Лекции №1. Нейронные сети. Основные положения.

Введение

Теория искусственных нейронных сетей включает в себя большой спектр вопросов из разных областей науки: биофизики, информатики, математики, схемотехники и т. д. Дадим следующее определение:

Искусственные нейронные сети – это совокупность моделей биологических нейронных сетей. Нейронные сети представляют собой совокупность элементов, связанных между собой синаптическими связями. Сеть обрабатывает входную информацию и в процессе изменения своего внутреннего состояния во времени формирует выходные воздействия.

Также можно сказать – «нейронные сети являются разделом большой науки – искусственного интеллекта, задача которой – разработка парадигм и алгоритмов, обеспечивающих компьютерное решение когнитивных задач, свойственных человеческому мозгу».

Искусственные нейронные сети имеют прямую аналогию со строением и принципом работы мозга, который представляет собой очень сложный, нелинейный, параллельный компьютер.

Краткая история развития искусственных нейронных сетей

Можно сказать, что искусственные нейронные сети появились в 1943 году, после того, как МакКаллок и Питтс опубликовали свою статью про моделирование принципа работы нейрона с помощью электрических схем (хотя… здесь есть момент, что нечто подобное публиковалось немного ранее, в частности немецкими учеными в 30-х годах). Но, по сути, эта пионерская работа, так и осталась пионерской, т.к. шла Вторая Мировая Война…

В 1949 году канадский физиолог и психолог Хебб вызказал идеи по поводу соединения нейронов в головном мозге и характере их работы. Главным выводом работы Хебба было, то, что обучение заключается главным образом за счет изменения силы синаптических связей (т.н. «правило Хебба»).

Первые (можно сказать «реальные») нейросети были созданы в конце 50-х гг. американскими учеными Ф. Розенблаттом (эксперимент датирован 1957 годом) и П. Мак-Кигюком. Это были попытки создать системы, моделирующие человеческий глаз и его взаимодействие с мозгом. Устройство, созданное ими, получило название перцептрона (персептрона). Оно умело различать буквы алфавита, но было чувствительно к их написанию, например, буквы А, А и А для этого устройства были тремя разными знаками. Постепенно в 70-80 гг. количество работ по этому направлению искусственного интеллекта стало снижаться (благодаря работе М.Мински). Слишком неутешительны оказались первые результаты. Авторы объясняли неудачи малой памятью и низким

быстродействием существующих в то время компьютеров. При этом Френк Розенблатт, можно сказать «трагическая фигура» в истории нейросетей, погиб в автокатастрофе в начале 70-х годов. А незадолго до этого в 1969 году, тогда еще не легенда, а начинающий кибернетик, Марвин Мински опубликовал (совместно с Пейпертом, причем оба были сокурсниками Розенблатта!) получившую широкую известность статью об ограниченной возможности к обучению персептрона… Это была проблема «исключающего ИЛИ», которая сейчас рассматривается в каждом уважающем себя учебнике по нейронным сетям. Да, это было действительно так, потому – что использовались линейные функции активации.

Да, кстати, Розенблатт был, кроме всего прочего, лектором по психологии в Корнеллском университете и ему принадлежит следующая интересная фраза: *«Еще никому не удалось обнаружить в отдельных элементах или клетках нервной сети какую – либо специфическую психологическую функцию, такую, как ‘память’, ’мышление’, ‘самосознание’ или ‘разум’. Это дает основание предполагать, что такие свойства присущи не отдельным элементам, а связаны с организацией и функционированием нервной системы в целом»* [7]. Также необходимо сказать, что Розенблатт родился в семье выходцев из Российской Империи.

Здесь стоит отметить, что и в СССР довольно плотно занимались искусственными нейронными сетями. В частности, монографии Мкртчяна и Сочивко об электрических схемах нейронов получили довольно широкую известность (это 60-е – 70-е года). Также выделим пионерские работы кибернетика Галушкина, который, как сейчас выясняется первым предложил градиентный способ обучения нейронных сетей.

Широкий интерес к нейронным сетям был инициирован после появления работы Джона Хопфилда (Hopfield J.J., 1982), который показал, что задача с «изинговскими» нейронами может быть сведена к обобщениям ряда моделей, разработанных к тому моменту в физике неупорядоченных систем. Работа сети Хопфилда (наиболее подробно обсуждаемая в физической литературе) состоит в релаксации начального "спинового портрета" матрицы двоичных кодов к одному из стационарных состояний, определяемых правилом обучения (правилом Хебба). Таким образом, данная сеть может применяться для задач распознавания. Далее вышла похожая рекуррентная нейронная сеть Хемминга.

В 1982 году появилась сеть Кохонена. Разработчик – уже «легендарный» финский ученый Тейву Кохонен. Сеть довольно простая, но хорошо зарекомендовала себя для достаточного круга практических задач нейроинформатики (о ней будет рассказано далее).

В 1986 году появилась работа Румельхарта, Хинтона и Вильямса (Rumelhart D.E., Hinton G.E., Williams R.J., 1986), содержавшая ответ на вопрос,

долгое время сдерживавший развитие нейроинформатики - как обучаются иерархические слоистые нейронные сети, для которых "классиками" еще в 40-50 х годах была доказана универсальность для широкого класса задач. В последующие годы предложенный Хинтоном алгоритм обратного распространения ошибок претерпел бесчисленное множество вариаций и модификаций. Появление данной работы сильно повлияло на развитие, а лучше сказать – на популярность искусственных нейронных сетей. Но следует отметить, что еще в 1974 году советский ученый Галушкин и параллельно ему американец Вербос – предложили практически то же самое – использование градиентных методов для обучения персептрона! Но слава досталась не им (хотя… и Галушкин и Вербос все равно стали очень известными учеными).

В 1975 году Фукусима разработал модель когнитрона – одну из первых многослойных нейросетевых моделей.

Прекрасный ретроспективный обзор нейросетевой науки до 1982 года можно найти в книге А.И. Галушкина и Я.З. Цыпкина [6].

1.1 Принцип работы нейрона человека

Термин «синапс» для обозначения предполагаемых связей между нервными клетками предложил сэр Чарльз Шеррингтон (Йельский университет) совместно с Фостером. Если упрощено то принцип работы можно описать следующим образом - По сути, это связь: одна ячейка «разговаривает» с другой. Клетка мозга или нейрон имеет большое основное тело с небольшими выступающими нитями. Итак, один нейрон, передатчик, использует очень тонкую нить, называемую аксоном. Второй нейрон, получатель, может получать контакты вдоль своего основного тела или вдоль ветвящихся, как дерево, ветвей, называемых дендритами. Когда кончик аксона передатчика подключается к приемнику, это синапс. Нейроны работают на электричестве. Если электрический сигнал проходит по аксону, его кончик выделяет в синапс химические вещества, называемые нейротрансмиттерами. Эти нейротрансмиттеры приказывают принимающей клетке либо активировать свой собственный электрический заряд, который посылает сигнал следующему нейрону в цепи, либо приказывают принимающей клетке оставаться в тишине.

Но, если по серьезному, то там сложные физико-химические процессы…

1.2 Принцип работы мозга человека

У всех позвоночных и у человека центральная нервная система состоит из головного и спинного мозга. Мозг содержит два типа клеток – нейроны и глиальные клетки. Количество глиальных клеток на порядок превосходит число нейронов. Раньше считали, что роль глиальных клеток сводится к доставке энергии (питания) к нейрону и механической поддержке всего основного

каркаса нейронов и их волокон, но уже и в позапрошлом столетии были ученые, которые прямо указывали, что размер глиальной клетки возрастает в сравнении низших с высшими животными. В нынешнее время, в неврологии уделяется огромное значение информационному значению глиальной клетки и ее участии в процессах обработки информации в нервных сетях.

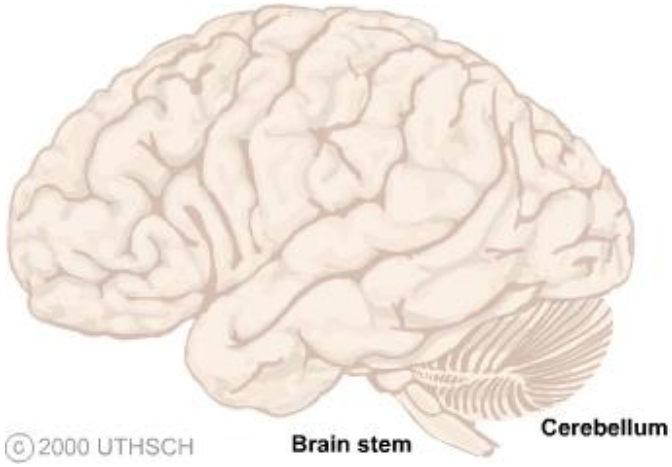


Рисунок 1. Мозг человека

Чем выше по своему эволюционному развитию позвоночное животное, тем больше развит его головной мозг. Например, головной мозг лягушек и рыб весит меньше чем их спинной мозг. Головной мозг человека в 50 раз превышает по весу спинной.

Все процессы жизнедеятельности человека, в том числе управления и регулирования, происходящие в организме человека, связаны с функционированием взаимодействующих нейронов. Нейрон, по сути и решатель и проводник сложно организованной нелинейной динамической нервной системы. Но кроме нейрона в нервной также существуют и глиальные клетки, жидкое межклеточное вещество – плазма крови, плотное основное вещество хряща и т.д.

Головной мозг человека занимает почти всю полость мозгового отдела черепа, кости которого защищают головной мозг от внешних механических повреждений. В процессе роста и развития головной мозг принимает форму черепа. Приводятся различные оценки количества нейронов, содержащихся в головном мозге человека. По одним оценкам головной мозг взрослого мужчины содержит в среднем 86,1 ±8,1 млрд нейронов и 84,6 ±9,8 млрд не нейронных клеток. При этом кора головного мозга содержит 19 % нейронов. По другим оценкам головной мозг человека содержит 90—95 миллиардов нейронов (согласно Википедии и Атласу мозга человека). Объём мозга большинства людей находится в пределах 1250—1600 кубических сантиметров и составляет 91—95 % ёмкости черепа. В головном мозге различают пять отделов: продолговатый мозг; задний, включающий в себя мост, мозжечок и эпифиз; средний; промежуточный; и передний мозг, представленный большими полушариями. Но обычно выделяют 3 больших части:

- полушария большого мозга;

- мозжечок;

- ствол мозга.

При этом мозг человека состоит из следующих отделов:

- продолговатый мозг. Частично ответственен за регуляцию обмена веществ, дыхание, кровообращение, равновесие и координацию движений;

- Варолиев мост. Передает информацию из спинного мозга в отделы головного. Также частично ответственен за глазные рефлексы, рефлекторное моргание, моторику кишечника и др.

- Мозжечок. Отвечает за координацию движений, координацию движений, регуляцию равновесия и мышечного тонуса. Непрерывно корректирует все произвольные и автоматические движения. Также отвечает за накопление опыта и памяти (возможно и мышления). Мозжечок важен для корректировки осанки и поддержания равновесия. Через входные данные вестибулярных рецепторов и проприорецепторов он модулирует команды *мотонейронам*, чтобы компенсировать сдвиги в положении тела или изменения нагрузки на мышцы. Пациенты с повреждением мозжечка страдают нарушениями равновесия и часто вырабатывают стереотипные стратегии осанки для компенсации этой проблемы (например, широкая стойка).

- Средний мозг. Ответственен за зрение, слух, контроль движений, регуляцию циклов сна и бодрствования, общего уровня возбуждения центральной нервной системы, концентрацию внимания, болевую чувствительность и многое другое (это довольно сложная структура, которая состоит из крыши (четверохолмие, среднемозговой части покрышки, красного ядра, черного вещества, ножек мозга и т.д.)). Отметим, что четверохолмие – это т.н. зрительные бугорки. Красное ядро ответственно за координацию движения и т.д. (подробно см. Атлас мозга человека).

Интересное замечание – мозг взрослого мужчины в среднем на 11% тяжелее и на 10% больше по объему, чем мозг женщины того же возврата (но это соответствует средней разнице между средним весом и средним ростом! И больше ни о чем не говорит).

Здесь стоит отметить (но лучше, конечно, пройти отдельный учебный курс) о кортексе и неокортексе, которые отвечают за мышление и память.

Рассмотрим принцип работы неокортекса предложенный Джефом Хокинсом, который является основателем Института нейронаук в Редвуде (RNI), а также соучредителем компании Numenta inc[1]. Джефом Хокинсом предложена модель работы мозга по принципу «память - предсказание» (далее без ссылок на Хокинса). Следует отметить, что до сих пор нет единой,

[1] Джеф Хокинс больше известен как руководитель группы разработчиком компьютеров Palm.

стройной и полностью подтвержденной экспериментально теории, как работает интеллект человека. Данная теория – одна из наиболее сильных и проработанных.

Место интеллекта человека находится в неокортексе. Хотя у интеллекта огромное число возможностей и колоссальная гибкость, неокортекс имеет на удивление регулярную структуру. Различные части неокортекса, ответственные ли за зрение, слух, осязание или языковые способности – все работают по одним и тем же принципам. Ключ к пониманию неокортекса – понять эти общие принципы и, в особенности, их иерархическую структуру.

