Экспертные системы и базы знаний

Тема 1. Тест Тьюринга, основные постулаты ИИ. Нейроинформатика. (4 часа)

Постулат нейроинформатики: Единственный объект, способный мыслить — это человеческий мозг. Поэтому любое «мыслящее» устройство должно каким-либо образом воспроизводить структуру человеческого мозга

Постулат кибернетики черного ящика (постулат логико-символьных систем): Для искусственного интеллекта не важно, как устроено «мыслящее» устройство. Главное, чтобы на заданные воздействия оно реагировало бы так же, как и мозг человека. По сути, это постулат А. Тьюринга, который считал, что все проблемы, решаемые людьми, могут быть сведены к набору алгоритмов.

Научные школы в ИИ:

Опираясь на постулаты в Искусственный Интеллект включены две научные школы: конвенциональный (традиционный) ИИ и вычислительный интеллект.

1. ***Конвенциональный (традиционный) ИИ***. Основанные на машинном обучении, формальной логике и статистическом анализе, подходы конвенционального ИИ включают:

• Экспертную систему (программы основаны на базе знаний и механизме рассуждений, по определенным правилам обрабатывают информацию и делают соответствующие выводы);

• рассуждения на основе аналогий;

• Байесовские сети (статистические методы анализа данных);

• поведенческие модели (включая программы-агенты, поведение которых зависит от изменений во внешней среде).

2. ***Вычислительный ИИ.*** Основанные на поэтапной разработке и обучении систем на не символьных эмпирических данных и «мягких» вычислениях, методы вычислительного ИИ включают:

• Нейросети;

• Нечеткие системы (рассуждения в условиях НЕ-факторов);

• Эволюционные вычисления (эволюционные алгоритмы, разновидности Монте-Карло, гармоничный поиск, роевые алгоритмы, меметика и т.д.);

• Гибридные системы (например, исходные экспертные правила генерируются нейросетью, а заключения - байесовской сетью).

Текущее состояние:

- Метод нейронных сетей очень привлекателен. Сложность связана с необходимостью переобучения сети, локальной оптимальностью решения, выбором соответствующей сетевой архитектуры, а самое главное, невозможно выяснить, как получается определенное решение проблемы. Методы нейронных сетей не подходят даже для моделирования мозга муравьев и пчел;

- Разработка гибридных моделей и алгоритмов для вывода знаний в условиях неопределенности в реальном времени, онтологий знаний, мультиагентных систем, роботов и игровых программ.

Наиболее значительные достижения в области теории:

- Формирование нейроматематики, нейроалгоритмов ритма и создание нейрокомпьютеров на их основе, разработка "мягких" вычислительных моделей мозга, связанных с восприятием, языком и знаниями, разработка эволюционных и квантовых алгоритмов обучения нейросетей, распознавание речи и роботы, соревнующиеся по футболу;

- Тест Тьюринга, модели и языки для представления знаний, разработка ЭС, автоматическое доказательство теорем, Deep Blue победил чемпиона мира по шахматам, Интернет-проект на сайте 20q.net по мотивам игры «20 вопросов».

Области практического применения:

-Система идентификации; бизнес-прогнозирование, экономическое, медицинское прогнозирование; управление космосом, робототехникой, металлургией, машиностроением и нефтегазовой промышленностью; решение сложных задач моделирования в режиме реального времени;

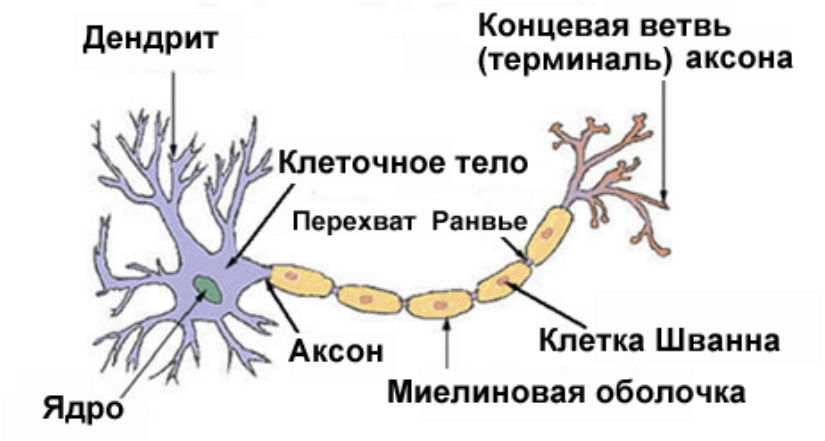
-Экспертные системы для медицины, геологии, бизнеса, экономики и промышленности; системы управления знаниями и принятия решений, системы обучения, системы интеллектуального анализа данных, семантический анализ и обработка информации на естественном языке; визуализация и интеллектуальность интерфейсов.

Нейроинформатика

Параметры мозга как системы обработки информации:

Количество нейронов мозга составляет около 10^{21}.

Структура биологического нейрона:



Объем коры: толщина 2-3 мм, площадь около 2200 $см^2$.

Потребляемая мощность: 1 – 25 Вт.

Нейрон имеет в среднем 10000 связей.

100 триллионов синапсов.

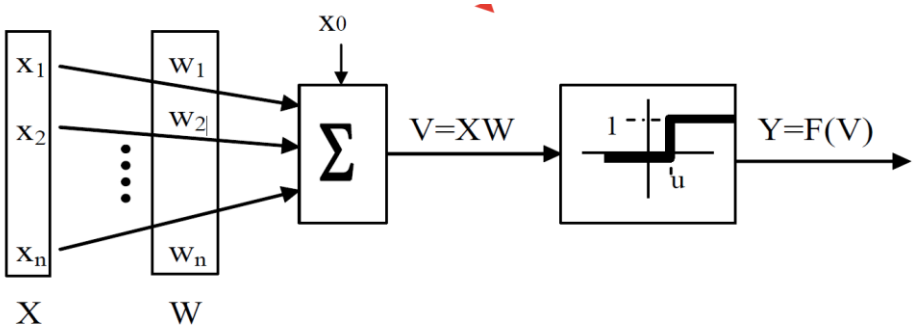
Огромные объемы информации мозг обрабатывает за доли секунды при том, что мозг – тихоход (время реакции нейрона несколько миллисекунд).

Объем передаваемой мозгом информации равен:

1012 (волокон)×103 (импульсов) = 1015.

В лучших ЭВМ: 10^3 (каналов)×10^9 (импульсов в сек) = 10^{12}, т.е. 1000 лучших ЭВМ равнозначны мозгу по производительности.

Нейроморфный чип IBM True North: 5,4 млрд. транзисторов (28nm), расход энергии 72 mW (примерно $400*10^9$ синаптических операций/с*W).



Искусственный нейрон МакКаллока - Питтса

На вход нейрона поступает некоторое множество сигналов x_j, j=1, 2... n, каждый из которых является выходом другого нейрона.

Нейрон вычисляет взвешенную сумму V=XW входных сигналов x_j и формирует на выходе сигнал величины 1, если эта сумма превышает определенный порог θ, и 0 - в противном случае.

Нейрон описывается математической моделью в виде уравнения

$$Y=F(V) = F(\Sigma x_j w_j > \theta),\ j = 1, 2,\ldots, n,$$

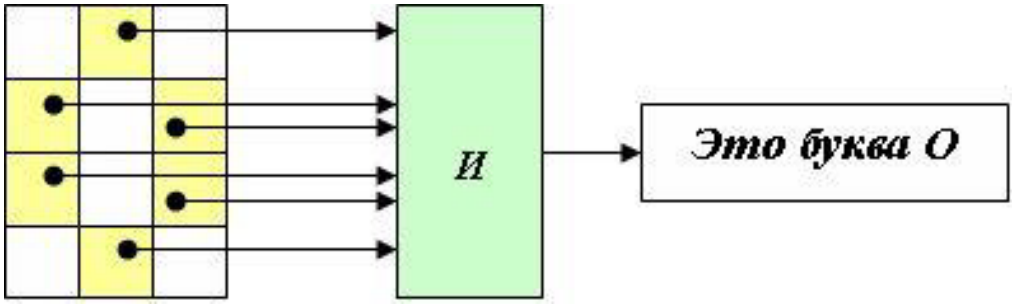
где Y – выходной сигнал нейрона, F – функция активации выхода нейрона, w_j – «вес» входа x_j, θ - пороговое значение.

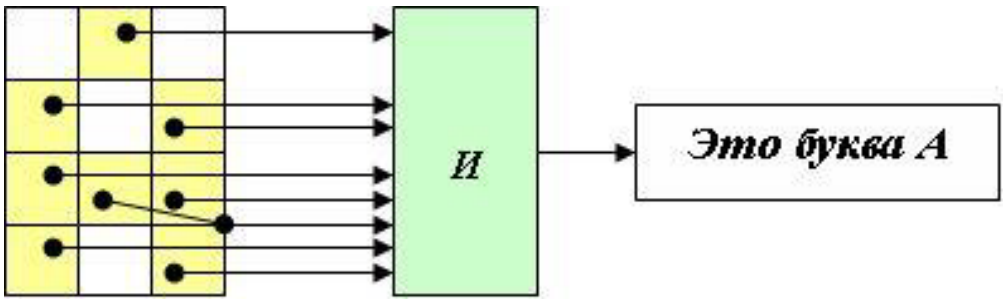
Добавив постоянный единичный вход $x_0=1$ и положив $w_0= - \theta$, получим:

$$Y= F(\Sigma xjwj + x_0),\ j=0, 1, 2,\ldots, n.$$

Нейронная сеть для распознавания символов

Представим экран, который стоит перед нами, он разбит на двенадцать клеток (4x3). Если мы сфокусируем изображение, то клетка либо засвечивается, либо нет. "Засветка" определяет единичное значение величины ее возбуждения, "не засветка" — нулевое. Так, буквы О и А определяют засветку клеток, как это показано на рисунках:





Можно задать вопрос- что же надо сделать, чтобы некоторый конструируемый нами прибор смог сказать, какая это буква?

Естественно, необходимо все сигналы возбуждения клеток экрана, засвечиваемые буквой О, подать на конъюнктор, который реализует схему И. Единичный сигнал на выходе конъюнктора, как мы можем видеть на рисунке, сформируется тогда и только тогда, когда засветятся все клетки экрана, на которое ложится изображение буквы О. Наличие единичного сигнала на выходе конъюнктора и определит ответ: "Это буква О".

То же необходимо сделать и для буквы А.

Пометим каждую клетку экрана ее координатами. Тогда на языке математической логики сделанное нами можно записать в виде логических высказываний — предикатов:

(1,2)& (2,1)& (2,3)& (3,1)& (3,3)& (4,2) → ***О***;
(1,2)& (2,1)& (2,3)& (3,1)& (3,2)& (3,3)& (4,1)& (4,3)→ ***А***.

Буквы не будут "мешать" друг другу, так как засветка соответствующих им клеток экрана частично не совпадает, и единичное значение конъюнкции определится только для одной из них.

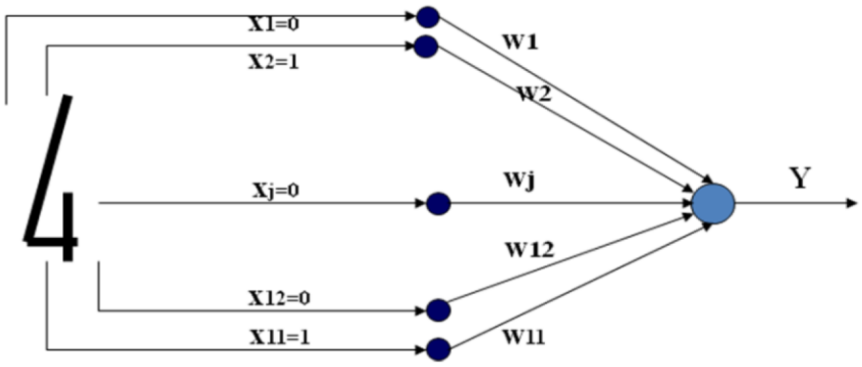
А если, предположить, что на экран мы можем подать букву ***К***? Тогда ни один из двух конъюнкторов не выработает единичное значение, так как не произойдет полное совпадение засветки соответствующих им клеток экрана. Чтобы "научить" систему букве ***К***, необходимо ввести еще один конъюнктор и проделать те же построения, что делались до этого. Основываясь на этом, мы можем утверждать, что построили систему распознавания двух "правильно" заданных букв.

Если буквы на экране пишутся дрожащей рукой, что делать в таком случае? В этом случае, нам необходимо разрешить альтернативную засветку каких-либо соседних клеток экрана и учитывать это с помощью операции дизъюнкции, ИЛИ. Например, можно построить предикат для распознавания буквы ***О***, допустив, например, возможность засветки клеток (1,1), (1,3), (4,1), (4,3). Аналогично, для буквы ***А*** можно допустить засветку клеток (1,1) и (1,3) и построить соответствующий предикат.

Объединив оба предиката, можно получить систему распознавания двух букв ***О*** и ***А***.

Персептрон Розенблатта и правило Хебба

Идея нейрона Мак-Каллоха–Питса была материализована Розенблаттом вначале в виде компьютерной программы, а затем в виде электронного устройства-персептрона, моделирующего глаз, который обучался распознаванию символов. Например, персептрон на 12 фотоэлементах для различения четных и нечетных цифр имеет вид:

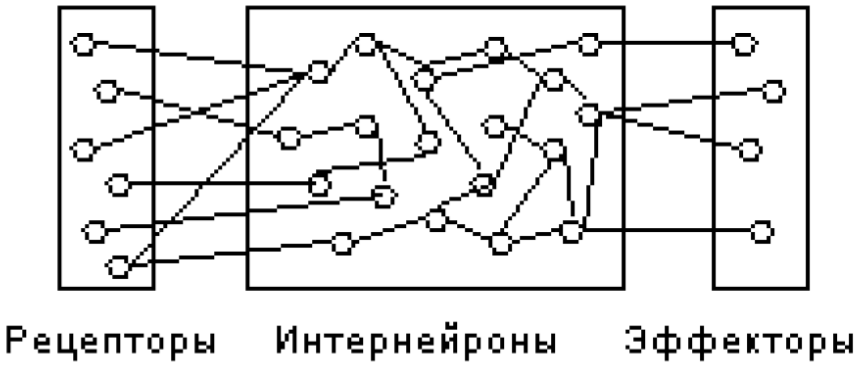


На матрицу накладывается карточка с изображением цифры 4. Фотоэлемент срабатывает (сигнал $x_j=1$), если на него попадает фрагмент цифры, иначе – $x_j=0$. Цель обучения: y=1, если на карточке четная цифра, y = 0 – в противном случае. Она достигается путем корректировки весов wj по правилам Хебба:

- если выход неправильный и равен 0, то увеличить веса для $x_j=1$;
- если выход неправильный и равен 1, то уменьшить веса для $x_j=1$.

Теорема (о сходимости персептрона): Если существует множество значений весов, обеспечивающих распознавание образов, то в конечном итоге алгоритм его обучения приведет к этому множеству.

Понятно, что пока никто не знает, как на самом деле функционирует мозг, как он строит связи внутри себя, как происходит процесс обучения нейтронной сети мозга? Есть лишь общее представление об этом:



Нейронов случайное число и соединены они случайно. Необходимо построить алгоритм соединения, после которого модель будет действовать целесообразно. Персептрон является лишь простейшей попыткой моделирования мозга.

Круг задач которые решает персептрон значительно увеличивается, если выходные сигналы будут не только 0 или 1, а могут принимать непрерывные

значения. Использование различных функций активации позволяет вносить нелинейность в работу нейрона и нейросети. На практике и в теоретических исследованиях используются различные функции активации: пороговая (функция Хевисайда), линейная, логистическая, гиперболический тангенс, радиально-базисная, экспонента, тригонометрический синус, модульная, квадратичная.

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.

У пороговой и линейной активационных функций есть свои недостатки. Ими можно назвать 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. Использование сигмоидальных функций, напротив, позволило перейти от бинарных выходов нейрона к аналоговым. Нейроны с сигмоидальными функциями, чаще всего используются в скрытых (внутренних) слоях многослойных нейросетей. Достоинством логистической функции является простота её производной:

$$\frac{dY(x)}{dx} = tY(x)(1 - Y(x)).$$

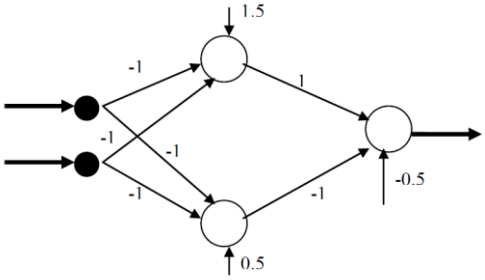
Это привело к новым подходам к обучению персептронов: минимизировать среднеквадратичную ошибку

$$\varepsilon = \frac{1}{2}\sum_i (Y_{i\ \textit{эталон}} - Y_{i\ \textit{факт}})^2.$$

Однако Минский и Пайперт доказали, что однослойные персептроны не могут решать многие простые задачи, например, реализовать логическую функцию «Исключающее ИЛИ» (XOR). Так появились многослойные

нейросети, в которых между слоями входных и выходных нейронов располагались нейроны скрытого слоя.

Проиллюстрируем это на примере. Рассмотрим модель многослойной нейросети следующего вида:



Модель содержит два входных элемента, два скрытых элемента и один выходной элемент. Все связи идут только в направлении от входного слоя к выходному. Скрытые элементы не получают данных от внешней среды непосредственно и не посылают данные непосредственно во внешнюю среду.

Функция XOR отображает пару двоичных входных сигналов:

Входы		Выход
x_1	x_2	XOR
1	1	0
1	0	1
0	1	1
0	0	0

В нашем случае значение комбинированных входов вычисляется по формуле

$$V_j = \sum_{i=0}^{n} x_i w_{ij},$$

а выходное значение нейрона есть результат применения пороговой функции:

$$Y(V_j) = \begin{cases} 1, & \text{если } V_j \geq 0, \\ 0, & \text{если } V_j < 0. \end{cases}$$

В качестве вводимых данных рассмотрим вначале пару $x_1=1$, $x_2=1$. Для первого скрытого элемента со смещением 1.5 получаем

$V=(x_0\times1.5)+(x_1\times(-1))+(x_2\times(-1)) = (1\times1.5)+(1\times(-1))+(1\times(-1))= -$ **0.5**.

Поэтому выходным значением элемента будет 0. Для второго скрытого элемента со смещением 0.5 получаем

$V=(x_0\times0.5)+(x_1\times(-1))+(x_2\times(-1)) = (1\times0.5)+(1\times(-1))+(1\times(-1))= -$ **1.5**.

Поэтому выходным значением будет 0. Если процедуру повторить для трех оставшихся пар входов, то можно убедиться в том, что вывод представленной выше сети соответствует функции XOR.

Основным алгоритмом обучения многослойных нейросетей является алгоритм обратного распространения ошибки (back propagation).

Его идея состоит в следующем. Мы знаем, что должно быть на выходе нейросети (по обучающей выборке) и постепенно вычисляем вход от слоя к слою по цепочке определенных формул. Реализуется как бы метод градиента. Задача стоит в следующем: найти все w_{ij}, то есть настроить веса всех связей так, чтобы нейросеть выдавала нужный выходной сигнал на соответствующий входной. Для настройки (обучения) нейросети на задачу необходимо реализовать множество итераций. Цель уменьшить ошибку ε до нуля. В результате находятся все лучшие значения w_{ij}.

Обучение происходит экспоненциально. Если ошибка E не приходит к нулю, то это означает, что сложности сети мало для обучения данному примеру (примерам), количество слоев или нейронов в слоях надо увеличить.

Общие признаки нейросетевой технологии следующие. Система начинает обнаруживать закономерности во входной информации. Система не знает, как она обучается - ей безразличен предмет рассуждений. Система легко доучивается и переучивается. Нейросеть можно "недокормить" примерами, но можно и «перекормить» (переобучить).

Оптимизация обучения нейросети.

Следовательно, существует две парадигмы обучения нейронных сетей — с учителем и без учителя. В первом случае входной вектор имеет готовые ответы, а во втором нейросеть обучается сама. Каждый вид обучения имеет свою область задач, и в целом они не пересекаются. В настоящее время изобретено и запатентовано большое количество архитектур нейросетей и методов их обучения. Но главное (исходное) - для обучения с учителем это "алгоритм обратного распространения ошибки", для обучения без учителя это алгоритмы Хебба и Кохонена. В последнее время появилась еще одна парадигма — обучение с подкреплением. В этой парадигме есть обучение с учителем и обучение без учителя.

Другие архитектуры для нейронных сетей.

В настоящее время не существует строгой теории выбора оптимального количества скрытых слоев для многослойной нейронной сети. Их способности математически не доказаны. Доказана эффективность гибридного алгоритма обучения, основанного на эволюционном алгоритме.

Кроме того, нейрофизиологи продемонстрировали, что существуют не только прямые, но и обратные связи между биологическими нейронами мозга, нейронные структуры способны к самообучению. Так появляются новые архитектуры: сети Хопфилда, сети Кохонена.

Нейропакеты, нейровычислители, нейрочипы

Сегодня в области проектирования нейросетей разработаны разнообразные нейропакеты (программные имитаторы), нейровычислители и нейрочипы.

Наиболее известными нейропакетами являются NeuNet Toolbox MatLab, Neuro Windows, Neuro Office, NNet+ (универсальные пакеты), ряд

специализированных нейропакетов для обработки изображений (ImageLib, Wisard NVision), распознавания образов, текстов и речи (NNetSheet-C, ICPAC, Propagator, Synaptics, NIntelligence Voc, AQIRE), управления динамическими объектами, обработки сигналов, финансового анализа, а также средства для проектирования гибридных нейросетевых ситем (G2, GURU, Nexpert Object, GeneHunter, Evolver, Genesis, Fuzzi Calc, Judgement Maker, Fuzzi Logic Toolbox Matlab).

Современный уровень теории ИНС, нейроматематики, схемотехники и микроэлектроники позволяет создавать на аппаратном уровне нейровычислители и нейрочипы. Они реализуются на заказных кристаллах, микроконтроллерах, программируемых логических интегральных схемах (ПЛИС), транспьютерах, цифровых сигнальных процессорах (DSP) или нейрочипах.

Нейрочип – это специализированная СБИС, ориентированная на реализацию нейросетевых алгоритмов (начало разработок – середина 90-х годов). По принципам построения, назначению и характеристикам они сильно отличаются друг от друга (аналоговые, цифровые, гибридные).

Недавно в США был создан прототип супернейрокомпьютера – системы обработки изображений с производительностью 80 pflops ($80*10^{15}$ операций с плавающей точкой в секунду) при объеме, равном мозгу, и потребляемой мощности 20 Вт.

Преимущества систем нейронных сетей

Резюмируем привлекательные черты распределенной обработки информации в нейросетевых системах:

- Параллелизм в обработке информации - глобальный характер связей между нейронами. До обучения эти связи произвольны и обычно малы. Обучение на примере «показывает» конкретную структуру сети для конкретной задачи;

- Единственным, но эффективным принципом обучения нейронных сетей является минимизация эмпирической ошибки с помощью обратного распространения ошибки в сети. Извне задается только цель обучения, т. е. способ определения ошибки по выходу сети. Более того, сеть постепенно изменяет свою конфигурацию, чтобы минимизировать эту ошибку, т.е. лучше справляться с поставленными перед ней задачами;

- Эксплуатационная надежность. Избыточность соединений приводит к тому, что значение каждого веса в отдельности неубедительно. Отключение ограниченного количества нейронов или отключение некоторых соединений серьезно не влияет на качество работы всей сети;

- Способность решать неформализованные задачи - проистекает из способности нейронных сетей самостоятельно разрабатывать очень сложные алгоритмы обработки данных, которые даже лучшие специалисты в данной области зачастую не в состоянии формализовать самостоятельно. Поэтому разработка нейронной сети стоит относительно дешево.

Почему, если у нейрокомпьютинга было так много преимуществ, но его начали активно использовать на практике только в 1990-х годах? Очевидно, что появление нейросетевых систем должно быть экономической необходимостью. 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, поднимает вопрос о человеко-машинном интерфейсе на ту высоту, которую он заслуживает. В результате растет интерес к задачам с искусственным интеллектом и нейронными сетями. Времена меняются. Вместо разрозненных персональных компьютеров появилась Сеть с ее неисчерпаемыми информационными ресурсами.

Тема 2. Данные – информация – знания. Свойства знаний, сходство/различие понятий. (4 часа).

Данные – информация – знания

Понятия "знание", "информация", "данные" часто отождествляются. В ИИ их необходимо различать. Взаимосвязь между понятиями «данные», «информация» и «знания» непростая. Для того чтобы уверенно оперировать понятиями "информация", "данные", "знание", необходимо не только понимать суть этих понятий, но и прочувствовать 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. Студент, который сдает экзамен по СИИ, нуждается в данных.
2. Студент, который сдает экзамен по СИИ, нуждается в информации.
3. Студент, который сдает экзамен по СИИ, нуждается в знаниях.

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

Во втором примере, очевидно, речь идет об информации в виде конспекта лекций или учебного пособия по СИИ.

Наиболее логичным выглядит третий вариант: студент при наличии соответствующих данных и информации, а также их использовании студент 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 упрощенном представлении данные можно рассматривать как полученные факты (сведения, сообщения, сигналы), характеризующие объект. Данные не равны информации. Станут ли данные информацией или нет, зависит от того, известны ли методы преобразования данных в известные понятия. Информация не является статическим объектом. Он динамичен и существует только в тот момент, когда взаимодействуют данные и методы их преобразования, то есть в процессе обработки информации. В остальное время он включается как данные. Данные объективны, методы субъективны, основаны на алгоритмах, придуманных людьми (испытуемыми).

Все остальное время она содержится в виде данных. Данные объективны, а методы субъективны, в их основе лежат алгоритмы, придуманные людьми (субъектами).

Информация

Прошло более 60 лет с тех пор, как Н. Винер в "Кибернетике" (1948 г.) 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.Э. Шеннон). Определений понятия информации достаточно много (известны более ста определений информации), но столь желаемого однозначного определения, устраивающего всех, так и не найдено.

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.

Мера Хартли. 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), а ***a*** - основание рассматриваемой системы.

Пример. Чтобы узнать положение точки в системе из двух клеток, т.е. получить некоторую информацию, необходимо задать 1 вопрос ("Левая или правая клетка?"). Узнав положение точки, мы увеличиваем суммарную информацию о системе на 1 бит ($I=\log_2 2$). Если система имеет n различных состояний, то максимальное количество информации равно $I=\log_2 n$.

Если измерение ведётся в экспоненциальной (натуральной) системе, то

$$k = 1\ ,\ I = \ln N \text{ (нат)};$$

Если измерение ведётся в двоичной системе, то

$$k = 1/\ln 2\ ,\ I = \log_2 N \text{ (бит)};$$

Если измерение ведётся в десятичной системе, то

$$k = 1/\ln 10\ ,\ I = \lg N \text{ (дит)}.$$

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 такова, чтобы информация была пропорциональна числу выборов.

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).

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.

Мера К. Шеннона. Формула 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***-е 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.

Формулы Хартли и Шеннона подтверждаются данными нейропсихологии.

Пример 1. Время *t* реакции испытуемого на выбор предмета из имеющихся *N* предметов линейно зависит от $\log_2 N$: $t=200+180\log_2 N$ *(мс)*. По аналогичному закону изменяется и время передачи информации в живом организме.

Пример 2. Один из опытов по определению психофизиологических реакций человека состоял в том, что перед испытуемым большое количество раз зажигалась одна из *n* лампочек, которую он должен указать. Оказалось, что среднее время, необходимое для правильного ответа испытуемого, пропорционально не числу *n* лампочек, а именно величине *I* определяемой по формуле Шеннона, где p_i - вероятность зажечь лампочку номер *i*.

Легко видеть, что в общем случае:

$$I=-\Sigma p_i \log_2 pi \le \log_2 N.$$

Если выбор *i*-го варианта предопределен заранее (выбора, собственно говоря, нет, $p_i=1$), то $I=0$.

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 ($H=Hmax$) и полной информационной неопределённости ($I=Imin$) до полного порядка ($H=Hmin$) и полной информационной определённости ($I=Imax$) в системе.

Но при этом информацию едва ли можно сравнивать с энтропией: принципиально необратимый процесс роста энтропии 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.

Большие количества материи и энергии управляются энергетически слабыми сигналами, несущими информацию в форме команд и данных. Такую информацию принято считать *функциональной*. Информация, хранимая (переносимая) информационным полем, представляет собой устойчивое свойство мира – его атрибут, и соответственно может быть названа *атрибутивной* информацией.

Сложность и многообразие информации в живой природе поражает воображение. Казалось бы, что принцип естественного отбора Ч.Дарвина в эволюции, дополненный случайностью мутаций, должен приводить именно к такому многообразию. Однако эпохальное открытие Э.Геккеля (1866), свидетельствует об ином. Зародыш высших живых существ, включая человека, повторяет в своём развитии все этапы эволюции живого в их хронологической последовательности!

