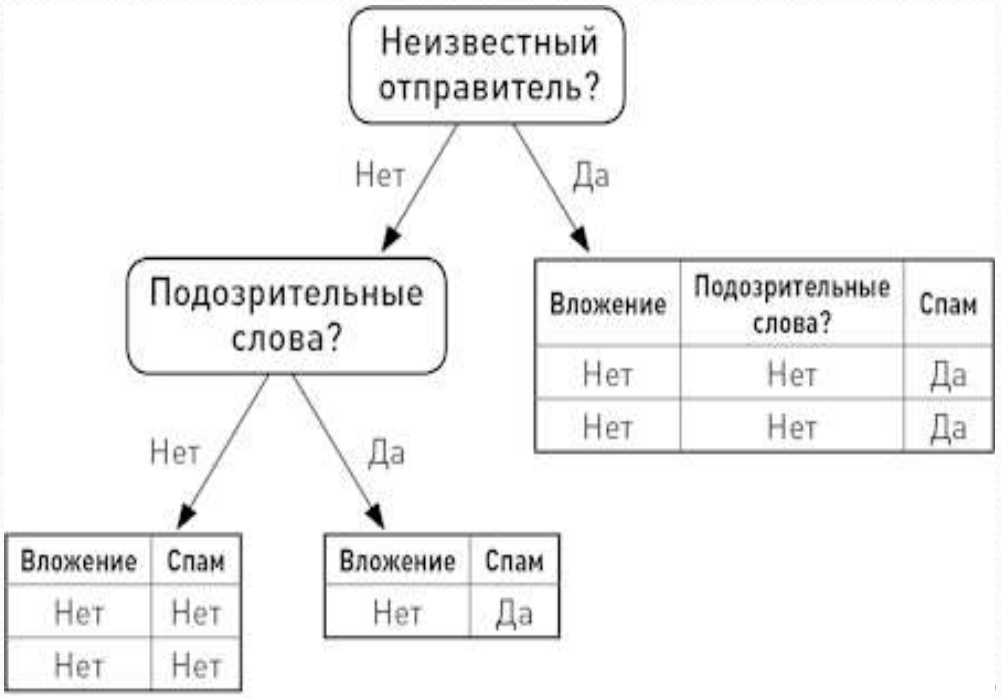
Приведен фрагмент списка электронных писем, в котором каждое описывается рядом атрибутов и тем, является оно спамом или нет.

Номинальные атрибуты «**Вложение**», «**Подозрительные слова**» и «**Неизвестный отправитель**» - входные атрибуты. Атрибут «**Спам**» – цель.

Вложение	Подозрительные слова	Неизвестный отправитель	Спам
Нет	Нет	Да	Да
Нет	Нет	Да	Да
Нет	Да	Нет	Да
Нет	Нет	Нет	Нет
Нет	Нет	Нет	Нет

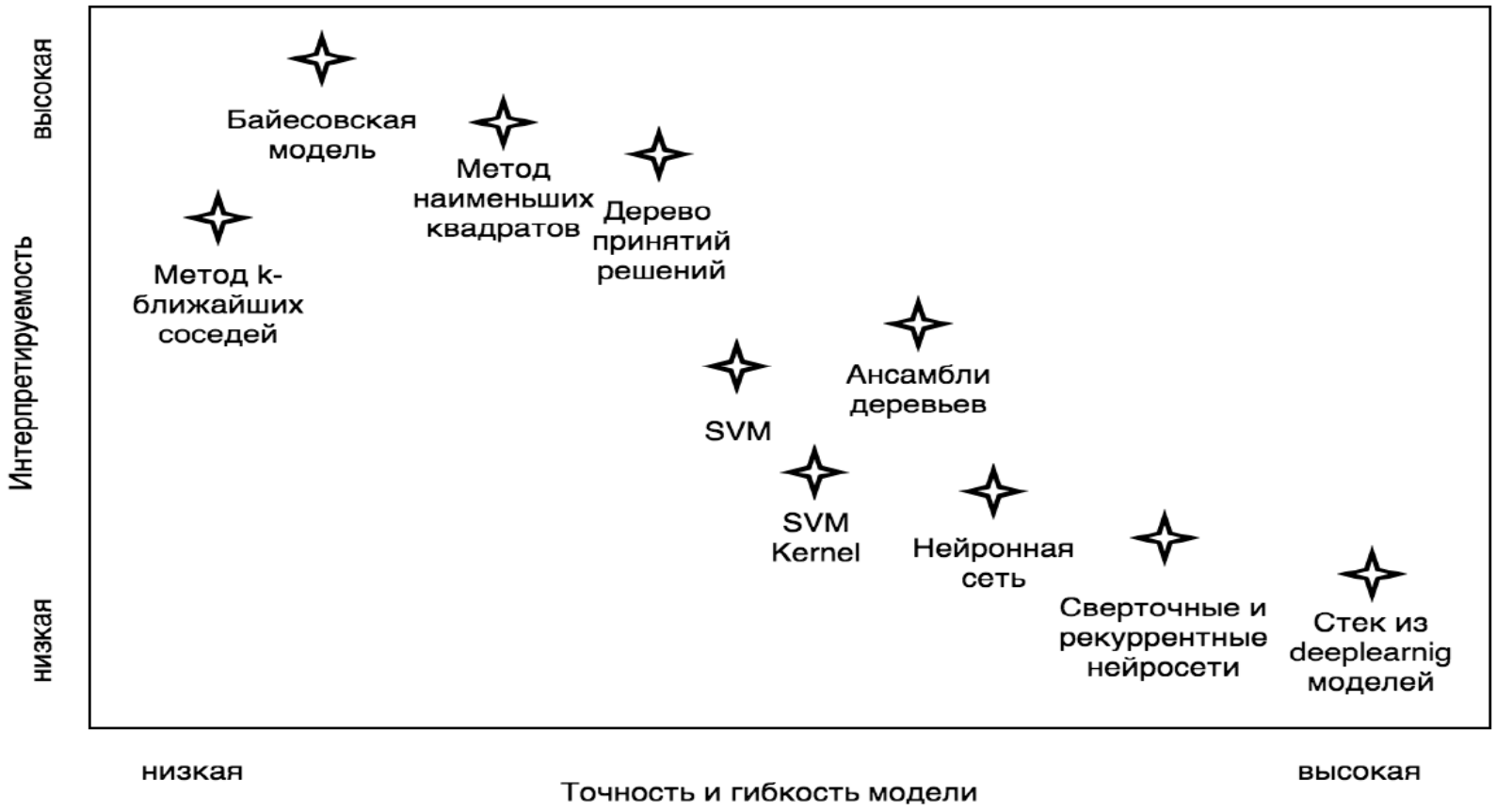
Атрибут «**Неизвестный отправитель**» разбивает набор данных на две группы (одна из них содержит большинство объектов, где Спам = И, а другая, где Спам = Л). «**Неизвестный отправитель**» помещается в корневой узел.