Паттерны. Паттерн – это образ (визуальный, звуковой, тактильный и т. д.), который подается на вход нейронов мозга или снимается с выхода. Паттерны бывают входные и выходные, то есть реакция.

Каким образом мозг решает задачи в миллионы раз быстрее компьютеров, например распознавание образов и логические задачи? Ответ, в том, что мозг не вычисляет решение – он достает ответ из памяти. Весь кортекс – это система памяти, это огромное количество всевозможных паттернов.

Неокортикальная память. Фундаментальное отличие неокортикальной памяти от компьютерной состоит в следующем:

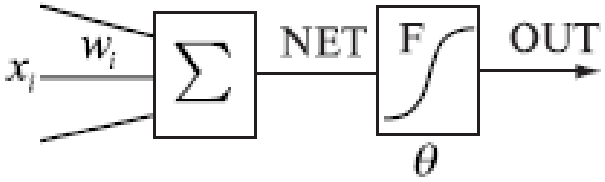
1. Неокортекс хранит последовательность паттернов.
2. Неокортекс вспоминает паттерны автоассоциативно.
3. Неокортекс хранит паттерны в инвариантной форме.
4. Неокортекс хранит паттерны иерархически.

Последовательность паттернов – пример: вспомните дом, где вы выросли. Вы начнете от входной двери или вашей комнаты и последовательно будете вспоминать обстановку. Или, например, алфавит – попробуйте произнести его задом наперед… Вы его помните в той последовательности в какой учили. Другой пример последовательности – любимый музыкальный сборник, когда заканчивается одна песня, мозг уже знает какая будет следующая, если, конечно, не включен Shuffle (далее см. книгу Хокинса).

1.3 Формальный персептрон

История искусственных нейронных сетей ведет свое начало с 1943 года с работы МакКаллока и Питтса, которые предложили формальный персептрон.

Рассмотрим подробнее формальный персептрон (иногда пишут *перцептрон* от английского *perception* – восприятие).



Нейрон состоит из взвешенного сумматора и нелинейного элемента. Функционирование нейрона определяется формулами:

$$NET = \sum_i w_i x_i$$

$$OUT = F(NET - \theta)$$

Где x_i - входные сигналы, совокупность которых формируют вектор x;

w_i - весовые коэффициенты, совокупность которых образуют вектор весов w;

NET - взвешенная сумма входных сигналов, значение NET передается на нелинейный элемент;

θ - пороговый уровень данного нейрона;

F - нелинейная функция, называемая функцией активации.

Нейрон имеет несколько входных сигналов x и один выходной сигнал OUT. Пороговый элемент отсекает небольшие значения входа NET, *как пример – студент на лекции слышит множество посторонних звуков, но концентрируется на голосе лектора*.

Также существует модель с дополнительным входом смещением (bias), которые отсекают малые значения сигналов, поступающих на нейрон, по сути случат фильтром.

Ограничения модели формального персептрона [5]. В теории нейронных сетей считается, что:

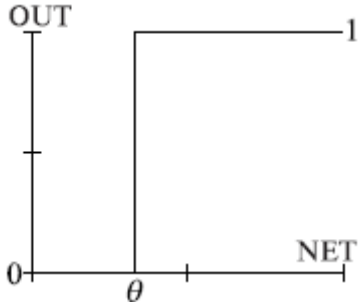
1. Вычисления нейрона происходят мгновенно, не внося задержку.
2. Нет четких алгоритмов для выбора функции активации.
3. Нет механизмов, регулирующих работу сети в целом, наподобии гормональной регуляции активности в нервных клетках.

Модель формального нейрона не является биоподобной и это скорее математическая абстракция.

Функции активации

Рассмотрим наиболее распространненые функции активации F (их также называют *функциями сжатия*, так как они ограничивают входной сигнал). Обычно нормализованный выход находится в диапазоне [0;1] или [-1;1].

1. Жесткая ступенька

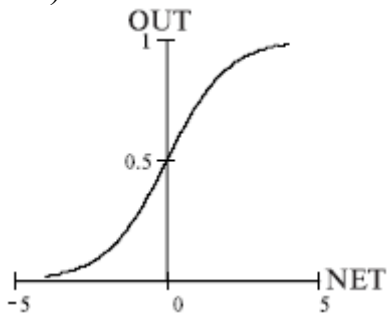


Работа описывается следующей формулой:

$$OUT = \begin{cases} 0, & NET < \theta \\ 1, & NET \geq \theta \end{cases}$$

Также ее еще называют функцией Хэвисайда (Heaviside function).

2. Сигмоида (функция Ферми)



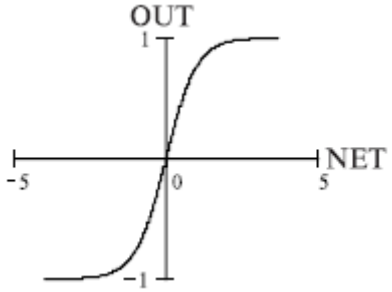
Функция описывающая сигмоиду:

$$OUT = \frac{1}{1 + e^{-NET}}$$

Часто применяются для многослойных перцептронов и других сетей с непрерывными сигналами. Гладкость, непрерывность функции — важные положительные качества. Непрерывность первой производной позволяет обучать сеть градиентными методами (например, метод обратного распространения ошибки).

Функция симметрична относительно точки (NET=0, OUT=1/2), это делает равноправными значения OUT=0 и OUT=1, что существенно в работе сети. Тем не менее, диапазон выходных значений от 0 до 1 несимметричен, из - за этого обучение значительно замедляется. Значение производной выражается через саму функцию, что приводит к увеличению скорости расчета производной при обучении сети (см. ниже) [5].

3. Гиперболический тангенс



Функция:

$$OUT = \text{th}(NET) = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}}$$

Также часто применяется в сетях с непрерывными сигналами. Производная выражается через саму функцию, что важно при градиентных методах обучения нейронной сети.

4. SOFTMAX – функция

$$OUT = \frac{e^{NET}}{\sum_i e^{NET_i}}$$

В данном случае суммирование ведется по всем нейронам данного слоя сети. Такой выбор обеспечивает сумму выходов слоя, равную единице при любых значениях сигнала NET_i данного слоя. Это позволяет трактовать выход OUT_i как вероятность событий, совокупность которых образует полную группу.

Выбор функции активации выбирается в зависимости от задачи, удобством программной (аппаратной) реализации. А также алгоритмом обучения и, в ряде случаев, топологией нейросети.

Подготовка входных данных

Существует а) проблема сбора данных (обучающей выборки), б) проблема нормирования данных (масштабирование). Проблему сбора данных (датасета) рассмотрим в более поздних лекциях.

Задача масштабирования. В связи с тем, что рабочий участок сигмоиды и тангенсоиды в пределах [-1;1], то желательно масштабировать входные данные в этот интервал. Большие входные значения дают ответы близкие к единице.

В качестве методов масштабирования можно предложить следующие (уйдут в «зону насыщения»):

1. Использовать формулу

$$\overline{X}_i = \frac{X_i}{|X_{\max}|}$$

2. Вычесть среднее и разделить на с.к.о.

3. В ряде случаев целесообразно использовать логарифмическую шкалу.

Основные черты искусственных нейронных сетей

Выделим основные характерные черты искусственных нейронных сетей [5]:

1. Универсальная модель для аппроксимации функций большой размерности.
2. Можно получить качественный прогноз для практически любого процесса с большим количеством переменных.
3. Хорошо зарекомендовавший себя инструмент для распознавания изображений
4. Некоторые нейронные сети хорошо работают по принципу автоассоциации
5. Хороший инструмент в области Data – mining и Text - mining

Основные (*положительные*) свойства искусственных нейронных сетей (далее просто *нейронные сети*):

- *нелинейность*. Современные сети используют нелинейные элементы, которые позволяют увеличивать объемы запоминаемой информации и применять сложные методы обучения;
- *адаптивность*. Нейронные сети позволяют адаптировать свои веса к изменению окружающей среды, то есть изменяемому пространству входов – выходов;
- *отказоустойчивость*. Нейронная сеть может потерять до 20-25% нейронов и продолжать правильно решать свою задачу (соответственно можно недообучить нейрнную сеть и она будет адекватно решать поставленную задачу). Это свойство также прослеживается у биологических нейронных систем – количество нейронов постоянно уменьшается, но тем не менее человек выполняет свои функции. Или, допустим, боксер пропускает удар в голову – нокдаун, огромное количество выбывших нейронов и связей, но спустя какое – то время мозг начинает работать в обычном режиме;
- *масштабируемость*. Возможность в ходе работы сети, добавить или удалить нейроны или даже слои с возможностью дообучения. Также выделяют понятие *пластичности* – «даже после удаления

определенного количества нейронов может быть проведено повторное обучение сети до ее первоначального уровня навыков, если осталось достаточное количество нейронов»[2];

- *единообразие анализа и проектирования*. Созданная для одной задачи сеть может успешно, возможно с небольшими изменениями, использоваться и для решения других задач. Это широко применяется при повторном использовании программного кода.

Задачи, решаемые с помощью нейронных сетей

Нейронные сети завоевывают все большую популярность, которая связана со следующими факторами:

- количество применений – практически в любой области либо уже применены, либо ведутся работы по использованию нейронных сетей;
- обычно реальные задачи нелинейные, а получить правильное решение упрощая их приводит к низкому качеству решения (т.к. получаются сильно упрощенные матмодели). Нейросети же могут работать с любыми данными (естественно после приведения их к численному виду). Плюс нейросети могут работать с длинными временными рядами;
- неклассические постановки задач;
- уникальные свойства искусственных нейронных сетей

В общем случае задачи, решаемые с помощью нейронных сетей делятся на следующие:

- распознавание образов (изображений и работа с видеопотоком);

- прогнозирование;

- классификация и кластеризация;

- сжатие, кодирование / декодирование;

Более сложный момент – применение в системах управления.

Валидационная выборка для нейронных сетей

В конец давайте рассмотрим вопрос подготовки валидационной выборки

Литература:

[2] Такая же особенность характерна и для мозга, в котором могут быть повреждены некоторые участки, но со временем за счет специальных тренировок часто достигаются первоначальные уровни навыков.

1. Хайкин С. Нейронные сети: полный курс. 2-е изд.: Пер. с. англ. – М.: Издательский дом «Вильямс», 2006 – 1104 с.
2. Осовский . Нейронные сети
3. Мкртчян С. О. Нейроны и нейронные сети. – М: Энергия, 1971. -232 с.
4. Суровцев И. С., Клюкин В. И., Пивоварова Р. П., Нейронные сети. – Воронеж: ВГУ, 1994. – 224с.
5. Заенцев И. В. Нейронные сети: основные модели. Учебное пособие . Воронеж: ВГУ. 1998.- 76с.
6. Нейронные сети: история развития теории. Кн. 5. Учеб. пособие для вузов. / Под общей ред. А.И. Галушкина и Я.З. Цыпкина. – М.: ИПРЖР. 2001. 840 с.
7. Розенблатт Ф. Принципы нейродинамики. Перцептроны и теория механизмов мозга. М., 1965.

ЛЕКЦИЯ 1-2. Топологии нейронных сетей и алгоритмы их обучения: многослойный персептрон и алгоритм обратного распространения ошибки, RBF – сети.

2.1 Многослойный персептрон

История искусственных нейронных сетей ведет свое начало с формального персептрона Мак-Каллока и Питтса, которые в 1943 году разработали компьютерную модель нейронной сети на основе математических алгоритмов и теории деятельности головного мозга (см. Лекция 1-1). Многослойный перцептрон состоит из взаимосвязанных нейронов, передающих информацию друг другу, во многом подобно человеческому мозгу. Каждому нейрону присваивается значение. Сеть можно разделить на три основных уровня.

Формальные персептроны (или, лучше говорить - нейроны) могут объединяться в сети различными способами. Рассмотрим простейшее объединение – многослойный персептрон прямого распространения сигнала (связи соотвественно идут направлено от i-го слоя к i+1 - му).

Различают:

Входной слой - это начальный уровень сети, который принимает входные данные, которые будут использоваться для создания выходных данных. Входной слой не производит никаких вычислений – просто передает сигнал на первый скрытый слой (в нейронах входного слоя нет функций активации).

Скрытые слои (может быть только один скрытый слой) - в сети должен быть хотя бы один скрытый слой. Скрытые слои выполняют вычисления и операции с входными данными, чтобы получить результат на выходе. Скрытые слои связаны между собой синаптическими связями – каждый нейрон предыдущего с слоя с каждым нейроном последующего слоя.

Выходной слой – собственно выдает результат вычисления сети. В нейронах выходного слоя также есть функции активации. Обычно функции активации одни и те же в нейронах скрытых слоев и выходном слое. Но могут быть и исключения (в данном случае нужен «хитрый» способ обучения).