Можно ли тогда сказать, что существует единая генеральная линия развития? Тогда почему сложность систем неуклонно увеличивается согласно закону роста информации?

Кроме того, существует огромный разрыв между ограниченной информационной ёмкостью гамет (половых клеток) организма и гигантским объёмом информации, необходимым для превращения зародыша во взрослую особь. Откуда берется эта информация?

Возникают также другие вопросы: как достигается фантастическая точность и устойчивость передачи сложной генетической информации от родителей к детям на протяжении тысячелетий? Где находится единая система контроля и штаб управления?

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, ценность информации относится к прагматическим отношениям между системой, информацией и целями системы. Ценность информации необходимо измерять, но по какой шкале: абсолютной или относительной? Практика показала, что во всей истории природы, жизни и разума ни одна информация не является абсолютной ценностью. Наоборот, очередной виток развития ставит новые цели, меняет эталон ценности, и то, что ранее казалось абсолютно ценным, приобретает статус относительной, мифотворческой ценности или вообще выбрасывается на свалку истории и науки. Можно ввести абсолютную меру ценности: считать, что информация, которая производит новую информацию с большей вероятностью, является более ценной. Но как оценить эту вероятность? Особенно сейчас, когда стало модным использовать так называемые оценки ценности научных работ- «Индекс цитирования» (чем больше раз цитируется работа, тем она ценнее и выше индекс). Но хорошо известно, что многие выдающиеся философы и ученые не издавали свои работы и «писали в стол» и стали известны миру только после своей смерти. Сократ был таким и, основываясь на свидетельствах своих учеников (Платона и др.), утверждал, что «письмо мертво». Это Г. Кавендиш, который занимался наукой вне официальной науки, в найденных после смерти бумагах были научные открытия и законы, которые известны науке в другом авторстве (закон Кулона и др.) Такими являются основатели многих религиозных учений и школ. Их учения обычно распространялись устно, подобно «модным словечкам» знаменитостей. Индекс цитирования не следует рассматривать как количественный показатель ценности научной информации. Накануне экзамена ценность информации о причуде учителя намного выше, чем в начале семестра, а ценность информации о своем учителе намного выше, чем чужого. Изменение целей изменяет и ценность информации, но не наоборот, поскольку ценность информации аксиологически вторична по отношению к целям, которые преследуют потребители информации.

Система ставит перед собой цели в каждый момент своего существования, она может достигать целей только за счет информации с необходимыми свойствами и является импульсом целенаправленной деятельности системы.. Информация ценна только в том случае, если она помогает достичь цели. Но в теории информации нет понятия о цели информационного процесса и его безопасности, из которого видно, что информация всегда выбирается из ложной информации и шума. Следовательно, метрика ценности, связанная с объемом информации, не предоставляют такой возможности. В том-то и проблема этой метрики, которая часто фатальна для системы — механизм отбора начинает работать, не дожидаясь решения о ценности полученной информации. Это происходит, например, в общественно-политических системах во времена бури и смуты. Стоит ли потом удивляться или кусать локоть, что сделанный выбор оказался неверным? В общем случае проблема сводится к оценке априорной

ценности информации. Не существует универсальной, важной априорной меры ценности информации.

Кодирование информации – обязательный подпроцесс любого информационного процесса. На каждом уровне иерархии систем возникает проблема выбора информационного кода. В современных компьютерах используется квазиоптимальный двоичный код, уступающий троичному коду по быстродействию (теоретически оптимален код значности e≈2,718281828459…). Оптимальный троичный код был реализован всего в двух компьютерах за всю историю вычислительной техники (СССР, Япония). Но троичный кодер оказался настолько ненадежным (в распознавании трех состояний), а троичная арифметика настолько не развитой, что от идеи троичного компьютера пришлось отказаться до лучших времен).

Знания

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, 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.

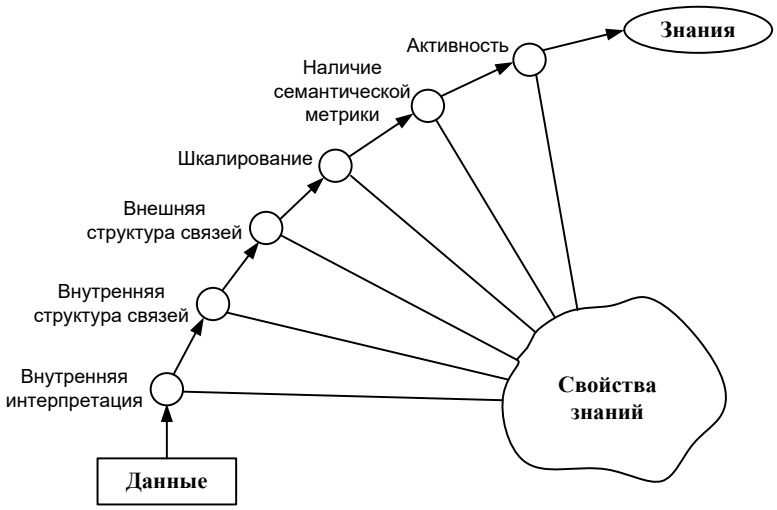
Несмотря на широкое распространение и использование понятия "знания" в различных научных дисциплинах и на практике, строгого определения данного термина нет.

Довольно часто используют так называемый *прагматический подход:* говорят, что *знания – это формализованная информация, на которую ссылаются и/или которую используют в процессе логического вывода.* Однако такое определение ограничено: оно фиксирует сознание на уже существующих методах представления знаний и, соответственно, механизмах вывода, не давая возможности представить себе другие ("новые").

Возможен и другой подход: попытаться на основе определения понятия *"данные"*, выявить их свойства и особенности, сформировать дополнительные требования к ним и уже затем перейти к понятию "знания".

Напомним, что данными называют формализованную информацию, пригодную для последующей обработки, хранения и передачи средствами автоматизации профессиональной деятельности.

Какие же свойства "превращают" данные в знания? На рис. 2.1 представлены шесть основных свойств знаний (часть из них присуща и данным).



2.1.Свойства знаний

Кратко охарактеризуем эти свойства.

- *Внутренняя интерпретация* (интерпретируемость). Это свойство предполагает, что в ЭВМ хранятся не только "собственно (сами) данные", но и "данные о данных", что позволяет содержательно их интерпретировать (см. рис. 1.3). Имея такую информацию, можно ответить на вопросы типа "Где находится НПО "Энергия"?" или "Какие предприятия выпускают космическую технику?". При этом в первой строке таблицы на рис. 2.2 находятся "данные о данных" (метаданные), а в остальных – сами данные.
- Наличие *внутренней структуры связей.* Предполагается, что в качестве информационных единиц используются не отдельные данные, а их упорядоченные определенными отношениями (родовидовыми, причинно-следственными и др.) структуры (эти отношения называют классифицирующими). Пример: факультет – курс – учебная группа – студент.

Предприятие	Место	Что выпускает
Завод им. Хруничева НПО "Энергия" НПО "Комета"	Москва Королев Москва	Космическую технику Космическую технику Конструкторскую документацию

Рис. 2.2. Внутренняя интерпретация

3. Наличие *внешней структуры связей.*

Внутренняя структура связей позволяет описывать отдельный объект (понятие). Однако объекты (понятия) способны находиться и в других отношениях (вступать в ситуативную связь). Пример: объекты "курс Государственного университета управления им. С. Орджоникидзе" и "урожай овощей в совхозе "Зареченский" могут находиться в ситуативной связи "принимает участие в уборке".

4. Возможность *шкалирования.*

Эта возможность предполагает введение соотношений между различными информационными единицами (т. е. их измерение в какой-либо шкале – порядковой, классификационной, метрической и т. п.) и упорядочение информационных единиц путем измерения интенсивности отношений и свойств. Пример: "97/ЭИ. 6-01 учебная группа занимает первое место на курсе по успеваемости".

5. Наличие *семантической метрики.* Шкалирование позволяет соотнести информационные единицы, но прежде всего для понятий, имеющих "количественное" толкование (характеристики). На практике довольно часто встречаются понятия, к которым не применимы количественные шкалы, но существует потребность в установлении их близости (например, понятия "искусственный интеллект" и "искусственный разум"). *Семантики* классифицируются следующим образом:

- *значение,* т. е. объективное содержание;
- *контекстуальный смысл,* определяемый связями данного понятия с другими, соседствующими в данной ситуации;
- *личностный смысл,* т. е. объективное значение, отраженное через систему взглядов эксперта;
- *прагматический смысл,* определяемый текущим знанием о конкретной ситуации (например, фраза "информация получена" может иметь как негативную, так и позитивную оценку – в зависимости от того, нужно это было или нет).

6. Наличие *активности.*

Данное свойство *принципиально отличает* понятие "знание" от понятия "данные". Например, знания человека, как правило, активны, поскольку ему свойственна познавательная активность (обнаружение противоречий в знаниях становится побудительной причиной их преодоления и появления новых знаний, стимулом активности является неполнота знаний, выражается в необходимости их пополнения). В отличие от данных, знания позволяют выводить (получать) новые знания. Будучи активными, знания позволяют человеку решать не только типовые, но и принципиально новые, нетрадиционные задачи.

Кроме перечисленных, знаниям присущи такие свойства, как *омонимия* (слово "коса" может иметь три смысла, связанных с определениями: девичья; песчаная; острая) и *синонимия* (знания "преподаватель читает лекцию" и

"студенты слушают лекцию" во многих случаях являются синонимами) и др. Классифицировать знания можно по самым различным основаниям.

По способу существования различают *факты* (хорошо известные обстоятельства) и *эвристики* (знания из опыта экспертов).

По *способу использования* в экспертных системах – *фактические знания* (факты) – знания типа "А – это А"; *правила* – знания для принятия решений ("Если... – то..."); *метазнания* (знания о знаниях – указывают системе способы использования знаний и определяют их свойства). Классическими примерами метазнаний являются народные пословицы и поговорки, каждая из которых характеризует знания (рекомендации по деятельности) в широком классе конкретных ситуаций (например, пословица "Семь раз отмерь, один – отрежь" применима не только в среде хирургов или портных).

По *формам представления* знания подразделяются на *декларативные* (факты в виде наборов структурированных данных) и *процедуралъные* (алгоритмы в виде процедур обработки фактов).

По *способу приобретения* знания бывают научные (полученные в ходе систематического обучения и/или изучения) и *житейские, бытовые* (полученные в "ходе жизни").

Дадим еще ряд определений, часто встречающихся в литературе.

Интенсиональные знания – знания, характеризующие или относящиеся к некоторому классу объектов.

Экстенсиональные знания – знания, относящиеся к конкретному объекту из какого-либо класса (факты, сведения, утверждения и т. д.)

Заметим: отношения интенсиональных и экстенсиональных знаний – это родовидовые отношения. Например, понятие "технологическая операция" – это интенсионал, а понятие "пайка" – это экстенсионал, так как пайка – одна из технологических операций. Очевидно, что эти *понятия относительны.* Так понятие "пайка", в свою очередь, можно считать интенсионалом по отношению к понятиям "пайка серебром" и "пайка оловом". Как правило, такого рода знания относятся к декларативным.

Физические знания – знания о реальном мире.

Ментальные знания – знания об отношениях объектов.

Мир задачи – совокупность знаний, используемых в задаче.

Мир пользователя – совокупность знаний пользователя.

Мир программы – совокупность знаний, используемых в программе.

Морфологические и синтаксические знания – знания о правилах построения структуры описываемого явления или объекта (например, правила написания букв, слов, предложений и др.).

Семантические знания – знания о смысле и значении описываемых явлений и объектов.

Прагматические знания – знания о практическом смысле описываемых объектов и явлений в конкретной ситуации. (Например, редкая монета для нумизмата и филателиста имеет различную прагматическую ценность.)

Предметные знания – знания о предметной области, объектах из этой области, их отношениях, действиях над ними и др.

3. Модели представления знаний. Задача вывода в базе знаний. (4 часа)

Для того *чтобы манипулировать всевозможными знаниями* из реального мира с помощью компьютера, необходимо осуществить их *моделирование.*

При проектировании модели представления знаний следует учесть *два требования:*

- *однородность представления;*
- простота *понимания.*

Выполнение этих требований позволяет упростить механизм логического вывода и процессы приобретения знаний и управления ими, однако, как правило, создателям интеллектуальной системы приходится идти на некоторый компромисс в стремлении обеспечить одинаковое понимание знаний и экспертами, и инженерами знаний, и пользователями.

Классификация методов моделирования знаний с точки зрения подхода к их представлению в ЭВМ показана на рис. 1.4.

Дадим общую характеристику основных методов представления знаний с помощью моделей, основанных на эвристическом подходе.

Рис. 3.1. Классификация моделей представления знаний

1. Представление знаний тройкой *"объект – атрибут – значение"* – один из первых методов моделирования знаний. Как правило, используется для представления *фактических знаний в простейших системах.*

Примеры:

Объект	Атрибут	Значение

Студент	Успеваемость	Отличник
Дом	Цвет	Белый
Пациент	Температура	Нормальная

Очевидно, что для моделирования знаний даже об одном объекте (например, о "студенте" или "доме") из предметной области *необходимо хранить значительное число "троек".*

2. Продукционная модель (модель правил; модель продукций – от англ, *production* – изготовление, выработка). В настоящее время наиболее проработанная и распространенная модель представления знаний, в частности – в экспертных системах.

Модель предусматривает разработку системы продукционных правил (правил продукций), имеющих вид:

ЕСЛИ A_1 И A_2 И ... A_n, ТО В1 ИЛИ B_2 ИЛИ...ИЛИ B_m, где Ai и Bj. – некоторые высказывания, к которым применены логические операции И и ИЛИ. Если высказывания в левой части правила (ее часто называют *антецедент* – условие, причина) истинно, истинно и высказывание в правой части *(консеквент* – следствие).

Полнота базы знаний (базы правил) определяет возможности системы по удовлетворению потребностей пользователей. Логический вывод в продукционных системах основан на построении прямой и обратной цепочек заключений, образуемых в результате последовательного просмотра левых и правых частей соответствующих правил, вплоть до получения окончательного заключения.

Пусть в некоторой области памяти хранятся следующие правила (суждения):

правило 1 – ЕСЛИ в стране происходит падение курса национальной валюты; ТО материальное положение населения ухудшается;

правило 2 – ЕСЛИ объемы производства в стране падают; ТО курс национальной валюты снижается;

правило 3 – ЕСЛИ материальное положение населения ухудшается; ТО уровень смертности в стране возрастает.

Если на вход системы поступит новый факт "В стране высокий уровень падения объемов производства", то из правил можно построить цепочку рассуждений и сформулировать два заключения:

факт 1 – правило 2 – правило 1 – заключение 1 – правило 3 – заключение 2,

где *заключение 1* (промежуточный вывод) – "Материальное положение населения ухудшается"; *заключение 2* (окончательный вывод) – "В стране возрастает уровень смертности".

В современных экспертных системах в базе знаний могут храниться тысячи правил, а коммерческая стоимость одного невыводимого (нового, дополнительного) правила очень высока.

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 ПС: 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.

Так что, если объект представляет собой небольшую задачу, он покажет только преимущества производственной системы. Базы данных необходимо структурировать при расширении знаний, решении сложных задач, выполнении гибких рассуждений или увеличении скорости рассуждений.

Кроме того, в конкретном случае факты не являются абсолютно верными и применение того или иного правила оправдано лишь частично. В производственных системах для этого используются термины «коэффициент достоверности», «обоснованность», «индекс уверенности» и т.п. В последние годы разработчики искали более формальную основу для работы с разнородными продуктами. Известны попытки использования нечеткой логики, описательной логики и других теорий. Эти попытки, однако, полностью отрицают тот факт, что оценки достоверности исходных фактов и правил в той или иной предметной области не являются достоверными по своей сути — они сделаны экспертами и часто ошибочны. Какой бы продуманной и взвешенной ни была реализация этой системы с элементами ИИ, на какой бы теоретической базе она ни базировалась, если исходный материал не совсем достоверен, найденные решения возможны, но не абсолютны!

В заключение отметим, что продукционные системы были исторически первым и до сих пор часто используемым способом представления знаний. Эффективная программная реализация продукционных правил лежит в основе многих практических программных систем; использование продукционных систем может быть полезным только в том случае, если все соответствующие ограничения и недостатки полностью понятны.

3. *Фреймовая модель.* Сравнительно новая модель представления знаний. Само понятие "фрейм" (англ, *frame* – рама, рамка, скелет, сгусток, сруб и т. д.) было введено в 1975 г. Марком Минским (M. Minsky, США).

Фрейм – это минимальная структура информации, необходимая для представления знаний о стереотипных классах объектов, явлений, ситуаций, процессов и др. С помощью фреймов можно моделировать знания о самых разнообразных объектах интересующей исследователя предметной области – важно лишь, чтобы эти объекты составляли класс концептуальных (повторяющихся: стереотипных) объектов, процессов и т. п. Примерами стереотипных жизненных ситуаций могут служить собрание, совещание; сдача

экзамена или зачета; защита курсовой работы и др. Примеры стереотипных бытовых ситуаций: отъезд в отпуск; встреча гостей; выбор телевизора; ремонт и др. Примеры стереотипных понятий: алгоритм; действие; методика и др. На рис. 3.2 представлен фрейм технологической операции "соединять".

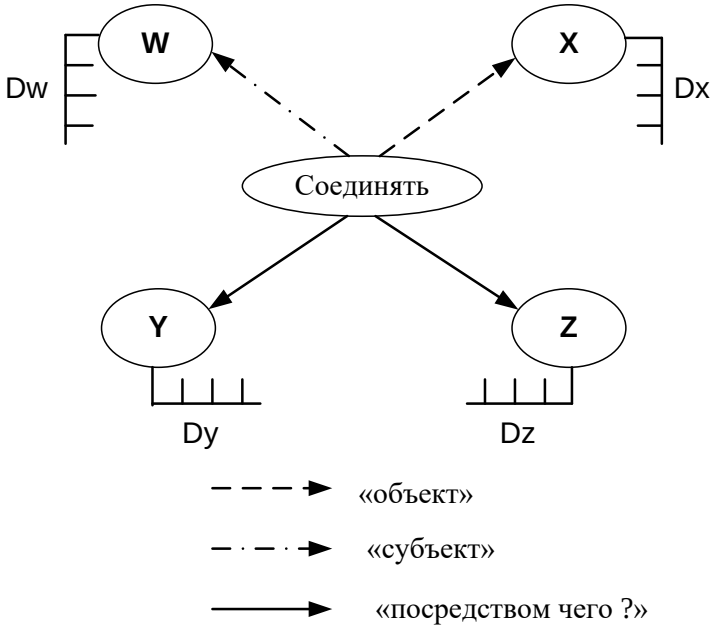


Рис.3.2. Фрейм ситуации «соединить»

Данный фрейм описывает ситуацию "Субъект *X* соединяет объект У с объектом Z способом W". На рисунке обозначены: вершины *X*, Y, Z, W – *слоты* (англ, *slot* – прорез; щель; пустота – составляющие фрейма); дуги – отношения; *Dx, Dy, Dz,* D_w – так называемые *шанции* – области возможных значений соответствующих слотов.

Наполняя слоты конкретным содержанием, можно получить фрейм конкретной ситуации, например; "Радиомонтажник соединяет микросхему с конденсатором способом пайки". Заполнение слотов шанциями называют *активизацией* фрейма,

С помощью фреймов можно моделировать как процедурные, так и декларативные знания. На рис. 3.2 представлен пример представления процедурных знаний. На рис. 3.3 приведен пример фрейма "технологическая операция", иллюстрирующий представление декларативных знаний для решения задачи проектирования технологического процесса.

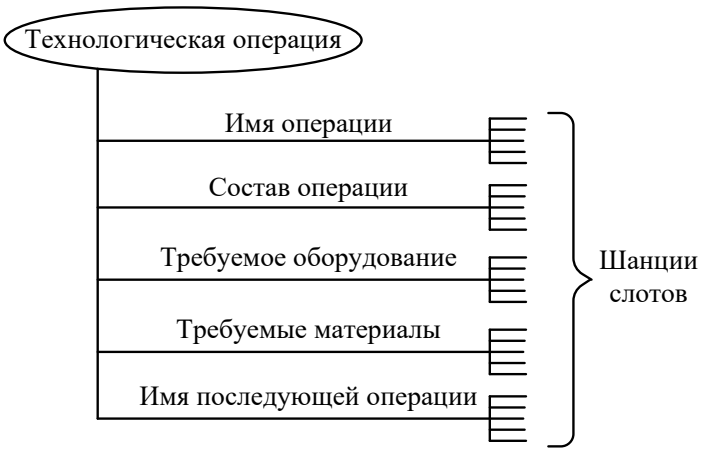


Рис. 3.3. Фрейм понятия "технологическая операция"

По содержательному смыслу фрейма выделяют:

- фреймы-понятия;
- фреймы-меню;
- фреймы с иерархически вложенной структурой.

Фрейм-понятие – это фрейм типа *И.* Например, фрейм "операция" содержит объединенные связкой *И* имена слотов "что делать", "что это дает", "как делать", "кто делает", "где делать" и т, д., а фрейм "предмет" – слоты с именами "назначение", "форма", "вес", "цвет" и т. д.

Фрейм-меню - это фрейм типа *ИЛИ.* Он служит для организации процедурных знаний с помощью оператора "выбрать". Например, фрейм "что делать" может состоять из объединенных связкой *ИЛИ* слотов "решить уравнение", "подставить данные", "уточнить задачу" и т, д., причем каждый из этих слотов может иметь несколько значений,

Фрейм с иерархически вложенной структурой предполагает, что в нем в качестве значений слотов можно использовать имена других фреймов, слотов и т, д., т, е, использовать иерархическую структуру, в которой комбинируются другие виды фреймов (в итоге получают так называемые фреймы-сценарии).

Значения слотов могут содержать ссылки на так называемые *присоединенные процедуры.* Различают два вида присоединенных процедур:

- Процедуры – демомы.
- Процедуры – слуги.

Процедуры-демоны присоединяются к слоту и активизируются при изменении информации в этом слоте (выполняют вспомогательные операции – например, автоматически корректируют информацию во всех других структурах, где используется значение данного слота) – см. рис. 3.4.

Процедуры-слуги активизируются при выполнении некоторых условий относительно содержимого слотов (часто по запросу). Данные процедуры определяются пользователем при создании фрейма. Например, во фрейме "Учебная аудитория" можно предусмотреть слоты "длина" и "ширина", а по их значениям вычислять значение слота "площадь".

Процедуры-демоны:

1	Процедура добавлено"	Выполняется, когда новая
2	Процедура удалено"	Выполняется, когда информация
3	Процедура нужно"	Выполняется, когда

Рис. 3.4. Типы присоединенных процедур

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.

4. *Модель семантической сети* (модель Куилиана).

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:

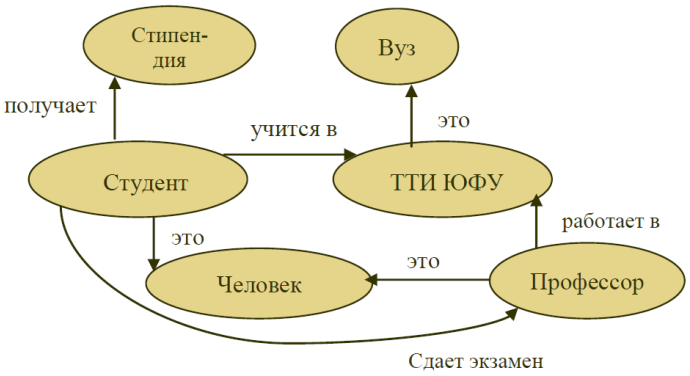


Рис.3.5.

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.

Была создана реальная система TLC, осуществлено моделирование человеческой памяти и разработана технологическая сторона концепции понимания смысла.

Таким образом, *семантическая сеть* – это *направленный граф с поименованными вершинами и дугами,* причем *узлы* обозначают конкретные *объекты,* а *дуги – отношения между ними*. Как следует из определения, данная модель представления знаний является более общей по отношению к фреймовой модели (иными словами, фреймовая модель – частный случай семантической сети). Семантическую сеть можно построить для любой предметной области и для самых разнообразных объектов и отношений.

В семантических сетях используют три типа вершин:

1. вершины-понятия (обычно это существительные);
2. вершины-события (обычно это глаголы);
3. вершины-свойства (прилагательные, наречия, определения).

Дуги сети (семантические отношения) делят на четыре класса:

1. *лингвистические* (падежные, глагольные, атрибутивные);
2. *логические* (И, ИЛИ, НЕ);
3. *теоретико-множественные* (множество – подмножество, отношения целого и части, родовидовые отношения);

4. *квантифицированные* (определяемые кванторами общности ∀ и существования *∃).*

Приведем два примера. На рис 3.6 представлена семантическая сеть для предложения (ситуации) "Студент Табуреткин добросовестно изучает новый план счетов на 2022 год перед сдачей экзамена по дисциплине "Бухгалтерский учет".

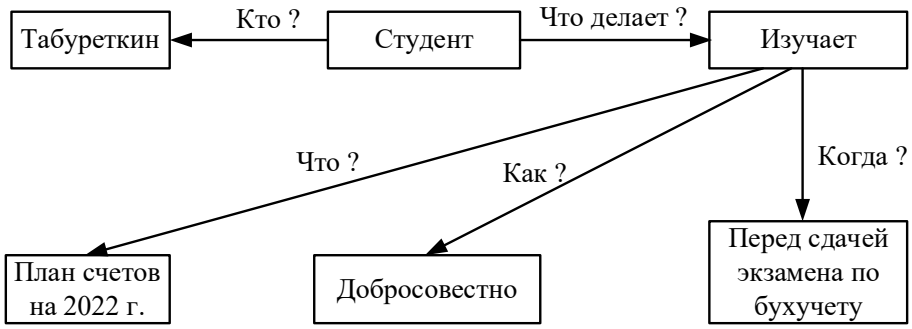


Рис.3.6. Семантическая сеть для предложения (ситуации)

Рисунок 3.7 содержит фрагмент семантической сети для понятия "автомобиль" (обозначения: IS-A – есть, является; HAS-PART – имеет часть).

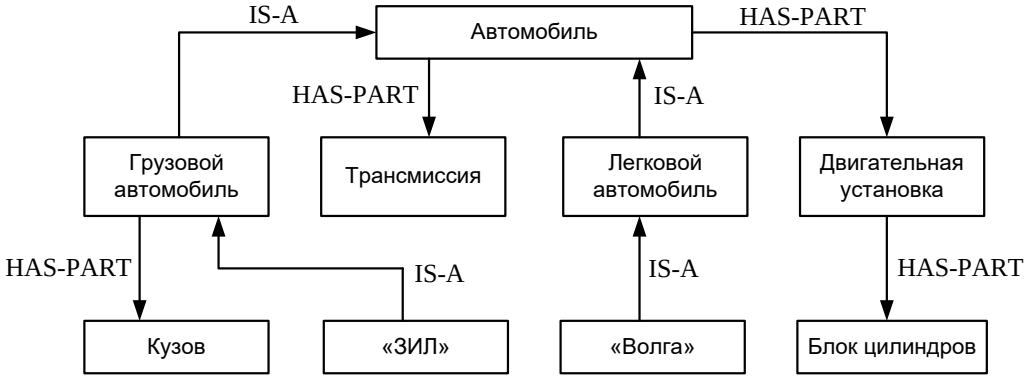


Рис. 3.7. Фрагмент семантической сети понятия "автомобиль"

Из приведенных примеров понятно, почему многие специалисты по искусственному интеллекту считают *фрейм частным случаем семантической сети со строго структурированными знаниями.*

Основное достоинство методов моделирования знаний с помощью семантических сетей и фреймов – универсальность, удобство представления как декларативных, так и процедуральных знаний. Имеют место и два недостатка:

1. громоздкость, сложность построения и *изменения;*

2. *потребность* в разнообразных *процедурах обработки,* связанная с разнообразием типов дуг и вершин.

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. В последних семантические сети используются для хранения знаний о структуре, форме и свойствах физических объектов. В области обработки естественного языка с помощью семантических сетей представляют семантические знания, знания о мире, эпизодические знания (т.е. знания о пространственно-временных событиях и состояниях).

В рамках реализации *теоретического подхода* применяют *логические модели,* прежде всего использующие представления знаний в системе *логики предикатов.* Преимущества такого подхода очевидны: единственность теоретического обоснования и возможность реализации системы путем введения формально точных определений и правил получения выводов. Однако в полной мере претворить в жизнь данный подход даже для "простых" задач оказалось весьма сложно. Поэтому появились *попытки перейти от формальной логики к так называемой человеческой логике* (модальной логике, многозначной логике и др.), модели которой в большей или меньшей степени учитывают "человеческий фактор", т. е. являются в определенном смысле компромиссными "в плане использования и теоретического, и эвристического подходов.

Очень коротко остановимся на предикатной модели представления знаний. Первые попытки использовать такую модель относятся к 50-м гг. прошлого века. Дадим несколько определений.