Все объекты в правой ветви имеют одинаковое целевое значение, а в левой – разное. Разделение объектов в левой ветви с использованием атрибута «**Подозрительные слова**» приводит к образованию двух наборов, где для одного Спам = Л, для другого Спам = И.



Зависимость гибкости алгоритма машинного обучения и интерпретируемости полученной модели

Самое важное: выбор модели среди множества алгоритмов машинного обучения: дерево решений, KNN (метод k-ближайших соседей), SVM (метод опорных векторов), NN (нейросеть). Основание: чего мы от модели хотим, насколько решения модели должны быть понятными (интерпретируемыми):



NFL-теорема и золотое правило оценки моделей

Среди алгоритмов машинного обучения нет одного наилучшего алгоритма. **NFL-теорема** утверждает, что не существует ни одного алгоритма машинного обучения, в среднем превосходящего все другие алгоритмы по всем возможным наборам данных. Поэтому этап моделирования обычно включает в себя построение нескольких моделей с использованием разных алгоритмов и их последующее сравнение, чтобы определить, какой из алгоритмов генерирует наилучшую модель.

Золотое правило оценки моделей заключается в том, что их **никогда не следует тестировать на тех же данных, на которых они были обучены**.

Как обеспечить выполнение золотого правила? - Данные разбиваются на **три части** – *обучающий набор, оценочный набор и тестовый набор*. Пропорции, обычно составляют **50:20:30** или **40:20:40**. Обучающий набор используется для обучения начальной группы моделей. Оценочный набор – для сравнения эффективности этих моделей на новых данных. После выявления алгоритма, который сгенерировал лучшую модель, используется тестовый набор.

Выводы. Мир меняется, а модели – нет. Предполагается, что будущее будет таким же, как и прошлое. Это предположение о стационарности. Алгоритмы машинного обучения ищут в прошлом закономерности, которые можно обобщить и интерполировать в будущее. Но это не всегда работает. По причине дрейфа модели перестают работать и нуждаются в перестройке.

Машинное обучение на языке Python

Процесс поиска подходящего алгоритма машинного обучения носит нелинейный характер. **Python**, будучи интерпретируемым высокоуровневым языком программирования, удобен для опробования разных вариантов. Он работает быстро, хотя и медленнее **C**, содержит огромное число библиотек.

Работая на **Python**, можно поручить численные расчеты расширениям, реализованным на **C** (библиотеки **NumPy** и **SciPy** http://scipy.org/Download). **NumPy** отвечает за многомерные массивы – основную структуру данных для современных алгоритмов. **SciPy** реализует быстрые численные алгоритмы. **Matplotlib** (http://matplotlib.org/) –библиотека для построения качественных графиков на **Python**.

В *Windows*, *Mac* и *Linux* имеются инсталляторы **NumPy, SciPy** и **matplotlib**.

Наборы данных для программирования задач машинного обучения можно использовать из репозитория наборов данных **UCI** (Калифорнийский университет в Ирвайне). Там свыше 200 наборов. Адрес репозитория http://archive.ics.uci.edu/ml/ .

Благодарю за внимание !