В то время как входные нейроны берут свои значения из окружения, значения всех других нейронов вычисляются с помощью математической функции, включающей веса и значения предшествующего слоя (см. предыдущую лекцию).

Работа многослойного персептрона описывается следующими формулами:

$$NET_{jl} = \sum_i w_{ijl} x_{ijl}$$

$$OUT_{jl} = F\left(NET_{jl} - \theta_{jl}\right)$$

$$x_{ij(l+1)} = OUT_{il}$$

где индексом i обозначается номер входа, j - номер нейрона в слое, l - номер слоя.

x_{ijl} - i-й входной сигнал j-го нейрона в слое l;

w_{ijl} - весовой коэффициент i - го входа нейрона номер j слоя l;

NET_{jl} - сигнал NET нейрона номер j слоя l;

Введем обозначения: w_{jl} — вектор-столбец весов для всех входов нейрона j в слое i. W_j — матрица весов всех нейронов в слоя j. В столбцах матрицы расположены вектора w_{jl}. Аналогично x_{jl} — входной вектор-столбец слоя l.

Каждый слой рассчитывает нелинейное преобразование от линейной комбинации сигналов предыдущего слоя. Многослойная нейронная сеть может формировать на выходе произвольную многомерную функцию при соответствующем выборе количества слоев, диапазона изменения сигналов и параметров нейронов. Многослойные нейронные сети являются универсальным аппроксиматором функций [5]. Общая формула, описывающая многослойный персептрон:

$$f(x) = \underbrace{F\left(\sum_{i_N} w_{i_N j_N N} \cdots \underbrace{\sum_{i_2} w_{i_2 j_2 2}\, \underbrace{F\left(\sum_{i_1} w_{i_1 j_1 1} x_{i_1 j_1 1} - \theta_{j_1 1}\right)}_{\text{слой 1}} - \theta_{j_2 2} \cdots - \theta_{j_N N}}_{\text{слой 2}}\right)}_{\text{слой } N}$$

За счет поочередного расчета линейных комбинаций и нелинейных преобразований достигается аппроксимация произвольной многомерной функции при соответствующем выборе параметров сети.

В некоторых источниках указывается, что способность выявлять статистические закономерности высокого порядка зависит от большого количества нейронов в скрытых слоях. Но не всегда сеть должна быть большого размера – все зависит от задачи, плюс необходима оптимизация работы сети. Нередко нейронная сеть может быть меньшего размера и может решать задачу с тем же показателем качества.

2.2 Парадигмы обучения нейронных сетей

Обычно выделяют две вида обучения нейронных сетей:

- с учителем – есть и входы и выходы;

- без учителя – есть только входы.

Но, есть еще промежуточная форма – обучение с подкреплением (о ней будет говориться в 3-х последних лекциях).

Остановимся на стандартных парадигмах. В первом случае есть и входные данные нейронной сети и соответствующие им выходы. Во втором случае есть только входные данные и нейронная сеть, обучающаяся без учителя, соответствующим образом интерпретирует входные данные на выходы сети. Обычно это используется в задачах кластеризации и классификации. Также обучение без учителя нередко используется в задачах сжатия данных.

Обучение с учителем. Пусть мы имеем входной вектор $X = \{x_1, x_{2,..,} x_n\}$ и ответ, то есть выходной вектор $Y = \{y_1, y_2, ..., y_n\}$. Тогда задача обучения нейросети сводится к тому, чтобы сеть на X^i выдавала на выходе Y^i или значение близкое к нему. Сеть обучается путем подстраивания весов синаптических связей w_{ij}^k. Для подстройки весов связей обычно используется один из градиентных методов.

В обычном многослойном персептроне всем весам связей присваиваются случайные веса. Эти случайные веса распространяют значения по сети для получения фактического результата (за счет их и функций активации сеть и функционирует, т.е. решает поставленную задачу). Естественно, этот результат будет отличаться от ожидаемого, т.к. веса связей «забиваются случайными значениями». Разница между двумя значениями на выходе – полученным и желаемым, называется ошибкой. Обратное распространение относится к процессу отправки этой ошибки обратно по сети с автоматической корректировкой весов, так что в конечном итоге ошибка между фактическим и ожидаемым выходом сводится к минимуму. Таким образом, выход текущей итерации становится входом и влияет на следующий выход. Это повторяется до тех пор, пока не будет получен правильный результат, ну или … необходимо заново переформировать нейронную сеть, произвести коррекцию входных временных рядов. Веса в конце процесса завершившегося с устраивающей разработчика ошибкой будут теми, на которых нейронная сеть работает правильно.

Обобщенный алгоритм обучения нейронных сетей с учителем следующий:

- Шаг 1. Подготовить обучающую выборку (входы - выходы);
- Шаг 2. Предварительно рассчитать структуру сети;

- Шаг 3. Инициализировать случайным образом матрицы весов синаптических связей нейронной сети;
- Шаг 4. Подать выбранный случайным образом пример из обучающей выборки и рассчитать выходы сети (прямой проход);
- Шаг 5. Рассчитать ошибку на выходе сети и с помощью специальных формул скорретировать веса синаптических связей. Если ошибка сети меньше заданной, то вернуться на шаг 3.
- (Шаг 6) Проверить на валидационной выборке, если ошибка удовлетворяет разработчика, то сеть – в реальный «боевой» режим;
- (Шаг 7) Оптимизация работы сети – сокращение числа слоев и нейронов;
- (Шаг 8). В режиме эксплуатации зачатую необходимо переобучение, т.к. появляются новые данные.

В скобках указаны не обязательные, но на практике необходимые шаги.

Немного про предварительный расчет структуры сети. Существует несколько подходов для выбора количества связей в нейронной сети. Например, использование формулы, основанной на теореме Колмогорова – Арнольда:

$$\frac{N_Y \cdot N_P}{1+\log_2(N_P)} \le N_W \le N_Y \cdot (\frac{N_P}{N_X}+1)\cdot(N_X+N_Y+1)+N_Y$$

Здесь необходимо упомянуть о процедуре Робинсона – Монро, датированной еще 1951 годом [1]!

А еще … Многие методы обучения идут от дельта – правила (или правила Видроу – Хоффа), принцип работы которого в следующем. После расчета выхода сети на выходе k - го нейрона на n - ой итерации получается значение $y_k(n)$ и есть желаемое значение выхода $d_k(n)$. Соответственно ошибка работы сети на n-ой итерации равна:

$$e_k(n) = d_k(n) - y_k(n)$$

Сигнал ошибки инициализует механизм управления, который применяет последовательность корректировок к синаптическим весам нейрона k. Эти изменения нацелены на пошаговое приближение выходного сигнала $y_k(n)$ к желаемому $d_k(n)$. Цель достигается за счет минимизации функции стоимости (еще называется индекс производительности) $E(n)$, определяемый обычно следующим образом:

$$E(n) = 0.5e_k^2(n)$$

где $E(n)$ - текущее значение ошибки. Пошаговая корректировка синаптических весов продолжается до тех пор, пока система не достигнет

устойчивого состояния, то есть пока веса не стабилизируются. При этом дельта правило изменения весов задается выражением:

$$\Delta w_{kj}(n) = \eta * e_k(n) * x_j(n)$$

где η - некоторая константа задающая скорость обучения. Дельта – правило можно сформулировать следующим образом: *«Корректировка, применяемая к синаптическому весу нейрона, пропорциональна произведению сигнала ошибки на входной сигнал, его вызвавший».* Тогда новое значение синаптической связи между нейронами:

$$\Delta w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

Для решения задач с помощью нейросетей необходимо:

- Всесторонне рассмотреть задачу – нейронные сети крайне тяжелый аппарат и, может быть, подобрать другой инструмент для ее решения;
- Собрать входную выборку и провести с ней набор необходимых процедур (шкалирование, масштабирование);
- Провести статистическую обработку входных данных. Как минимум, нужен корреляционный и факторный анализ;
- После проведенного анализа выбрать входы (входные переменные) и выходы (выходные переменные);
- Рассчитать количество скрытых слоев и нейронов в них по специальным формулам;
- Сформировать сеть, инициализировать веса связей
- Обучить нейронную сеть и протестировать на валидационной выборке

Примечания:

- сильно коррелированные входы будут только мешать нейронной сети «понимать» задачу;

- в случае прогнозирования нейронная сеть должна иметь только один вход (по опыту автора), т.к. в противном случае выходы будут «тянуть одеяло на себя»;

- количество слоев и нейронов в них проще брать либо с заведомо недостаточного количества, либо с заведомо большего и далее динамически по результатам работы сети их изменять.

Самый применяемый алгоритм обучения - алгоритм обратного распространения ошибки (backpropagation) будет рассмотрен далее.

Обучение без учителя

Обучение без учителя, также известное как машинное обучение без учителя, использует немаркированные наборы данных (входы - выходы) для

нейронной сети. Эти алгоритмы обнаруживают скрытые закономерности или группировки данных без необходимости вмешательства человека (нет, человек – эксперт потом оценивает качество обучения). Способность нейронных сетей, обучающихся без учителя, обнаруживать сходства и различия в информации делает его идеальным решением для исследовательского анализа данных, стратегий перекрестных продаж, сегментации клиентов и распознавания изображений и прочих задач кластеризации и классификации.

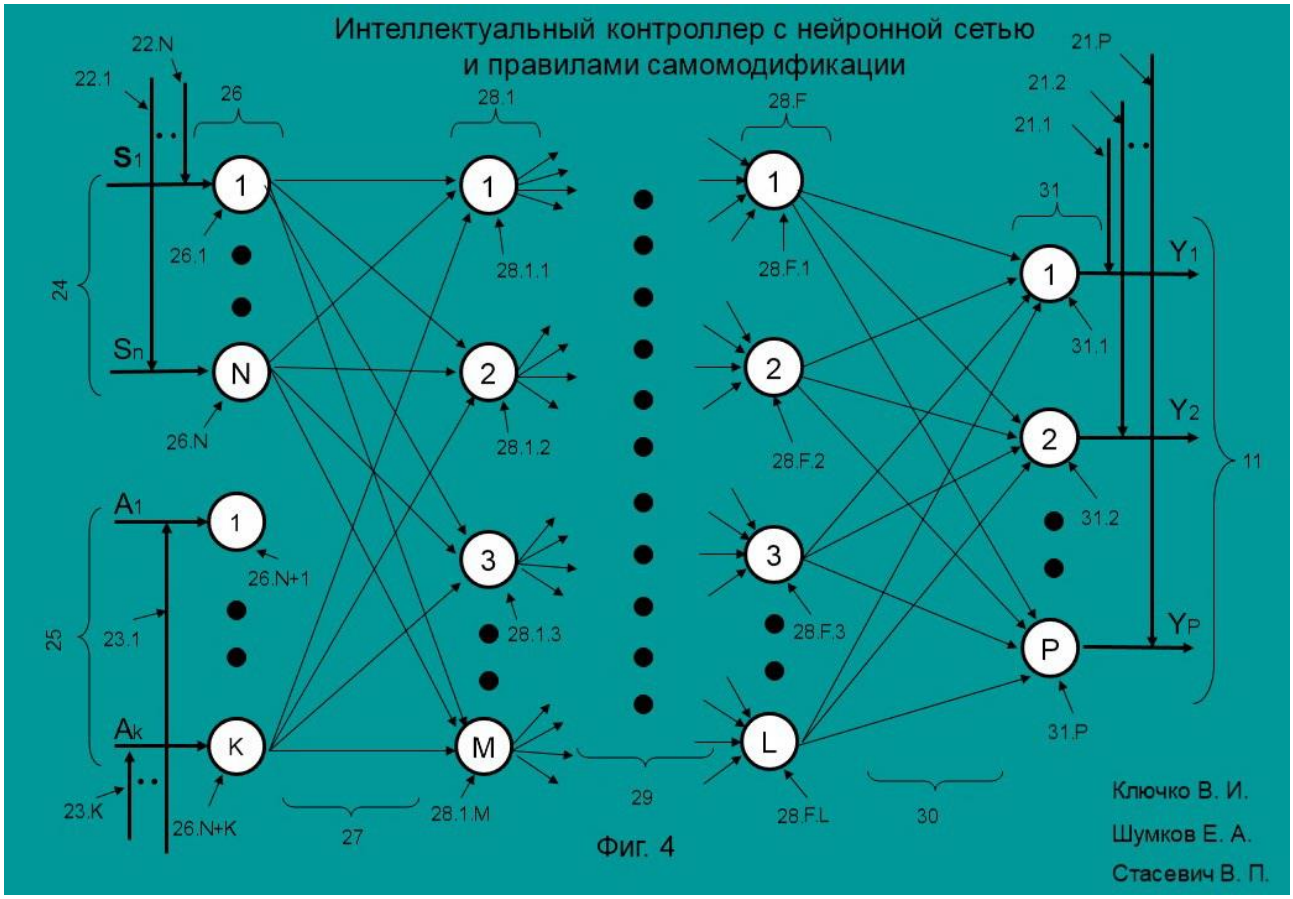


Рисунок 2. Пример нейронной сети

Сигнальный метод обучения Хебба.