Пусть имеется некоторое *множество объектов,* называемое *предметной областью.* Выражение $P(x_1, x_2,...,x_n)$, где x_i (i = 1,..,n) – так называемая предметная переменная, а P принимает значения 0 или 1, называется *логической функцией* или *предикатом.*

Предикат $P(x_1,x_2,...,x_n)$ задает *отношение* между элементами x_1 $x_2,...,x_n$ и обозначает высказывание, что "x_1, $x_2,...,x_n$ находятся между собой в отношении P".

Из подобного рода элементарных высказываний с помощью *логических связок* образуют *более сложные высказывания,* которые могут принимать те же значения – "истина" и "ложь". В качестве связок используются *конъюнкция, дизъюнкция, импликация, отрицание, эквивалентность.*

Предикат от *n* переменных называют n-местным.

Одноместные (унарные) предикаты отражают свойства *определенного объекта* или класса *объектов.* Многоместные предикаты позволяют записывать *отношения,* которые существуют *между группой элементов.*

Если *a* – тоже предикат, то P(*a*) – предикат 2-го порядка, и т. д. до n~го порядка.

Приведем примеры различных предикатов.

1. Унарный предикат (высказывание) "Река впадает в Каспийское море" имеет значение 1, если "Река" = "Волга", и значение 0, если "Река" = "Днепр".

2. Двухместный предикат *"x_1* не меньше x_2" может иметь значение 1 или 0 в зависимости от значений x_1 и x_2. Если значение предиката *тождественно равно* 1 при любых значениях предметных переменных, он называется *тавтологией.*

В аппарат исчисления предикатов входят также *символы функций* (обычно обозначаемые латинскими буквами /, *g, h* и т. д.), задаваемых на множестве предметных переменных, и *кванторы общности* ∀ и *существования* ∃.

3. Представление с помощью предиката знаний, заключенных в теореме Пифагора: P{g[f(x), f(y)], f(z)}, где предикат P – "быть равным", функция $g(x, y) = x + y;$ функция $f(x) = x^2$.

Иногда используется такая форма записи:

РАВНЫ [СУММА (КВАДРАТ *(x),* КВАДРАТ *(y)),* КВАДРАТ (z)].

Предикат P равен 1, если *x, y, z* – соответственно длины катетов и гипотенузы прямоугольного треугольника.

Как уже отмечалось, предикаты удобны для описания *декларативных* знаний (фактов, событий и т. п.). Их *главные достоинства – возможность реализации строгого вывода знаний* (исчисления предикатов) и сравнительная *компактность модели.* К сожалению, предикаты мало пригодны для записи *процедуралъных* знаний. Кроме того, опыт показал, что человеческое знание по своей структуре много сложнее структуры языков предикатного типа, поэтому требуются специальные навыки *"подгонки"* структуры реального знания под структуру модели (как правило, значительно обедняющей исходные знания).

4. Достоверный вывод, метод резолюций. Правдоподобные методы вывода (4 часа)

Логический 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 *вывод по аналогии* (от частного к частному), *индуктивный* (от частного к общему), *нечёткий* (основанный на логике Заде), *нейросетевой* и некоторые другие методы вывода. Единственным достоверным методом вывода является *дедуктивный логический вывод* (рассуждения от общего к частному на основе аксиом).

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.

С позиции теории информации каждый ход при поиске решения должен приводить к уменьшению энтропии в системе от максимального значения до нуля. Согласно Шеннону, энтропия (неопределённость) – это вероятность $p(m_i)$ получения определённого сообщения из множества $M=\{m_1,\ldots, m_n\}$. Объем информации, содержащейся в этом сообщении, будет равен

$$I(m_i) = -\log p(m_i).$$

Здесь объем информации в сообщении и вероятность получения этого сообщения связаны обратной монотонной зависимостью. Энтропия множества сообщений M является суммой вида

$$U(M) = -\Sigma_i p(m_i)\log p(m_i),\ i=1,\ldots, n.$$

Чем более неожиданным является сообщение, тем оно информативнее. Отсюда следует эвристика выбора при построении вершин дерева поиска: выбирается то состояние (правила, действие), которое сулит наибольший прирост информации.

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). Пусть, например, необходимо проверить общезначимость формулы

$$F = ((A+B)\rightarrow C)\rightarrow(A\rightarrow(B\rightarrow C)),$$

где *A, B, C* – булевы переменные (1 или 0); операция импликация (→) является ложной тогда и только тогда, когда посылка истинна, а заключение ложно. Проверим общезначимость формулы несколькими методами.

В *алгоритме Квайна* для проверки используется семантическое дерево из $\mathbf{2^k}$ висячих вершин (***k*** – число переменных). Надо пройти все маршруты от корня до этих вершин.

Например, пусть A = 1. Тогда F = ((1+B)→C)→(1→(B→C)) = =C→ (B→C). Если B = 1, то F = C→(1→C) = C→C = 1. Если B = 0, то F = C→(0→C) = C→1 = 1.

Пусть теперь A = 0. Тогда F = ((0+B)→C)→(0→(B→C)) = (B→C)→1. Здесь, если B = 1, то F = 1; если B = 0, то F = 1. Следовательно, F ≡ 1.

Алгоритм редукции методом от противного использует свойства операции импликации. Предположим, что формула F ложна на некотором наборе A, B, C. Тогда f = ((A+B)→C) = 1, g = (A→(B→C)) = 0, откуда следует, что A = 1, B = 1, C = 0, однако f = ((1+1)→0) = 0. Получили противоречие. Следовательно, F является общезначимой формулой.

Наконец, если переменных в проверяемой на общезначимость формуле немного, то доказательство можно провести с использованием булевых таблиц истинности.

Ньюэлл и Саймон обосновали две гипотезы, на которых базируется большинство исследований в области ИИ. В соответствии с этими гипотезами человеческое мышление представляет собой систему, оперирующую с символами, отображающими действительность.

Гипотеза 1. Необходимым и достаточным условием для осуществления интеллектуальных действий в символьных системах является универсальность формальных манипуляций над конкретными символами (достаточно прочесть строку символов, разделить её на компоненты и переупорядочить, добавив или удалив какие-то символы без учёта семантики символов).

Гипотеза 2. Символьные системы решают задачи при помощи поиска, т.е. они генерируют потенциальные решения и модифицируют их до тех пор, пока решения не будут удовлетворять заданным условиям поиска. Поиск - это движение по дереву решений от одних узлов этого пространства к другим.

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 (под знаком квантора не могут находиться символы предикатов).

В основе языка Пролог лежит вывод по методу резолюций, который разработали Робинсон и Маслов. Рассмотрим вначале метод резолюций в исчислении высказываний.

В исчислении высказываний из простых высказываний можно составлять сложные высказывания посредством применения логических операций, например:

Высказывание	Название операции	Обозначение
X и Y	Конъюнкция	X & Y
X или Y	Дизъюнкция	X v Y
не X	Отрицание	¬X
если X, то Y	Импликация	X→Y

Истинность высказываний, получающихся в результате этих операций, определяется по следующей таблице:

X	Y	X & Y	X v Y	¬X	X→Y
0	0	0	0	1	1
0	1	0	1	1	1
1	0	0	1	0	0
1	1	1	1	0	1

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 ***F1,F2,..., Fk***, если при любой интерпретации ***j*** из ***j(F1) = j(F2) = ... = j(Fk) = 1*** следует, что ***j(G) = 1***.

Понятие логического следствия тесно связано с понятием выполнимости.

Множество формул {***F1, F2,..., Fk***} называется *выполнимым*, если существует интерпретация ***j*** такая, что ***j(F1) = j(F2) = ... = j(Fk) = 1***.

Проверить выполнимость множества формул {***F1, F2,..., Fk***} можно, построив для них таблицы истинности. Если найдется хотя бы одна строка, в которой в столбцах формул ***F1, F2,..., Fk*** 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 ***F1, F2,..., Fk*** тогда и только тогда, когда множество формул ***L = {F1, F2,..., Fk, ¬G}*** невыполнимо.

Доказательство теоремы сводится к тому, что некоторая формула ***G*** (ее называют *гипотезой* теоремы), является логическим следствием множества формул ***F1,..., Fk*** (*допущения* теоремы). Текст теоремы может быть сформулирован следующим образом: если формулы ***F1,..., Fk*** истинны, то истинна и формула ***G***.

Метод резолюций:

Задача логического вывода сводится к задаче проверки выполнимости множества формул {***F1, F2,..., Fk, ¬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 ■ (или #)**.** Считается, что пустой дизъюнкт ложен при любой интерпретации;
- *противоположные литералы* - литералы ***X*** и ***¬X***.

Правило резолюции. Из дизъюнктов (***X∨F***) и (***¬X∨G***) выводим дизъюнкт (***F∨G***). Другими словами, дизъюнкт (***F∨G***) является логическим следствием дизъюнктов (***X∨F***) и (***¬X∨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*. 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 ***D1, D2,..., Dn*** 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.

Алгоритм резолютивного вывода состоит в доказательстве того, что формула **G** является логическим следствием множества формул ***F1,..., Fk*** и включает следующую последовательность шагов.

1. Составляется множество формул $\{F_1,..., F_k, \neg G\}$.

2. Каждая из этих формул приводится к конъюнктивной нормальной форме (КНФ)- конъюнкции элементарных дизъюнкций. 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 $F_1,..., F_k$. Если из **S** нельзя вывести ■, то **G** не является логическим следствием формул $F_1,..., F_k$.

Рассмотрим несколько примеров вывода по методу резолюций.

Пример 1. Пусть даны два утверждения:

1) «Яблоко красное и ароматное»,

2) «Если яблоко красное, то яблоко вкусное».

Докажем утверждение, что «Яблоко вкусное».

Для этого введем множество формул, описывающих простые высказывания, соответствующие приведённым выше утверждениям:

X1 – «Яблоко красное»,

X2 – «Яблоко ароматное»,

X3 – «Яблоко вкусное».

Исходные утверждения запишем в виде следующих формул:

X1 & **X2** — «Яблоко красное и ароматное»,

X1→X3 — «Если яблоко красное, то яблоко вкусное».

Утверждение, которое надо доказать, выражается формулой **X3**.

Докажем, что **X3** является логическим следствием формул (**X1** & **X2**) и (**X1→X3**). Для этого составляем множество формул с отрицанием доказываемого высказывания:

{(**X1**&**X2**), (**X1**→**X3**), ¬**X3**}.

Приводим все формулы к КНФ:

{(**X1**&**X2**), (¬**X1** v **X3**), ¬**X3**}.

Зачёркивая конъюнкции, получаем следующее множество дизъюнктов:

S = {**X1**, **X2**, **(**¬**X1** v **X3**), ¬**X3**}.

Ищем вывод пустого дизъюнкта:

X1, **(**¬**X1** v **X3**), **X3**, ¬**X3**, ■.

Получили пустой дизъюнкт. Следовательно, утверждение о том, что яблоко вкусное, верно.

Пример 2. Пусть даны два утверждения:

1) **X1**→**X2** ,

2) **X2**→**X3** .

Докажем, что утверждение **X1**→**X3** является логическим следствием из этих утверждений. Для этого составляем множество формул с отрицанием доказываемого высказывания:

{(**X1**→**X2**), (**X2**→**X3**), ¬(**X1**→**X3**)}.

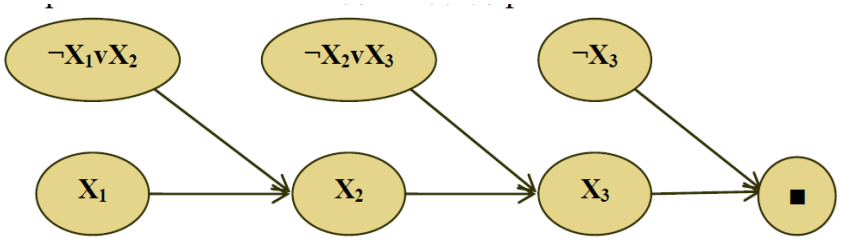
Устраняем импликации и приводим все формулы к КНФ:

{(¬**X1** v **X2**), (¬**X2** v **X3**), ¬(¬**X1** v **X3**)} = {(¬**X1** v **X2**), (¬**X2** v **X3**), (**X1**&¬**X3**)}.

Зачёркивая конъюнкции, получим множество из четырёх дизъюнктов:

S = {(¬**X1** v **X2**), (¬**X2** v **X3**), **X1**, ¬**X3**}.

Представим резолютивный вывод в виде дерева поиска:



Получен пустой дизъюнкт. Следовательно, утверждение ***X1→X3*** является логическим следствием утверждений ***X1→X2*** и ***X2→X3*** .

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∨G***) выводим дизъюнкт ***G***. Кроме того, метод резолюций является обобщением метода доказательства от противного.

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 еще два правила *склейки*.

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

Например, пусть даны два утверждения:

1) «Существуют студенты, которые любят всех преподавателей»,

2) «Ни один из студентов не любит невежд».

Докажем утверждение «Ни один преподаватель не является невеждой».

Введем множество формул, описывающих следующие простые утверждения:

C(x) – **x** есть студент;

P(y) – **y** есть преподаватель;

H(y) – **y** есть невежда;

L(x, y) – **x** любит **y**.

Тогда два исходных утверждения можно записать в виде формул:

1) **$\forall x(C(x)\&\forall y(P(y)\rightarrow L(x,y)))$**;

2) **$\forall x(C(x)\rightarrow\forall y(H(y)\rightarrow\neg L(x,y)))$**.

Утверждение, которое надо доказать, выражается следующей формулой:

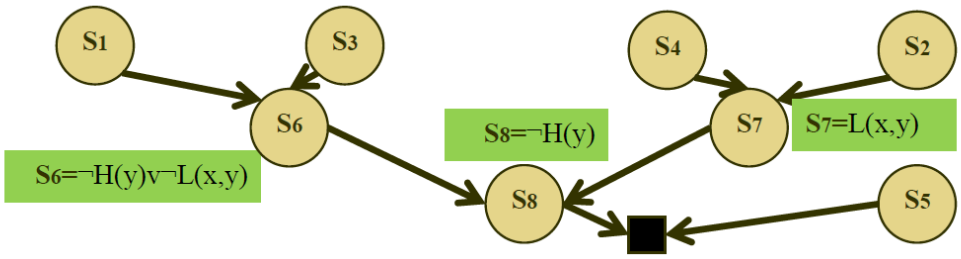
$\forall y(P(y)\rightarrow\neg H(y))$.

Применяя специальный алгоритм, приводим утверждения к набору дизъюнктов **S** = {**S1, S2, S3, S4, S5**}:

S1 = C(x); **S2** = ¬P(y) v L(x, y); **S3** = ¬C(x) v ¬H(y) v ¬L(x, y));

S4 = P(y); **S5** = H(y).

Дерево вывода для данного примера, построенное согласно методу резолюций для логики предикатов 1-го порядка, имеет вид, представленный на рисунке:



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.

Например, 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:

$$Q \leftarrow P1 \& P2 \& \ldots \& Pn,$$

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.

Решением последней задачи является алгоритм пересечений реки в виде графа возможных состояний, на котором необходимо найти путь, удовлетворяющий исходным условиям. Граф состояний строится при условии, что лодка может быть использована для перевоза человека и волка, человека и козы, человека и капусты и только человека.

Задача Стимроллер

Бенчмаркой для оценки эффективности алгоритмов вывода является *задача Стимроллер.* Эта задача была сформулирована Шубертом в 1978 г. в качестве теста для систем достоверного логического вывода и доказательства теорем.

Задача состоит в определении логической корректности рассуждения:

1. Существуют волки, лисы, птицы, гусеницы и улитки.
2. Волки, лисы, птицы, гусеницы и улитки – животные.
3. Существуют злаки.
4. Злаки – растения.
5. Каждое животное либо ест все растения, либо ест всех животных, которые меньше него и едят некоторые растения.
6. Гусеницы и улитки меньше птиц.
7. Птицы меньше лис.
8. Лисы меньше волков.
9. Волки не едят лис и злаки.
10. Птицы едят гусениц.
11. Птицы не едят улиток.
12. Гусеницы и улитки едят некоторые растения.
13. Следовательно, существует животное, которое ест некоторых питающихся злаками животных.

Чтобы решить задачу на Прологе надо записать ее условия на языке исчисления предикатов 1-го порядка. Для этого каждое их понятий представляется предикатом:

A(X): X – животное; B(X): X – птица; C(X): X – гусеница; F(X): X – лиса; G(X): X – злак; P(X): X – растение; S(X): X – улитка; W(X): X – волк; E(X, Y): X ест Y; M(X, Y): X меньше Y.

С их помощью описывается множество посылок, например:

1. W(w) (Существуют волки)
2. ⌐F(X) v A(X) (Лисы - животные)
3. ⌐G(X) v P(X) (Злаки - растения)
4. ⌐A(X)v⌐A(Z)vP(Y)v⌐P(V)vE(X,Y)v⌐M(Z,X)v⌐E(Z,V)vE(X,Z) (Каждое животное либо ест все растения, либо ест всех животных, которые меньше него и едят некоторые растения)
5. ⌐C(X) v ⌐B(Y) v M(X, Y) (Гусеницы меньше птиц)
6. ⌐W(X) v ⌐G(Y) v ⌐E(X, Y) (Волки не едят злаки)
7. ⌐C(X) v E(X, h(X)) (Гусеницы едят некоторые растения) и т.д.

Здесь где X, Y, Z, V – переменные; *w, f, b, c, s, g* – константы; *h u i* – сколемовские функции. Эти функции служат для удаления кванторов существования (∃ или A^{-1}).

Заключение задачи можно записать следующим образом:

E-1XE-1Y (A(X))&A(Y)&(E(X,Y)&E-1Z(G(Z)&E(Y,Z)))) или его отрицание

⌐A(X) v ⌐A(Y) v ⌐G(Z) v ⌐E(Y,Z) (Не существует животного, которое ест некоторых, питающихся злаками, животных).

Задача характеризуется экспоненциальным ростом пространства поиска. Анализ хода решения задачи «Стимроллер» на Прологе показывает, что «ход рассуждений» программы мало похож на человеческий.

Мы разбиваем задачу на подзадачи и стремимся делать общие, а не частные выводы. Например, «Волки не едят растения». Пролог-программа пытается получить частные факты. Например, «Лиса *f* ест птицу *b*» и пробует найти доказательство путем подстановки частных фактов в исходные посылки.

Для эффективного решения задачи в процессе логического вывода необходимо сокращать множество дизъюнктов и находить наилучшие связи для резольвент.

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 типа алгоритма Тарского, 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 тема ИИ.

Если проанализировать механизмы ПР человека или компьютерного алгоритма, использующих информацию о среде ПР, то можно заметить, что в их основе, в том или ином виде лежит одно из следующих правил вывода:

1. Дедуктивное правило modes ponens (или правило отделения):

$$\begin{array}{l}\text{если Н то А;}\\ \underline{\text{Н - истинно;}}\\ \text{А – истинно.}\end{array} \qquad (1)$$

2. Правило вывода по индукции:

$$\frac{\begin{array}{c}\text{если Н то А;}\\ \text{А - истинно;}\end{array}}{\text{Н – более правдоподобно.}} \qquad (2)$$

3. Правило вывода по аналогии:

$$\frac{\begin{array}{c}\text{если Н то А;}\\ \text{если Н то В;}\\ \text{А – истинно;}\end{array}}{\text{В – истинно по аналогии.}} \qquad (3)$$

Здесь посылка **Н** и следствия **А** и **В** – некоторые высказывания (факты) о процессе ПР в конкретных ситуациях.

Первое правило вывода лежит в основе рассмотренного выше материала и является достоверным методом вывода. Второе и третье правило относится к правдоподобным правилам вывода.

Пример 1. Согласно ОСТ для того, чтобы нарезать резьбу М6 шагом 1мм в стальном листе необходимо просверлить отверстие диаметром 4,95 мм. Отвечая на вопрос «каким должно быть отверстие в стальном листе, чтобы нарезать резьбу М6 шагом 1мм», человек по справочнику или информационно-справочная система по своей базе данных определяя ответ «диаметр 4,7мм» реализуют дедуктивное правило modus ponens. Здесь в качестве высказывания **Н**

выступает предложение «резьба М6 и шаг 1мм», а в качестве высказывания **А** – предложение «диаметр 4,7мм». Здесь используется дедуктивный вывод.

Пример 2. Автомобиль плохо заводится (или не заводится вообще) если неисправен аккумулятор. Когда владелец автомобиля обнаруживает, что последний не заводится, он проверяет аккумулятор. В этой ситуации реализуется вывод по индукции. Здесь посылкой **Н** выступает предложение «неисправен аккумулятор», а следствием **А** - предложение «автомобиль плохо заводится». По наблюдаемому следствию **А** владелец автомобиля делает предположение о правдоподобности посылки **Н**. Здесь используется андуктивный вывод.

Пример 3. Написав письмо (ручкой на бумаге) Вы решили переслать его по электронной почте с помощью компьютера, стоящего на Вашем столе. В это время настольная лампа, также стоящая на столе, начинает мигать. Вы делаете вывод, что компьютер включать не стоит. В этом примере в неявном виде реализуется вывод по аналогии. Здесь посылкой **Н** является высказывание типа «сбой в электросети», а следствиями **А** – «лампа мигает» и **В** – «компьютер не включать». Наблюдая факт «лампа мигает» Вы без вольтметра и осциллографа делаете предположение о посылке **Н** «сбой в электросети», а затем по факту **Н** и правилу «если **Н** то **В**» делаете вывод о следствии **В** – «компьютер не включать». Здесь используется вывод по аналогии.

Пример 4. Если на улице идет дождь, то тротуар будет влажным. Если на улице идет дождь, то выходя из квартиры вы берете зонт. Утром вы смотрите в окно и замечаете, что тротуар влажный и выходя из квартиры вы берете зонт. Здесь посылкой **Н** является высказывание типа «идет дождь», а следствиями **А** – «влажный тротуар» и **В** – «взять зонт». Наблюдая факт «влажный тротуар» Вы делаете предположение о посылке **Н** «идет дождь», а затем по факту **Н** и правилу «если **Н** то **В**» делаете вывод о следствии **В** – «взять зонт». Здесь в рассуждениях также был использован вывод по аналогии.

5. Технологии инженерии знаний

Знания являются основой любой интеллектуальной информационной системы (ИИС). Научное направление, занимающееся вопросами формализации, представления и обработки знаний в информационных системах, называют инженерией знаний. Этот термин предложил в 1977 г. американский ученый Э. Фейгенбаум. Он писал: «По опыту нам известно, что большая часть знаний в конкретной предметной области остается личной собственностью эксперта. И это происходит не потому, что он не хочет разглашать своих секретов, а потому, что он не в состоянии сделать этого, — ведь эксперт знает гораздо больше, чем сам осознает». Данное высказывание констатирует существование так называемых неформальных, или неявных, знаний, связанное с рядом факторов, в частности:

- часть экспертных знаний носит неосознаваемый характер;
- эксперт не всегда способен оценить важность тех или иных знаний для принятия решения;
- опыт, накопленный экспертом, сложно вербализовать и представить в формализованном виде.

Таким образом, основная задача инженерии знаний — извлечение и формализация знаний с целью их последующего использования. Сам Э.Фейгенбаум определил инженерию знаний как «привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов».

Инженерия знаний изучает следующие вопросы:

- извлечение знаний из экспертов и (или) литературы;
- формализация и обработка знаний;
- проектирование и разработка баз знаний.

Знания представляют собой более сложную информационную категорию по сравнению с данными. Данные несут в себе фактическую информацию, связанную с состоянием объектов, процессов, явлений предметной области. Знания описывают не только данные, но и взаимосвязи между ними, поэтому их иногда называют структурированными данными. Знания представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта.

Рисунок 5.1 иллюстрирует предлагаемую классификацию методов извлечения знаний, в которой используются наиболее употребительные термины, что позволит инженерам по знаниям в зависимости от конкретной задачи и ситуации выбрать подходящий метод.

Из предложенной схемы классификации видно, что основной принцип деления связан с источником знаний.

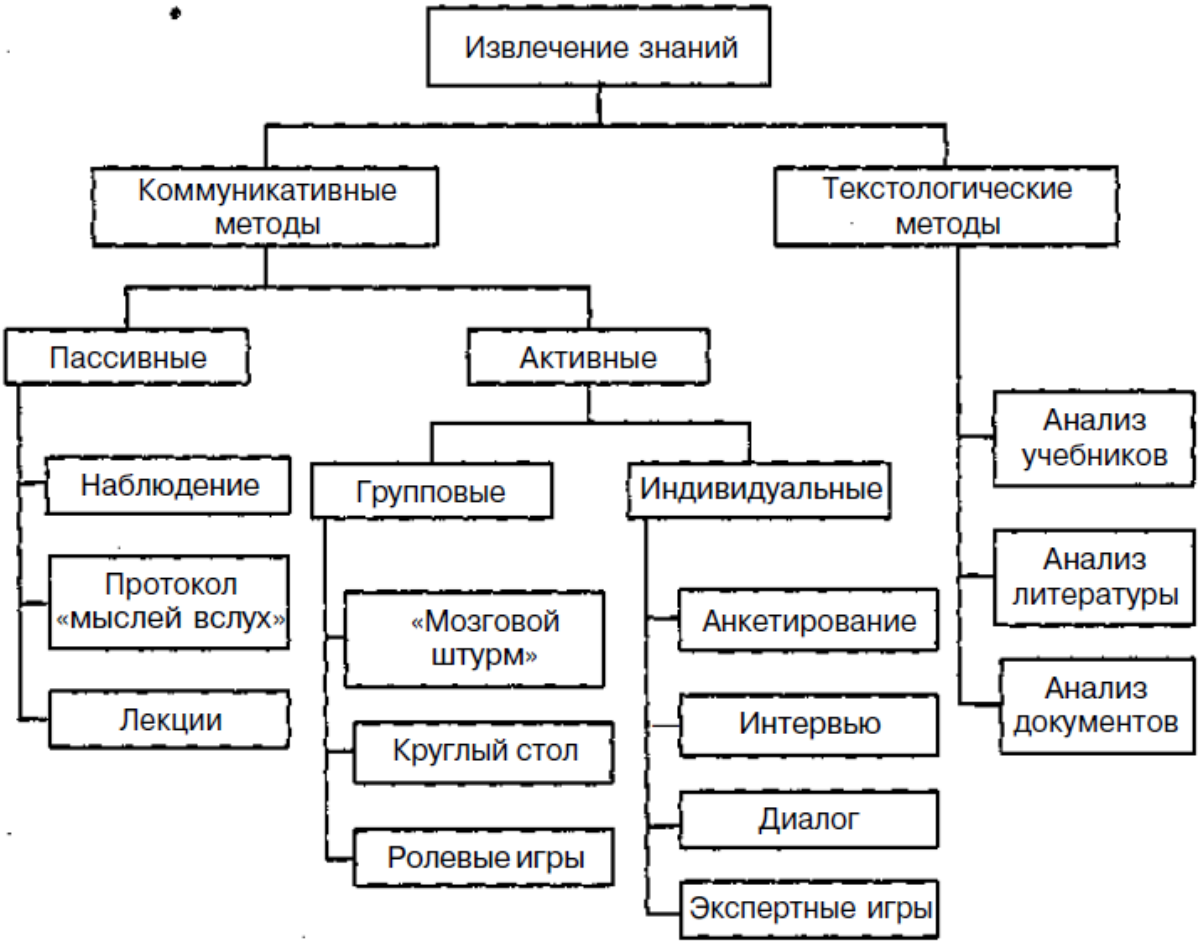


Рис.5.1. Классификация методов извлечения знаний

Разделение этих групп методов на верхнем уровне классификации не означает их антагонистичности, обычно инженер по знаниям комбинирует различные методы, например сначала изучает литературу, затем беседует с экспертами, или наоборот.

В свою очередь, *коммуникативные* методы можно также разделить на две группы: активные и пассивные. Пассивные методы подразумевают, что ведущая роль в процедуре извлечения как бы передается эксперту, а инженер по знаниям только протоколирует рассуждения эксперта во время его реальной работы по принятию решений или записывает то, что эксперт считает нужным самостоятельно рассказать в форме лекции. В активных методах, напротив, инициатива полностью в руках инженера по знаниям, который активно контактирует с экспертом различными способами — в играх, диалогах, беседах за круглым столом и т. д.

Следует еще раз подчеркнуть, что и активные и пассивные методы могут чередоваться даже в рамках одного сеанса извлечения знаний. Например, если инженер по знаниям застенчив и не имеет большого опыта, то вначале он может использовать пассивные методы, а постепенно, ближе знакомясь с экспертом, захватывать инициативу и переходить «в наступление».

Пассивные методы на первый взгляд достаточно просты, но на самом деле требуют от инженера по знаниям умения четко анализировать поток сознания

эксперта и выявлять в нем значимые фрагменты знаний. Отсутствие обратной связи (пассивность инженера по знаниям) значительно ослабляет эффективность этих методов, чем и объясняется их обычно вспомогательная роль при активных методах.