Contact Information:

E-mail: srodzin@sfedu.ru

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Кафедра МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ И ПРИМЕНЕНИЯ ЭВМ

Дисциплина:

Машинное обучение и биоинспирированная оптимизация

ЛЕКЦИЯ 4

Решение задач классификации, кластеризации, прогнозирования, обнаружения аномалий методами машинного обучения

Родзин Сергей Иванович,
профессор ИКТИБ ЮФУ,
srodzin@sfedu.ru

Стандартные задачи анализа данных

Одним из важнейших навыков специалиста по данным является способность **сформулировать проблему как стандартную задачу анализа данных**. Большинство проектов в этой области можно отнести к одному из четырех основных классов задач:

- кластеризация (или сегментация);
- обнаружение аномалий (или выбросов);
- поиск ассоциативных правил;
- прогнозирование (классификация и регрессия).

Существует множество алгоритмов машинного обучения, и каждый предназначен для конкретной задачи анализа данных. Например, алгоритмы, генерирующие модели дерева решений, в первую очередь предназначены для решения задач прогнозирования.

Поскольку задача влияет как на структуру набора данных, так и на выбор алгоритмов машинного обучения, определиться с ее типом необходимо на раннем этапе жизненного цикла проекта, в идеале – на этапе понимания бизнес-целей **CRISP-DM**. Рассмотрим стандартные типы задач.

Задача кластеризации (сегментация)

- ***Кластеризация*** – разбиение совокупности объектов на однородные группы (*кластеры*).
- ***Кластер*** (cluster – «скопление», «гроздь») можно охарактеризовать как группу объектов, имеющих общие свойства.
- Задача кластеризации сводится к определению "сгущений точек".
- ***Цель*** кластеризации – поиск существующих структур.

Кластеризация: задача поддержки маркетинговых кампаний и продаж

Как сформулировать проблему как задачу кластеризации?

Диапазон атрибутов, которые можно использовать для описания клиентов в процессе кластеризации, огромен, но есть наиболее типичные:

- демографическая информация (возраст, пол и т.д.),
- место жительства (почтовый индекс, адрес и т.д.),
- транзакционная информация (продукты или услуги, которые приобретал клиент, доход от него, жаловался ли на услугу и проч.).

Большая проблема – определить, какие атрибуты должны быть включены, а какие исключены, чтобы добиться наилучших результатов. Принятие решения о выборе атрибутов основано на итерациях экспериментов, их анализе специалистом.

Наиболее известным алгоритмом машинного обучения для кластеризации является алгоритм **k-средних**. Буква **k** в названии указывает количество кластеров, которые алгоритм ищет в данных.

Кластеризация методом к-средних (k-means)

Основные «правила игры»:

1. k – число кластеров – выбирается заранее
2. Начальные координаты центров кластеров выбираются случайным образом (рис.1)
3. Основная идея – минимизировать целевую функцию $\sum_{i=1}^{n} {d_i}^2$, где n – число объектов в кластере, а d_i – расстояние между i-ым объектом и центром кластера (рис.2)
4. На каждой итерации d – центр кластера – сдвигается в центр масс (точку, каждая координата которой – среднее соответствующих координат объектов кластера) (рис.3)