Рассмотрим хорошо зарекомендовавший себя алгоритм обучения по Хеббу (кстати, сам Хебб, в разработке не участвовал).

Веса по данному методу корректируются исходя из формулы:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha * y_i^{n-1} * y_j^n$$

где y_i^{n-1} - выходное значение нейрона i слоя $n-1$, y_j^n - выходное значение j-го нейрона слоя n. α - коэффициент скорости обучения. n - здесь и далее,

номер слоя сети. При обучении данным методом усиливаются связи между возбужденными нейронами – это видно из второго слагаемого формулы.

Также есть и дифференциальный метод обучения Хеба.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot \left[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1)\right] \cdot \left[y_j^{(n)}(t) - y_j^{(n)}(t-1)\right]$$

Т.е. к предыдущему значению веса синаптической связи добавляется перемноженное изменение выходов на последней и предпоследней итерациях. Соответственно необходимо хранить предыдущее значение в памяти (по всем нейронам).

Как видно из формулы сильнее всего обучаются те синапсы, которые наиболее динамично изменились в сторону увеличения. Т.е. это придает некоторую эластичность процессу обучения.

Шаги алгоритма обучения методами Хебба (и сигнального и дифференциального):

1. Подготавливаем входную выборку.
2. Определяемся с входными переменными.
3. Решаем сколько будет выходов – нейронной сети «все равно сколько их будет и что они означают».
4. Подбираем функцию активации – либо сигмоида, либо таншенсоида
5. Инициализируем случайными числами в интервале либо [0;1], либо [-1;1] матрицы весовых коэффициентов;
6. Выбираем случайным образом пример из обучающей выборки.
7. Подаем выбранный пример на входы сети и рассчитываем выходы многослойного персептрона;
8. По вышеобозначенным формулам (либо для сигнального, либо дифференциального способа) корректируем веса синаптических связей.
9. Пока веса не застабилизируются – цикл на шаг 6.

Замечание:

- в ряде задач необходимо перебирать количество выходов, т.к. заранее не известно, на сколько классов необходимо разбивать выходное пространство;

- застабилизирование весов – здесь все ложится на плечи разработчика, либо за последние 10 итераций, либо за последние 100…

Отметим, что вид откликов на каждый класс входных образов заранее не известен и будет представлять собой произвольное сочетание состояний нейронов выходного слоя, обусловленное случайным распределением весов на стадии инициализации. При этом сеть способна обобщать схожие образцы, относя их к одному классу. Отметим, что алгоритм Хебба берет свое начало в книге «The Organization of Behavior» (Хебб, 1946 год), в которой приводится четкое определение физиологического правила обучения для синаптической

модификации «эффективность переменного синапса между двумя нейронами повышается при многократной активации этих нейронов через данный синапс».

Другой распространённый алгоритм обучения без учителя – алгоритм Кохонена (не путать с нейронной сетью Кохонена!). Данный алгоритм работает по нижепредставленной формуле. Суть – к предыдущему значению весового коэффициента добавляется разница между выходом нейрона предыдущего слоя и значением весового коэффициента с поправкой на скорость обучения. На начальных этапах скорость обучения можно сделать большой, потом снижать, используя расписание обучения.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot \left[y_i^{(n-1)} - w_{ij}(t-1) \right]$$

Из формулы видно, что обучение сводится к минимизации разницы между входными сигналами нейрона, поступающими с выходов нейронов предыдущего слоя и весовыми коэффициентами его синапсов. У алгоритма абсолютно те же шаги, что и алгоритма обучения без учителя Хебба, но на третьем шаге выбирается нейрон с наибольшим выходным значением и корректировка по формуле Кохонена производится только для него. То есть выбранный нейрон на данной итерации уходит в насыщение, а остальные по сути, притормаживаются. Выбор такого нейрона может происходить, например, расчетом скалярного произведения вектора весовых коэффициентов с вектором входных значений (выигравший нейрон дает максимальное значение произведения).

Есть и другой вариант – расчет расстояния между этими векторами в n-мерном пространстве, где p - размер векторов.

$$D_j = \sqrt{\sum_{i=0}^{p-1} (y_i^{(n-1)} - w_{ij})^2}$$

j - индекс нейрона в слое n, i - индекс суммирования по нейронам слоя $(n-1)$. В данном случае используется принцип «победитель забирает все». Иногда, слишком часто побеждающий нейрон принудительно исключают из рассмотрения, т.к. он постоянно будет «победившим» и изменяться будут только его веса. Можно также использовать так называемые жадные алгоритмы, которые позволяют с небольшой вероятностью выбирать нейрон с не самым большим значением на выходе.

Под **классификацией** обычно понимают задачу в определении принадлежности входного образа, представленного вектором признаков одному или нескольким предварительно определенным классам. Т.е. мы знаем классы, на которые разбивать до решения задачи и сколько их будет.

Под **кластеризацией** обычно понимают процесс классификации «без учителя», т.е. при отсутствии обучающей выборки с метками классов. Алгоритм кластеризации основан на подобии образов и размещает близкие

объекты в один кластер. Т.е. мы не знаем классы и их количество. Обычно необходимо несколько итераций, чтобы прийти к оптимальному количеству классов.

Проблемы обучения нейронных сетей:

Одной из главных проблем использования нейронных сетей, для задачи прогнозирования в частности, является длительное время обучения нейронной сети. Автором совместно с Д. Карловым был предложен скоростной алгоритм обучения нейронной сети. Существуют также и другие методы повышения скорости обучения нейронной сети. Отметим, что повышение скорости, зачастую ведет к снижению качества решения задачи, в случае прогнозирования – к снижению точности прогноза на неизвестных во время обучения примерах. Одним из распространенных способов обучения нейронной сети, альтернативно градиентным методам, является использование генетических алгоритмов. Но, в случае большой размерности нейронной сети, процесс адаптации синаптических весов сети становится громоздким. Но основные проблемы:

- медленная скорость сходимости градиентных методов;

- невозможно получить один и тот же ответ одной и той же нейронной сетью дважды обученной одним и тем же алгоритмом обучения (тут можно говорить о надежности функционирования).

RBF – сети

Сеть радиально – базисных функций (Radial Basis Function Neural Networks, RBFN) – это искусственная нейронная сеть, использующая радиально базисные функции в качестве функций активации. Выходом сети является линейная комбинация радиальных базисных функций входов и параметров нейрона. Нейроны данной сети обладают выраженными локальными характеристиками. Сети радиальных базисных функций имеют множество применений, в том числе функции приближения, прогнозирования временных рядов, классификации и системы управления [Википедия]. Основная идея близка (или, можно сказать, восходит) к методу потенциальных функций и оценке по Парзену (машинное обучение или теория распознавания образов). Радиально – базисная сеть является универсальным аппроксиматором подобно многослойному персептрону.

Сети RBF аналогичны сетям кластеризации K-средних и сетям PNN / GRNN. Базовое отличие состоит в том, что сети PNN / GRNN имеют по одному нейрону для каждой точки в обучающей выборке, тогда как сети RBF имеют переменное количество нейронов, которое обычно намного меньше, чем количество обучающих точек. Для задач с обучающими наборами малого и

среднего размера сети PNN / GRNN обычно более точны, чем сети RBF, но сети PNN / GRNN плохо работают на больших обучающих наборах.

Впервые сформулированы в 1988 Брумхедом и Лоу (но есть данные, что ее первыми предложили Moody и Darken также в 1988 году, но немного ранее). При этом реализуются идеи, сформированные Т.Кавером, которую можно, немного упрощенно, озвучить следующим образом: «линейно неразделимая задача в пространстве размерностью n, может стать линейно разделимой в пространстве с большей размерностью h» (очевидно, что $h>n$). Но вначале про радиально – базисные функции (РБФ).

Радиальная функция – это функция $f(x)$, зависящая только от расстояния между x и фиксированной точкой пространства X. Можно также сформулировать следующим образом: «Радиальная базисная функция – это функция, которая изменяется при удалении от местоположения». Линейные комбинации РБФ можно использовать для аппроксимации заданной функции. Их характерное свойство (радиальных функций) заключается в том, что отклик функции монотонно убывает (возрастает) с удалением от центральной точки. Типичный пример такой функции – функция Гаусса:

$$h(x) = \exp(-\frac{(x-c)^2}{r^2})$$

или

$$h(x) = \phi(\frac{x^2}{d^2})$$

где x – вектор входных сигналов нейрона, а d – ширина окна функции. При этом h – убывающая функция.

Для $c=0$ и $r=1$ функция Гаусса представлена на Рисунке

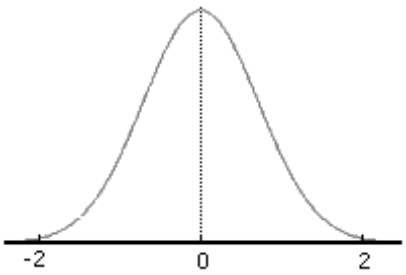


Рисунок 3. Функция Гаусса

Кроме функции Гаусса, также существуют мультиквадратичная, обычная квадратичная, обратная мультиквадратичная, полигармонический сплайн и другие радильно – базисные функции [2].

(Кстати, радиальные базисные функции используются в машинах опорных векторов в машинном обучении!)

Структура радиально – базисной сети – это нейронная сеть, которая содержит слой скрытых нейронов с радиально симметричной активационной, каждый из которых предназначен для хранения отдельного эталонного вектора (в виде вектора весов).

Для построения RBFN необходимо выполнение следующих условий.

Во-первых, наличие эталонов, представленных в виде весовых векторов нейронов скрытого слоя. Во-вторых, наличие способа измерения расстояния входного вектора от эталона. Обычно это стандартное евклидово расстояние. В-третьих, специальная функция активации нейронов скрытого слоя, задающая выбранный способ измерения расстояния. Обычно используется функция Гаусса, существенно усиливающая малую разницу между входным и эталонным векторами.

Или, другими словами, выходной сигнал эталонного нейрона скрытого слоя y_i - это функция (гауссиан) только от расстояния p_i между входными и эталонными векторами.

Обучение слоя образцов-нейронов сети подразумевает предварительное проведение кластеризации для нахождения эталонных векторов и определенных эвристик для определения значений δ_i.

Нейроны скрытого слоя соединены по полно-связной схеме с нейронами выходного слоя, которые осуществляют взвешенное суммирование. Для нахождения значения весов от нейронов скрытого к выходному слою используется линейная регрессия.

Каждый радиально – базисный нейрон производит следующее нелинейное преобразование:

$$y_i = F_i(x) = w_{j0} + \sum_{i=1}^{h} w_{ji}\phi_i(x)$$

где $\phi_i(x)$ – радиально – базисные функции, которые определяют характер отображения из пространства входов в пространство выходов.

Недостатки радиально – базисных сетей:

- предварительно должно быть известно число эталонов, а также эвристики для построения активационных функций нейронов скрытого слоя;

Достоинства радиально – базисных сетей:

- в них нет этапа обучения в обычном понимании (т.е. повышенная скорость).

И еще немного **про историю** нейронных сетей (они уже довольно давно активно применяются…):

- американский банк «Chase Manhattan Bank» в 1985 году начал внутреннюю программу по созданию новых инструментов анализа финансовых рынков, в качестве подрядчика (или аутсорсера) была выбрана компания «Inductive Inference Inc.», возглавляемая в то время др. Д. Ротенбергом. В результате в 1987 году была представлена нейросетевая программа по оценке корпоративных рисков[3] [Trippi];
- американский банк «Citibank» использует нейронные сети для предсказания событий их последствий на финансовых рынках с 1990 года;
- «Chemical Bank» в начале 90-х годов прошлого века использовал нейросистему фирмы “Neural Data” для предварительной обработки транзакций на валютных биржах 23 стран, фильтруя подозрительные сделки;
- «Fidelity of Boston» в 90 – е годы прошлого века использовал нейросети для управления портфелем ценных бумаг с суммарным объемом свыше $3 миллиарда;
- Полностью автоматизированные системы управления портфелями ценных бумаг на основе нейросетей в 90-е годы применяли такие компании, как “Deere & Co” и “LBS Capital”;

Литература:

1. Метод потенциальных функций в теории обучения машин, Айзерман М. А., Браверман Э. М., Розоноэр Л. И., Главная редакция физико-математической литературы изд-ва «Наука», М., 1970, - 384 с.
2. Buhmann, Martin D. (2003), Radial Basis Functions: Theory and Implementations, Cambridge University Press, ISBN 978-0-521-63338-3.
3. Хайкин С. Нейронные сети: полный курс. 2-е изд.: Пер. с. англ. – М.: Издательский дом «Вильямс», 2006 – 1104 с.
4. Заенцев И. В. Нейронные сети: основные модели. Учебное пособие . Воронеж: ВГУ. 1998.- 76с.

[3] Этот пример можно также трактовать, как пример применения нейронных сетей для фундаментального анализа.