Активные методы можно разделить на две группы в зависимости от числа экспертов, отдающих свои знания. Если их число больше одного, то целесообразно помимо серии индивидуальных контактов с каждым применять и методы групповых обсуждений предметной области. Такие групповые методы обычно активизируют мышление участников дискуссий и позволяют выявлять весьма нетривиальные аспекты их знаний. В свою очередь, индивидуальные методы на сегодняшний день остаются ведущими, поскольку столь деликатная процедура, как «отъем знаний», не терпит лишних свидетелей.

Отдельно следует сказать об играх. Игровые методы сейчас широко используются в социологии, экономике, менеджменте, педагогике для подготовки руководителей, учителей, врачей и других специалистов. Игра — это особая форма деятельности и творчества, где человек раскрепощается и чувствует себя намного свободнее, чем в обычной трудовой деятельности.

На выбор метода влияют три фактора: личностные особенности инженера по знаниям, личностные особенности эксперта и характеристика предметной области.

Одна из возможных классификаций людей по психологическим характеристикам делит всех на три типа:

• мыслитель (познавательный тип);

• собеседник (эмоционально-коммуникативный тип);

• практик (практический тип).

Мыслители ориентированы на интеллектуальную работу, учебу, теоретические обобщения и обладают такими характеристиками когнитивного стиля, как поле независимость и рефлексивность (см. параграф 3.3). *Собеседники* — это общительные, открытые люди, готовые к сотрудничеству. *Практики* предпочитают действие разговорам, хорошо реализуют замыслы других, направлены на результативность работы.

Для характеристики предметных областей можно предложить следующую классификацию:

• хорошо документированные;

• средне документированные;

• слабо документированные.

Эта классификация связана с соотношением двух видов знаний $\boldsymbol{Z_1}$, и $\boldsymbol{Z_2}$, где $\boldsymbol{Z_1}$ — это экспертное «личное» знание, а $\boldsymbol{Z_2}$ — материализованное в книгах «общее» знание в данной конкретной области. На рисунке 5.2 показана данная классификация.

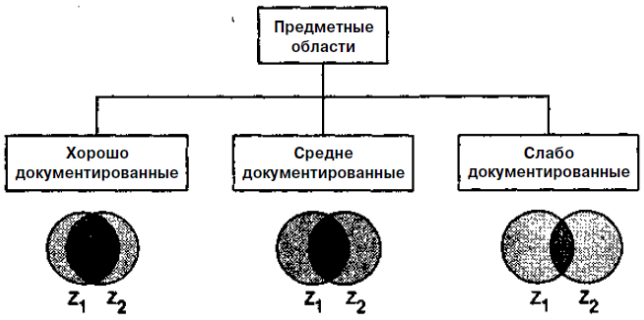


Рис.5.2. Классификация предметных областей

Предметные области можно разделить по критерию структурированности знаний. Под структурированностью будем понимать степень теоретического осмысления и выявленности основных закономерностей и принципов, действующих в данной предметной области. И хотя ЭС традиционно применяются в слабо структурированных предметных областях, сейчас наблюдается тенденция расширения сферы внедрения экспертных систем.

По степени структурированности знаний предметные области могут быть:

• хорошо структурированными — с четкой аксиоматизацией, широким применением математического аппарата, устоявшейся терминологией;

• средне структурированными — с определившейся терминологией, развивающейся теорией, явными взаимосвязями между явлениями;

• слабо структурированными — с размытыми определениями, богатой эмпирикой, скрытыми взаимосвязями, с большим количеством «белых пятен».

Рассмотренная классификация методов и предметных областей поможет инженеру по знаниям, четко определив свою предметную область, соотнести ее с предложенными типами и наметить подходящий метод или группу методов извлечения знаний.

Коммуникативные методы. Рассмотрим подробнее обе разновидности коммуникативных методов: ***пассивные и активные***.

Согласно классификации, к пассивным методам относятся: *наблюдения; анализ протоколов «мыслей вслух»; лекции.*

Наблюдения. В процессе наблюдений инженер по знаниям находится непосредственно рядом с экспертом во время его профессиональной деятельности или имитации этой деятельности. При подготовке к сеансу извлечения эксперту необходимо объяснить цель наблюдений и попросить максимально комментировать свои действия.

Во время сеанса аналитик записывает все действия эксперта, его реплики и объяснения. Может быть сделана и видеозапись в реальном масштабе времени. Непременное условие этого метода — невмешательство аналитика в работу эксперта хотя бы на первых порах. Именно метод наблюдений является единственно «чистым» методом, исключающим вмешательство инженера по знаниям и навязывание им каких-то своих структур представлений.

Существуют две основные разновидности проведения наблюдений:

• наблюдение за реальным процессом;

• наблюдение за имитацией процесса.

Обычно используются обе разновидности. Сначала инженеру по знаниям полезно наблюдать за реальным процессом, чтобы глубже понять предметную область и отметить все внешние особенности процесса принятия решения. Это необходимо для проектирования эффективного интерфейса пользователя. Кроме того, только наблюдение позволит аналитику увидеть предметную область.

Наблюдение за имитацией процесса проводят обычно также за рабочим местом эксперта, но сам процесс деятельности запускается специально для аналитика. Наблюдения за имитацией проводят также и в тех случаях, когда наблюдения за реальным процессом по каким-либо причинам невозможны (например, профессиональная этика врача-психиатра может не допускать присутствия постороннего на приеме).

Протоколы наблюдений после сеансов в ходе домашней работы тщательно расшифровываются, а затем обсуждаются с экспертом.

Таким образом, наблюдения является одним из наиболее распространенных методов извлечения знаний на начальных этапах разработки. Обычно он применяется не самостоятельно, а в совокупности с другими методами.

Протоколирование «***мыслей вслух***» отличается от наблюдений тем, что эксперта просят не просто прокомментировать свои действия и решения, но и объяснить, как это решение было найдено, то есть продемонстрировать всю цепочку своих рассуждений. Во время рассуждений эксперта все его слова, весь «поток сознания» протоколируется инженером по знаниям, при этом полезно отметить даже паузы и междометия. Иногда этот метод называют «вербальные отчеты».

Основной трудностью при протоколировании «мыслей вслух» является принципиальная сложность для любого человека объяснить, как он думает. При этом существуют экспериментальные психологические доказательства того факта, что люди не всегда в состоянии достоверно описывать мыслительные процессы. Кроме того, часть знаний, хранящихся в невербальной форме вообще слабо коррелируют с их словесным описанием.

От инженера по знаниям метод «мысли вслух» требует тех же умений, что и метод наблюдений. Обычно «мысли вслух» дополняются потом одним из активных методов для реализации обратной связи между интерпретацией инженера по знаниям и представлениями эксперта.

Лекция является самым старым способом передачи знаний. В лекции эксперту также предоставлено много степеней свободы для самовыражения; при этом необходимо сформулировать эксперту тему и задачу лекции.

Хороший вопрос по ходу лекции помогает и лектору и слушателю. Серьезные и глубокие вопросы могут существенно поднять авторитет инженера по знаниям в глазах эксперта.

Метод извлечения знаний в форме лекций, как и все пассивные методы, используют в начале разработки как эффективный способ быстрого погружения инженера по знаниям в предметную область.

Активные индивидуальные методы извлечения знаний — наиболее распространенные. К основным активным методам можно отнести:

- анкетирование;

- интервью;
- свободный диалог;
- игры с экспертом.

Во всех этих методах активную функцию выполняет инженер по знаниям, который пишет сценарий и режиссирует сеансы извлечения знаний. Игры с экспертом существенно отличаются от трех других методов. Три оставшихся метода похожи между собой и отличаются лишь по степени свободы, которую может себе позволить инженер по знаниям при проведении сеансов извлечения знаний. Их можно назвать вопросными методами поиска знаний.

Анкетирование — наиболее стандартизированный метод. В этом случае инженер по знаниям заранее составляет вопросник или анкету, размножает ее и использует для опроса нескольких экспертов. Это основное преимущество анкетирования.

Сама процедура может проводиться двумя способами:

1. Аналитик вслух задает вопросы и сам заполняет анкету по ответам эксперта.

2. Эксперт самостоятельно заполняет анкету после предварительного инструктирования.

Выбор способа зависит от конкретных условий (например, от оформления анкеты, готовности эксперта). Второй способ кажется предпочтительным, так как у эксперта появляется неограниченное время на обдумывание ответов.

Под ***интервью*** будем понимать специфическую форму общения инженера по знаниям и эксперта, в которой инженер по знаниям задает эксперту серию заранее подготовленных вопросов с целью извлечения знаний о предметной области.

Наибольший опыт в проведении интервью накоплен, наверное, в журналистике и социологии. Большинство специалистов этих областей отмечают тем не менее крайнюю недостаточность теоретических и методических исследований по тематике интервьюирования.

Интервью очень близко тому способу анкетирования, когда аналитик сам заполняет анкету, занося туда ответы эксперта. Основное отличие интервью в том, что оно позволяет аналитику опускать ряд вопросов в зависимости от ситуации, вставлять новые вопросы в анкету, изменять темп, разнообразить ситуацию общения. Кроме этого, у аналитика появляется возможность «взять в плен» эксперта своим обаянием, заинтересовать его самой процедурой и тем самым увеличить эффективность сеанса извлечения.

На Рисунке 5.3. представлена подробная классификация вопросов для интервью.

Три основные характеристики вопросов, которые влияют на качество интервью:

• язык вопроса (понятность, лаконичность, терминология);
• порядок вопросов (логическая последовательность и немонотонность);
• уместность вопросов (этика, вежливость).

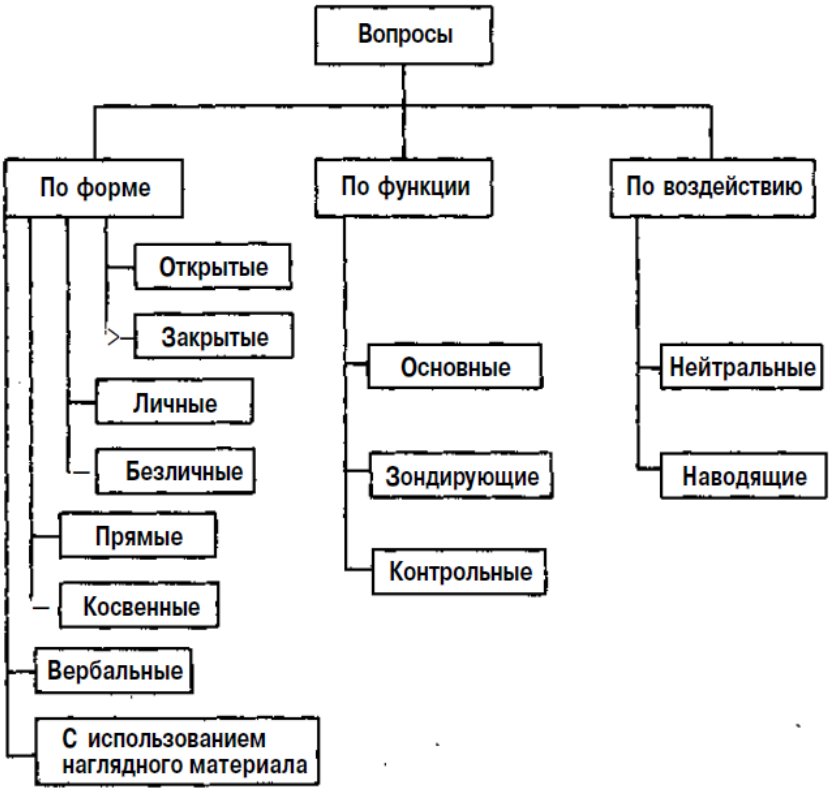


Рис.5.3.

Свободный ***диалог*** — это метод извлечения знаний в форме беседы инженера по знаниям и эксперта, в которой нет жесткого регламентированного плана и вопросника. Это определение не означает, что к свободному диалогу не надо готовиться. Рисунок 5.4 графически иллюстрирует схему такой подготовки, дополненную в связи со спецификой инженерии знаний. Подготовка занимает разное время в зависимости от степени профессионализма аналитика, но в любом случае она необходима, так как несколько уменьшает вероятность самого нерационального метода — метода проб и ошибок.

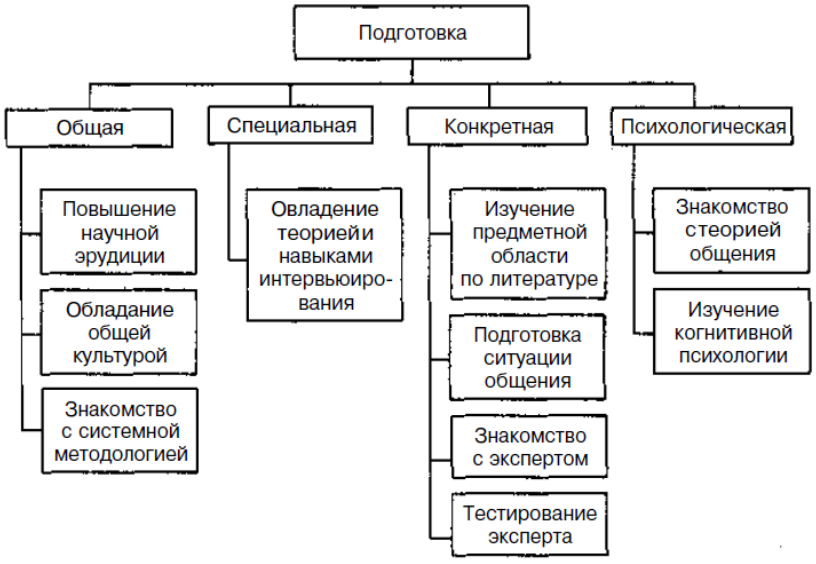


Рис.5.4.

Активные групповые методы. К групповым методам извлечения знаний относятся *ролевые игры*, *дискуссии за «круглым столом»* с участием нескольких экспертов и «*мозговые штурмы*». Основное достоинство групповых методов — это возможность одновременного «поглощения» знаний от нескольких экспертов, взаимодействие которых вносит в этот процесс элемент принципиальной новизны от наложения разных взглядов и позиций.

Метод круглого стола предусматривает обсуждение какой-либо проблемы из выбранной предметной области, в котором принимают участие с равными правами несколько экспертов. Обычно вначале участники высказываются в определенном порядке, а затем переходят к живой свободной дискуссии. Число участников дискуссии колеблется от трех до пяти - семи.

Большинство общих рекомендаций по извлечению знаний, предложенных ранее, применимо и к данному методу.

Однако существует и специфика, связанная с поведением человека в группе. Во-первых, от инженера по знаниям подготовка «круглого стола» потребует дополнительных усилий: как организационных, так и психологических. Во-вторых, большинство участников иногда будет говорить совсем не то, что они сказали бы в другой обстановке, то есть желание произвести впечатление на других экспертов будет существенно «подсвечивать» их высказывания.

Задача дискуссии — коллективно, с разных точек зрения, под разными углами исследовать спорные гипотезы предметной области.

«***Мозговой штурм***» или «мозговая атака» — один из наиболее распространенных методов раскрепощения и активизации творческого мышления.

Впервые этот метод был использован в 1939 г. в США как способ получения новых идей в условиях запрещения критики. Замечено, что боязнь критики мешает творческому мышлению, поэтому основная идея штурма - это отделение процедуры генерирования идей в замкнутой группе специалистов от процесса анализа и оценки высказанных идей.

Основной девиз штурма — «чем больше идей, тем лучше».

Игрой называют такой вид человеческой деятельности, который отражает (воссоздает) другие ее виды. При этом для игры характерны одновременно условность и серьезность.

Понятие экспертной игры или игры с экспертами в целях извлечения знаний восходит к трем источникам — это деловые игры, широко используемые при подготовке специалистов и моделировании, и компьютерные игры, часто применяемые в обучении.

Под *деловой игрой* чаще всего понимают эксперимент, где участникам предлагается производственная ситуация, а они на основе своего жизненного опыта, своих общих и специальных знаний и представлений принимают решения. Решения анализируются, и вскрываются закономерности мышления участников эксперимента. Именно эта анализирующая часть деловой игры полезна для получения знаний. И если участниками такой игры становятся эксперты, то игра из деловой превращается в экспертную. Из трех основных типов деловых игр (учебных, планово-производственных и исследовательских)

к экспертам ближе всего исследовательские, которые используются для анализа систем, проверки правил принятия решений.

Диагностическая игра — это та же деловая игра, но применяемая конкретно для диагностики методов принятия решения в медицине (диагностика методов диагностики). Эти игры возникли при исследовании способов передачи опыта от опытных врачей новичкам.

Плодотворность моделирования реальных ситуаций в играх подтверждается сегодня практически во всех областях науки и техники. Они развивают логическое мышление, умение быстро принимать решения, вызывают интерес у экспертов.

6. Гипертекстовые, многоагентные и онтологические системы.

Что такое гипертекст. 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.

Библия состоит из книг Старого и Нового Завета. Книги разбиты на перенумерованные "стихи" без повторений. Каждый стих - целостная тема. Многие стихи сопровождаются гиперссылками на другие стихи, что позволяет читать Библию «нелинейно».

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- 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.

Тим 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.

Модели гипертекста:

Формализованная модель. В основе модели 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 2-х 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 **<x0, x1,…, x11>, где**

x0 –имя ИСС; x1 – заголовок;

x2 – аннотация;

x3 – точка входа в ИСС;

x4 – текст-фрагменты ИСС;

x5 – цифровые объекты ИСС;

x6 – программные объекты;

x7 – справка по ИСС;

x8 – признак ускоренного просмотра;

x9 – признак детального просмотра;

x10 – список локальных гиперссылок;

x11 – список глобальных гиперссылок.

Основной недостаток модели - отсутствие возможности явного определения типов гиперссылок.

Тезаурусная модель гипертекста. Она имеет следующий вид:

$\boldsymbol{\Gamma_0 = (T, I, S, Q)}$

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

Тезаурус Т – это семантически упорядоченный перечень терминов предметной области;

Тезаурусная статья **ti** = {**mi , Ami**}, где **mi** - термин**, Ami -** множество терминов, с которыми **mi** связан семантическими отношениями **R**.

Лингвистами выделено ***200*** *семантических типов отношений.* Наиболее часто употребляются **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, т.е. тезаурус гипертекста, 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.

Гипертекстовые информационные системы

Автоматический анализ текстов. Попробуем разобраться с основами работы поисковых систем. 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 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».

Дж.Зипф (G. Zipf) 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.

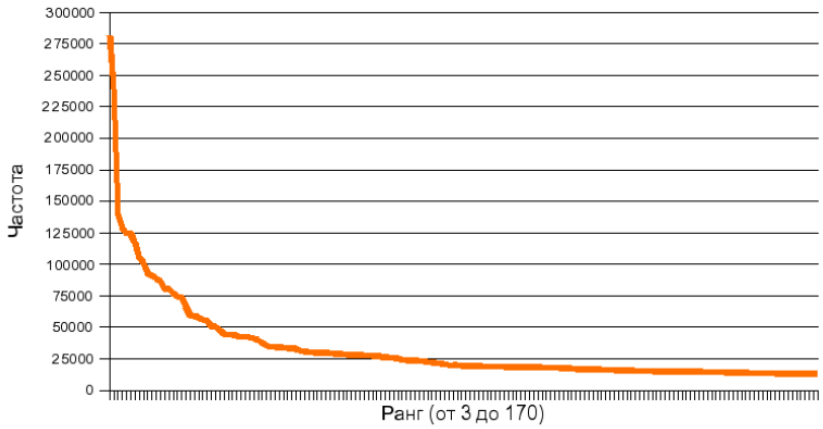
Первый закон Зипфа. Выбрав любое слово можно подсчитать, сколько раз оно встречается в тексте. Эта величина называется частотой вхождения слова. Далее, можно измерить частоту каждого слова текста. Некоторые слова будут иметь одинаковую частоту. Их нетрудно сгруппировать, пронумеровать и расположить группы 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

Зипф обнаружил интересную 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 (x) и частота (y) связаны формулой вида $y = k/x$. Её график, как известно, - равносторонняя гипербола:

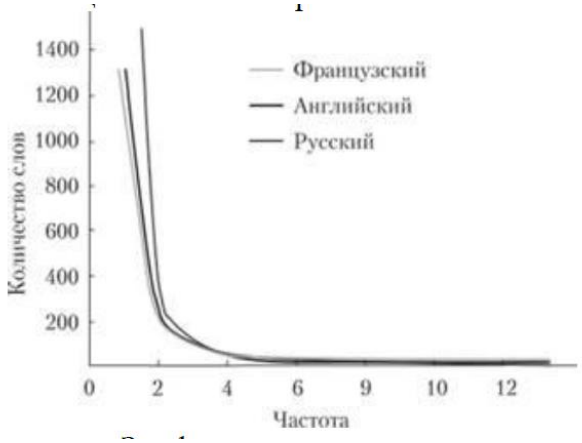


Следовательно, по 1-му закону Зипфа, если самое распространенное слово встречается в тексте, например, 100 раз, то следующее по частоте слово вряд ли встретится 99 раз. Частота вхождения второго по популярности слова, с высокой долей вероятности, окажется на уровне 50.

Значение константы 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 Зипфа, не выглядят исключением закон безупречен, и тут коэффициент Зипфа равен 0,06-0,07.

Американский биолог В.Ли попытался опровергнуть закон Зипфа, строго доказав, что случайная последовательность символов подчиняется закону Зипфа. Он сделал вывод, что закон Зипфа 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 одной оси (оси X) частоту вхождения слова, а по другой (оси Y) -- 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 кривой Зипфа останется неизменной. Могут немного отличаться лишь коэффициенты, отвечающие за наклон кривой:

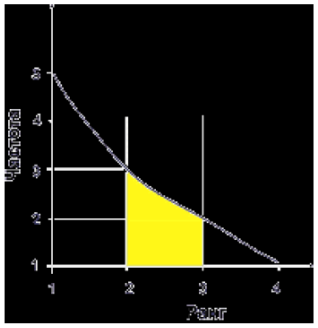


Как с помощью законов Зипфа извлечь слова, отражающие смысл текста? - Воспользуемся 1-м законом. Наиболее значимые слова лежат в средней части гиперболы. Это понятно, потому что это часто встречающиеся предлоги, местоимения, артикли и т.п. Редко встречающиеся слова тоже зачастую не имеют решающего смыслового 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, словарями и т.п.

Проведем эксперимент. Подвергнем абзац текста математическому анализу и попытаемся определить список значимых слов. В качестве примера возьмем следующий абзац:

«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».

Предположим, что областью значимых слов являются слова из диапазона частот от 2 до 3:



Анализ показывает, что не все слова, которые попали в диапазон значимых, отражают смысл текста. Смысл абзаца очень точно выражают слова: Зипфа, манускриптов, Войнича, законам. Интернет-запрос типа: «закон&Зипфа+манускрипт&Войнича» непременно найдет этот документ. Однако в область значимых слов попали и слова: на, не, для, например, это. Эти слова являются «шумом», который затрудняет правильный выбор. «Шум» можно уменьшить путем предварительного исключения из исследуемого текста некоторых слов. Для этого создается словарь ненужных слов (стоп-лист).

Например, для английского текста стоп-словами станут термины: the, a, an, in, to, of, and, that и т.д. Для русского текста в стоп-лист могли бы быть

включены все предлоги, частицы, личные местоимения и т.п. Есть и другие способы повысить точность оценки значимости терминов.

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.

Матричное представление базы документов.

Пусть необходимо организовать коллекцию документов так, чтобы можно было легко отыскать в ней нужный материал. Запросы могут быть простыми (одно слово) и сложными (несколько слов, связанных логическими операторами). Взаимодействие со сложными запросами требует изощренной организации базы документов.

Наиболее простой способ представить элементы базы - создать матрицу «Документ-термин».

Предположим, база включает 8 документов (Д1, Д2, …, Д8), в которых содержатся 11 терминов. Если термин входит в документ, в соответствующей клетке ставится 1, в противном случае – 0:

Документы / *Термины*	**Д1**	**Д2**	**Д3**	**Д4**	**Д5**	**Д6**	**Д7**	**Д8**
Алкоголизм	0	1	0	0	1	0	0	0
Бутылка	1	1	0	1	0	0	1	0
Вино	0	1	0	1	0	0	1	0
Корабль	1	0	0	0	0	0	0	1
Модель	1	0	0	0	0	1	0	1
Море	0	1	1	0	0	1	0	0
Парус	0	0	1	0	0	1	0	1
Пиво	0	0	0	1	1	0	0	0
Судомоделизм	0	0	1	0	0	0	0	0
Урожай	0	0	0	1	1	0	1	0
Хобби	0	0	1	0	0	0	0	1

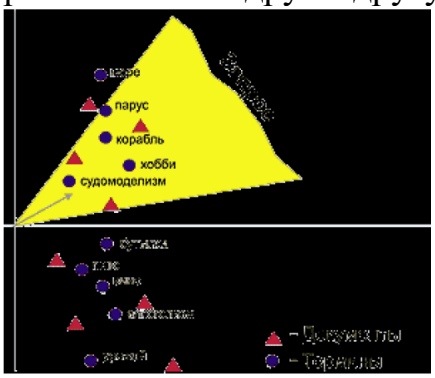
Составим, например, такой запрос: корабли в бутылках. Система обработает запрос: удалит стоп-слова и, возможно, проведет морфологический анализ. Останется два термина: корабль и бутылка. Система будет искать все документы, где встречается хотя бы один из терминов. Посмотрим на матрицу. Указанные в запросе термины есть в документах: Д1, Д2, Д4, Д7, Д8. Они и будут выданы в ответ на запрос. Однако нетрудно заметить, что документы Д4 и Д7 из области виноделия и никакого отношения к постройке моделей кораблей в бутылках не имеют. Впрочем, система все сделала правильно, ведь, с ее точки зрения, термины корабль и бутылка равноценны.

Пространственно-векторное представление базы документов.

Пространственно-векторная модель позволяет получить результат, хорошо согласующийся с запросом. Причем документ может оказаться полезным, даже не имея 100% соответствия. В найденном документе может вовсе не оказаться

одного или нескольких слов запроса, но при этом его смысл будет запросу соответствовать. Как достигается такой результат?

Все документы в базе размещаются в воображаемом многомерном пространстве. Координаты каждого документа зависят от структуры терминов, содержащихся в них. В результате окажется, что документы с похожим набором терминов разместятся в пространстве ближе друг к другу:



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 значительно сложнее. Однако полученных знаний достаточно, чтобы попытаться выработать оптимальную стратегию поиска информации в сети Интернет.

Методика поиска информации в сети Интернет. Теперь понятно, как поисковая система выделяет ключевые слова. Как воспользоваться этим знанием, чтобы сформировать оптимальный запрос?

Алгоритм:

- Удаляем из текста стоп-слова.
- Вычисляем частоту вхождения каждого термина без учета морфологии слов и регистра.
- Сформируем лист терминов в порядке убывания их частоты вхождения.
- Выбираем примерно посередине диапазон частот, ориентируясь на конкретный смысл текста.
- Из выбранного диапазона выбираем 10-20 терминов.
- Составляем запрос и отправляем его поисковой системе.

В ответ вы можете получить несколько миллионов ссылок. Поисковая машина ранжирует результаты, на первых страницах окажутся практически стопроцентно релевантные документы.

Модели автоматизации поиска. 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).*

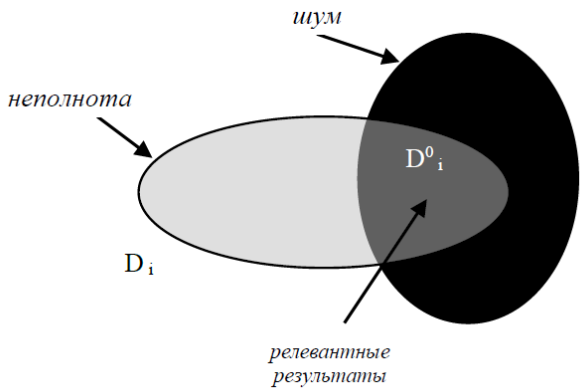
Коэффициенты информационной *полноты* и *шума* определяются следующим образом:

$$k_{П} = \lim_{k \to m} \frac{1}{k} \sum_{i=1}^{k} \frac{| D_i \cap D_i^0 |}{| D_i |}$$

и

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

где m – достаточно большое число (по теореме о больших числах), $D^0{}_i$– подмножество релевантных документов из Di, полученных по i-запросу. Смысл коэффициентов полноты и шума иллюстрирует рисунок:



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 $k_{Ш}$ часто используют обратный ему показатель – *коэффициент точности*: $k_T = 1 - k_{Ш}$.

Тогда оценка *эффективности информационного поиска* равна:

$$E_1 = 2k_T k_{П} / (k_T + k_{П})$$

В теории информационного поиска предложен *обобщенный комплексный показатель эффективности* E_β (*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 ($\beta \in [0, \infty]$). 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).