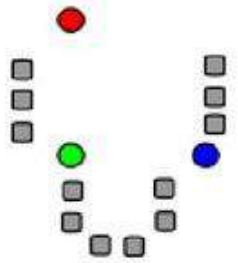


Рис.1

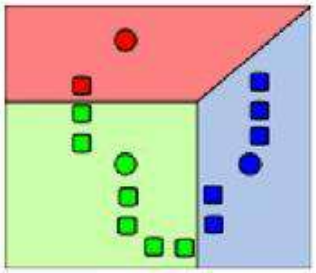


Рис.2

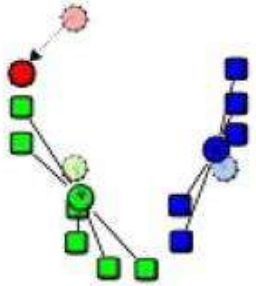
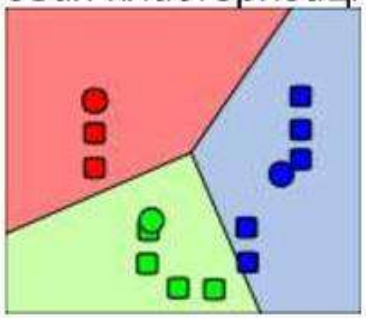
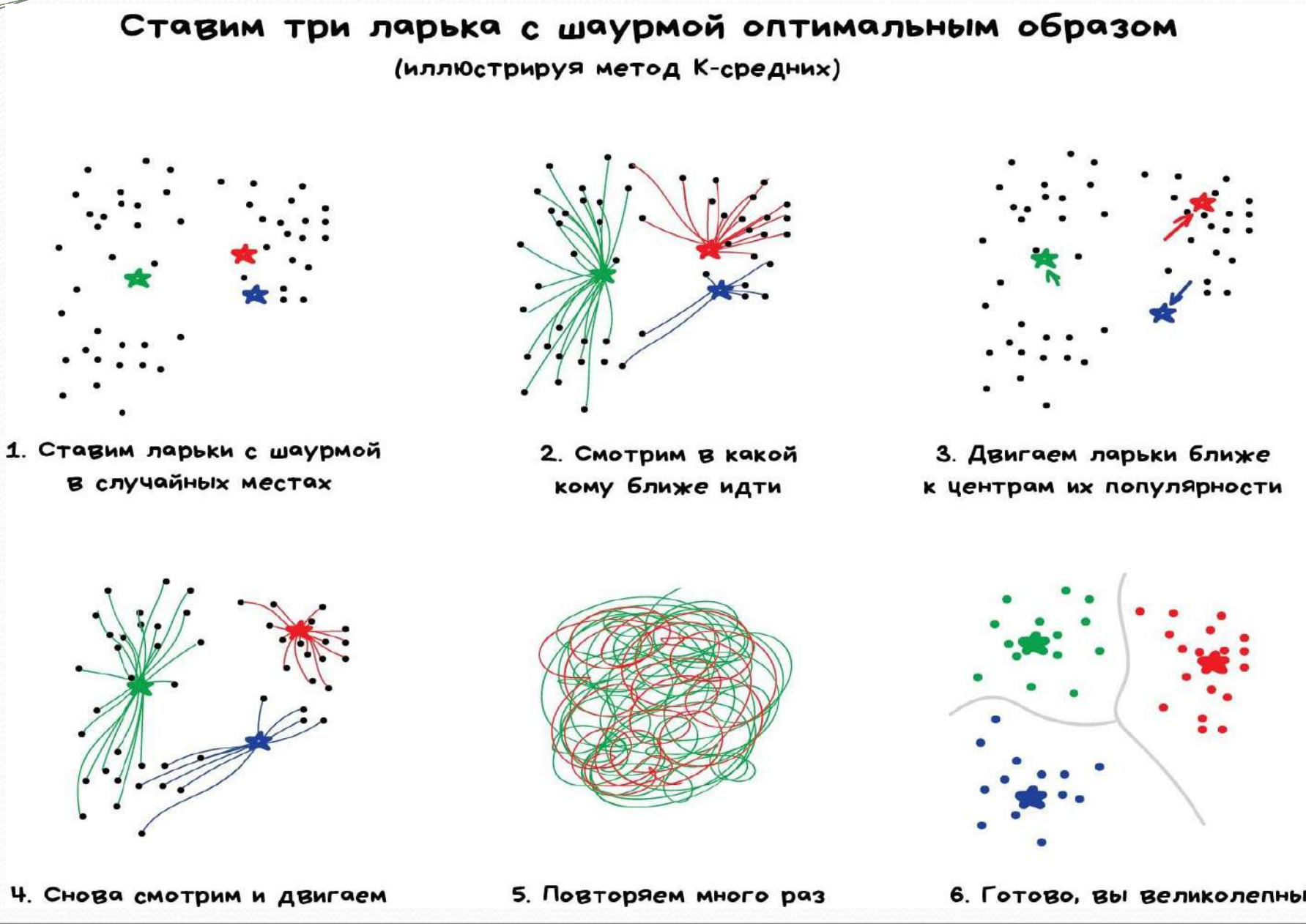


Рис.3

итоговая кластеризация



Кластеризация методом к-средних (*пример*)



Особенности применения алгоритма k-средних

Алгоритм **k-средних** недетерминированный - разные начальные позиции кластерных центров, вероятно, будут давать и разные кластеры.

Поэтому алгоритм обычно запускается несколько раз, а затем результаты этих прогонов сравниваются, чтобы увидеть, какие кластеры выглядят наиболее адекватными с учетом предметной области и ее понимания специалистом по данным.

В алгоритме **k-средних** нет обязательного условия, что все кластеры должны быть одного размера.

Кластеризации может применяться к большинству типов данных. Например, для анализа учебных курсов с целью выявления групп студентов, которые предпочитают разные методы обучения; для идентификации групп похожих документов в корпусе текстов; в биоинформатике для анализа последовательностей генов в процессе, называемом микрочиповым анализом.

Области применения кластеризации

- Анализ данных (Data mining)
  - Упрощение работы с информацией
  - Визуализация данных
- Группировка и распознавание объектов
  - Распознавание образов
  - Группировка объектов
- Извлечение и поиск информации
  - Построение удобных классификаторов

Задача обнаружения аномалий в данных

Обнаружение аномалий (анализ выбросов) включает в себя поиск и выявление объектов, которые не соответствуют типичным данным в наборе. Эти несоответствующие объекты называют аномалиями или выбросами.

Обнаружение аномалий используется сейчас при анализе финансовых транзакций с целью выявления мошенничества и запуска расследований. Например, оно позволяет определить мошеннические действия по кредитным картам, выявляя транзакции, происходящие в необычном месте или на необычно большую сумму по сравнению с другими транзакциями по этой кредитной карте.

Вначале эксперт определяет ряд правил, которые помогают идентифицировать аномальные события.

Затем эти правила описываются, например, на SQL и запускаются в базах или хранилищах данных.

Схема мошенничества с кредитными картами заключается в том, что вор проверяет, работает ли украденная карта, совершая по ней небольшую покупку, а затем, если транзакция проходит, как можно быстрее покупает что-нибудь дорогое, прежде чем карта будет аннулирована.