ЛЕКЦИЯ 1-3. Задачи, решаемые с помощью нейронных сетей

Пример на обучение без учителя

Рассмотрим такой пример – в Российской Федерации есть много коммерческих и государственных банков и организации необходимо выбрать несколько надежных для ведения своих счетов, осуществления платежей и т.д. У каждого банка есть показатели финансовой деятельности (несколько сотен показателей), которые публикуются в официальных отчетах ЦБ РФ. Как понять – какой банк надежный? (Здесь, конечно, возникает вопрос наличия банков в местности, где находится организация, которая ищет банк. Но будем рассматривать весь набор банков). Для решения данной задачи можно нанять группу экспертов в банковской сфере. А можно решить с помощью обучения без учителя с помощью стандартного многослойного персептрона, т.е. на входе будут показатели банков (естественно все показатели подавать не надо – необходимо провести предварительно факторный и корреляционный анализ), а выходов будет, допустим 4 – т.е. категории на которые делим банки: «надежные», «хорошие», «средние» и «ненадежные».

Пример применения нейронной сети для поиска групп специалистов

Пример применения нейронной сети для поиска характеристик специалистов. Суть заключается в следующем – есть входной поток задач, которые должен выполнить, допустим, отдел разработчиков программного обеспечения. У каждой задачи есть своя спецификация в развернутом виде - ТЗ (т.е. то, что должно быть разработано). Есть группа специалистов (программистов, инженеров, архитекторов СУБД, тестировщиков и т.д.). У каждого специалиста есть свои характеристики по набору навыков, личностные характеристики (например, умение работать в команде), а также есть знания (показатели) о том, как данные специалисты выполняли определенные задачи в прошлом. Задача нейронной сети или их конгломерата (а вообще говоря, менеджера проекта) разбить поступившую задачу на подзадачи, определить сроки выполнения и подобрать специалистов под те или иные подзадачи, чтобы выполнить проект в указанные сроки с заданным показателем качества. Данную задачу, кроме многослойного персептрона, можно решать с помощью сети Кохонена.

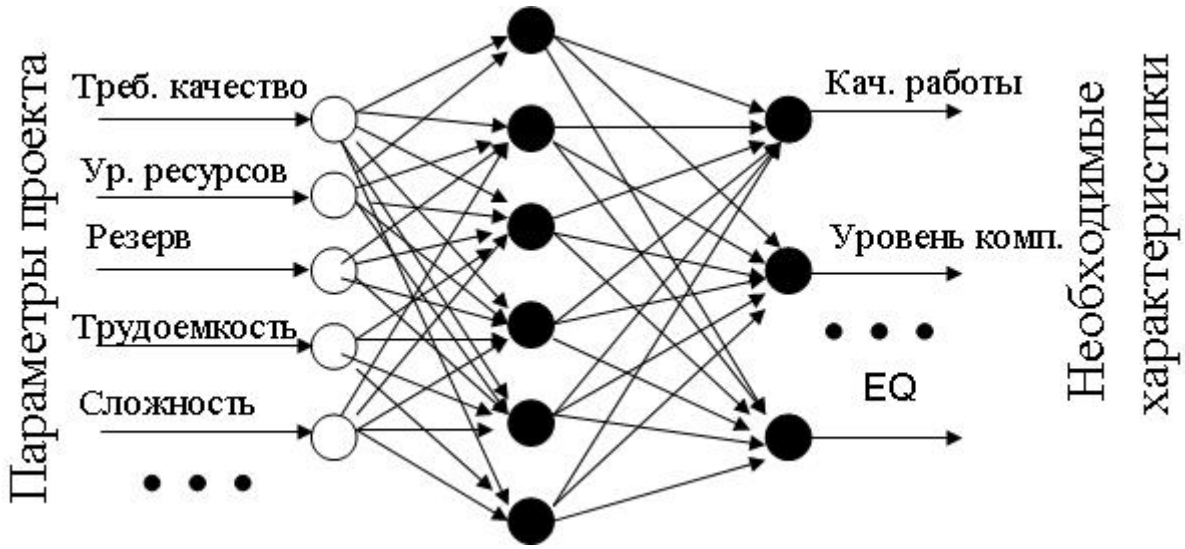
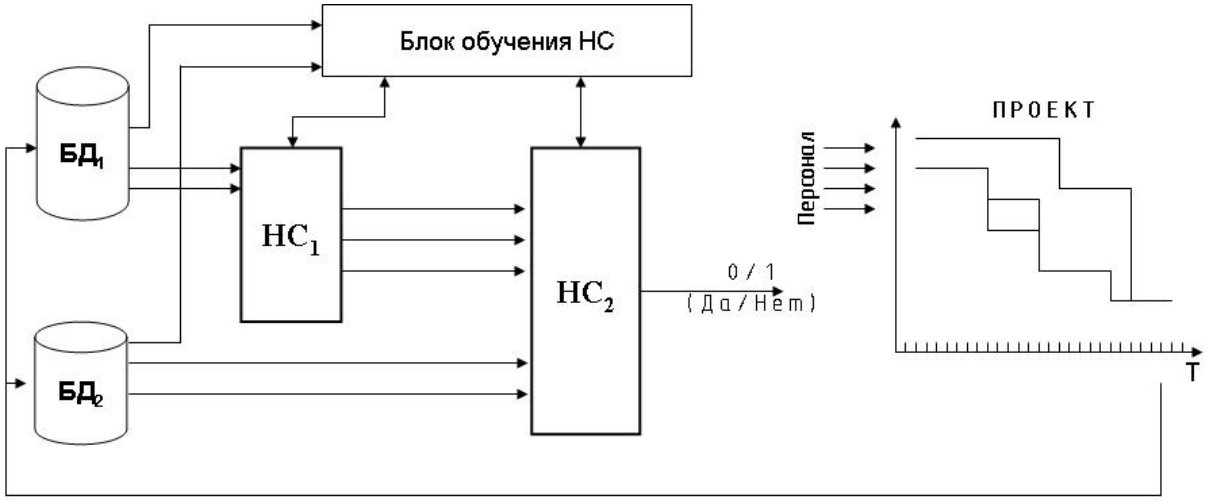


Рисунок 4. Схема нейронной сети для поиска характеристик специалистов

В той же работе предложен и вариант с использованием каскада нейронных сетей для решения данной задачи. Т.е. можно говорить о сложной нейросетей структуре [1].



$БД_1$ – база данных проектов
$БД_2$ – база данных персонала

Рисунок 5. Каскад нейронных сетей для поиска специалистов

Применение нейронных сетей в экспертных системах

Одним из узких мест при использовании ЭС в реальных задачах является их статичность, т.е. жестко определенные базы знаний и правила вывода. Есть, конечно, динамические экспертные системы, блоки приобретения знаний, а также нечеткие, нейронные (НЭС) и нейро – нечеткие варианты экспертных систем, но такие системы сильно индивидуальны и к их построению нет единого подхода. Существует также проблема «приспособления фрейма к реальной ситуации», описанная самим М. Мински. Рассмотрим возможные

варианты использования искусственных нейронных сетей в экспертных системах – на взгляд автора перспективном направлении их развития (входят в т. н. гибридные интеллектуальные системы). Данные системы могут в ряде случаев решать задачи, которые не под силу отдельным «чистым» подходам. В данной работе представим варианты применения нейронных сетей в фреймовых экспертных системах.

Применение нейронных сетей в экспертных системах дает следующие особенности:

- способность обучения на примерах при неизвестных закономерностях между входными и выходными данными без фрагментации выборки данных;

- эффективное сжатие информации за счет построения нелинейных отображений;

- возможность работы с зашумленными данными;

- быстрой и качественной адаптации к новым данным (в т.ч. и введению новых параметров) и др.

В то же время применение НС несет следующие потенциальные опасности:

- могут возникнуть проблемы с обучением нейронной сети;

- учитывая тот факт, что механизм получения решения нейронной сетью обычно рассматривается, как вывод «черного ящика» (за исключением нескольких типов нейронных сетей и алгоритмов их работы, например, М – сети, ALM или KBANN), то применение нейронных сетей в экспертных системах не желательно для критически важных задач.

Одной из проблем в работе фреймовых систем является распознавание текущего входного образа и сопоставление его с конечным фреймом - экземпляром. Проблема возникает (при работе в реальных условиях) в связи с неточными измерениями, зашумленными входами, неточностью модели, стохастическим поведением внешней среды для ряда задач, плохо формализованными знаниями эксперта и т.д.. Для решения данной задачи можно использовать искусственные НС в различных комбинациях. В частности, если входные данные (параметры слотов) можно выразить численно, то возможно применение стандартного многослойного полносвязного персептрона, который необходимо обучить по данным из базы знаний на имеющихся примерах. Учитывая хорошую аппроксимирующую способность нейронных сетей, можно достаточно хорошо бороться с неточностями и зашумленностями входных данных. Понятно, что по мере пополнения базы знаний необходимо дообучать нейронную сеть. Здесь, конечно, может присутствовать ошибка обучения нейронной сети (она может «по своему обобщить» имеющиеся данные) и поэтому участие эксперта все равно необходимо. Отметим, что нейронная сеть в данной комбинации частично

заменяет блок приобретения знаний. Варианты использования нейронных сетей во фреймовых экспертных системах показаны на рисунке 1 пунктиром.

Кроме стандартного многослойного персептрона, учитывая особенности комбинированной задачи классификации и распознавания, можно использовать сеть Кохонена, возможно модифицированную. В простых случаях можно использовать сеть Хемминга или сеть Хопфилда (учитывая ограничения по используемому алфавиту). Для определенных задач, например, прогнозирования, применяются радиально – базисные сети (RBF) или вероятностные нейронные сети (PNN).

Другой интересной комбинацией использования нейронных сетей в фреймовых экспертных системах, является внедрение сети встречного распространения в решатель экспертной систем, т.е. нейронная сеть обучается делать экспертный вывод на базе правил экспертной системы. Как вариант - можно реализовать нейро-нечеткий вывод.

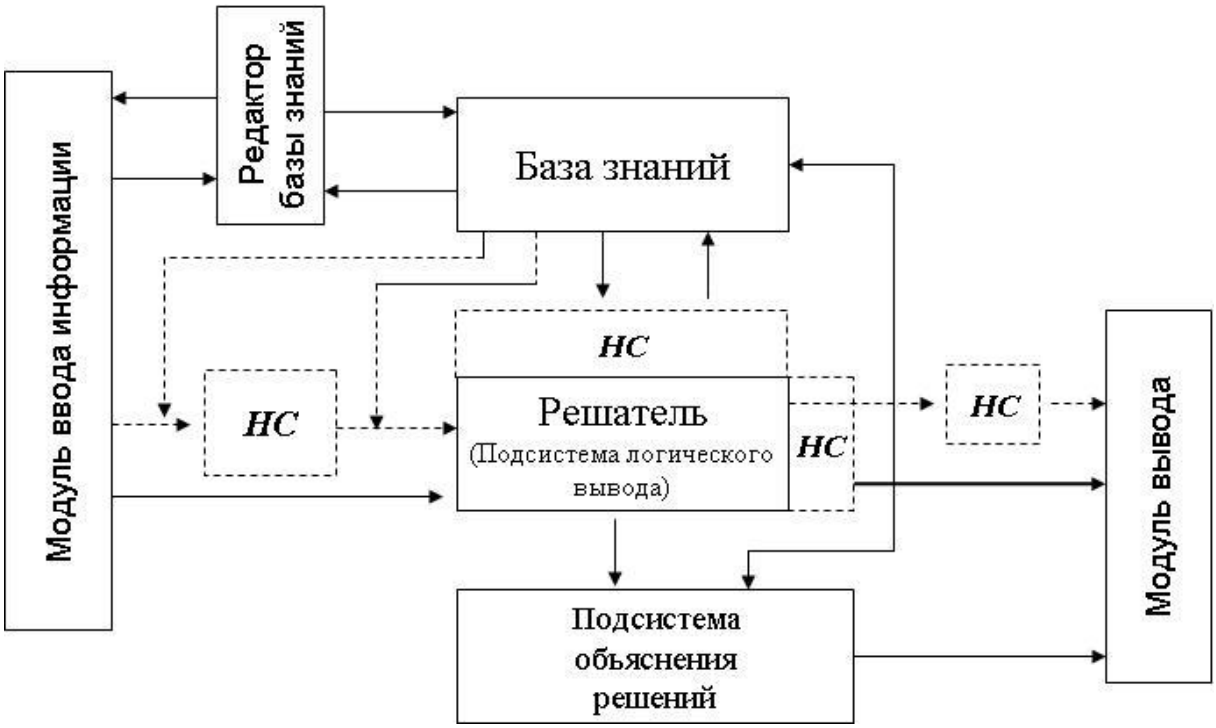


Рисунок 6. Варианты использования нейронных сетей во фреймовых экспертных системах

Еще одним вариантом использования является внедрение НС в качестве прогнозирующего элемента по уже сделанному выводу экспертной системы, если необходим прогноз развития ситуации (фрейм - сценарий). Но учитывая «непрозрачный» вывод нейронной сети, возникает проблема в трассировке результата – по сути будет отсутствовать подсистема объяснения решений.