Сравнение гипертекстовых, фактографических и документальных информационно-поисковых систем

Информационно-поисковые системы (ИПС) бывают 3-х типов:

• *документальные* (хранят и выдают сведения о документах),

• *фактографические* (хранят не документы, а факты о предметной области, например в виде реляционных БД). Поиск в них в отличие от документального является полным и точным,

• *гипертекстовые* (хранят документы и их семантическую структуру). • 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: $\mathbf{k_{Пmax}} = \mathbf{0{,}9 \div 1}$; $\mathbf{k_{Шmax}} = \mathbf{0{,}1 \div 0{,}2}$.

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

Методы автоматизации поиска реализованы в ИПС, БД, БЗ и поисковых машинах Интернет:

• в ИПС 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 БД и БЗ , наряду с перечисленными методами, применяются наряду с перечисленными 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 ИИ. 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 – самый 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.

Автоматическое реферирование и аннотирование

Реферат – это доклад, излагающий краткое содержание работы на определенную тему с обзором библиографических источников.

Аннотация – это краткий реферат с указанием категории читателей, для которых предназначено аннотируемое произведение.

Основные методы автоматического реферирования и аннотирования: поверхностные (извлекают из текста обороты типа «идея состоит…» и т.п.), объединяют их в реферат; глубинные (используют тезаурусы и синтаксический разбор текста).

Проблемы: составление реферата из разноязычных и разнотипных работ, сборника докладов конференции или газетных статей, семантика графики и т.п.

Системы машинного перевода

Машинный перевод - это процесс перевода текстов (письменных, а в идеале и устных) с одного естественного языка на другой с помощью специальной компьютерной программы. Так же называется направление научных исследований в ИИ, связанных с построением подобных систем.

Мысль использовать ЭВМ для перевода была высказана в 1947 году, сразу после появления первых ЭВМ. Первая публичная демонстрация машинного перевода состоялась в 1954 году. Несмотря на примитивность той системы (словарь в 250 слов, грамматика из 6 правил, перевод нескольких простых фраз), этот эксперимент получил широкий резонанс: начались исследования по всему миру.

Потребность в переводах растет, а критерии оценки результатов ясные. Однако, несмотря на многолетние совместные усилия лингвистов, математиков и программистов удалось лишь увеличить производительность перевода, но не его качество. Машинный перевод художественных текстов практически всегда оказывается неудовлетворительного качества. 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.

За 3 десятилетия сменились 3 поколения систем машинного перевода: системы прямого перевода (перевод получается по качеству немного лучше подстрочного перевода), системы-преобразователи (используют знания о морфологии и синтаксисе языка), системы с языком посредником (используют знания о семантике языка, правила перевода, сложное математическое, информационное и программное обеспечение).

Автоматическая классификация документов

Классификация позволяет сузить поиск, увеличить его скорость и точность в системах документооборота, каталогах Интернет, каналах вещания, службах электронной почты, электронных библиотеках, информагентствах, Интернет-порталах и др.

Цель классификации – автоматическое распределение поступающих в систему документов в зависимости от их типа и содержания по рубрикам (классам) по заданным критериям (библиотечные системы).

Наиболее эффективный метод группировка и поиск ближайшего соседа путем вычисления «расстояния» между парами документов и объединения ближайших соседей в кластеры. Наиболее известными технологиями являются фильтрация в Microsoft Outlook, рубрицирование в продукте Inxight Categorizer и др.

Программные продукты, реализующие технологии обработки текстов на ЕЯ

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

• Смысловой анализатор текста Text Analyst;

• ИПС 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 ЕЯ, запатентованный компанией «НейрОК Интелсофт». Реализует несколько видов поиска: по SQL-запросам; по ключевым словам; ассоциативный и др.

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

Инженерия знаний - это одно из важнейших направлений ИИ и современного программирования, связанное с разработкой экспертных систем и баз знаний. Изучает методы и средства извлечения, представления, структурирования и использования знаний.

Справедливости ради, отметим, что в последние годы появляются языки и системы, например, Knowledge.NET, в основе которых лежит принцип расширения современных систем программирования средствами и библиотеками КЕ. Как это выглядит на практике? - Инженер по знаниям использует весь арсенал моделей представления знаний (онтологии, фреймы, продукции, сети и др.) для формализации ПО. Для реализации алгоритмов он использует конструкции Java, C# и др. Для реализации расширений,

предназначенных для представления знаний, используется их конвертирование в программу на базовом языке. Для компиляции полученных текстов в объектный код используется «штатный» компилятор.

Интеллектуальный агент

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. ИИ. Современный подход, 2006]. 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 информатике и 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.

Программа-агент может быть математически описана как агентская функция:

f: P (результат восприятия) → A (действия).

Иными словами, программный 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.

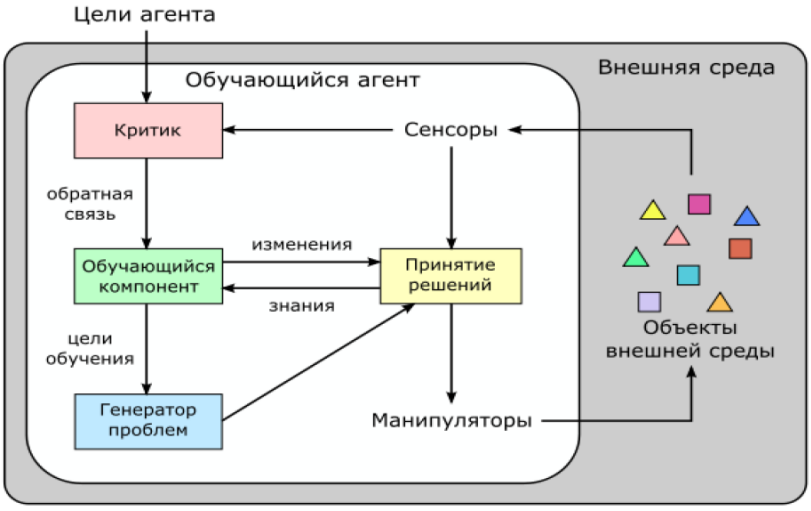
Практичные агенты различают, когда цель достигнута, и когда не достигнута, а также насколько желательно для них текущее состояние с помощью «функции полезности».

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.

Американские ученые из Политехнического института Ренсселера совместно с компанией IBM создали программу-интеллектуального агента **Эдди** (**Eddie**). В агенте реализованы технологии ИИ 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 информатике и программной инженерии

В компьютерной 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*, 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.

Характеристики агентов в многоагентной системе

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. Агенты в МАС должны обладать:

•• *Автономностью* – агенты работают без непосредственного вмешательства со стороны

•• *Интерактивностью* – взаимодействуют с другими агентами

•• *Реактивностью* – воспринимают окружающую среду и взаимодействуют с ней

•• *Проактивностью* – сами являются источником возмущения для окружающей среды, проявляя целеустремленное поведение

•• *Целеустремленностью* – агенты способны выполнять высокоуровневые задачи и проявлять 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).

Современные направления изучения МАС:

- знания, желания и намерения;
- координация
- самоорганизация;
- мультиагентное обучение;
- надежность и устойчивость к сбоям;

Общие принципы координации в МАС

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 (1970) сформулировал три базовых принципа координации в МАС: прогнозирование, развязывание и оценка взаимодействий.

В основе большинства известных методов координации МАС, 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.

Пример 1: Управление воздушным движением в районе аэропорта и разрешение непредвиденных конфликтов.

Пример 2: Команда агентов, которые функционируют в динамической внешней среде в присутствии шума и противодействия со стороны других команд соперников (команды роботов, принимавшие участие в соревнованиях RoboCup).

Во многих приложениях агенты, так или иначе, конкурируют. Это "эгоистичные" (self–interested) агенты, каждый из них стремится максимизировать свою функцию полезности, игнорируя интересы других агентов. В таких МАС взаимодействие агентов выражается в форме переговоров. Правила ведения переговоров должны быть предварительно установлены в виде протокола переговоров.

Протоколы основываются или на теоретико-игровой арбитражной схеме Нэша, или на модели аукциона. Наиболее часто здесь используется стандартный протокол контрактных сетей (*FIPA*).

Самоорганизация и эмерджентность

Сложность современных приложений такова, что всякое централизованное управление просто невозможно. Как управлять системой, в которой информация поставляется миллионами сенсоров?

Именно поэтому сейчас активно развиваются исследования в области самоорганизующихся систем. Самоорганизация – это динамические и адаптивные процессы, ведущие к поддержанию системы без внешнего управления. В качестве прообразов часто используются примеры из области

эволюционных вычислений (роевые алгоритмы). Например, основные принципы роевых муравьиных алгоритмов состоят в следующем:

1. Взаимодействия между индивидуумами 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 источнику пищи).

Основными чертами самоорганизации являются:

1. *Автономность* – взаимодействие в внешним миром допустимо, но недопустимо управление из внешнего мира

2. *Адаптивность и робастность* по отношению к изменениям – способность реагировать подходящим образом на изменения среды

3. *Возрастание порядка* – в соответствии с возрастанием организации системы (уменьшение числа состояний системы, появление пространственной, временной и функциональной структур; появление состояний системы на метауровне - аттракторов)

4. *Динамика* – самоорганизация есть процесс, но не какое-либо конечное состояние.

Говорят, что система проявляет *эмерджентность*, если на макроуровне в системе возникают новые свойства, паттерны, и т.д., и эти процессы являются следствием взаимодействий на микроуровне.

Эмерджентность и самоорганизация являются процессами, которые могут встречаться по отдельности, а также совместно. Основное сходство в том, что они являются динамическими процессами, обусловлены сложными взаимодействиями на микроуровне, а проявляются на макроуровне. Оба процесса робастны.

Примерами сложного взаимодействия являются аукционы и биржи.

Сложные взаимодействия агентов. Аукционы.

Проведение аукционов является примером сложного взаимодействия агентов. *Аукцион* (лат. «повышаю») − метод проведения торговли каким-либо товаром, ценными бумагами и т.д. Совсем не обязательно аукционы проходят с повышением цены. Аукцион – способ продажи , основанный на конкуренции.

Для каждого участника аукциона ценности делятся на три вида: личная, общая и коррелированная ценность. Существуют 4 основных вида аукционов:

•• прямой (английский),

•• голландский (оптовый),

•• янки (своей цены),

•• обратный.

Английский аукцион является самым распространенным видом. Он проводится с гласными торгами и поднятием цены, начинается с минимальной цены, причём покупатели выставляют по очереди более высокие цены. Товар

достается покупателю, давшему максимальную цену. Покупатель оказывается в невыгодном положении, если вещь представляет собой общую или коррелированную ценность. Не всегда торги заканчиваются продажей.

На *обратном аукционе* покупатели выставляют запросы на требуемые товары, а продавцы соревнуются, предлагая лучшую цену и условия.

Главная особенность *аукционов янки* − закрытые от других участников торги. Каждый участник подает свою цену в конверте, продавец выбирает наибольшую, а покупатель покупает товар по той цене, которую он назвал.

Данный вид аукционов не имеет доминирующих стратегий, считается, что способствует шпионажу.

Название голландского аукциона происходит от рынка цветов в Голландии. Это оптовый аукцион, на котором продавец может выставлять много единиц товара одновременно. Покупатели могут претендовать на покупку многих единиц товара. Все выигравшие покупатели платят только минимальную из выигравших цен.

Оригинальным видом аукционов считается также *аукцион Викри*. В нем побеждает вторая по величине цена. Доминантной стратегией для частных ценностей является называние честной цены. Этот аукцион не способствует построению выигрышных стратегий, а люди их не любят.

Биржи

Биржи являются естественным выбором в случае, когда много продавцов и много покупателей. В отличие от аукционов здесь нет понятия аукционист. Все участники могут быть как продавцами, так и покупателями.

Аукционы – это частный случай биржи.

Администратор биржи определяет набор предметов $M = \{1, 2, \ldots, m\}$, которые доступны для торговли.

Ставка (*bid*) – это совокупность вида

$Bj=((\lambda_{1j},\lambda_{2j},\ldots\lambda_{mj}),p_j)$ где λ_{ij} – количество предметов вещи i, цена p_i.

Если $p_j>0$, то ставка – это запрос на продажу, иначе – на покупку.

Должен быть максимизирован остаток при условии, что спрос не превышает предложение. Проблема состоит в разбиении всех биржевых ставок на выигравшие и проигравшие.

Биржевые аукционы широко применяются в самых различных областях: при продаже недвижимости, при распределении радиочастот, времени прилета/вылета самолетов, при составлении маршрута движения транспортных средств и т.д.

Основные сферы применения МАС

Многоагентные системы применяются 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.

В заключение отметим следующее. В ранние годы ИИ были популярными споры на тему: «Может ли машина мыслить?» и «Что такое интеллектуальная

система?». Аналогично, распространение концепции агентов и МАС привело к появлению таких вопросов как «Что такое интеллектуальный агент?», «Чем отличаются агенты от обычных программ или манипуляционных роботов?», «В чем различие между МАС и ЭС?».

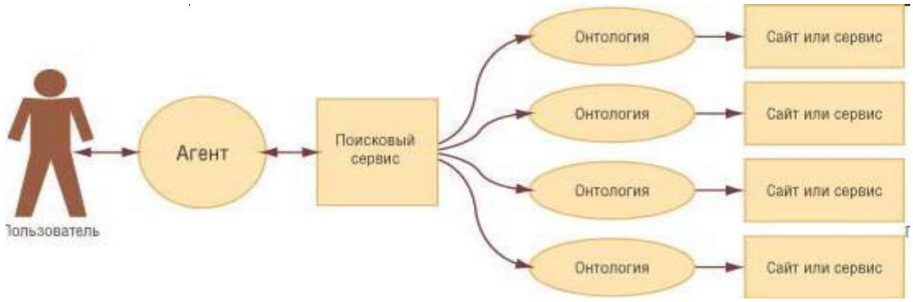
Компьютерная программа или робот обретает статус агента тогда, когда у них имеются средства оперативного восприятия и интерпретации изменений среды, а также планирования и организации действий. Но это предполагает наличие механизмов мотивации, целеобразования, предвидения и пр.

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.

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.



Для чего нужны онтологии? -

- 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;
- для анализа знаний в предметной области.

Формальное определение онтологии

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:

• Простой 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.

Пример. 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).

(в скобках обычно указываются типы значений и разрешенные значения).

Класс *«Книга»* имеет следующие атрибуты:

• название (строка);

• автор (строка);

• ISBN (строка специального формата);

• число страниц (натуральное число);

• тип обложки (строка; возможные значения: мягкая, твердая, суперобложка);

• издательство (экземпляр класса «издатель»);

• год издания (натуральное число — четыре цифры);

• описание (текст);

• цена (число с плавающей точкой — два знака после запятой).

Класс *«Журнал»:*

• 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:

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.

Процесс разработки онтологии

В общем виде процесс разработки онтологии включает следующие этапы:

• составляется глоссарий терминов (понятий);

• на ЕЯ 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;

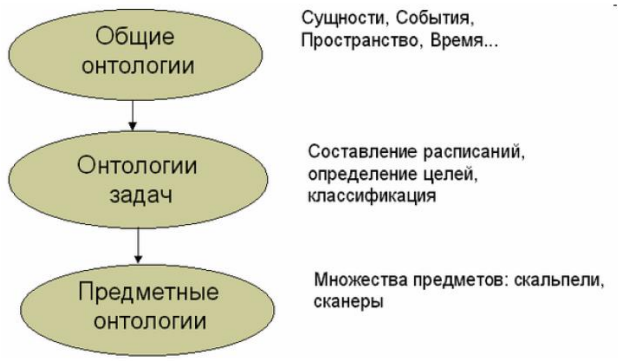
• в зависимости от целей, для которых разрабатывается онтология, в нее могут добавляться экземпляры классов;

• 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.

Классификация онтологий по содержимому

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



В общем случае онтологическая система представляет собой совокупность трех множеств:

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

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 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.

Образ, класс *– классификационная группировка в системе классификации, объединяющая (выделяющая) определенную группу объектов по некоторому признаку*.

Образы обладают характерным свойством, проявляющимся в том, что ознакомление с конечным числом явлений из одного и того же множества дает возможность узнавать сколь угодно большое число его представителей. Примерами образов могут быть: река, море, жидкость, музыка Чайковского, стихи Маяковского, буквы А, Б и т. д. В качестве образа можно рассматривать и некоторую совокупность состояний объекта управления, причем вся эта совокупность состояний характеризуется тем, что для достижения заданной цели требуется одинаковое воздействие на объект. Образы обладают характерными объективными свойствами в том смысле, что разные люди, обучающиеся на различном материале наблюдений, большей частью одинаково и независимо друг от друга классифицируют одни и те же объекты.

7. Системы и методы распознавания образов.

В целом проблема распознавания образов состоит из двух частей: обучения и распознавания. Обучение осуществляется путем показа отдельных объектов с указанием их принадлежности тому или другому образу. В результате обучения распознающая система должна приобрести способность реагировать одинаковыми реакциями на все объекты одного образа и различными – на все объекты различных образов. Очень важно, что процесс обучения должен завершиться только путем показов конечного числа объектов без каких-либо других подсказок. В качестве объектов обучения могут быть либо картинки, либо другие визуальные изображения (буквы), либо различные явления внешнего мира, например звуки, состояния организма при медицинском диагнозе, состояние технического объекта в системах управления и др. Важно, что в процессе обучения указываются только сами объекты и их принадлежность образу. За обучением следует процесс распознавания новых объектов, который характеризует действия уже обученной системы. Автоматизация этих процедур и составляет проблему обучения распознаванию образов. В том случае, когда человек сам разгадывает или придумывает, а затем навязывает машине правило классификации, проблема распознавания решается частично, так как основную и главную часть проблемы (обучение) человек берет на себя.

Проблема обучения распознаванию образов интересна как с прикладной, так и с принципиальной точки зрения. С прикладной точки зрения решение этой проблемы важно прежде всего потому, что оно открывает возможность автоматизировать многие процессы, которые до сих пор связывали лишь с деятельностью живого мозга. Принципиальное значение проблемы тесно связано с вопросом, который все чаще возникает в связи с развитием идей кибернетики: что может и что принципиально не может делать машина?

Круг задач, которые могут решаться с помощью распознающих систем, чрезвычайно широк. Сюда относятся не только задачи распознавания зрительных и слуховых образов, но и задачи распознавания сложных процессов и явлений, возникающих, например, при выборе целесообразных действий руководителем предприятия или выборе оптимального управления технологическими, экономическими, транспортными или военными операциями. В каждой из таких задач анализируются некоторые явления, процессы, состояния внешнего мира, всюду далее называемые объектами наблюдения.

При решении задач управления методами распознавания образов вместо термина "изображение" применяют термин "состояние". Состояние – это определенной формы отображение измеряемых текущих (или мгновенных) характеристик наблюдаемого объекта. Совокупность состояний определяет ситуацию. Понятие "ситуация" является аналогом понятия "образ". Но эта аналогия не полная, так как не всякий образ можно назвать ситуацией, хотя всякую ситуацию можно назвать образом.

Классификация на основе решающих функций

Основным назначением системы распознавания образов является отыскание решений о принадлежности предъявляемых ей образов некоторому классу. Один из важных подходов к этой задаче предполагает использование решающих функций. Проиллюстрируем этот подход с помощью рисунка 7.1:

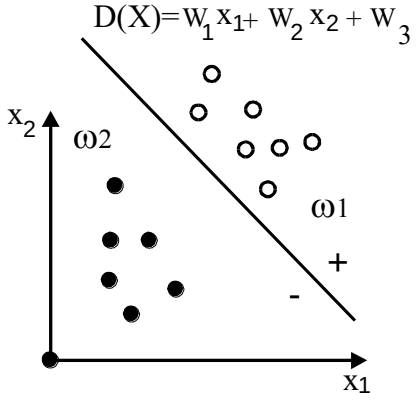


Рис.7.1. Использование решающих функций

Из рисунка видно, что две совокупности образов удобно разделить прямой.

Пусть $d(X)=w_1x_1+w_2x_2+w_3=0$ - уравнение разделяющей прямой, где w_i - параметры, а x_1 и x_2 -переменные. Из рисунка следует, что подстановка в $d(X)$ любого образа X, принадлежащего классу ω_1, даст положительное значение. Отрицательное значение функция $d(X)$ примет при подстановке образа, относящегося к классу ω_2. Таким образом, функцию $d(X)$ можно использовать в качестве решающей (или дискриминантной) функции, поскольку, рассматривая образ X, классификация которого неизвестна, можно утверждать, что образ X принадлежит классу ω_1, если $d(X)>0$, и классу ω_2, если $d(X)<0$. Если образ лежит на разделяющей границе, то имеет место случай, соответствующий неопределенности $d(X)=0$.

Успех применения описанной схемы распознавания образов зависит от двух факторов: 1) вида функции $d(X)$ и 2) практической возможности определения ее коэффициентов. Первый из них непосредственно связан с геометрическими свойствами рассматриваемых классов.

Рассмотренный вариант линейной решающей функции легко обобщить на n–мерный случай. Общий вид линейной решающей функции задается формулой:

$$d(X)=w_1x_1+w_2x_2+...+ w_nx_n+w_{n+1}=W'_0X+ w_{n+1} , \qquad (7.1)$$

где вектор $W_0 = (w_1, w_2,..., w_n)'$ называется весовым или параметрическим.

Выражение (2.1) лучше представлять в виде:

$$d(X)= W'X, \qquad (7.2)$$

где $X=(x_1,x_2,...,x_n,1)'$ и $W=(w_1,w_2,...,w_n,w_{n+1})'$ - пополненные векторы образов и весов соответственно.

Предполагается, что в случае разбиения на два класса решающая функция $d(X)$ обладает свойством:

$$d(X) = W'X \begin{cases} > 0, \textbf{if } \boldsymbol{X} \in \omega_1, \\ < \boldsymbol{0}, \textbf{if } \boldsymbol{X} \in \omega_2. \end{cases} \qquad (7.3)$$

Рассмотрим случаи разбиения на несколько классов ω_1, ω_2, ..., ω_M, т.е. предполагается, что объекты принадлежат более чем двум классам.

Случай 1. Каждый класс отделяется от всех остальных одной разделяющей поверхностью. В этом случае существует M решающих функций, обладающих свойством:

$$d_i(X) = W_i' X \begin{cases} > 0, \textbf{if } \boldsymbol{X} \in \omega_i, \\ < \boldsymbol{0}, \textbf{if } \boldsymbol{X} \notin \omega_i. \end{cases}, \; i=1, 2,\ldots,M, \qquad (7.4)$$

где $W_i=(w_{i1},w_{i2},...,w_{in},w_{i,n+1})'$ – весовой вектор, соответствующий i-й решающей функции.

Пример 1. Пример, иллюстрирующий случай 1, приведен на Рис.7.2.

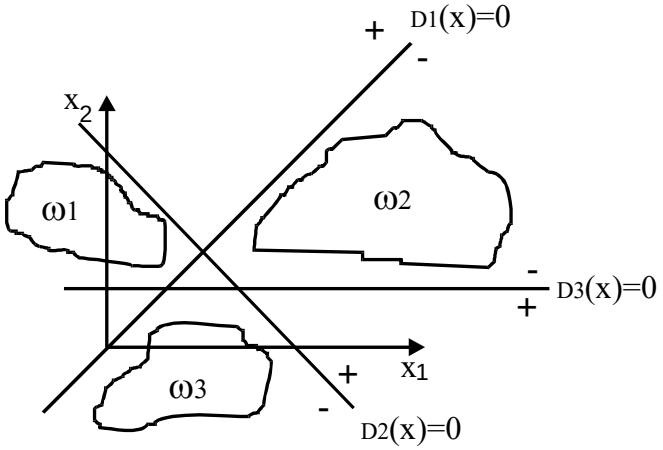


Рис.7.2.

Здесь каждый класс можно отделить от всех остальных с помощью одной разделяющей границы. Так, если некоторый образ X принадлежит классу ω_1, на основании геометрических соображений заключаем, что $d_1(X)>0$, а $d_2(X)<0$ и $d_3(X)<0$. Граница, отделяющая класс ω_1 от остальных, определяется значениями X, при которых $d_1(X)=0$.

Пусть решающие функции, соответствующие рис.2.2, имеют вид:

$$d_1(X)= -x_1+x_2,\ d_2(X)=x_1+x_2-5,\ \text{ и } d_3(X)=-x_2+1.$$

Следовательно, три разделяющие границы определяются уравнениями:

$$-x_1+x_2=0,\quad x_1+x_2-5=0,\quad -x_2+1=0.$$

Итак, любой образ, для которого выполняются условия $d_1(X)>0$, а $d_2(X)<0$ и $d_3(X)<0$ автоматически зачисляется в класс ω_1. (Рис.7.3). Хотя класс $\boldsymbol{\omega_1}$ занимает сравнительно небольшой участок, в действительности область, соответствующая решению об отнесении объекта к данному классу, безгранична.

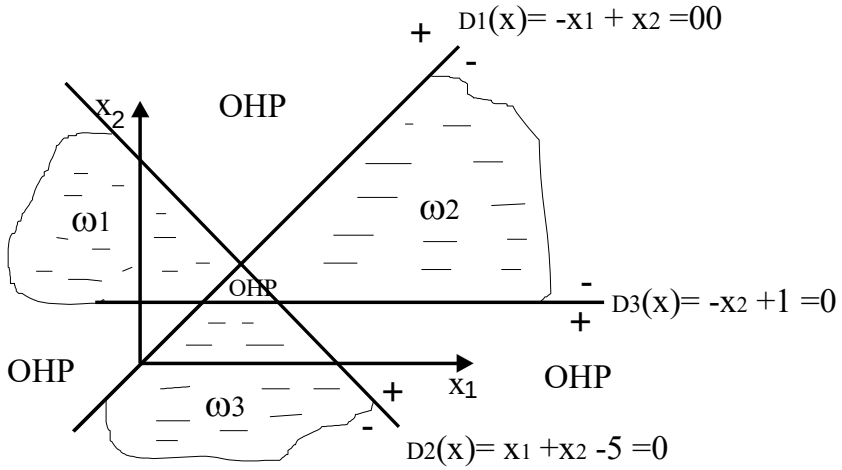


Рис.7.3.

Отметим, что если функция $d_i(X)$ больше нуля при более одном значении i, рассматриваемая схема классификации не позволяет найти решение. Это справедливо также и при $d_i(X)$ для всех i. В данном примере существуют 4 области неопределенности, соответствующие одной из этих ситуаций.

Отнесение неклассифицированного объекта к одному из трех классов производится непосредственным образом. Пусть нам надо классифицировать образ $X=(6,5)'$. Подставляем его параметры в наши три решающие функции, получаем:

$$d_1(X)=-1<0,\ d_2(X)=6>0,\ d_3(X)=-4<0.$$

Вывод: образ X зачисляется в класс ω_2.

Случай 2. Каждый класс отделяется от любого другого взятого в отдельности класса «индивидуальной» разделяющей поверхностью, т.е. классы попарно разделимы. В этом случае существует $C(M,2)=M\times(M-1)/2$ разделяющих поверхностей. Решающие функции имеют вид: $d_{ij}(X)= W_{ij}'X$ и обладают тем свойством, что если образ X принадлежит классу ω_i, то

$d_{ij}(X)>0$ для всех $j\neq i$, и кроме того, $d_{ij}(X)= -d_{ji}(X)$.

Также не редки случаи, представляющие собой комбинацию случаев 1 и 2. Для их решения требуется менее $M\times(M-1)/2$ разделяющих поверхностей, необходимых в той ситуации, когда все классы разделимы только попарно.

Пример 2. На рис. 7.4 представлены три класса образов, разделимых согласно случаю 2.

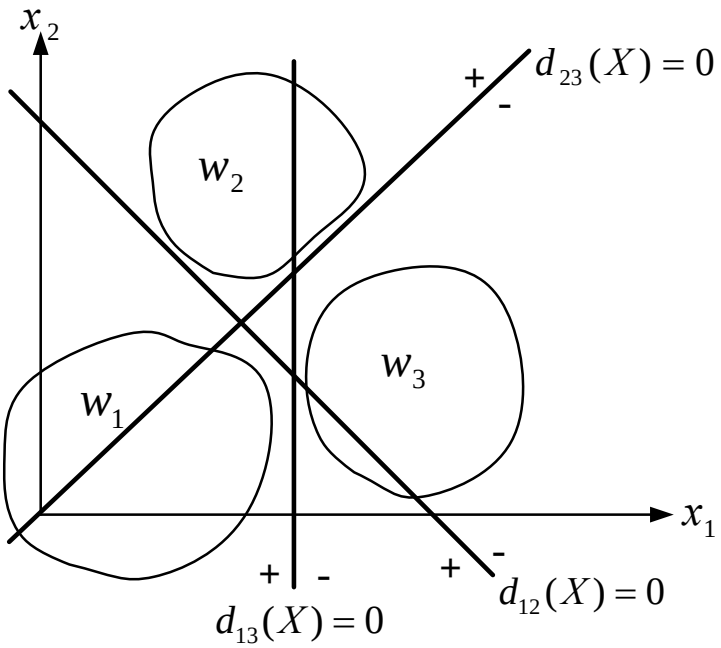


Рис.7.4.