Программист БД пишет сценарии, которые выявляют последовательности транзакций по кредитной карте, соответствующие этой закономерности, и либо автоматически блокируют карту, либо предупреждают компанию-эмитента.

Подходы к обнаружению аномалий и области применения

Основным недостатком подхода, основанного на правилах, является то, что он может идентифицировать аномальные события только после того, как они произошли и попали в поле внимания. В идеале желательно выявлять аномалии, когда они происходят впервые.

Обнаружение аномалий является противоположностью кластеризации. Цель кластеризации найти группы схожих элементов, цель обнаружения аномалий – поиск элементов, непохожих на остальную часть набора данных.

Существует два подхода к обнаружению аномалий:

- группировка нормальных данные вместе, а аномальные помещаются в отдельные кластеры;
- измеряется расстояние между объектом и центром кластера. Чем дальше объект находится от центра кластера, тем выше вероятность того, что он окажется аномальным и требует расследования.

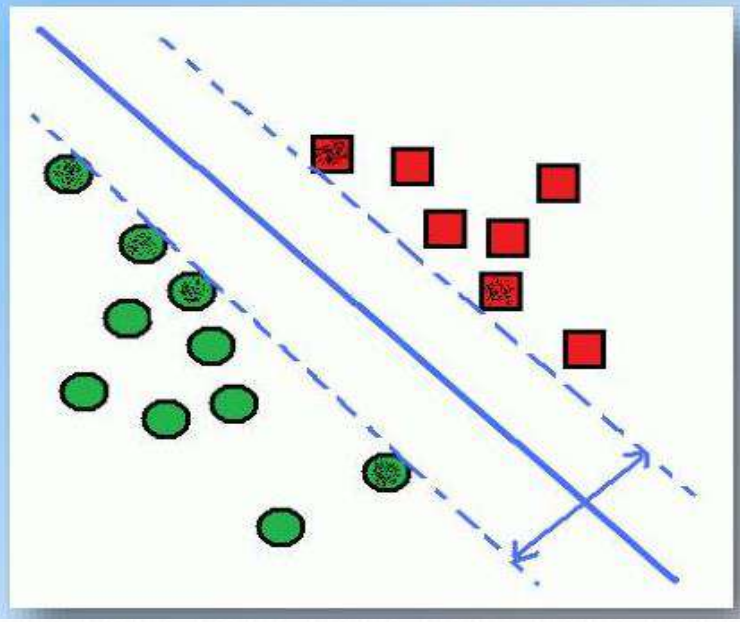
Основные области применения задачи обнаружения аномалий:

- мониторинг финансовых транзакций;
- анализ страховых претензий для выявления нетипичных;
- обнаружение возможных взломов или нетипичного поведения сотрудников в сети;
- в медицине выявление аномалий в историях болезней пациентов;
- в интернете вещей (IoT) обнаружение аномалий при мониторинге ситуаций.

Выявление аномалий: метод опорных векторов (SVM-метод)