Необходимо отметить и более «тяжелое» применение нейронных сетей в экспертных системах – генерация и поиск структуры и параметров экспертных систем.

Поиск нейросетевых моделей с помощью генетических алгоритмов

Технология совместного использования искусственных нейронных сетей и генетических алгоритмов известна, как технология COGANN (Combinations of Genetic Algorithms and Neural Networks).

Считается, что совместное использование искусственных нейронных сетей и генетических алгоритмов при правильной комбинации может дать синергетический эффект, т.к. эти два направления обладают существенным потенциалом для решения сложных, трудноформализуемых задач. Но в данном подходе существует еще много нерешенных задач.

Существуют два основных подхода к совместному применению данных технологий, которые можно обозначить, как «обучение нейронных сетей с помощью генетических алгоритмов» и «использование нейронных сетей в генетических алгоритмах». Для обучения нейронных сетей генетические алгоритмы обычно используются двумя способами:

а) поиск субоптимального набора весов синаптических связей искусственной нейронной сети с заданными параметрами с помощью основных операций генетических алгоритмов: скрещивания, мутации и инверсии, а также отбора в следующее поколение;

б) поиск структуры нейронной сети (входы, выходы, количество скрытых слоев и нейронов в них и др.) с использованием основных операций генетических алгоритмов с дальнейшим обучением нейронной сети в отдельном блоке (при этом может также искаться подходящий метод обучения нейронной сети).

Рассмотрим более подробно два варианта объединения нейронных сетей и генетических алгоритмов. Основной задачей можно считать – оптимизация количества слоев и нейронов в них при заданном показателе качества работы нейронной сети. Здесь уместно ввести функцию приспособленности особи.

Вообще говоря, в идеале каждое последующее поколение должно иметь меньшее среднее количество нейронов и соответственно связей при повышении качества работы нейросетей.

А). Поиск топологии нейронной сети и метода ее обучения для конкретной задачи с помощью генетического алгоритма. Сразу отметим, что для некоторых архитектур нейронных сетей существует только один метод обучения и в данном подходе, для сокращения временных издержек, очень важен предварительный анализ и фильтрация входных данных.

Возможны несколько вариантов.

А1). Согласно [5] «Проектирование оптимальной топологии нейронной сети может быть представлено в виде поиска такой архитектуры, которая обеспечивает наилучшее решение конкретной задачи (относительно выбранного критерия). Такой подход предполагает перебор пространства архитектур, составленного из всех возможных вариантов, и выбор точки этого пространства, наилучшей относительно заданного критерия оптимальности».

А2). Генерация определенного количества банков нейронных сетей одного типа, но с различными параметрами и ведение данных банков по эпохам, возможно, с удалением «нежизнеспособных» банков особей в течении эволюции.

А3). Генерация нескольких банков особей с различными архитектурами нейронных сетей и методами обучения, с последующим ведением их через эпохи.

Конечно, было бы расточительно не использовать предыдущие запуски поиска нейронных сетей с помощью генетического алгоритма, но здесь необходим сбор данных, как по обучающим выборкам, так и по ходу работы генетического алгоритма, но данный момент будет рассмотрен в дальнейших работах.

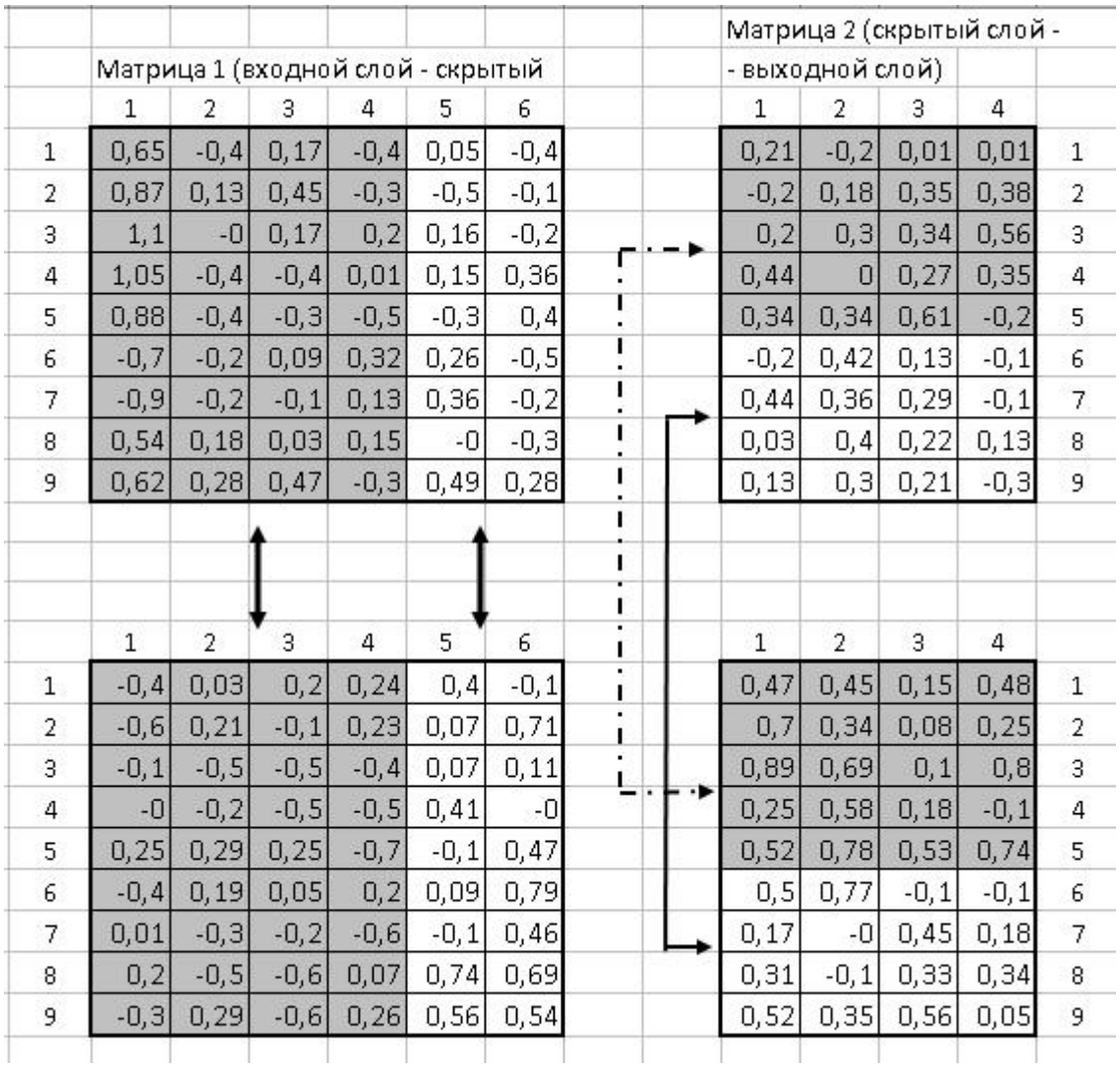
Б). Рассмотрим теперь вариант поиска весов синаптических связей нейронной сети с помощью генетического алгоритма. Возможно, самый простой вариант из технологии COGANN, но также требует значительных вычислительных ресурсов. Для определенности выберем стандартный многослойный персептрон прямого распространения сигнала. Очевидно, что без дополнительных затрат операцию скрещивания для двух нейронных сетей в данном подходе, можно проводить только для нейронных сетей одного размера. Данное ограничение можно, правда частично, обойти следующим образом — генерировать большое количество особей с разными параметрами (количество входов и выходов, количество скрытых слоев и нейронов в них), а далее для операции скрещивание выбирать нейронные сети с одинаковыми параметрами (по т. н. «маске»). Покажем, как работает стандартный вариант на многослойном персептроне с одним скрытым слоем:

а) генерируется банк особей нейронных сетей с одной и той же структурой;

б) далее происходит заданное количество операций скрещивания, мутации и инверсии (количество, впрочем, может быть получено случайным образом в заданном интервале).

в) отбор в следующую эпоху стандартным образом, но, конечно, нужно оценивать приспособленность особи, т.е. нейронной сети. Самый простой способ – оценка точности обучения нейронной сети на валидационной выборке.

Операция мутации в данном случае происходит случайным образом – сначала генерируется случайное число *K* (интервал [0; количество син. связей]), показывающее, сколько синаптических весов нужно изменить. А далее в цикле последовательно генерируется *K* пар случайных чисел M – номер синаптической связи (интервал как у *K*), N – новое значение синаптической связи. Здесь мы уходим от классического кодирования в генетическом алгоритме через 0 и 1. Отметим, что матрицы весов связей можно рассматривать как в общей совокупности (т.е. одна последовательность весов связей в хромосоме), так и раздельно, т.е. каждый шаг операции мутация разбивается на несколько шагов, число которых равно числу скрытых слоев + 1.



	1	2	3	4	5	6
1	0,65	-0,4	0,17	-0,4	0,05	-0,4
2	0,87	0,13	0,45	-0,3	-0,5	-0,1
3	1,1	-0	0,17	0,2	0,16	-0,2
4	1,05	-0,4	-0,4	0,01	0,15	0,36
5	0,88	-0,4	-0,3	-0,5	-0,3	0,4
6	-0,7	-0,2	0,09	0,32	0,26	-0,5
7	-0,9	-0,2	-0,1	0,13	0,36	-0,2
8	0,54	0,18	0,03	0,15	-0	-0,3
9	0,62	0,28	0,47	-0,3	0,49	0,28

	1	2	3	4	5	6
1	-0,4	0,03	0,2	0,24	0,4	-0,1
2	-0,6	0,21	-0,1	0,23	0,07	0,71
3	-0,1	-0,5	-0,5	-0,4	0,07	0,11
4	-0	-0,2	-0,5	-0,5	0,41	-0
5	0,25	0,29	0,25	-0,7	-0,1	0,47
6	-0,4	0,19	0,05	0,2	0,09	0,79
7	0,01	-0,3	-0,2	-0,6	-0,1	0,46
8	0,2	-0,5	-0,6	0,07	0,74	0,69
9	-0,3	0,29	-0,6	0,26	0,56	0,54

	1	2	3	4	
1	0,21	-0,2	0,01	0,01	1
2	-0,2	0,18	0,35	0,38	2
3	0,2	0,3	0,34	0,56	3
4	0,44	0	0,27	0,35	4
5	0,34	0,34	0,61	-0,2	5
6	-0,2	0,42	0,13	-0,1	6
7	0,44	0,36	0,29	-0,1	7
8	0,03	0,4	0,22	0,13	8
9	0,13	0,3	0,21	-0,3	9

	1	2	3	4	
1	0,47	0,45	0,15	0,48	1
2	0,7	0,34	0,08	0,25	2
3	0,89	0,69	0,1	0,8	3
4	0,25	0,58	0,18	-0,1	4
5	0,52	0,78	0,53	0,74	5
6	0,5	0,77	-0,1	-0,1	6
7	0,17	-0	0,45	0,18	7
8	0,31	-0,1	0,33	0,34	8
9	0,52	0,35	0,56	0,05	9

Рисунок 7. Операция скрещивание

Операцию инверсия для данного подхода, на взгляд автора удобно трактовать, как стандартную инверсную перестановку хромосомы. Но можно вводить элемент случайности – для каждой матрицы весов синаптических связей генерировать случайное число (0 или 1) и если сгенерированное число

равно 1, тогда делать перестановку соответствующей матрицы весов связи (точнее – того участка хромосомы, который за нее отвечает). И здесь также можно делать инверсию лишь части хромосомы, выбранной случайным образом.

А вот с операцией скрещивания возможно большее количество вариантов за счет использования одного из главных принципов генетических алгоритмов – случайности. Рассмотрим некоторые из них:

Б.СК.1) берется две полных хромосомы весов синаптических связей нейронной сети, далее в случае многослойного персептрона с одним скрытым слоем генерируется два числа показывающих разрез в матрицах, по которым необходимо провести обмен генами (пояснение см. рисунок 1).

Б.СК.2) генерируется случайное число (0 или 1), которое показывает - производить обмен между матрицами весов синаптических связей разных слоев или между одинаковыми. Затем, в случае разных, генерируется число в интервале [0; min размер числа нейронов], показывающее, сколькими весами связей можно обменяться. Здесь размер возможно лучше задавать двумя числами, чтобы выделять сектор.

Мы здесь не упомянули операцию изъятия из популяции – она остается стандартной.

Понятно, что процесс генетического поиска нейросетевой модели может быть бесконечным, т.к. получить нейронную сеть со стопроцентной точностью воспроизведения обучающей выборки (не говоря уже о работе с неизвестными образцами) для практически значимых задач практически нереально. И необходимо сверху (разработчиком) ограничить данный процесс. Это лучше делать двумя способами:

- введением предельного количества эпох;

- анализом выхода функции приспособленности на т.н. «плато», т.е. отсутствию изменения в течении заданного количества эпох.

При этом необходимо учитывать, что и сам процесс обучения нейронной сети может быть «бесконечным» и его нужно также ограничивать стандартными способами.