Очевидно, что ни один класс нельзя отделить от всех остальных с помощью единственной разделяющей поверхности. Каждая из приведенных на рисунке границ обеспечивает разделение точно двух классов. Так, например, хотя граница $d_{12}(X)=0$ проходит через класс ω_3, она дает эффективное разделение лишь для классов ω_1, и ω_2.

Пусть решающие функции имеют вид: $d_{12}(X)= -x_1-x_2+5$, $d_{13}(X)= -x_1+3$, $d_{23}(X)= -x_1-+x_2$.

Разделяющие границы снова получим, приравнивая решающие функции к нулю. Области решений, однако, теперь могут содержать несколько зон, где соответствующие функции положительны. В частности, область, отвечающая классу ω_1, определяется значениями образа X, при которых $d_{12}(X)>0$ и $d_{13}(X)>0$. Значение решающей функции $d_{23}(X)$ в этой области не существенно, поскольку эта функция никак не связана с классом ω_1. Области, определяемые тремя указанными решающими функциями, представлены на рис.7.5, причем для выделения областей, соответствующих разным классам, использовано условие $d_{ij}(X)= -d_{ji}(X)$.

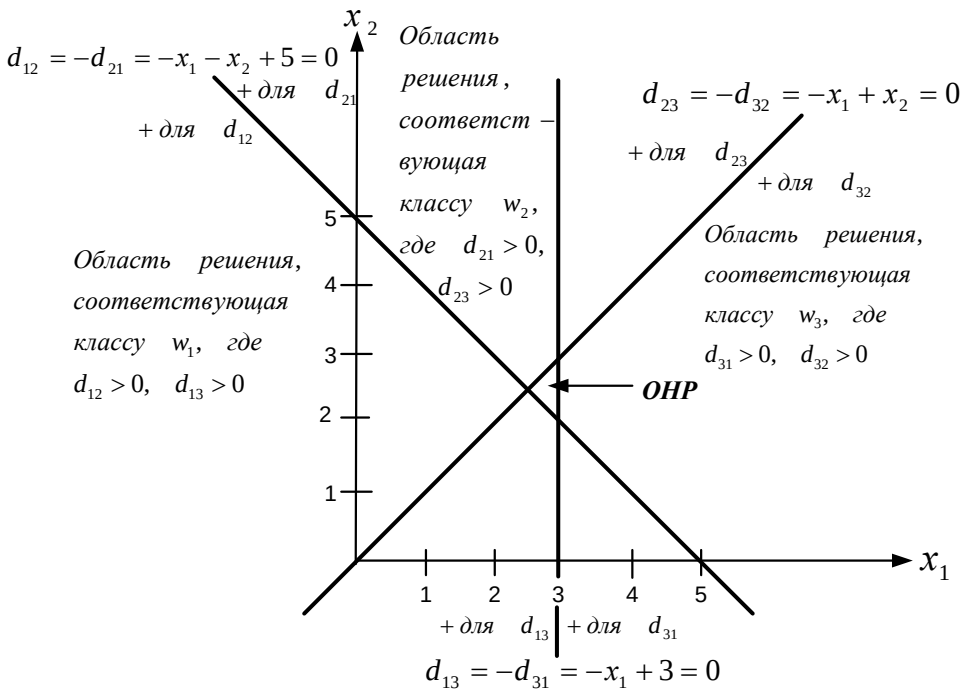


Рис.7.5.

Так, поскольку $d_{12}(X)= -x_1-x_2+5$, то $d_{21}(X)= +x_1+x_2-5$ и зона положительности функции $d_{12}(X)$ совпадает с зоной отрицательности функции $d_{21}(X)$. Как и в случае 1, оказывается, что области решения безграничны и существуют области неопределенности, в которых условия случая 2 не выполняются.

Отнесение неклассифицированного объекта к одному из трех классов также производится непосредственным образом. Пусть нам надо классифицировать образ $X=(4,3)'$. Подставляем его параметры в наши решающие функции, получаем: $d_{12}(X)=-2$, $d_{13}(X)=-1$, $d_{23}(X)=-1$. Отсюда автоматически следует, что $d_{21}(X)=+2$, $d_{31}(X)=+1$, $d_{32}(X)=+1$. Так как $d_{3j}(X)>0$ для всех $j=1,2$ и в область неопределенности мы не попали, то образ X зачисляется в класс ω_3.

Случай 3. Существует M решающих функций: $d_k(X)= W_k'X$, $k=1,2,...,M$, таких, что если образ X принадлежит классу ω_i, то

$$d_i(X)> d_j(X) \text{ для всех } j \neq i. \qquad (7.5)$$

Эта ситуация является разновидностью случая 2, поскольку можно положить

$$d_{ij}(X)=d_i(X)-d_j(X) = (w_i-w_j)'X = w'_{ij}X, \qquad (7.6)$$

где $w_{ij}=w_i-w_j$. Можно убедиться, что если $d_i(X)>d_j(X)$ для всех $j \neq i$, то $d_{ij}(X)>0$ для всех $j \neq i$, т.е. если классы разделимы, как в случае 3, то они автоматически разделимы и как в случае 2. Обратное, вообще говоря, не верно.

Пример 3. На рис. 7.6 представлены три класса образов, разделимых согласно случаю 3.

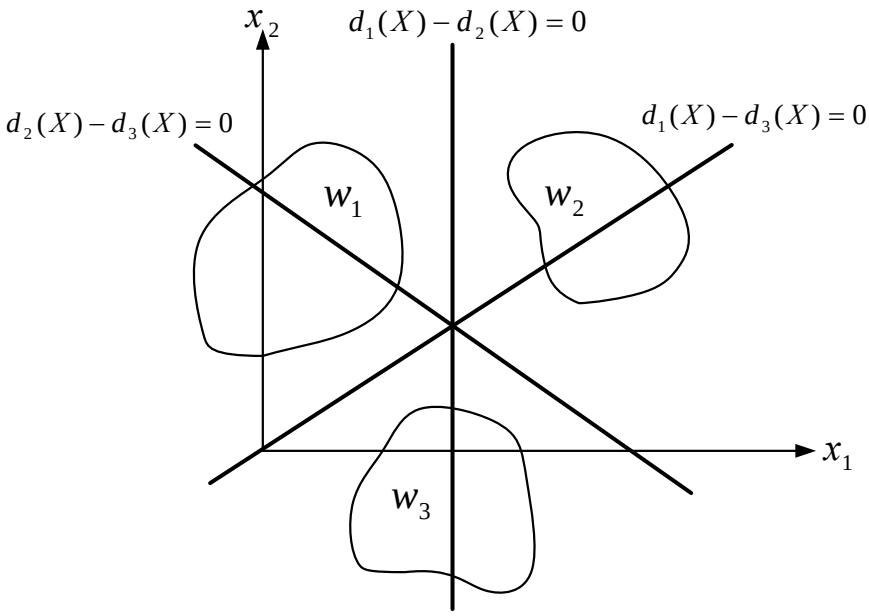


Рис.7.6.

Для образов, принадлежащих классу ω_1, должны выполняться условия $d_1(X)>d_2(X)$ и $d_1(X)>d_3(X)$. Это эквивалентно требованию того, чтобы входящие в данный класс образы располагались в положительных зонах поверхностей $d_1(X)-d_2(X)=0$ и $d_1(X)-d_3(X)=0$.

Пусть в качестве решающих функций выбраны следующие:

$$d_1(X)= -x_1+x_2,\ d_2(X)=x_1+x_2-1,\ d_3(X)= - x_2.$$

Разделяющие границы для трех классов выглядят при этом так:

$$d_1(X)-d_2(X)=-2x_1+1=0,$$
$$d_1(X)-d_3(X)=-x_1+2x_2=0,$$
$$d_2(X)-d_3(X)=x_1+2x_2-1=0.$$

Для того, чтобы определить область решений, соответствующую классу ω_1, необходимо выделить область, в которой выполняются неравенства $d_1(X)>d_2(X)$ и $d_1(X)>d_3(X)$. Эта область совпадает с положительными зонами для прямых $-2x_1+1=0$ и $-x_1+2x_2=0$. (Рис.7.7):

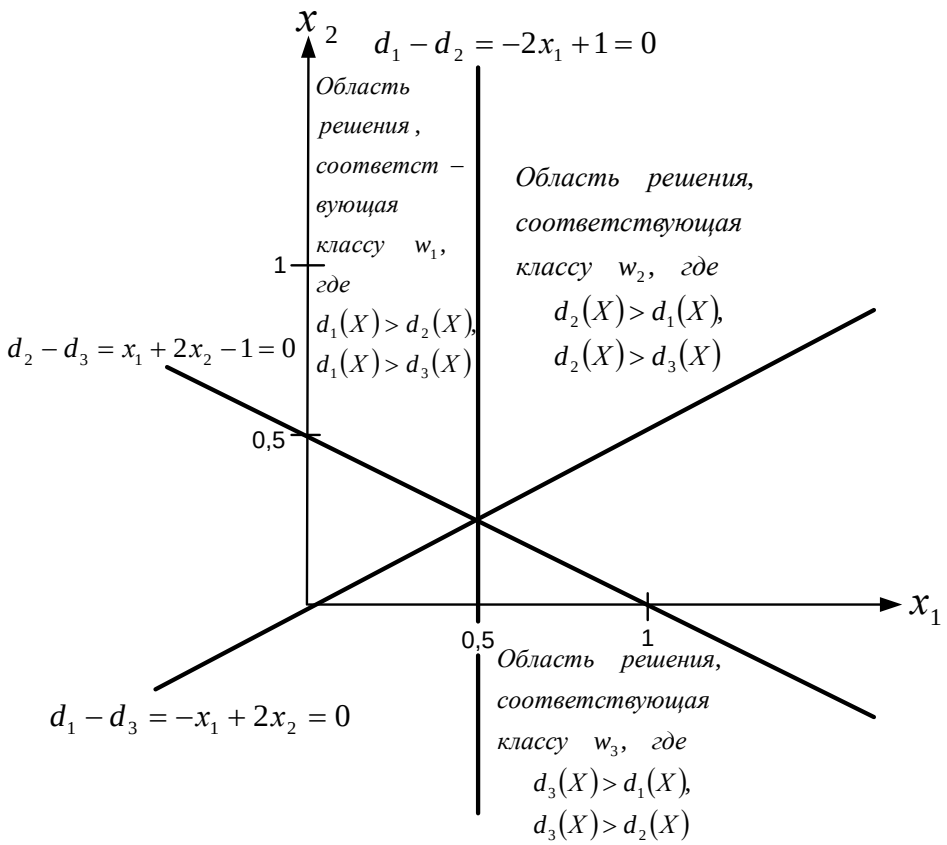


Рис.7.7.

Область принятия решения о принадлежности образа классу ω_2 совпадает с положительными зонами для прямых $2x_1-1=0$ и $x_1+2x_2-1=0$. Область, отвечающая классу ω_3 совпадает с положительными зонами для прямых $x_1-2x_2=0$ и $-x_1-2x_2+1=0$. Также отметим, что области неопределенности как таковые в случае 3 отсутствуют (за исключением собственно разделяющих границ).

Пусть нам надо классифицировать образ $X=(1,1)'$. Подстановка компонент этого вектора в выбранные решающие функции дает: $d_1(X)=0$, $d_2(X)=1$, $d_3(X)= -1$. Вывод: образ относится к классу ω_2.

Классификация образов с помощью функций расстояния

Выбор функций расстояния в качестве инструмента классификации является естественным следствием того, что наиболее очевидный способ введения меры сходства для образов, интерпретируемых нами также как точки в евклидовом пространстве, - определение их близости. Рассмотрим рис.7.9:

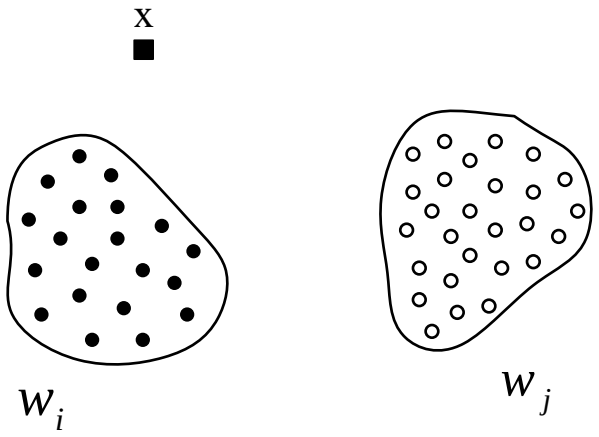


Рис.7.9.

Здесь можно прийти к интуитивному выводу о принадлежности вектора X классу ω_i исключительно из тех соображений, что этот вектор находится ближе к векторам образов класса ω_i.

Можно рассчитывать на получение удовлетворительных результатов при классификации образов с помощью функций расстояния только в тех случаях, когда классы образов обнаруживают тенденцию к проявлению кластеризационных свойств. Это обстоятельство можно оценить, сопоставив рис.7.9 и рис.7.10, на котором представлены образы, классификация которых с помощью понятия близости вызывает затруднения.

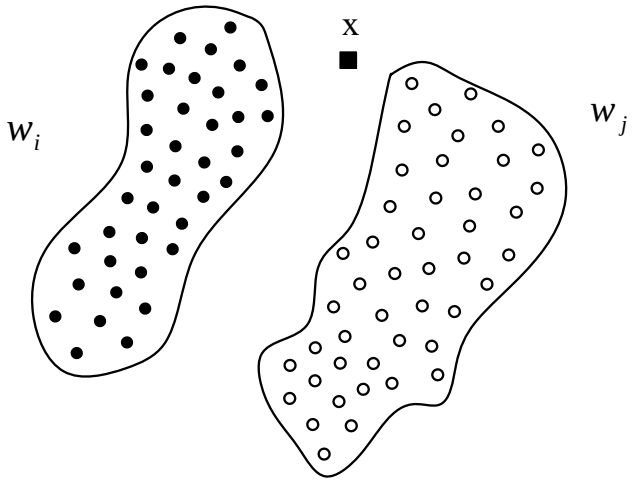


Рис.7.10.

Поскольку близость классифицируемого образа к образам некоторого класса используется здесь в качестве критерия для его классификации, такой подход называется *классификацией образов по критерию минимума расстояния*.

1.Случай единственности эталона. В некоторых случаях образы любого из рассматриваемых классов проявляют тенденцию к тесной группировке вокруг некоторого образа, являющегося типичным или репрезентативным для соответствующего класса. Подобные ситуации возникают, если изменчивость

образов невелика, а помехи легко поддаются учету. Типичным примером этого служит задача считывания банковских (и других) чеков с помощью ЭВМ.

Рассмотрим M классов. Пусть эти классы допускают представление с помощью эталонных образов z_1, z_2,…, z_M. Евклидово расстояние между произвольным вектором образа X и i–м эталоном определяется выражением:

$$D_i = \|X - z_i\| = \sqrt{(X - z_i)' \times (X - z_i)}\,. \qquad (7.7)$$

Классификатор, построенный по принципу минимума расстояния, вычисляет расстояние, отделяющее неклассифицированный образ X от эталона каждого класса, и зачисляет этот образ в класс, оказавшийся ближайшим к нему. Другими словами, образ X приписывается к классу ω_i, если условие $D_i<D_j$ выполняется для всех $j\neq i$. Случаи равенства расстояний разрешаются произвольным образом.

Формуле (2.7) можно придать более удобный вид. Возведение D_i в квадрат дает:

$$\begin{aligned} D^2{}_i &= \|X - z_i\|^2 = (X - z_i)' \times (X - z_i) = \\ X'X - 2X'z_i + z_i'z_i &= X'X - 2(X'z_i - \frac{1}{2} z_i'z_i). \end{aligned} \qquad (7.8)$$

Выбор минимального значения D_i^2 эквивалентен выбору минимального D_i, поскольку все расстояния – положительные величины. Формула (7.8) показывает, что выбор минимального значения D_i^2 эквивалентен выбору максимального значения разности $(X'z_i - \frac{1}{2} z_i'z_i)$,поскольку при вычислении любых D_i^2, член $X'X$ не зависит от значения i. Следовательно, решающие функции можно определить как:

$$d_i(X)= X'z_i-(1/2)z_i'z_i, \qquad i=1,2,\ldots,M, \qquad (7.9)$$

где образ X относится к классу ω_i, если условие $d_i(X)>d_j(X)$ справедливо для всех $j\neq i$.

Отметим, что $d_i(X)$ - линейная решающая функция, т.е, если z_{ij}, $j=1,2,\ldots,n$, - компоненты вектора z_i , причем:

$$\begin{aligned} w_{ij}&=z_{ij}, \ j=1,2,\ldots,n, \\ w_{i,n+1}&=-(1/2)z_i'z_i \end{aligned} \qquad (7.10)$$

и вектор $X=(x_1,x_2,..,x_n,1)'$, то (2.9) можно представить в обычной линейной форме:

$$d_i(X) = w_i'X, \ i=1,2,\ldots,M, \qquad (7.11)$$

где $w_i=(w_{i1},w_{i2},\ldots,w_{i,n+1})'$.

На рис.7.11 изображена разделяющая граница для примера с двумя классами, каждый из которых задается единственным эталоном.

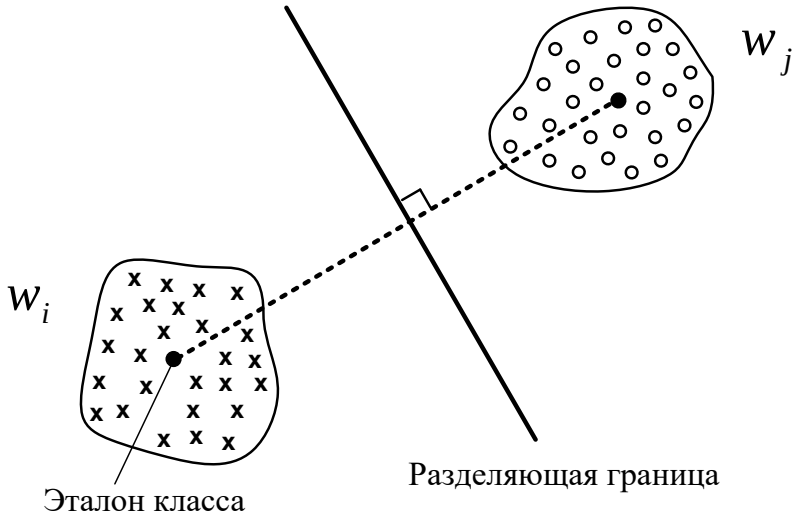


Рис.7.11.

Можно показать, что линейная разделяющая поверхность, обеспечивающая разделение всех пар эталонных точек z_i и z_j , является гиперплоскостью, которая представляет собой *геометрическое место точек, равноудаленных от этих двух эталонных точек*. Т.о., классификаторы, основанные на принципе минимального расстояния, представляют собой частный случай линейного классификатора, разделяющие границы которого должны обладать этим свойством.

2.Множественность эталонов. Допустим, что каждый класс можно охарактеризовать не единственным, а несколькими эталонными образами, т.е. любой образ, принадлежащий классу ω_i, проявляет тенденцию к группировке вокруг одного из эталонов z_i^1, z_i^2,…, z_i^{Ni}, где N_i– количество эталонных образов, определяющий i–й класс. В этом случае можно воспользоваться классификатором, подобным рассмотренному в предыдущем пункте. Запишем функцию, определяющую расстояние между произвольным образом X и классом ω_i, в виде:

$$D_i = \min_l \| X - z_i^l \|, \quad l = 1,2,...,N_i . \tag{7.12}$$

это означает, что D_i - наименьшее из расстояний от образа X до каждого эталона класса ω_i. Как и ранее вычисляются значения расстояний D_i, $i=1,2,…,M$, и классифицируемый образ зачисляется в класс ω_i, если условие $D_i < D_j$ справедливо для всех $j \neq i$. В случае равенства расстояний решение принимается произвольным образом.

Учитывая выражение (7.9), получаем:

$$d_i(X) = \max_l \{(X' z_i^l) - (1/2)(z_i^l)' z_i^l\}, \quad l = 1,2,...,N_i . \tag{7.13}$$

Как и раньше, образ X зачисляется в класс ω_i, если условие $d_i > d_j$ справедливо для всех $j \neq i$.

На рис.7.12 представлены разделяющие границы для случая двух классов, когда класс имеет два эталона.

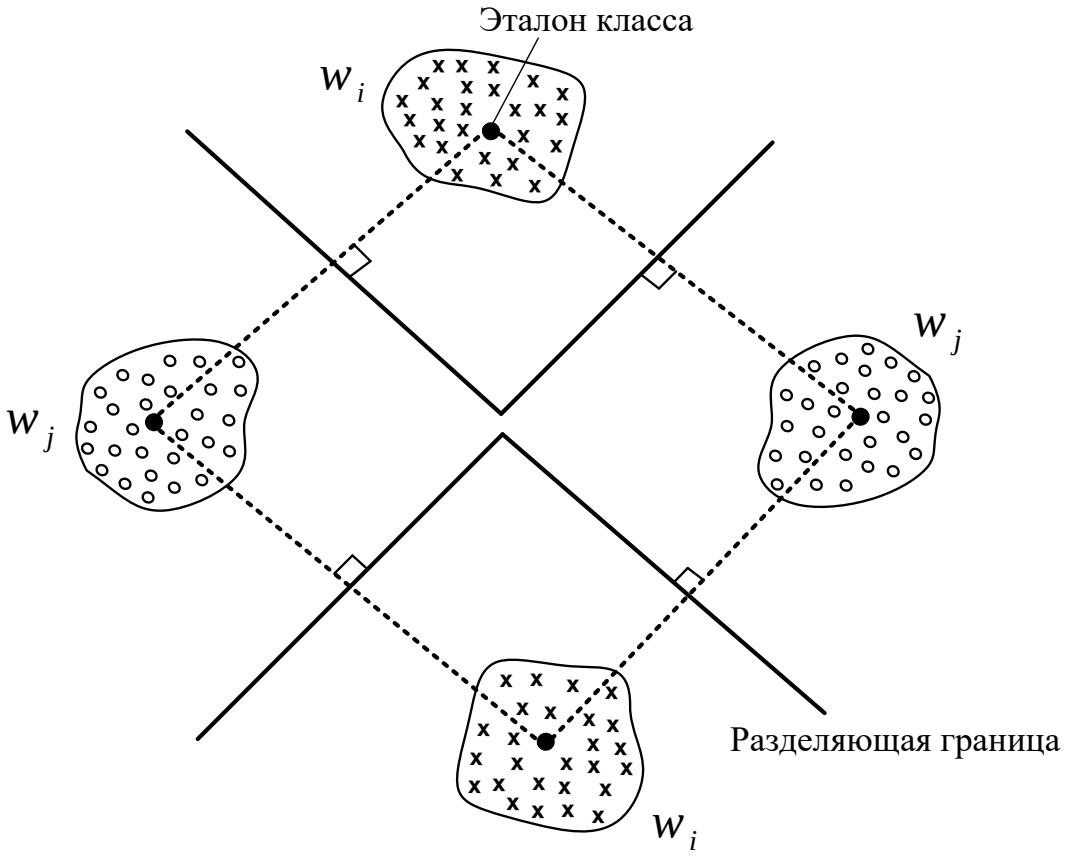


Рис.7.12.

Здесь границы, разделяющие классы ω_i и ω_j, являются кусочно-линейными. Этот случай можно было бы интерпретировать как задачу о разбиении на четыре класса, каждый их которых обладает единственным эталоном, тогда участки границ представляют собой геометрические места точек, равноудаленных от прямых, соединяющих эталоны различных классов.

Обобщение принципов классификации по минимуму расстояния

Рассмотрим выборку образов с известной классификацией $\{s_1, s_2, \ldots, s_N\}$, причем предполагается, что каждый образ выборки входит в один из классов $w_1, w_2, \ldots, w_M$. Можно определить правило классификации, основанное на принципе *ближайшего соседа* (БС-правило). Это правило относит классифицируемый образ к классу, к которому принадлежит его ближайший сосед, причем образ s_i называется ближайшим соседом образа X, если

$$D(s_i, X) = \min_l \{D(s_l, X)\}, \; l = 1,2,\ldots,M, \qquad (7.14)$$

где D - любое расстояние (в частности евклидово), определение которого допустимо на пространстве образов.

Эту процедуру классификации иногда называют 1-БС-правилом, т.к. при ее применении учитывается принадлежность некоторому классу только одного

ближайшего соседа образа *X*. Также можно ввести *q*–БС-правила, предусматривающие определение *q* ближайших к *X* образов и зачисление его в тот класс, к которому относится наибольшее число образов, входящих в эту группу.

Выявление кластеров

Умение находить в заданном наборе данных эталоны или центры кластеров играют главную роль в построении классификаторов образов по принципу манимума расстояния. Выявление кластеров во многих отношениях является «искусством» довольно эмпирическим, т.к. качество работы определенного алгоритма зависит не только от характера анализируемых данных, но также определяется выбранной мерой подобия образов и методом, используемым для идентификации кластеров в системе данных.

Меры сходства. Для того чтобы определить на множестве данных кластер, необходимо вначале ввести меру сходства, которая может быть положена в основу правила отнесения образов к области, характеризуемой некоторым центром кластера. Ранее рассматривалось евклидово расстояние между образами *X* и *z*:-

$$D=||X-z||. \qquad (7.15)$$

Эта характеристика использовалась в качестве меры сходства соответствующих объектов: чем меньше расстояние, тем больше сходство.

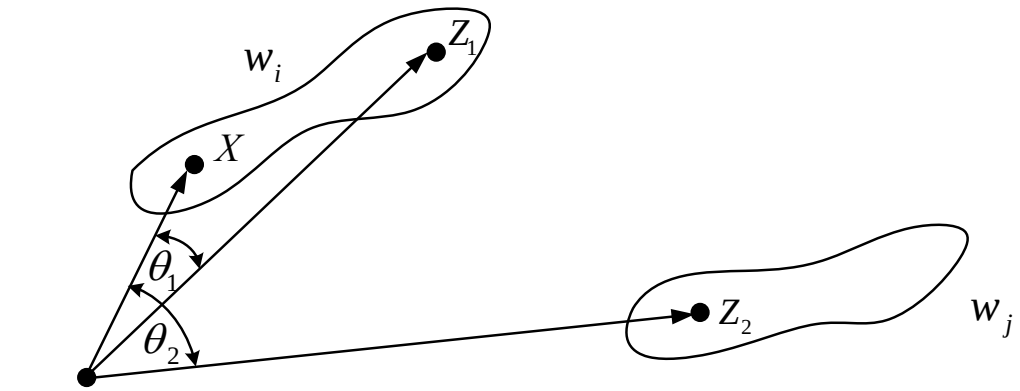
Меры сходства не исчерпываются расстояниями. В частности, может использоваться непараметрическая функция сходства:

$$s(X,z)=\frac{X'\times z}{\| X \| \times \| z \|}, \qquad (7.16)$$

представляющая собой косинус угла, образованного векторами X и z, и достигающую максимума, когда их направления совпадают. Этой мера сходства используется в тех случаях, когда кластеры обнаруживают тенденцию располагаться вдоль главных осей (Рис.7.13).

Из рис.7.13 видно, что образ z_1 обладает большим сходством с образом X, чем образ z_2, поскольку значение функции $s(X,z_1)$ больше значения $s(X,z_2)$. Использование данной меры сходства связано с определенными ограничениями, в частности, достаточное отстояние кластеров друг от друга и от начала координат.

Если рассматриваются двоичные образы и их элементы принимают значения {0,1}, то функции сходства можно дать следующую интерпретацию.



$$S(X,Z_1)=\cos\theta_1=\frac{X'Z_1}{\|X\|\|Z_1\|}$$

$$S(X,Z_2)=\cos\theta_2=\frac{X'Z_2}{\|X\|\|Z_2\|}$$

Рис. 7.13.

Если x_i=1, считается что двоичный образ X обладает i–м признаком. Тогда числитель в (7.16) определяет число общих для образов X и z признаков, а знаменатель - среднее геометрическое числа признаков, которыми обладает образ X, и числа признаков, которыми обладает образ z. Поэтому, функция s(X,z) есть мера наличия общих признаков у двоичных векторов X и z.

Критерии кластеризации

Поскольку близость двух образов является относительной мерой их подобия, обычно вводят порог, чтобы установить приемлемые степени сходства для процесса отыскания кластеров. Подход к кластеризации, предусматривающий использование показателя качества, связан с разработкой процедур, которые обеспечат минимазацию или максимизацию выбранного показателя качества. Одним из таких показателей является сумма квадратов ошибки:

$$J=\sum_{j=1}^{N_C}\sum_{X\in S_j}\| X-m_i \|^2 , \qquad (7.17)$$

где N_C – число кластеров, S_j- множество образов, относящихся к j–му кластеру, а величина

$$m_j=\frac{1}{N_j}\sum_{X\in S_j}X , \qquad (7.18)$$

- вектор выборочных средних значений для множества S_j, величина N_j характеризует количество образов, входящих во множество S_j.