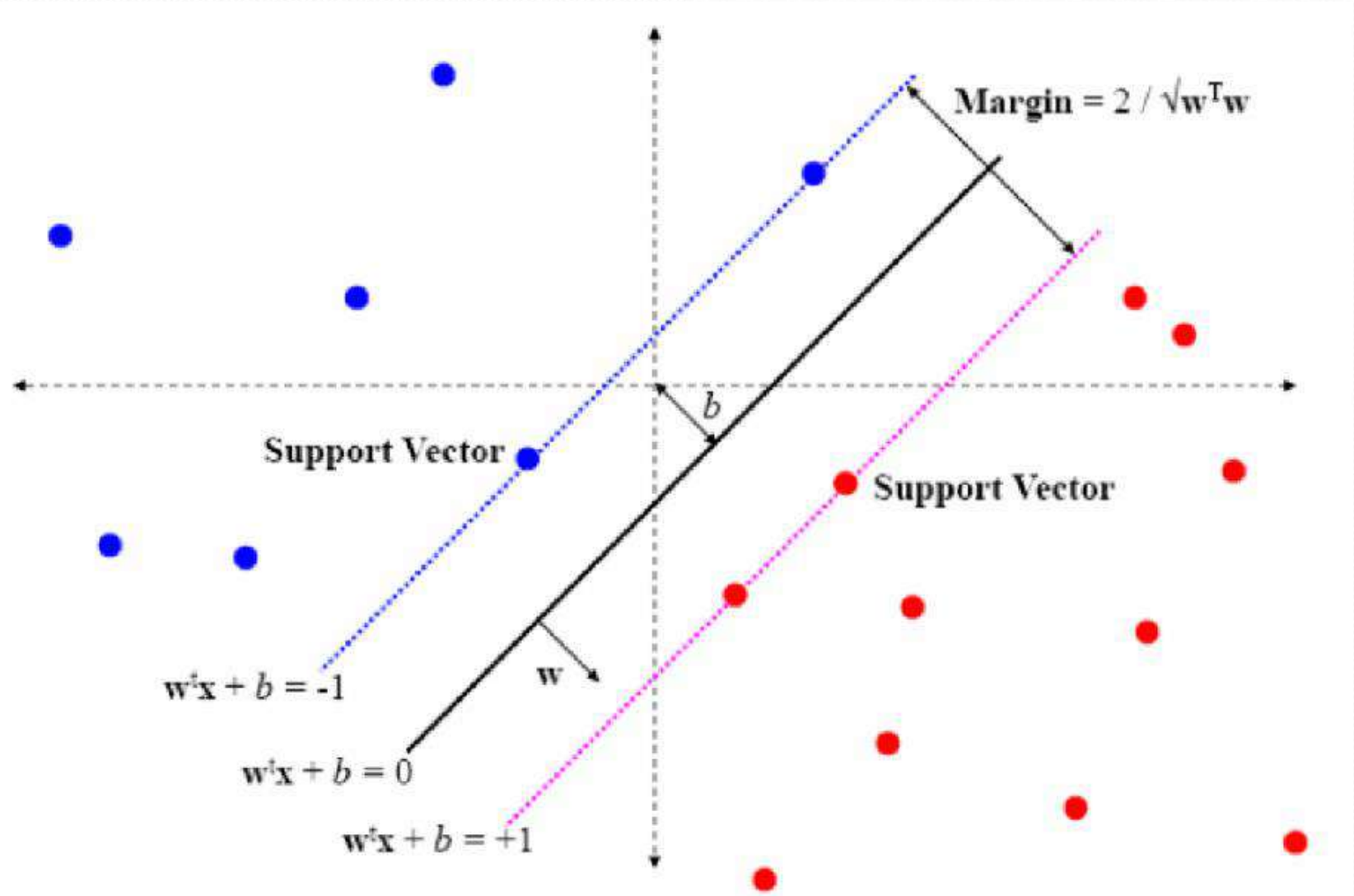
Идея метода. Анализирует данные как один кластер и выявляет основные характеристики и ожидаемое поведение объектов. Затем алгоритм маркирует каждый объект, чтобы указать, насколько он похож или отличен от основных характеристик и ожидаемого поведения. С помощью этой информации выявляют аномалии. Чем больше объект не похож на остальные, тем выше необходимость его исследования.

- Нахождение плоскости, разделяющей два множества объектов
- Опорные вектора - объекты множества, лежащие на границах областей



- Область между границами должна быть как можно больше
- Достаточно небольшого набора данных
- Используется не все множество образцов для классификации

SVM-метод *(иллюстрация)*



Задача поиска ассоциативных правил

Поиск ассоциативных правил (АП) – это метод анализа данных при обучении без учителя. Ассоциативное правило: «**Из события А следует событие В**».

Идея. Найти группу элементов, часто встречающихся вместе.

Например, для такого анализа данных бизнес отслеживает корзину товаров каждого покупателя при каждом посещении магазина. При поиске АП каждая строка в наборе данных описывает содержимое корзины, оплаченной конкретным покупателем в конкретное время. Атрибуты в этом наборе данных – приобретенные товары. На основе данных алгоритм поиска ассоциативных правил ищет товары, которые встречаются в каждой корзине.

В отличие от кластеризации и обнаружения аномалий, которые фокусируются на выявлении сходств или различий между объектами (или строками) в наборе данных, поиск АП фокусируется на рассмотрении связей между атрибутами (или столбцами) в наборе данных. Этот тип анализа ищет корреляции между продуктами.

Основные характеристики АП: *поддержка, достоверность* и *улучшение*.
Поддержка показывает какой процент транзакций поддерживает данное правило.
Достоверность показывает какова вероятность того, что из **А** следует **В**.
Улучшение показывает полезнее ли правило, нежели случайное угадывание.

Алгоритм Apriori: создание ассоциативных правил

(1) Найти все комбинации объектов (например, товаров в наборе транзакций), которые случаются с заданной минимальной частотой. Эти комбинации называются *частыми предметными наборами*.

(2) Рассчитать правила, которые отражают совместное вхождение товаров в частые предметные наборы:

ЕСЛИ {предпосылка} – ТО {следствие}.

Алгоритм **Apriori** вычисляет вероятность появления элемента в частом предметном наборе с учетом присутствия в нем других предметов. Например, правило, выведенное из частых предмет наборов, содержащих предметы A, B и C:

ЕСЛИ {хот-доги, кетчуп} – ТО {пиво}.

P.S. *Один из супермаркетов в 1980-х гг. использовал алгоритм* ***Apriori*** *и выявил неожиданную ассоциацию клиентов, покупающих вместе подгузники и пиво: семьи с маленькими детьми готовились к уик-энду и знали, что им нужно запастись подгузниками и купить пиво, чтобы дома было что выпить. Магазин разместил эти два товара рядом, и продажи выросли.*

Особенности алгоритма Apriori

Даже небольшой набор данных может содержать большое количество АП. Алгоритм ограничивается только правилами, которые имеют высокие значения поддержки и достоверности. Тривиальные правила не принимаются во внимание.

После сокращения набора правил *аналитик данных* анализирует оставшиеся и дает рекомендации для сайтов, рекламы в магазинах, рассылок, перекрестных продаж и т.д.