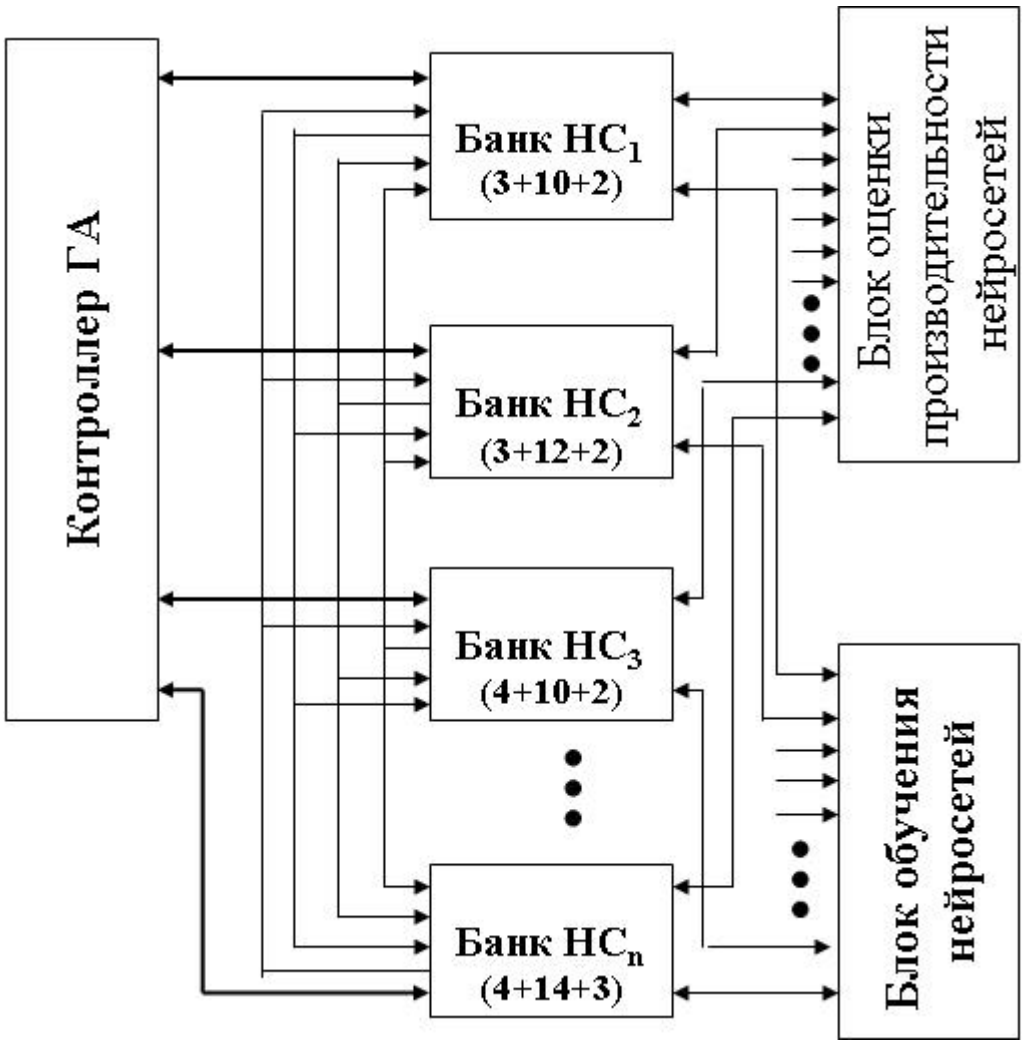


Рисунок 8. Архитектура системы поиска нейросетевых моделей с помощью генетического алгоритма

В конце предложим оригинальную топологию для поиска нейронной сети с помощью генетического алгоритма. Принцип работы следующий – формируется несколько банков особей нейронных сетей, каждый банк содержит особи нейронных сетей одинаковой размерности, например, «4 – 10 – 2», т.е. 4 входа, 10 нейронов в скрытом слое и 2 выхода. Генетический поиск весов синаптических связей идет в каждом банке обособлено, но особи могут перемещаться между банками особей, при этом. если особь структуры «$X_1 – Y_1 – Z_1$» перемещается в банк структуры «$X_2 – Y_2 – Z_2$», то структура данной особи «подгоняется» под структуру нового банка, т.е. если, допустим $X_1 < X_2$, то случайным образом добавляется ($X_2 – X_1$) нейронов во входной слой с соответствующим количеством синаптических связей между новыми нейронами входного слоя и нейронами скрытого слоя. Если, например, $Y_1 > Y_2$, тогда у передаваемой особи удаляется случайным образом выбранных ($Y_1 – Y_2$) нейронов в скрытом слое с соответствующими связями с нейронами во входном и выходном слое и т.д. [6]

Литература:

1. Шумков Е.А. Применение нейронных сетей для подбора состава группы проекта // Электронный журнал КубГАУ, 2019, №153
2. Шумков Е.А. Фреймовые экспертные системы с использованием нейронных сетей // Политематический сетевой электронный научный журнал КубГАУ. 2019, №154. сс. 226 – 232.
3. Minsky M. A. Framework for representing knowledge. Cambridge: MIT Press. 1974.
4. Бычков А.В. Нейросетевое управление рентабельностью предприятия. Дисс. канд. техн. наук. Краснодар: КубГТУ. 2001. 155 с.
5. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. М.: Горячая линия – Телеком. 2006. с. 452.
6. Шумков Е.А. Поиск нейросетевых моделей с помощью генетических алгоритмов // Электронный сетевой политематический журнал «Научные труды КубГТУ». 2021, №4. сс. 44-55

ЛЕКЦИЯ 2-1. Алгоритм обратного распространения ошибки

Нейронные сети с позиций системного подхода можно рассматривать как черный ящик, на вход которого поступает некоторый векторный сигнал, а на выходе формируется сигнал с заданными характеристиками, который обеспечивается в результате обучения сети.

Выделим следующие архитектуры нейронных сетей:

- послойно – полносвязные прямого распространения сигнала;
- с динамическими обратными связями;
- рекуррентные;
- радиально – базисные;
- сверточные;

Для каждой архитектуры нейросетей разработан один или несколько алгоритмов обучения. При этом ряд нейросетей может обучаться только методами с учителем, а ряд только без учителя. Наибольшее распространение получили алгоритмы обучения многослойного послойно – полносвязного персептрона.

Часто для каждой архитектуры свой специальный метод обучения, но есть несколько универсальных. Базовым является следующий метод обновления весов связей нейронных сетей

2.3 Алгоритм обратного распространения ошибки

Алгоритм обратного распространения ошибки (обратное распространение) - важный математический инструмент для повышения точности прогнозов при интеллектуальном анализе данных и машинном обучении. По сути, обратное распространение - это алгоритм, используемый для быстрого вычисления производных. На самом деле этот градиентный метод известен очень давно – см. любой учебник по теории оптимизации.

Данный алгоритм был опубликован в 1986 году в работе Руммельхарта, Хинтона и Вильямса, которые считаются первооткрывателями применения градиентного спуска при обучении нейронных сетей, но первым данный алгоритм предложил Вербос в своей кандидатской диссертации в 1974 году, а также наш, советский кибернетик Галушкин в том же году.

Искусственные нейронные сети используют обратное распространение в качестве алгоритма обучения для вычисления градиентного спуска с учетом весов [1, 2, 5]. Желаемые выходы сравниваются с достигнутыми выходами системы, а затем системы настраиваются путем регулировки веса соединений, чтобы максимально сократить разницу между ними. Алгоритм получил свое название, в связи с тем, что веса обновляются в обратном направлении, от

вывода к вводу. Т.е. нам известна ошибка на выходе, далее считаем изменение весов на последнем слое, затем, через ошибку на последнем слое, пересчитываем матрицу перед предпоследним слоем и т.д.

В этом алгоритме происходит распространение ошибки от выходов нейронной сети ко входам, то есть в направлении обратном распространению сигналов в обычном режиме работы (обычный режим работы - прямой). Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки нейронной сети является величина:

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2$$

где $y_{j,p}^{(N)}$ реальное выходное состояние нейрона j выходного слоя N нейронной сети при подаче на ее входы p - го образа; d_{jp} - желаемое (идеальное) выходное состояние этого нейрона. Суммирование происходит по всем нейронам выходного слоя и по всем обрабатываемым сетью образам.

Важное замечание:

- если у нас в выборке, допустим, 1000 образцов или больше, то такой подход приведет к большому числу вычислений, что сильно увеличит время обучения нейросети. Поэтому для больших выборок используют т.н. «пакетный» способ расчета ошибки – берется определенный процент от выборки (допустим 10%).

- можно считать ошибку только по одному примеру, но здесь можем сильно выиграть в скорости, но проиграть в итоге в точности обучения;

- не забываем, что примеры выбираются случайным образом, т.к. нейронная сеть будет обучаться именно последовательности! Которую мы ей предлагаем.

Алгоритм обратного распространения ошибки приводить не будем – он хорошо показан во всех учебниках по нейронным сетям (Хайкин, Заенцев, Осовский и др.).

Приведем таблицу, в которой собраны наиболее распространенные алгоритмы обучения многослойного персептрона.

Т.к. самым распространенным является метод обратного распространения ошибки, то сравнение обычно проводится именно с ним.

Таблица

Распространенные алгоритмы обучения многослойного персептрона

№	Название	Краткий принцип работы	Плюсы - минусы

1	Многослойный персептрон и алгоритм обратного распространения ошибки (BackProp)	Градиентный способ обучения	«+»: - может аппроксимировать любую функцию; - большое кол-во успешных реализаций; «-»: - медленная скорость обучения[4]; - трудность подбора: кол – ва слоев, нейронов в них, входов и выходов. - трудность отделения локальных и глобальных минимумов.
2	МП и алгоритм Левенберга – Марквардта[5]	Градиентный способ обучения	«+»: - высокая скорость обучения; - точность прогнозирования обычно выше, чем у BackProp; «-»: - может быть только один выход; - трудность реализации.
	Многослойный персептрон и QProp	Модернизированный алгоритм обратного распространения ошибки	«+» Скорость обучения «-»: - невысокая точность

[4] Некоторые недостатки алгоритма обратного распространения ошибки нередко можно обойти специальными методами: динамическим добавлением нейронов, метод эластичного изменения весов и т.д. [**Ошибка! Источник ссылки не найден., Ошибка! Источник ссылки не найден.**]

[5] Иногда называется как «Левенберга - Маркуарда».

			обучения;
3	МП и RProp	Модифицированный градиентный способ обучения	«+»: - очень высокая скорость обучения; - не требователен к вычислительным ресурсам; «-»: - меньшая точность обучения, чем у BackProp
4	МП и алгоритм сопряженных градиентов	Направление градиентного спуска выбирается таким образом, чтобы оно было ортогонально и сопряжено ко всем предыдущим направлениям	«+»: - имеет сходимость близкую к линейной; - заметно быстрее BackProp; - Невысокие требования к памяти; - Эффективен при большом количестве переменных. «-»: - После определенного количества итераций необходим рестарт процедуры расчета; - Необходимы дополнительные расчеты при рестарте
5.	МП и Quikprop	Главное отличие в «факторе момента», который адаптируется к текущему результату процесса обучения.	«+»: - Содержит элементы, предотвращающие зацикливание в локальных минимумах; - Высокая скорость.

			«-»: - Сильно упрощенная формула изменения весов; - Высокая ошибка обучения
6.	Метод Ньютона	Аналогично Ньютона решения систем уравнений	«+» «-» - Требует вычисления второй производной;
	Метод Ньютона с регулируемым шагом	$x^{k+1} = x^k + \alpha_k h^k$, при $\alpha_k = 1$ совпадает с классическим методом Ньютона.	«+» Нередко дает лучшие результаты, чем классический метод «-» - Необходимо вычисление второй производной;
	Метод покоординатного спуска	Стандартный покоординатный спуск	«+»: «-»: - больше вычислений, чем у BackProp
	Метод случайного поиска	Задается начальный вектор параметров. Новый вектор ищется как начальный плюс случайный, умноженный на радиус. Если после определенного количества итераций не уменьшилась ошибка, то радиус сужается и т.д.	«+»: - возможно быстрое обучение «-»: - высокая вероятность не обучить нейронную сеть;

Есть некоторые ограничения на использование того или иного вышеперечисленного алгоритма для многослойного персептрона.

Возникает вопрос – какой задать ошибку обучения нейронной сети? Вообще говоря, все зависит от решаемой задачи. Допустим, мы прогнозируем курс доллара к рублю, тогда ошибка в 5%, если мы угадали направление движения курса, это будет существенный результат. Да даже 10%. Если мы распознаем 10 образов и сеть в итоге ошибается в одном, то это тоже не плохо (если это не система контроля управления доступом). И так далее.. все зависит от задачи.