Показатель качества (7.17) определяет общую сумму квадратов отклонений характеристик всех образов, входящих в некоторый кластер, от соответствующих средних значений по кластеру.

Помимо рассмотренного существует масса других показателей качества. Среди них: среднее квадратов расстояний между образами в кластере, среднее квадратов расстояний между образами, входящими в разные кластеры и др.

Простой алгоритм выявления кластеров. Пусть задано множество *N* образов $\{x_1,x_2,\ldots,x_n\}$. Пусть также центр первого кластера z_1 совпадает с любым из заданных образов и определена произвольная неотрицательная пороговая величина *T*. Для удобства можно считать, что $z_1=x_1$. После этого вычисляется расстояние D_{21} между образом x_2 и центром кластера z_1 по формуле $D_{21}=\|x_2-z_1\|$. Если это расстояние больше значения пороговой величины *T*, то учреждается новый центр кластера $z_2=x_2$. В противном случае образ x_2 включается в кластер, центром которого является z_1. Пусть условие $D_{21}>T$ выполнено, т.е. z_2 – центр нового кластера. На следующем шаге вычисляются расстояния D_{31} и D_{32} от образа x_3 до центров кластеров z_1 и z_2. Если оба расстояния оказываются больше порога *T*, то учреждается новый центр кластера $z_3=x_3$. В противном случае образ x_3 зачисляется в тот кластер, чей центр к нему ближе. Подобным же образом расстояния от *каждого* нового образа до каждого известного центра кластера вычисляются и сравниваются с пороговой величиной – если все эти расстояния превосходят значение порога *T*, учреждается новый центр кластера. В противном случае образ зачисляется в кластер с самым близким к нему центром.

Результаты описанной процедуры определяются выбором первого центра, порядком осмотра образов, значением пороговой величины *T* и, конечно, геометрическими характеристиками данных. Несмотря на эти недостатки, алгоритм позволяет просто и быстро получить приблизительные оценки основных характеристик заданного набора данных. Кроме того, этот алгоритм привлекателен с вычислительной точки зрения, т.к. для выявления центров кластеров, соответствующих определенному значению порога *T*, ему требуется только однократный просмотр выборки.

Алгоритм максиминного расстояния. Данный алгоритм представляет собой еще одну эвристическую процедуре, использующую евклидово расстояние. Этот алгоритм в принципе аналогичен предыдущему, за исключением того обстоятельства, что он в первую очередь выявляет наиболее удаленные кластеры. Рассмотрим алгоритм на конкретном примере.

Возьмем выборку, состоящую из 10-ти двумерных образов (Рис.7.***14***).

Первая из них содержит выборочные образы, другая – список центров кластеров, установленных алгоритмом. До начала работы алгоритма вторая таблица – пуста.

На первом шаге алгоритма один из объектов, например x_1, произвольным образом назначается центром первого кластера (обозначим его z_1). Над стрелками будем ставить цифры, обозначающие порядковый номер шага, на котором производится выделение соответствующего центра кластера.

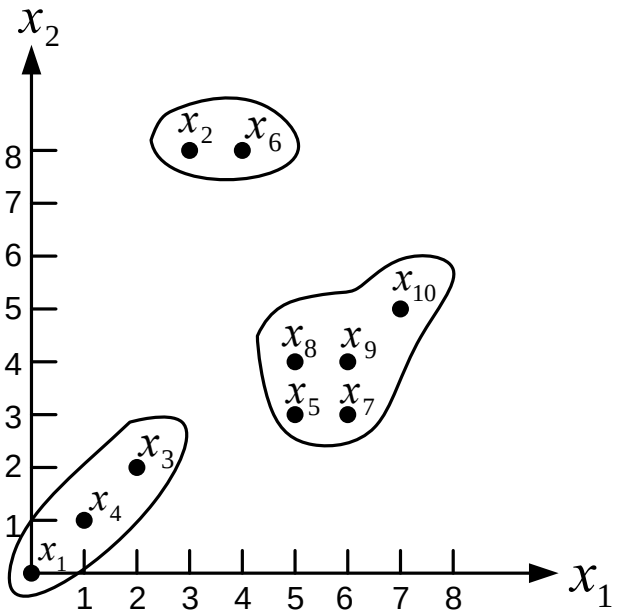


Рис.7.14.

Рассмотрим таблицы, приведенные на рис.7.15:

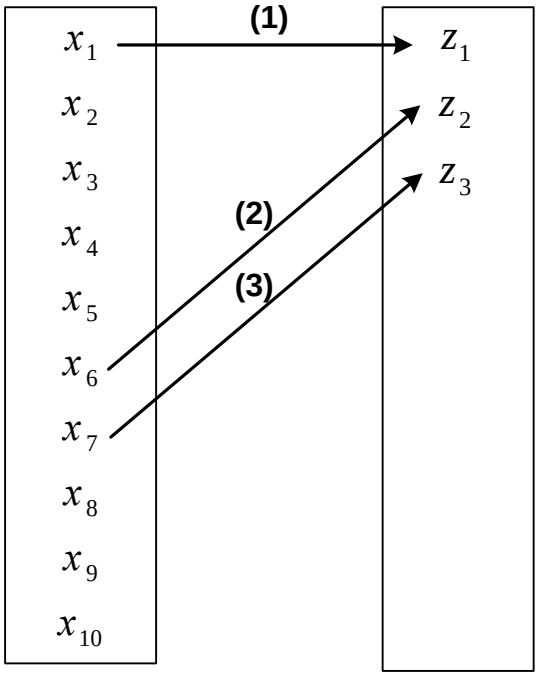


Рис.7.15.

Затем отыскиваем образ, отстоящий от образа x_1 на наибольшее расстояние. В нашем случае это – x_6, который назначается центром кластера z_2.

На третьем шаге алгоритма производится вычисление расстояний между всеми остальными образами выборки и центрами кластеров z_1 и z_2. В каждой паре этих расстояний выделяется *минимальное*. После этого выделяется *максимальное* из этих минимальных расстояний. Если это последнее составляет значительную часть расстояния между центрами кластеров z_1 и z_2 (скажем, не менее половины этого расстояния), соответствующий образ назначается центром

кластера z_3. В противном случае выполнение алгоритма прекращается. Если воспользоваться таким критерием в нашем примере, то получаем, что центром кластера z_3 становится образ x_7.

На следующем шаге алгоритма вычисляется расстояние между тремя выделенными центрами кластеров и всеми остальными образами; в каждой группе из *трех расстояний* выбирается минимальное. После этого находится максимальное из этих минимальных расстояний. Если последнее составляет значительную часть «типичных» предыдущих максимальных расстояний, то соответствующий образ назначается центром кластера z_4. В противном случае выполнение алгоритма прекращается. Хорошей мерой для оценки типичных предыдущих расстояний является их среднее значение. Если воспользоваться этим критерием в нашем примере и потребовать, чтобы новое максимальное расстояние составляло по меньшей мере половину среднего, то из рисунка видно, что очередное максимальное расстояние – расстояние между образами x_1 и x_3 - не удовлетворяет этому условию. Поэтому, на данном шаге работа алгоритма прерывается. В общем случае описанная процедура повторяется до тех пор, пока на каком-либо шаге не будет получено максимальное расстояние, для которого условие, определяющее выделение нового кластера, не выполняется.

В нашем примере были выделены три кластерных центра - x_1, x_6 и x_7. При отнесении остальных образов к одному из кластеров используется критерий, предусматривающих введение классифицируемого образа в тот кластер, центр которого для него ближайший. В нашем примере мы получаем три кластера: $\{x_1, x_3, x_4\}$, $\{x_2, x_6\}$ и $\{x_5, x_7, x_8, x_9, x_{10}\}$. Результаты соответствуют нашим интуитивным представлениям об этих данных.

Алгоритм К внутригрупповых средних. Рассмотренные алгоритмы являются в сущности эвристическими процедурами. Рассмотрим теперь алгоритм, который минимизирует показатель качества, определяемый как сумма квадратов расстояний всех точек, входящих в кластерную область, до центра. Он состоит из следующих шагов.

Шаг 1. Полагаем k=1. Определяем К исходных центров кластеров $z_1(1)$, $z_2(1),\ldots,z_K(1)$. Этот выбор производится произвольно, и обычно в качестве исходных центров используются первые К выборки из заданного множества образов.

Шаг 2. На k–м шаге итерации заданное множество образов {X} распределяется по К кластерам по следующему правилу:

$$X \in S_j(k), \text{ если } \|X-z_j(k)\|<\|X-z_i(k)\| \qquad (7.19)$$

Для всех $i=1,2,\ldots,K$, $i\neq j$, где $S_j(k)$ - множество образов, входящих в кластер с центром $z_j(k)$. В случае равенства в (2.19) решение принимается произвольным образом.

Шаг 3. На основе результатов шага 2 определяются новые центры кластеров zj(k+1) исходя из условия, что сумма квадратов расстояний между всеми образами, принадлежащими множеству Sj(k), и новым центром кластера должна быть минимальной. Другими словами, новые центры кластеров zj(k+1) выбираются таким образом, чтобы минимизировать показатель качества:

$$J_j = \sum_{X \in S_j(k)} \| X - z_j(k+1) \|^2, j = 1,2,K. \quad (7.20)$$

Центр $z_j(k+1)$, обеспечивающий минимизацию показателя качества, является в сущности, выборочным средним, определенным по множеству $S_j(k)$. Следовательно, новые центры кластеров определяются как:

$$z_j(k+1) = \frac{1}{N_j} \sum_{X \in S_j(k)} X, j = 1,2,K. \quad (7.21)$$

Здесь N_j - число выборочных образов, входящих в множество $S_j(k)$.

Шаг 4. Равенство $z_j(k+1) = z_j(k)$ при $j=1,2,\ldots,K$ является условием сходимости алгоритма, и при его достижении выполнение алгоритма заканчивается. В противном случае алгоритм повторяется от шага 2.

Отметим, что качество алгоритма зависит от выбираемых центров кластеров, от выбора исходных центров кластеров и от последовательности осмотра образов.

Пример. В качестве простой иллюстрации алгоритма *К* внутригрупповых средних рассмотрим образы, представленные на рис.10. (N=6, K=2, $X_1(0,0)$, $X_2(1,0)$, $X_3(0,1)$, $X_4(1,1)$, $X_5(3,2)$, $X_6(4,2)$. Принять $Z_1(1)=X_1$, $Z_2(1)=X_2$).

Перцептроны

Одним из методов решения задач обучения распознаванию образов основан на моделировании гипотетического механизма человеческого мозга. Структура модели заранее постулируется. При таком подходе уровень биологических знаний или гипотез о биологических механизмах является исходной предпосылкой, на которой базируются модели этих механизмов. Примером такого направления в теории и практике проблемы ОРО является класс устройств, называемых перцептронами. Нужно отметить, что перцептроны на заре своего возникновения рассматривались только как эвристические модели механизма мозга. Впоследствии они стали основополагающей схемой в построении кусочно-линейных моделей, обучающихся распознаванию образов.

В наиболее простом виде перцептрон (Рис.7.17) состоит из совокупности чувствительных (сенсорных) элементов (S-элементов), на которые поступают входные сигналы. S-элементы случайным образом связаны с совокупностью ассоциативных элементов (А-элементов), выход которых отличается от нуля только тогда, когда возбуждено достаточно большое число S-элементов, воздействующих на один А-элемент. А-элементы соединены с реагирующими элементами (R-элементами) связями, коэффициенты усиления которых переменны и изменяются в процессе обучения. Взвешенные комбинации выходов R-элементов составляют реакцию системы, которая указывает на принадлежность распознаваемого объекта определенному образу. Если распознаются только два образа, то в перцептроне устанавливается только один R-элемент, который обладает двумя реакциями – положительной и отрицательной. Если образов больше двух, то для каждого образа устанавливают

свой R-элемент, а выход каждого такого элемента представляет линейную комбинацию выходов A-элементов:

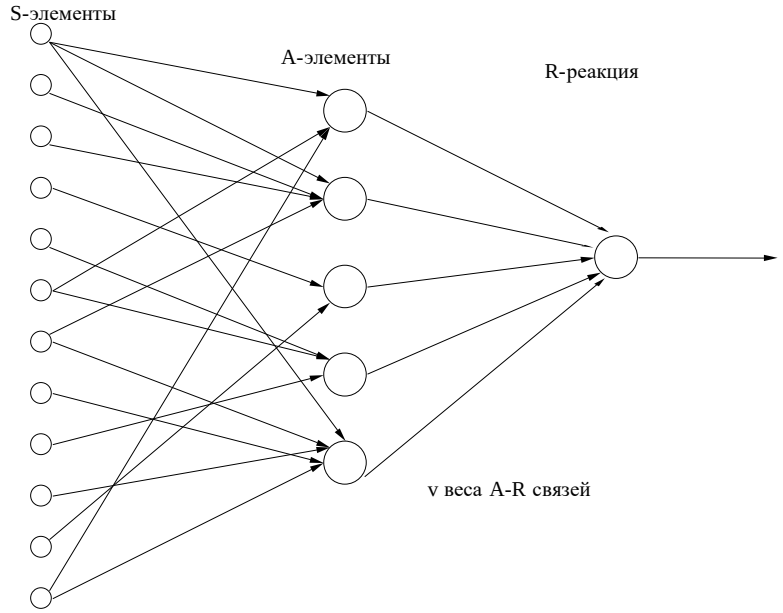


Рис. 7.17.

Реакция всей системы пропорциональна сумме взятых с определенными весами реакций элементов ассоциативной сетчатки. Обозначим через xi реакцию i -го ассоциативного элемента и через wi - соответствующий вес. Тогда реакцию системы можно записать как:

$$R = \sum_{i=1}^{n+1} w_i \times x_i = w'X \tag{7.22}$$

Если R>0, то предъявленный системе образ принадлежит классу w1. Если R<0, то образ относится к классу w2.

Данная перцептронная модель представляет собой не что иное, как реализацию линейной решающей функции.

Данный подход легко распространить на случай разделения на несколько (M) классов посредством увеличения числа реагирующих элементов в R-сетчатке. Классификация проводится аналогично: рассматриваются значения реакций R1, R2,…,RM и образ причисляется к классу ωi, если Ri>Rj для всех j≠i.

Если в перцептроне допускаются лишь связи, идущие от бинарных S-элементов к A-элементам и от A-элементов к единственному R-элементу, то такой перцептрон принято называть элементарным α-перцептроном.

О перцептронах было сформулировано и доказано несколько основополагающих теорем, две из которых, определяющие основные свойства перцептрона, приведены ниже.

Теорема 1. Класс элементарных α-перцептронов, для которых существует решение для любой задуманной классификации, не является пустым.

Эта теорема утверждает, что для любой классификации обучающей последовательности можно подобрать такой набор (из бесконечного набора) A-

элементов, в котором будет осуществлено задуманное разделение обучающей последовательности при помощи линейного решающего правила.

Теорема 2. Если для некоторой классификации решение существует, то в процессе обучения α-перцептрона с коррекцией ошибок, начинающегося с произвольного исходного состояния, это решение будет достигнуто в течение конечного промежутка времени.

Смысл этой теоремы состоит в том, что если относительно задуманной классификации можно найти набор А-элементов, в котором существует решение, то в рамках этого набора оно будет достигнуто в конечный промежуток времени.

Обычно обсуждают свойства бесконечного перцептрона, т. е. перцептрона с бесконечным числом А-элементов со всевозможными связями с S-элементами (полный набор А-элементов). В таких перцептронах решение всегда существует, а раз оно существует, то оно и достижимо в α-перцептронах с коррекцией ошибок.

Очень интересную область исследований представляют собой многослойные перцептроны и перцептроны с перекрестными связями, но теория этих систем практически еще не разработана.

Принцип покрепления – наказания. Обучающий алгоритм для перцептрона, приведенного на рис.2.17, сводится к простой схеме итеративного определения весов *W*. Рассмотрим описание этой схемы, которую называют *алгоритмом перцептрона*.

Заданы два обучающих множества, представляющие классы ω_1 и ω_2 соответственно. Пусть *W(1)* – начальный вектор весов, который выбирается произвольно. В таком случае *k*-й шаг обучения выглядит следующим образом.

Если $X(k) \in \omega_1$ и $W'(k) \times X(k) \leq 0$, то вектор весов *W(k)* заменяется вектором

$$W(k+1) = W(k) + cX(k), \qquad (7.23)$$

где с - корректирующее приращение.

Если $X(k) \in \omega_2$ и $W'(k) \times X(k) \geq 0$, то вектор весов *W(k)* заменяется вектором

$$W(k+1) = W(k) - cX(k), \qquad (7.24)$$

В противном случае не изменяется, т.е.

$$W(k+1) = W(k). \qquad (7.25)$$

Иными словами, алгоритм вносит изменения в вектор весов W в том и только том случае, если образ, предъявленный на k-м шаге обучения был классифицирован неверно. Корректирующее приращение **c** должно быть положительным, и в данном случае предполагается, что оно постоянно.

Алгоритм перцептрона является процедурой типа «подкрепление-наказание», причем, подкреплением за правильную классификацию образа служит отсутствие наказания. Т.е., если образ классифицирован правильно, то система подкрепляется тем, что в вектор весов W не вносится никаких изменений. С другой стороны, если образ классифицируется неправильно и произведение W'(k)×X(k) оказывается меньше нуля, когда оно должно быть больше нуля, система «наказывается» увеличением значения вектора весов W(k) на величину, пропорциональную X(k). Точно также, если произведение

W’(k)×X(k) оказывается больше нуля, когда оно должно быть меньше нуля, система наказывается противоположным образом. *Сходимость* алгоритма наступает при правильной классификации *всех образов* с помощью некоторого вектора весов. Можно *доказать* (*мы этого делать не будем*), что алгоритм перцептрона сходится за *конечное число итераций*, если заданные классы *линейно разделимы*.

Пример. Рассмотрим образы, приведенные на рис.2.18. Следует применить к этим образам алгоритм перцептрона с тем, чтобы с его помощью определить весовой вектор решения.

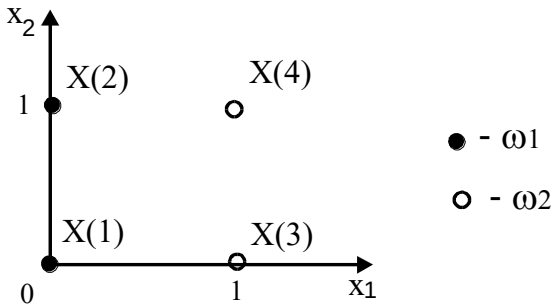


Рис.7.18.

Из рисунка видно, что два заданных класса линейно разделимы и, следовательно, применение алгоритма окажется успешным.

До начала применения алгоритма пополним все образы. При этом рассматриваемые классы обратятся в ω_1: {(0,0,1)’, (0,1,1)’} и в ω_2: {(1,0,1)’, (1,1,1)’} . Задав c=1 и W(1)=0 и предъявив образы в указанном порядке, получим (по шагам):

$$W'(1)\times X(1)=(0,0,0)\times\begin{pmatrix}0\\0\\1\end{pmatrix}=0\text{, следовательно: } W(2)=W(1)+X(1)=\begin{pmatrix}0\\0\\1\end{pmatrix},$$

$$W'(2)\times X(2)=(0,0,1)\times\begin{pmatrix}0\\1\\1\end{pmatrix}=1\text{, следовательно: } W(3)=W(2)=\begin{pmatrix}0\\0\\1\end{pmatrix},$$

$$W'(3)\times X(3)=(0,0,1)\times\begin{pmatrix}1\\0\\1\end{pmatrix}=1\text{, следовательно: } W(4)=W(3)-X(3)=\begin{pmatrix}-1\\0\\0\end{pmatrix},$$

$$W'(4)\times X(4)=(-1,0,0)\times\begin{pmatrix}1\\1\\1\end{pmatrix}=1\text{, следовательно: } W(5)=W(4)=\begin{pmatrix}-1\\0\\0\end{pmatrix}.$$

Коррекция вектора весов проводилась на 1 и 3 шагах. Т.к. полученный результат можно считать искомым решением только в том случае, когда алгоритм осуществит без ошибок полный цикл итерации по всем образам, обучающее множество следует предъявить еще раз.

Процесс обучения системы продолжается при X(5)=X(1), X(6)=X(2), X(7)=X(3) и X(8)=X(4). Второй цикл итерации приводит к следующим результатам:

$$W'(5)\times X(5)=(-1,0,0)\times\begin{pmatrix}0\\0\\1\end{pmatrix}=0,\ \text{следовательно: } W(6)=W(5)+X(5)=\begin{pmatrix}-1\\0\\1\end{pmatrix},$$

$$W'(6)\times X(6)=(-1,0,1)\times\begin{pmatrix}0\\1\\1\end{pmatrix}=1,\ \text{следовательно: } W(7)=W(6)=\begin{pmatrix}-1\\0\\1\end{pmatrix},$$

$$W'(7)\times X(7)=(-1,0,1)\times\begin{pmatrix}1\\0\\1\end{pmatrix}=0,\ \text{следовательно: } W(8)=W(7)-X(7)=\begin{pmatrix}-2\\0\\0\end{pmatrix},$$

$$W'(8)\times X(8)=(-2,0,0)\times\begin{pmatrix}1\\1\\1\end{pmatrix}=-2,\ \text{следовательно: } W(9)=W(8)=\begin{pmatrix}-2\\0\\0\end{pmatrix}.$$

Поскольку в данном цикле итерации опять совершено 2 ошибки, все образы предъявляются еще раз для значений X(9)=X(1), X(10)=X(2), X(11)=X(3) и X(12)=X(4):

$$W'(9)\times X(9)=(-2,0,0)\times\begin{pmatrix}0\\0\\1\end{pmatrix}=0,\ \text{следовательно: } W(10)=W(9)+X(9)=\begin{pmatrix}-2\\0\\1\end{pmatrix},$$

$$W'(10)\times X(10)=(-2,0,1)\times\begin{pmatrix}0\\1\\1\end{pmatrix}=1,\ \text{следовательно: } W(11)=W(10)=\begin{pmatrix}-2\\0\\1\end{pmatrix},$$

$$W'(11)\times X(11)=(-2,0,1)\times\begin{pmatrix}1\\0\\1\end{pmatrix}=-1,\ \text{следовательно: } W(12)=W(11)=\begin{pmatrix}-2\\0\\1\end{pmatrix},$$

$$W'(12)\times X(12)=(-2,0,1)\times\begin{pmatrix}1\\1\\1\end{pmatrix}=-1,\ \text{следовательно: } W(13)=W(12)=\begin{pmatrix}-2\\0\\1\end{pmatrix}.$$

Можно убедиться, что в следующем итерактивном цикле все образы классифицируются правильно, т.е. вектор решений имеет вид: W=(-2,0,1)’ . Следовательно решающая функция будет: d(X)=-2x_1+1.

Алгоритм перцептрона можно представить в другой, эквивалентной форме, умножив пополненные образы одного из классов на –1. Таким образом, умножив все образы, например класса ω_2, на -1, алгоритм перцептрона можно переписать как:

$$W(k+1)=\begin{cases}W(k), & \text{если } W'(k)\times X(k)>0,\\ W(k)+cX(k), & \text{если } W'(k)\times X(k)\le 0,\end{cases} \qquad (7.26)$$

где с – положительное корректирующее приращение.

Разновидности перцептронного подхода. Варьируя способ выбора корректирующего приращения c, можно получить несколько модификаций алгоритма перцептрона. К наиболее распространенным алгоритмам обучения относятся алгоритм фиксированного приращения, алгоритм коррекции абсолютной величины и алгоритм дробной коррекции.

В алгоритме фиксированного приращения корректирующее приращение **c** является константой, большей нуля (наш пример).

В алгоритме *коррекции абсолютной величины* c выбирается достаточно большим, для того чтобы гарантировать правильную классификацию образа после коррекции весов. Другими словами, если значение W'(k)×X(k)≤0, то коэффициент c выбирается так, чтобы выполнялось:

$$W'(k+1)\times X(k) = [W(k)+cX(k)]'\times X(k)>0. \qquad (7.27)$$

Один из способов, обеспечивающий неравенство (1) состоит в выборе в качестве **c** *наименьшего целого числа*, превышающего значение |W'(k)×X(k)|/ [X'(k)×X(k)].

В алгоритме дробной коррекции c выбирается таким, чтобы величина |W(k)×X(k)-W'(k+1)×X(k)| составляла некоторую долю λ от величины |W(k)×X(k)|, т.е.

$$|W(k)\times X(k)-W'(k+1)\times X(k)| = \lambda\times|W(k)\times X(k)|, \qquad (7.28)$$

Подстановка W(k+1)= W(k)+c×X(k)| в выражение (7.26) дает:

$$c = \lambda\frac{|W'(k)\times X(k)|}{X'(k)\times X(k)}.$$

Этот алгоритм, очевидно, требует, чтобы начальный вектор весов отличался от 0.

Синтаксическое распознавание образов

Возникновение теории формальных языков в середине 50-х годов связано с разработкой Ноамом Хомским математических моделей грамматик при исследовании естественных языков. Одной из первоначальных задач лингвистов, работающих в этой области, было создание «вычислительных» грамматик, способных описывать естественные языки, например английский. Была надежда на то, что если это удастся, то не составит большого труда научить машину «понимать» естественные языки. И хотя эти надежды пока не оправдались, побочные результаты этих исследований оказали важное влияние в других областях, например, при разработке компиляторов, в языках программирования и распознавания образов.

Под строкой будем понимать конечное множество символов, каждый из которых является элементом некоторого заранее определенного произвольного непустого конечного множества символов, называемого *алфавитом.*

Если строка содержит *m* символов (подсчитываются все символы строки независимо от их повторений), то о такой строке говорят, что она имеет длину *m.* Например, строка 001110 имеет длину 6. Длина строки обозначается |x|. Пустой строкой называют строку длины 0, т.е. строку, не содержащую ни одного символа. Пустую строку принято обозначать буквой ε.

Пусть Σ - некоторый алфавит. Обозначим через Σ^* множество определенных над этим алфавитом строк. Например, если Σ = {0,1}, тогда Σ^*={ε,0,1,00,01,10,11,000,001,010,...}.

Если $x \in \Sigma^*$ - строка длины m, а $y \in \Sigma^*$- строка длины n, то их *объединение,* обозначаемое xy, может быть определено как строка длины $m + n$, причем первые m символов составляют строку, совпадающую со строкой x, а последние n символов составляют строку, совпадающую со строкой y.

Формальным языком L над алфавитом Σ называется произвольное подмножество множества Σ^*. Если L_1 и L_2 -два формальных языка, то их объединение $L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$ также является формальным языком.

Как и в случае объединения строк операция объединения формальных языков ассоциативна, но не коммутативна.. Для произвольного формального языка L справедливо следующее утверждение: $L\{\varepsilon\} = \{\varepsilon\}L = L$.

Объединение формального языка L с самим собой записывается как L^2 или в общем случае как $L^0=\{\varepsilon\}$, $L^1=L$, $L^i = LL^{i-1} = L^{i-1}L$ для $i \geq 2$. Замыкание Клини формального языка L, обозначаемое как L^*, может быть определено как $\bigcup_{i=0}^{\infty} L^i$. Если определить L^+ как $\bigcup_{i=1}^{\infty} L^i$, то определение замыкания Клини формального языка L можно записать как $L^*=L^+ \cup \{\varepsilon\}$ Обычно совокупность строк, принадлежащих множеству Σ^* и имеющих длину 2, обозначают Σ^2, и имеющих длину 3 – соответственно Σ^3, и т.д. Совокупность строк, принадлежащих множеству Σ^* и имеющих длину, большую или равную 1, обозначают Σ^+. Воспользуемся принятыми обозначениями и запишем следующее утверждение: $\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i$ и $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$.

Контекстная грамматика G может быть представлена совокупностью четырех объектов (N, T, P, S), где

N - конечное множество нетерминальных символов языка. Согласно принятому соглашению элементы множества N обозначаются прописными буквами (иногда с нижними индексами);

T - конечное множество терминальных символов {поэтому $N \cap T = \varnothing$}. Согласно принятому соглашению элементы множества T обозначаются строчными буквами (иногда с нижними индексами);

P - конечное множество *продукций* вида $\alpha \to \beta$, где α – строка в левой части продукции, такая, что $\alpha \in (N \cup T)^+$, а β - строка в правой части продукции, такая, что $\beta \in (N \cup T)^*$; элемент $S \in N$ - начальный символ грамматики.

Символ $\to$ используется для обозначения отношения «определяется как».

Таким образом, простой пример грамматики G может иметь следующий набор правил вывода:

$$S \to A|B,$$
$$A \to aA|a,$$
$$B \to bB|b.$$

Из первого правила следует, что *S* либо совпадает с *A*, либо совпадает с *B*. Из второго правила следует, что *A* представляет собой либо строку вида *a,* либо строку вида *aa,* либо строку вида *aa...a.* Аналогичные рассуждения справедливы для B.