Алгоритм работает эффективнее, если, например, корзины товаров связаны с демографическими данными клиента. По этой причине многие ритейлеры используют программы лояльности.

Например, АП, учитывающего демографические данные:

ЕСЛИ **[пол (мужской) & возраст (<35) & {хот-доги, кетчуп}]**

ТО **{пиво}[***поддержка* **= 2 %,** *доверие* **= 90 %]**

Пример алгоритма Apriori

TID	Приобретенные покупки
100	a, b, c
200	b, d
300	b, a, d, c
400	e, d
500	a, b, c, d
600	f

Формирование 1-элементных кандидатов

Itemset	Support
a	3
b	4
c	4
d	3
e	1
f	1

Часто встречающиеся 1-элементные наборы

Itemset	Support
A	3
B	4
C	4
D	3

Сканирование базы данных D

Формирование 2-элементных кандидатов

Itemset
ab
Ac
ad
Bc
Bd
Cd

Подсчет 2-элементных кандидатов

Itemset	Support
ab	3
ac	3
ad	2
bc	2
bd	3
cd	2

Часто встречающиеся 2-элементные наборы

Itemset	Support
ab	3
ac	3
bd	3

Формирование 3-элементных кандидатов

Itemset
abc
abd
bcd
acd

Подсчет 3-элементных кандидатов

Itemset	Support
abc	3
abd	2
bcd	2
acd	2

Min_sup=3

Часто встречающиеся 3-элементные наборы

Itemset	Support
abc	3

Формирование 3-элементных кандидатов

Itemset
abc

Часто встречающиеся 3-элементные наборы

Itemset	Support
abc	3

Min_sup=3

Часто встречающиеся наборы

Itemset	Support
abc	3

Задача прогнозирования оттока клиентов

Стандартной бизнес-задачей в сфере управления взаимоотношениями с клиентами является оценка вероятности того, что отдельный клиент предпримет какое-либо действие. Привлечение нового клиента обходится в **пять-шесть раз дороже**, чем удержание постоянного.

Удержание клиентов за счет снижения тарифов для всех и замены старых телефонов на новые – неподходящий вариант. Цель - идентифицировать клиента, который **собирается сменить оператора**, и попытаться убедить его остаться, предлагая новый телефон взамен старого или выгодный тарифный план.

Проблема выявления клиентов, которые могут уйти в ближайшем будущем, называется **прогнозированием оттока**. Задача прогнозирования состоит в том, чтобы **классифицировать** клиента, подпадает он под риск оттока или нет.

Классификация – это метод машинного обучения с учителем, в ходе которого берется набор данных с помеченными экземплярами и строится модель классификации. Помеченный набор данных называется обучающим. Он состоит из объектов, целевой результат которых уже известен.

После создания помеченного набора данных начинается построение модели классификации с использованием алгоритма машинного обучения.

Пример задачи прогнозирования оттока клиентов

Задача ценового прогнозирования: регрессионная модель

Ценовое прогнозирование – это задача оценки стоимости товара в определенный момент времени. Востребовано любым, кто рассматривает возможность покупки товара.

Ценовое прогнозирование включает в себя оценку значения непрерывного атрибута. Решается как проблема регрессии. Проблема регрессии похожа на проблему классификации – предполагается построение модели, которая может предсказать будущее значение на основании набора входных атрибутов. Отличие в том, что классификация оценивает значения категориального атрибута, а регрессия – значения непрерывного. Регрессионный анализ требует набора данных, в котором указано значение целевого атрибута для каждого из объектов.

Структура регрессионных моделей прогнозирования цены одинакова независимо от товара – меняется только имя и количество атрибутов.

При наличии соответствующих данных алгоритм регрессии определяет, какое влияние каждый из атрибутов оказывает на окончательную цену.

Конфиденциальность и этика

(1) Самый большой вопрос, стоящий сегодня перед наукой о данных, – как найти баланс между свободой частной жизни отдельных лиц, безопасностью и интересами всего общества.

(2) Что считать разумными способами использования персональных данных в таких контекстах, как борьба с терроризмом, улучшение медицины, борьба с преступностью, выявление мошенничества, оценка кредитного риска, страхование и таргетированная реклама?

(3) Наука о данных – это палка о двух концах. Правда в том, что алгоритмы науки о данных скорее аморальны, чем объективны.

(4) Может возникнуть ощущение, что любая проблема может быть решена с использованием технологий анализа данных. – Нужен лишь достаточный объем корректных данных.

(5) Звучат призывы объединить данные из разных источников с целью, например, поддержки исследований в области здравоохранения или удобства для государства и граждан. Жизнь в таком обществе превратит нас из свободных граждан в заключенных паноптикума (музей, коллекция разнообразных необычайных предметов).

Конфиденциальность и этика *(продолжен*

(6) Внедрение видеонаблюдения означает, что данные о человеке могут собираться в любое время и где бы он ни был – на улице, в магазине, на парковке, не говоря уже о возможности отслеживания мобильных телефонов.

(7) Реальные примеры сбора данных: учет покупок по кредитным картам, снятие наличных в банкоматах, звонки по мобильному телефону и проч.

В интернете данные о людях собираются, когда они посещают сайты или входят в систему, отправляют e-mail, совершают онлайн-покупки, назначают даты, пользуются устройством для чтения электронных книг, смотрят лекцию на открытых онлайн-курсах или публикуют что-то в социальной сети. Собранные вместе точечные данные и определяют **цифровой след человека**.

(8) Персональные данные собираются двумя способами, и оба являются проблематичными с точки зрения конфиденциальности:

- данные могут собираться без ведома человека («**цифровая тень**»);
- человек решил оставить свои данные («**цифровой след**»), но он не знает, как они будут использованы, будут ли переданы третьим сторонам.

Конфиденциальность и этика *(продолжение)*

(9) Мощь современных методов выявления закономерностей такова, что даже собранные с ведома и согласия человека они могут иметь для него негативные и непредсказуемые последствия. Методы науки о данных способны на основе открытой информации, которую мы охотно публикуем, например, в социальных сетях, вывести другую информацию, сугубо личную, которой мы делиться не планировали.

(10) Пример. Многие пользователи **Facebook** ставят лайки потому, что хотят продемонстрировать поддержку. Модели, разработанные **Facebook**, могут довольно точно предсказывать вашу сексуальную ориентацию, политические и религиозные взгляды, умственные способности, особенности личности, склонность к употреблению алкоголя, наркотиков и сигарет.

Примеры неконтекстных связей, установленных этими моделями:

- поддержка кампании за права человека как предиктор гомосексуальности (и для мужчин, и для женщин);
- симпатия к бренду Honda как вероятный признак того, что человек не курит.

Вычислительные методы и законы сохранения конфиденциальности

Наиболее известны два метода: **дифференциальная приватность** и **федеративное машинное обучение**.

Дифференциальная приватность – это математический метод получения полезной информации о населении, без изучения отдельных людей. В основе лежит идея добавления шума на этапе сбора данных,

Пример. Шум вводится в данные методом случайного ответа. При анкетировании респондентам предлагается ответить «Да» или «Нет» на какой-нибудь деликатный вопрос (касающийся нарушения закона, состояния здоровья и т.д.), используя следующую процедуру: 1. Подбросьте монету и держите результат в секрете. 2. Если выпал орел, отвечайте «Да». 3. Если выпала решка, отвечайте правдиво.

Половина, которым выпадет орел, отвечает «Нет», другая половина ответит правдиво. Число, ответивших «Нет» в общей численности приблизительно вдвое превысит количество правдивых ответов «Нет». Зная это, можно вычислить истинное число ответов «Да». Однако, невозможно определить, для кого конкретно из респондентов это условие выполняется.

Apple внедрила дифференциальную приватность в **iOS 10**, чтобы защитить конфиденциальность отдельных пользователей, сохраняя возможность выявлять закономерности в данных.

Федеративное машинное обучение — это методика заключения модели в защищенную среду и ее обучение без перемещения данных куда-либо.

Этика в науке о данных

Анкетирование интернет-пользователей показывает:

- более половины опрошенных заявляют, что не знают, как ограничить информацию, собираемую о них сайтами,
- им не нужна целевая реклама, потому что она отслеживает и анализирует их поведение в интернете,
- обеспокоены тем, что их информация используется для целей, отличных от той, для которой она была собрана.

Интересанты: рекламные, страховые и интернет-компании, спецслужбы, органы полиции, правительства, медицинские и социальные организации. У них разные взгляды на проблему конфиденциальности данных.

Лучше всего действовать консервативно и этично. Если мы разрабатываем новые решения в области данных, то должны учитывать и этические вопросы.

Наверное, это не просто. Представьте себе, что вы работаете в важном проекте специалистом по данным. Анализируя данные, вы обнаружили ряд зависимых атрибутов, которые в совокупности оказались связаны с расой, религией, сексуальной ориентацией или иным конфиденциальным признаком. Вы знаете, что по закону не можете использовать расовый атрибут в вашей модели. Вы также полагаете, что включение этих атрибутов в модель заставит ее работать.

Спросите себя: «**Что я буду делать?**»

Заключение

Люди всегда пытались понять мир, выявляя закономерности в собственном опыте. Наука о данных – воплощение этого поиска.

Слова «точный», «умный», «интеллектуальный», «персонализированный» являются частью отраслевых названий науки о данных: *точная медицина*, *умный город*, *интеллектуальный транспорт*, *целевая реклама*, *персонализированные развлечения*. Все эти сферы человеческой жизни объединяет **необходимость принятия решений**. - Как лечить пациента? - Сколько школ необходимо построить в ближайшие четыре года? - Какой фильм или книгу порекомендовать?

Наука о данных помогает принимать такие решения.

Наука о данных в ее современном виде представляет собой смесь больших данных, компьютерных мощностей и человеческой изобретательности в целом ряде технологических областей (от глубинного анализа данных и исследования баз данных до машинного обучения).

Джинн уже выпущен из бутылки: наука о данных оказывает и будет оказывать существенное влияние на нашу повседневную жизнь.

В лекциях дан краткий обзор основных идей для понимания науки о данных.