Если строка β может быть получена из строки α с помощью одного или нескольких правил вывода, то будем записывать это следующим образом: $\alpha \Rightarrow \beta$. Если $\alpha_1 \Rightarrow \alpha_2$, $\alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{n-1} \Rightarrow \alpha_n$, $(n \geq 1)$, то эту последовательность выводов можно кратко записать так: $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \ldots \Rightarrow \alpha_n$, или, еще более кратко, $\alpha_1 \overset{*}{\Rightarrow} \alpha_n$.

Воспользуемся теперь формальным определением грамматики для строгого определения выводимой строки. Пусть *G=(N,T,P,S).* - контекстная грамматика и пусть $\gamma_1\alpha\gamma_2 \in (N \cup T)^+$ - строка терминальных и нетерминальных символов длиной ≥ 1. Если $\alpha \rightarrow \beta$ - продукция из *P,* то строка α в строке $\gamma_1\alpha\gamma_2$ может быть заменена строкой β, и в результате получим $\gamma_1\beta\gamma_2$ Это записывают следующим образом:

$$\gamma_1\alpha\gamma_2 \underset{G}{\Rightarrow} \gamma_1\beta\gamma_2 .$$

В этом случае говорят, что строка $\gamma_1\alpha\gamma_2$ генерирует строку $\gamma_1\beta\gamma_2$, или что строка $\gamma_1\beta\gamma_2$ выводится из строки $\gamma_1\alpha\gamma_2$.

Если $\alpha_1, \alpha_2, \ldots, \alpha_n \in (N \cup T)^*$ и $\alpha_1 \underset{G}{\Rightarrow} \alpha_2$, $\alpha_2 \underset{G}{\Rightarrow} \alpha_3, \ldots, \alpha_{n-1} \underset{G}{\Rightarrow} \alpha_n$ $(n \geq 1)$, то обычно пишут сокращенно $\alpha_1 \underset{G}{\Rightarrow} \alpha_2 \underset{G}{\Rightarrow} \alpha_3 \ldots \underset{G}{\Rightarrow} \alpha_{n-1} \underset{G}{\Rightarrow} \alpha_n$, или еще короче, $\alpha_1 \overset{+}{\underset{G}{\Rightarrow}} \alpha_n$, и при этом говорят, что строка α_n выводится из строки α_1 за один или более шагов. Далее, $\overset{+}{\underset{G}{\Rightarrow}}$ обозначает транзитивное замыкание отношения $\underset{G}{\Rightarrow}$, а $\overset{*}{\underset{G}{\Rightarrow}}$ обозначает рефлексивное замыкание отношения $\overset{+}{\underset{G}{\Rightarrow}}$. В таком случае $\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_n \Leftrightarrow \alpha_1 = \alpha_n$ или $\alpha_1 \overset{+}{\underset{G}{\Rightarrow}} \alpha_n$.

Если строка $\alpha \in (N \cup T)^*$ такая, что $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$, то строку α называют *сентенциальной формой* контекстной грамматики *G: Сентенцией* грамматики *G* называют произвольную сентенциальную форму из *T**, т.е. произвольную строку терминальных символов, которая может быть выведена из начального символа *S.* В этом случае множество всех сентенций грамматики *G* называют *языком, порождаемым грамматикой G,* и обозначают *L(G).*

Контекстно-свободные грамматики. Очень часто вид левой части каждой продукции грамматики может быть ограничен лишь единственным нетерминальным символом. Контекстные грамматики, продукции которых удовлетворяют такому ограничению, в теории формальных языков носят

название *контекстно свободных грамматик.* Строгое определение контекстно-свободной грамматики может быть таким:

Контекстно-свободной грамматикой называют такую контекстную грамматику $G= (N,T,P,S)$, каждая продукция которой имеет вид $A\to\beta$, где $A\in N$, $\beta\in(N\cup T)^*$.

Любой язык, порожденный контекстно-свободной грамматикой, называют *контекстно свободным языком.* Поводом к возникновению термина "контекстно свободный" послужил тот факт, что любой символ $A\in N$ в сентенциальной форме грамматики G может быть раскрыт согласно продукции $A\to\beta$ независимо от того, какими строками он окружен внутри самой сентенциальной формы.

Если $G = (N, T,P, S)$ – контекстно-свободная грамматика, то любой нетерминальный символ $x\in L(G)$ может быть выведен из начального символа S. Общепринятым методом представления вывода некоторой непустой строки считается *дерево вывода* (иногда его называют деревом грамматического разбора). Корнем этого дерева будет начальный символ S, остальные его узлы пометим слева направо следующим образом: $a_1,a_2,\ldots,a_n$, где $a_i\in T$, $1\le i\le n$ и $x = a_1a_2\ldots a_n$; при этом внутренние узлы дерева будут соответствовать нетерминальным символам языка. Пусть A - нетерминальный символ, который может быть расширен с помощью продукции вида $A\to x_1\,x_2\ldots x_m$ ($x_i\in N\cup T$, $i=1,2,\ldots,m$). В таком случае узел дерева, соответствующий нетерминальному символу A, является родительским узлом для m узлов дерева, соответствующих m нетерминальным символам $x_1\,x_2\ldots x_m$ (рис.7.26).

Рассмотрим в качестве примера дерево вывода (рис. 7.27), иллюстрирующее вывод сентенции a^3b^2 контекстно-свободной грамматики G, имеющей продукции вида

$$S \to AB,$$
$$A \to aA|a,$$
$$B \to bB|b.$$

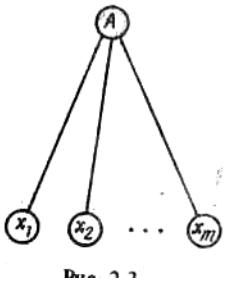


Рис.7.26.

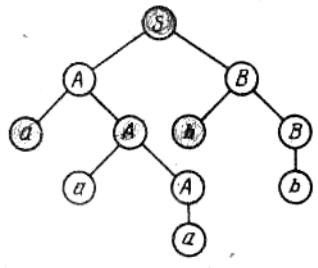


Рис.7.27.

Это дерево вывода в точности соответствует последовательности выводов:

$$S \Rightarrow AB,$$
$$\Rightarrow aAB,$$
$$\Rightarrow aaAB,$$
$$\Rightarrow aaaB,$$
$$\Rightarrow aaabB,$$
$$\Rightarrow aaabb.$$

Если $x \in L(G)$ то, как правило, можно найти несколько возможных вариантов вывода. Например, сентенцию a^3b^2 можно вывести в рассматриваемой грамматике следующим образом:

$S \Rightarrow AB \Rightarrow AbB \Rightarrow aAbB \Rightarrow aaAbB \Rightarrow aaAbb \Rightarrow aaabb,$

$S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow aAbb \Rightarrow aaAbb \Rightarrow aaabb,$

$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaAbB \Rightarrow aaabB \Rightarrow aaabb.$

Понятно, что все эти схемы вывода соответствуют одному и тому же дереву вывода.

Если для любого элемента $x \in L(G)$ все возможные схемы его вывода соответствуют одному и тому же дереву вывода, то такую контекстно свободную грамматику называют *однозначной грамматикой.* Если же некоторому элементу $x \in L(G)$ соответствует несколько несовпадающих деревьев вывода, то контекстно свободная грамматика G называется *неоднозначной.*

Схема вывода сентенции $x \in L(G)$ называется *левосторонней,* если раскрывается всегда самый левый нетерминальный символ сентенциальной, формы. Поскольку теперь каждому дереву вывода соответствует единственная левосторонняя схема вывода, то можно переопределить неоднозначную грамматику следующим образом: контекстно свободная грамматика G называется неоднозначной, если существует такая сентенция $x \in L(G)$, которой можно поставить в соответствие две различные левосторонние схемы вывода.

Рассмотрим неоднозначную грамматику G', имеющую продукцию

$$S \to SbS|ScS|a.$$

Пусть сентенции $abaca \in L(G')$ соответствуют две различные левосторонние схемы вывода

$$S \Rightarrow SbS \Rightarrow abS \Rightarrow abScS \Rightarrow abacS \Rightarrow abaca,$$
$$S \Rightarrow ScS \Rightarrow SbScS \Rightarrow abScS \Rightarrow abacS \Rightarrow abaca.$$

Понятия формальных грамматик могут быть связаны с распознаванием следующим образом:

Пусть у нас имеются два класса образов ω_1 и ω_2 и пусть образы этих классов могут быть построены из признаков, принадлежащих некоторому конечному множеству $\boldsymbol{T}$. Каждый образ может рассматриваться как цепочка или предложение некоторого языка.

Допустим, что существует грамматика $\boldsymbol{G}$, такая, что порождаемый ею язык состоит из предложений (образов), принадлежащих только классу ω_1.

Очевидно, что эта грамматика может быть использована в целях классификации, т.к. заданный образ неизвестной природы может быть отнесен к ω_1, если он является предложением языка $\boldsymbol{L(G)}$. В противном случае образ приписывается классу ω_2.

Образ попадает в класс ω_2 исключительно потому, что он не принадлежит классу ω_1. Тем не менее не исключено, что он не принадлежит и классу ω_2.

Он может представлять собой зашумленную или искаженную цепочку, которую лучше всего изъять из распознавания.

Чтобы обеспечить эту возможность, необходимо задать две грамматики G_1 и G_2, порождающие языки $L(G_1)$ и $L(G_2)$ соответственно.

Образ зачисляется в класс, язык которого считает этот образ правильным предложением. Если он не является предложением ни $L(G_1)$ ни $L(G_2)$, образ изымается.

В случае M классов мы рассматриваем M грамматик и связанных с ними языков $L(G_i)$, $i=1,2,\ldots,M$.

Распознаваемый объект относится к классу ω_i в том и только том случае, если он является предложением языка $L(G_i)$. Если объект является предложением более одного языка или не принадлежит ни одному из них, он может быть изъят из рассмотрения или произвольно отнесен к одному из классов.

Чтобы получить реальную пользу от структурных свойств объекта в процессе синтаксического распознавания, понятие цепочки должно быть обобщено на двумерный случай. Правила подстановки в грамматиках цепочек заключаются в простом соединении цепочек с целью формирования новых.

Рассмотрим позиционный дескриптор ***НАД(a,b)***, обозначающий, что структура, представленная символом ***a***, расположена над структурой, представленной символом ***b***. И рассмотрим позиционный дескриптор ***СЛЕВА(a,b)***, обозначающий, что структура, представленная символом ***a***, расположена слева от ***b***. Квадратная структура □, составленная из непроизводных элементов | и -, описывается предложением: ***НАД(- , НАД(СЛЕВА(|, |), -)).***

Основная трудность при подобном подходе заключается в определении содержания дескрипторов ***НАД*** и ***СЛЕВА***. Так, предыдущему описанию «квадрат» удовлетворяет и структура:

В таких случаях накладываются ограничения. Так, для дескриптора ***НАД(a,b)*** таким ограничением является требование, чтобы хотя бы часть элемента ***a*** находилась над элементом ***b***. В этом случае данная структура не будет считаться допустимой, т.к. элемент – не находится над элементом | | и элемент | | не находится над элементом -.

В некоторых работах используется схема, заключающаяся в соединении структур только в особых точках. Например, чтобы каждая структура имела две выделенные точки. И соединение структур должно происходить только в этих точках (рис.2.28.а).

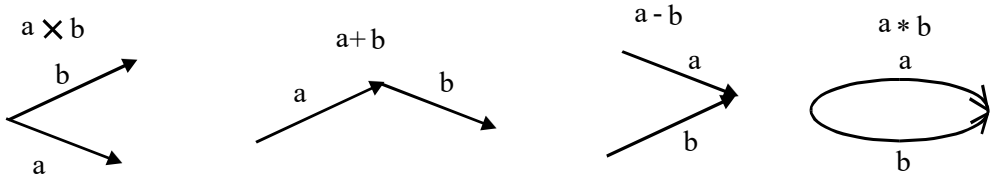


Рис.7.28.а

Две выделенные точки интерпретируются как «головной» и «хвостовой» концы стрелы. Типичные правила соединения показаны на рис.2.28.б:

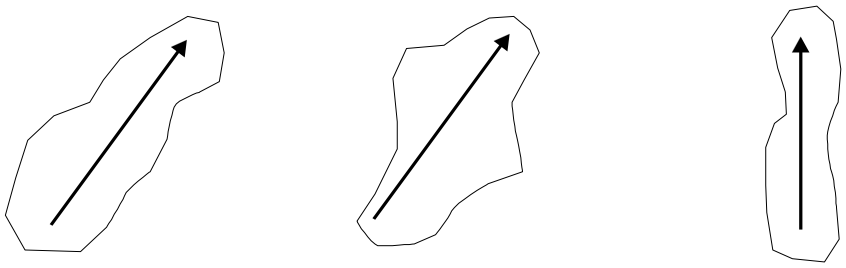


Рис.7.28.б

Пример 1. Рассмотрим опять структуры типа квадрат. Непроизводными элементами служат вертикальный и горизонтальный отрезки, (Рис.7.29). Контекстно-свободная грамматика ***G***, способная порождать квадраты, задается как:

G=(N,T,P,S,) при **T={a_1, a_2}, V_N={S,O_1, O_2};**

P: S→A(a_1,O_2);

O2→A(O_1,a_1);

O1→L(a_2,a_2),

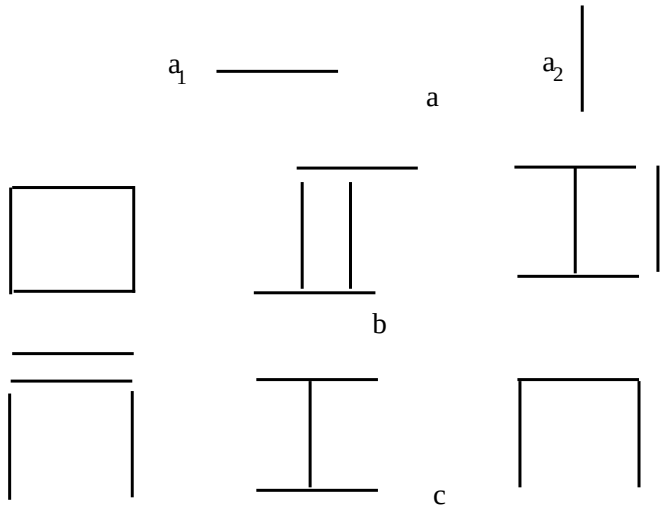


Рис.7.29

Здесь **A(x,y)** и **L(x,y)** читаются соответственно «x расположен над y» и «x расположен слева от y». При этом, позиционный дескриптор **A(x,y)** считается допустимым только в том случае, если часть **x** находится непосредственно над **y**, а дескриптор **L(x,y)** допустим тогда, когда часть **y** находится непосредственно справа от **x**.

Грамматический разбор представляет здесь простую процедуру, поскольку используется только одна последовательность правил подстановки. Предположим надо установить, принадлежит или нет данная структура к классу «квадрат». Синтаксический разбор «сверху- вниз» будет производиться

следующим образом. Первое правило подстановки начинается с **S** и предполагает поиск некоторого объекта $\mathbf{O_2}$ ниже непроизводного элемента $\mathbf{a_1}$. Если ниже некоторого объекта $\mathbf{a_1}$ не найдено ни одного элемента, грамматический разбор прерывается и образ отклоняется. Если же правило применено успешно, на следующем шаге отыскивается некоторый объект $\mathbf{O_1}$ над другим непроизводным элементом $\mathbf{a_1}$. Если $\mathbf{O_1}$ обнаружен, грамматический разбор продолжается, в противном случае образ отклоняется. Наконец, объект $\mathbf{O_1}$, обнаруженный на предыдущем шаге, должен для принятия образа разделиться на два непроизводных элемента $\mathbf{a_2}$ по условию $\mathbf{L(a_2,a_2)}$. Этой схеме грамматического разбора удовлетворяют структуры, показанные на Рис.7.28.(б) и не удовлетворяют на Рис.7.29.(в).

Пример 2. Одним из приложений лингвистических понятий в распознавании образов является язык PDL (Picture Description Language) – язык описания изображений. Здесь непроизводным элементом служит любая n-мерная структура с двумя выделенными точками – хвостовой и головной (Рис.2.28.а – для двумерных структур). Непроизводный элемент может примыкать к другим непроизводным элементам ***только*** в хвостовой и\или головной точке. Т.о. структуры языка PDL представляют собой ориентированные графы и для обработки этих структур можно использовать грамматики.

Основные правила соединения непроизводных элементов приведены ранее (Рис.7.28.б). Также могут быть использованы ***пустые*** непроизводные элементы для порождения внешне разъединенных структур, а также ***нулевые*** точки – непроизводный элемент с идентичными головной и хвостовой точками.

Рассмотрим простую грамматику языка PDL: **G=(N,T,P,S,)**,

где $\mathbf{N=\{S, A_1, A_2\ A_3, A_4, A_5\};}$

$\mathbf{T=\{a, b, c, d\},}$

P: $\mathbf{S \rightarrow d + A_1;}$

$\mathbf{A_1 \rightarrow c + A_2;}$

$\mathbf{A_2 \rightarrow \sim d * A_3;}$

$\mathbf{A_3 \rightarrow a + A_4;}$

$\mathbf{A_4 \rightarrow b * A_5;}$

$\mathbf{A_5 \rightarrow c.}$

Причем, **~d** означает перемену мест головной и хвостовой точек непроизводного элемента **d**.

Непроизводные элементы показаны на Рис.2.30:

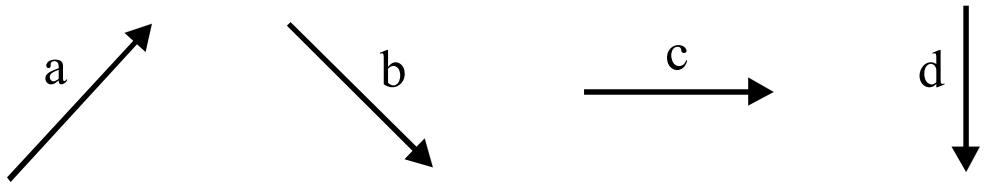


Рис.7.30.

Применение первого правила постановки приводит к получению непроизводного элемента ***d***, сопровождаемого еще не определенной переменной. На этом этапе мы знаем только, что хвостовая точка структуры, представленная символом A_1, будет связана с головной точкой элемента ***d***, потому что этот непроизводный элемент сопровождается оператором «+». Переменная A_1 разлагается на $c+A_2$, причем A_2 пока не определена. Аналогичным образом A_2 разлагается на $\sim d*A_3$.

Результаты применения этих трех правил показаны на Рис.7.31.(a,b,c). Из определения оператора «*» мы знаем, что при разложении элемента A_3 происходит его соединение с составной структурой ***8.7.c*** путем присоединения хвостовой точки к хвостовой, а головной точки к головной. Конечный результат показан на Рис.7.31.f.

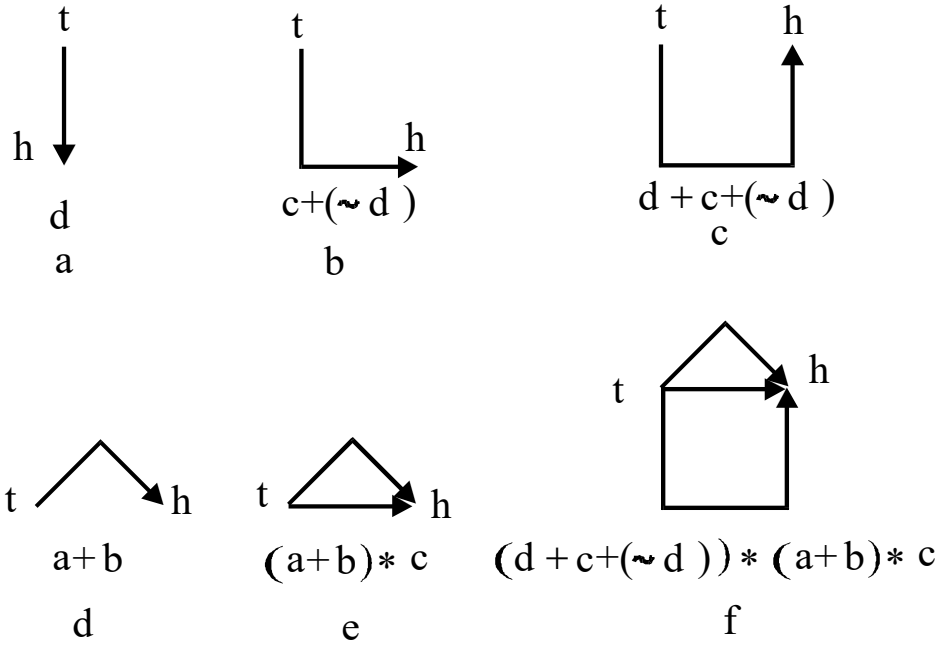


Рис.7.31.

Грамматика языка PDL, описанная выше, способна порождать только одну структуру. Однако, можно расширить число структур, введением рекурсивности – способности переменной замещаться этой переменной. Например:

P: $S \rightarrow d + A_1$;
$A_1 \rightarrow c + A_1$;
$A_1 \rightarrow \sim d * A_2$;
$A_2 \rightarrow a + A_2$;
$A_2 \rightarrow b * A_2$;
$A_2 \rightarrow c$.

Результатом применения этих правил в указанном порядке будет структура «домик» (Рис.7.31.f). Кроме этого, новое множество правил допускает возможность порождать и другие (бесконечные) структуры.

Грамматический разбор ***сверху вниз*** будет происходить в следующей последовательности. Предположим наш объект – «домик». Грамматический анализатор начинает с корневого символа ***S.*** В данном случае ***S*** заменяется на **d**

+ $\mathbf{A_1}$ и т.д. приходим к выводу, что образ поддается грамматическому разбору. При наличии двух и более грамматик анализ производится до тех пор, пока образ не идентифицируется либо пока не исчерпаются возможности грамматики; в последнем случае образ отклоняется.

Обучение и грамматический вывод

Используя лингвистическую терминологию, процедуру получения решений с помощью обучающей выборки легко интерпретировать как задачу получения грамматики из множества выборочных предложений. Эту процедуру обычно называют ***грамматическим выводом*** (или *восстановление грамматики*).

Модель вывода грамматики:

Рис.7.32.

Здесь задача заключается в том, что множество выборочных цепочек ***{x_i}*** подвергается обработке с помощью адаптивного обучающего алгоритма. На выходе в конечном счете воспроизводится грамматика ***G***, согласованная с данными цепочками, т.е. множество цепочек ***{x_i}*** является подмножеством языка ***L(G)***. К сожалению ни одна из известных схем не в состоянии решить эту задачу в общем виде.

Рассмотрим алгоритм, являющийся модификацией процедуры, разработанной Фельдманом в 1969г. Этот алгоритм для заданного множества терминальных цепочек выводит ***автоматную грамматику***. (Автоматная грамматика – грамматика вида **A→aA** или **A→a**, т.е. частный случай формы Грейбаха).

Основная идея метода Фельдмана заключается в том, чтобы сначала построить нерекурсивную грамматику, порождающую в точности заданные цепочки, а затем, сращивая нетерминальные элементы, получить более простую рекурсивную грамматику, порождающую бесконечное число цепочек. Алгоритм можно разделить на три части. Первая часть формирует нерекурсивную грамматику. Вторая часть преобразует ее в рекурсивную грамматику. Затем в третьей части происходит упрощение этой грамматики.

Рассмотрим эту процедуру на примере.

Пусть задано выборочное множество терминальных цепочек ***{caaab, bbaab, caab, bbab, cab, bbb, cb}***. Требуется построить автоматную грамматику, способную порождать эти цепочки. Алгоритм построения грамматики состоит из следующих этапов.

Часть1. Строится нерекурсивная грамматика, порождающая в точности заданное множество выборочных цепочек. Выборочные цепочки

обрабатываются в порядке уменьшения длины. Правила подстановки строятся и прибавляются к грамматике по мере того, как становятся нужны для построения соответствующей цепочки из выборки. Заключительное правило постановки, используемое для порождения самой длинной выборочной цепочки, называется ***остаточным правилом***, а длина его правой части ***равна 2*** (т.е. имеет вид $\mathbf{A \rightarrow a_1 a_2}$).

В нашем примере первой цепочкой максимальной длины является ***caaab***. Для ее порождения строятся следующие правила подстановки:

$\mathbf{S \rightarrow cA_1}$
$\mathbf{A_1 \rightarrow aA_2}$
$\mathbf{A_2 \rightarrow aA_3}$
$\mathbf{A_3 \rightarrow ab}$ где $\mathbf{A_3}$ - правило остатка.

Вторая цепочка - ***bbaab***. Для порождения этой цепочки к грамматике добавляются следующие правила:

$\mathbf{S \rightarrow bA_4}$
$\mathbf{A_4 \rightarrow bA_5}$
$\mathbf{A_5 \rightarrow aA_6}$
$\mathbf{A_6 \rightarrow ab}$.

Для порождения третьей цепочки требуется добавление к грамматике одного правила

$$\mathbf{A_3 \rightarrow b.}$$

Рассмотрев остальные цепочки, получаем, что множество правил подстановки, построенных для порождения обучающей выборки, имеет вид:

$\mathbf{S \rightarrow cA_1}$	$\mathbf{A_3 \rightarrow ab}$
$\mathbf{S \rightarrow bA_4}$	$\mathbf{A_4 \rightarrow bA_5}$
$\mathbf{A_1 \rightarrow b}$	$\mathbf{A_5 \rightarrow b}$
$\mathbf{A_1 \rightarrow aA_2}$	$\mathbf{A_5 \rightarrow aA_6}$
$\mathbf{A_2 \rightarrow b}$	$\mathbf{A_6 \rightarrow b}$
$\mathbf{A_2 \rightarrow aA_3}$	$\mathbf{A_6 \rightarrow ab}$
$\mathbf{A_3 \rightarrow b}$	

Часть 2. В этой части, соединяя каждое правило остатка ***длины 2*** другим (неостаточным) правилом грамматики, получаем рекурсивную автоматную грамматику. Это происходит в результате слияния каждого нетерминального элемента правила остатка с нетерминальным элементом неостаточного правила, который может порождать остаток. Так, например, если $\boldsymbol{A_r}$ - остаточный нетерминал вида $\mathbf{A_r \rightarrow a_1 a_2}$ и $\boldsymbol{A_n}$ - неостаточный нетерминал вида $\mathbf{A_n \rightarrow a_1 A_m}$, где $\mathbf{A_m \rightarrow a_2}$, все встречающиеся $\boldsymbol{A_r}$ заменяются на $\boldsymbol{A_n}$, а правило подстановки $\mathbf{A_r \rightarrow a_1 a_2}$ отбрасывается. Таким образом создается автоматная грамматика, способная порождать данную обучающую выборку, а также обладающая общностью, достаточной для порождения бесконечного множества других цепочек.

В нашем случае $\mathbf{A_6}$ может слиться с $\mathbf{A_5}$, а $\mathbf{A_3}$ - с $\mathbf{A_2}$. В результате имеем:

$\mathbf{S \rightarrow cA_1}$	$\mathbf{A_2 \rightarrow b}$

$S \to bA_4$ $A_4 \to bA_5$
$A_1 \to b$ $A_5 \to b$
$A_1 \to aA_2$ $A_5 \to aA_5$
$A_2 \to b$ $A_5 \to b$
$A_2 \to aA_2$

Здесь рекурсивными являются правила: $A_2 \to aA_2$ и $A_5 \to aA_5$.

Часть 3. Полученная грамматика упрощается объединением эквивалентных правил постановки. Два правила с левыми частями A_i и A_j эквивалентны, если соблюдены следующие условия. Предположим, что, начиная с A_i можно породить множество цепочек $\{x\}_i$, а начиная с A_j - множество цепочек $\{x\}_j$. Если $\{x\}_i=\{x\}_j$, то два правила постановки считаются эквивалентными, и каждый символ A_j может быть заменен на A_i без ущерба для языка, порождаемого этой грамматикой.

В нашем примере эквивалентны правила с левыми частями A_1 и A_2. После слияния A_1 и A_2 получаем:

$S \to cA_1$ $A_4 \to bA_5$
$S \to bA_4$ $A_5 \to b$
$A_1 \to b$ $A_5 \to b$
$A_1 \to aA_1$ $A_5 \to aA_5$

Точно также эквивалентны правила с левыми частями A_1 и A_5. После слияния A_1 и A_5 получаем:

$S \to cA_1$, $S \to bA_4$, $A_1 \to b$, $A_1 \to aA_1$, $A_4 \to bA_1$.

Дальнейшее слияние правил невозможно, поэтому алгоритм в процессе обучения строит следующую грамматику:

G=(N,T,P,S), **N={S,A,B},** **T={a,b,c}** где

P: $S \to cA$
$S \to bB$
$A \to aA$
$B \to bA$
$A \to b$

Эта грамматика может порождать обучающую выборку, использованную в процессе ее вывода.