Проблемы алгоритма обратного распространения ошибки

Главные недостатки алгоритма обратного распространения ошибки следующие:

- алгоритм сходится при бесконечно малых изменениях весов;
- эффект обучения может не доходить до дальних слоев, т.е. алгоритм эффективен для сетей с небольших количеством скрытых слоев (1 - 2);
- для данного алгоритма необходим хороший генератор случайных чисел;
- необходима значительная обучающая выборка;
- невозможно вербализовать результаты работы нейронной сети;
- нет никакой гарантии, что многослойный персептрон обучится определенной задаче за приемлемое время.

Некоторые моменты могут быть убраны за счет использования специальных вариантов предобработки данных.

Методы повышения скорости обучения

Расписание обучения. Веса и пороговые уровни инициализируются случайными значениями. Созданная таким образом сеть абсолютно неадекватна решаемой задаче и может генерировать на выходе только шум. Поэтому ошибка в начале обучения очень велика, и есть смысл вводить большие коррекции параметров. Ближе к концу обучения ошибка значительно снижается, и коррекции должны быть малыми. Чтобы менять длину шагов по параметрам, используют расписание обучения (т.н. learning schedule). Выберем скорость обучения зависящей от времени обучения: $\varepsilon(t)$. Обычно скорость монотонно убывает с ростом времени. Для сходимости алгоритма необходимо:

$$\varepsilon(t) \underset{t\to\infty}{\to} 0 \quad \text{и} \quad \int_0^{\infty} \varepsilon(t) = +\infty$$

Часто выбирают $\varepsilon(t) \to 0$ и $\int_0^\infty \varepsilon(t) = +\infty$ или аналогичные функции.

Считается, что алгоритмы с расписанием обучения сходятся гораздо быстрее, т.к. при старте обучения используются большие коррекции весов за счет большого значения скорости обучения, а в конце небольшой скоростью обучения персептроны плавно доводятся до высоких показателей обучения за счет небольшой скорости обучения. Суть коррекции большой скоростью обучения – функция ошибки проскакивает небольшие локальные минимумы, к которых бы она «застряла» при небольшой скорости обучения. И наоборот, алгоритм «нащупав» глобальный минимум, снижает скорость обучения и ищет оптимум именно в этой «ямке».

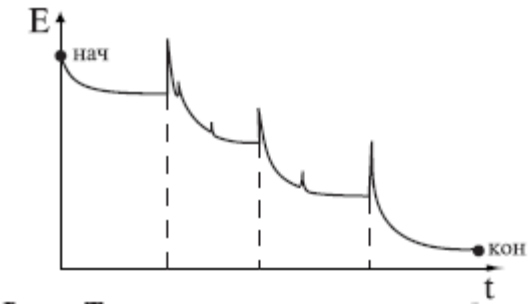
Динамическое добавление нейронов

Адекватный выбор количества нейронов и слоев — серьезная и нерешенная проблема для нейронных сетей. Основным способом выбора остается прямой перебор различного количества слоев и определение лучшего. Для этого требуется каждый раз по новому создавать сеть. Информация, накопленная в предыдущих сеансах обучения, теряется полностью. Начинать перебор количества нейронов можно как с заведомо избыточного, так и с недостаточного. Независимо от этого, новая созданная сеть с другим количеством нейронов требует полного переобучения.

Динамическое добавление нейронов состоит во включении нейронов в действующую сеть без утраты ее параметров и частично сохраняет результаты, полученные в предыдущем обучении. Сеть начинает обучение с количеством нейронов, заведомо недостаточным для решения задачи. Для обучения используются обычные методы. Обучение происходит до тех пор, пока ошибка не перестанет убывать и не выполнится условие [5]:

$$\begin{cases} \dfrac{E(t)-E(t-\delta)}{E(t_0)} < \Delta_T \\ t \geq t_0 + \delta \end{cases}$$

где t — время обучения; Δ_T — пороговое значение убыли ошибки; δ — минимальный интервал времени обучения между добавлениями новых нейронов; $t0$ — момент последнего добавления нейрона; Когда выполняются оба условия, добавляется новый нейрон. Веса и порог нейрона инициализируются небольшими случайными числами. Обучение снова повторяется до тех пор, пока не будут выполнены заданные условия. Типичная зависимость ошибки от времени обучения приведена на рисунке ниже. Моменты добавления новых нейронов отмечены пунктиром. После каждого добавления ошибка сначала резко возрастает, т.к. параметры нейрона случайны, а затем быстро сходится к меньшему значению.



Алгоритм RProp

Данный алгоритм является модернизацией алгоритма обратного распространения ошибки. Принцип действия алгоритма и количество шагов то же. За исключением формул корректировки. RProp хорошо работает во многих ситуациях, потому что он динамически адаптирует размер шага для каждого веса независимо. Большинство вариантов градиентного спуска используют знак и величину градиента. Градиент указывает направление наискорейшего подъема. Поскольку обычно мы хотим найти минимум, мы следуем градиенту в противоположном направлении и это направление полностью определяется знаком градиента. Формулы:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ \Delta_{ij}^{(t)}, & \frac{\partial E^{(t)}}{\partial w_{ij}} \frac{\partial E^{(t-1)}}{\partial w_{ij}} > 0 \\ \eta^- \Delta_{ij}^{(t)}, & \frac{\partial E^{(t)}}{\partial w_{ij}} \frac{\partial E^{(t-1)}}{\partial w_{ij}} < 0 \end{cases},$$

$$0 < \eta^- < 1 < \eta^+$$

Т.е. если знак производной не изменился, то продолжаем изменять в том же направлении для быстрой сходимости, если изменился, то в противоположном. Данный алгоритм в несколько раз быстрее, чем backprop, но менее точен. Его можно использовать для «макетирования», т.е. если с помощью RProp задача решается с приемлемой точностью, то можно для большей точности применить стандартный алгоритм обратного распространения ошибки. Алгоритм хорошо разобран на сайте www.basegroup.ru (доступен также исходный код).

Прогнозирование с помощью нейронных сетей

Нейронные сети – это очень мощный и гибкий механизм прогнозирования. Подробно рассматривать данный способ применения нейронной сети не будем, можно посмотреть, допустим здесь http://apsheronsk.bozo.ru/Neural/Lec9.htm . Приведем только пример скользящего окна

Таблица №1

Структура «скользящего окна»

Вход №1	Вход №2	Вход №3	Вход №4
$P(t)-P(t-1)$	$P(t-1)-P(t-2)$	$P(t-2)-P(t-3)$	$P(t-3)-P(t-4)$
$P(t-1)-P(t-2)$	$P(t-2)-P(t-3)$	$P(t-3)-P(t-4)$	$P(t-4)-P(t-5)$
$P(t-2)-P(t-3)$	$P(t-3)-P(t-4)$	$P(t-4)-P(t-5)$	$P(t-5)-P(t-6)$
…	…	…	…

Еще немного про обучение без учителя

При обучении без учителя существует проблема фиксированных центров, к которым стремятся выходы нейронов последнего слоя нейросети. Суть проблемы – часто недостаточно обучающих примеров. Для избегания данной проблемы, если не увеличивать количество примеров во входной выборке, существует способ с использованием гибридного обучения (не путать с гибридными нейронными сетями):

- 1 момент. использование радиально базисных нейронов в скрытом слое;

- 2 момент. Использование оценки линейных весов в выходном слое.

Литература:

1. Рутковская Д., Пилиньский М.. Рутковский Л. «Нейронные сети, генетические алгоритмы и нечеткие системы». Пер. с польск., И.Д. Рудинского. М.: Горячая линия – Телеком, 2006. 452 с.
2. Хайкин С. Нейронные сети: полный курс. 2-е изд.: Пер. с. англ. – М.: Издательский дом «Вильямс», 2006 – 1104 с.
3. Заенцев И. В. Нейронные сети: основные модели. Учебное пособие . Воронеж: ВГУ. 1998.- 76с.
4. Шумков Е.А. Система поддержки принятия решений предприятия на основе нейросетевых технологий. Дисс. канд. техн. наук. Краснодар: КубГТУ. 2004.
5. Ботин В.А. Адаптивный критик с фильтром Калмана. Дисс. канд. техн. наук. Краснодар: КубГТУ. 2011. 123 с.

Лекция 2-2. Карты Кохонена и другие топологии нейронных сетей

Карты Кохонена

Сети Кохонена относятся к области искусственных топографических карт, обладающих свойства самоорганизации (т.н. карты самоорганизации – SOM, self – organizing map). В них используются модели отображения признаков. Но сети Кохонена также подразумевают под собой следующие разновидности: сети векторного квантования (LVO – learning vector quantization) и сети встречного распространения.

Различают два основных подхода к построению карт саморганизации:

- по модели Уилшоу – ван дер Мальсбурга (близка принципу представления информации о геометрической близости – работы зрительного участка коры головного мозга).

- по модели Кохонена (является более общей – работает по принципу векторного кодирования);

Основной принцип, взятый из нейробиологии – использование двумерной решетки нейронов, первая решетка – состоит из предсинаптических нейронов, а вторая из постсинаптических (входные и выходные нейроны). Первая решетка работает по принципу возбуждающего механизма близкого радиуса действий, а вторая по принципу тормозящего механизма дальнего действия. Между решетками синапсы Хеббовского принципа действия (см. Хайкин).

SOM или нейронная сеть Кохонена - один из основных типов искусственных нейронных сетей. Ее архитектура представляется двумерной сеткой связанных нейронов, которые, по сути, есть многомерные векторы. (Размерность векторов равно количеству дескрипторов.) Обучение сети выполняется в два этапа, первый шаг - это выбор победившего нейрона, а второй шаг - это самоорганизация карты. Задача стандартной сети Кохонена – это разделение множества объектов по кластерам[6]. Количество кластеров указывается количеством выходных нейронов сети Кохонена. Отметим различие между классификацией и кластеризацией – в первом случае количество классов заранее определено, во втором – нет и, собственно, формирование кластеров (их геометрического положения) происходит именно в процессе разнесения объектов (можно сказать, что здесь есть динамика и кластеры могут «плыть» при добавлении новых объектов или удалении старых из выборки. А в классификации классы четко раз и навсегда, в рамках задачи, строго определены).

В общем случае задача кластеризации на сетях Кохонена представляется следующим образом [5]:

[6] Англ. Cluster – пучок, сгусток, группа.

- есть набор объектов, которые описываются вектором параметров $x^p \in X$, имеющих N компонент $x^p = (x_1^p, ..., x_N^p)$. Естественно, все компоненты числового характера[7];

- есть заранее введенное множество классов $C^1, ..., C^M = \{C^m\}$ в пространстве классов C (обычно количество кластеров меньше, чем количество объектов, в случае их равенства, задача сводится к тривиальной);

Необходимо определить ядра классов $\{c^m\} = c^1, ..., c^m$ в пространстве классов C, так что бы меры близости $d(x^p, c^m)$ были минимальны, то есть:

$$\sum_p d(x^p, c^{m(p)}) \to \min$$

Обычно d есть евклидова мера $d(x, y) = \sum_i (x_i - y_i)^2$ - частный случай расстояния Минковского. Еще часто используют Манхетеннское расстояние. Если данное расстояние меньше, чем заданная δ, то объекты считаются близкими и считаются, как расположенные в одном кластере, в противном случае – в разные кластеры.

Функция $m(p)$, определяющая номер класса по индексу p множества объектов $\{x^p\}$, задает разбиение на классы и является решением задачи классификации.

Например, для задачи разбиения банков на категории {надежный, хороший, посредственный, ненадежный} по К параметрам банковской финансовой отчетности, центрами групп будут {ОПУ^1, ОПУ^2 , ОПУ^3, … ОПУ^k }, {ОВП^1 , ОВП^2 , … ОВП^k } и так далее.

Сеть Кохонена выглядит, как показано на рисунке:

[7] Часто возникает задача приведения нечисловых переменных к числовым, но об этом в дальнейшем.

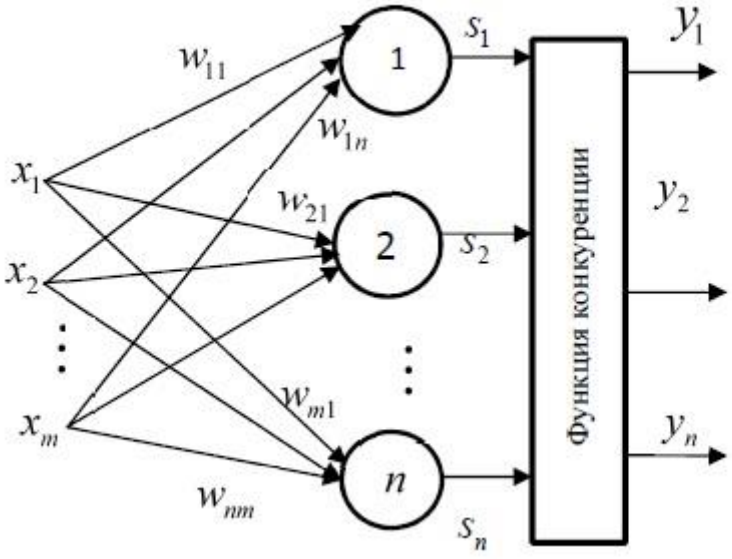


Рисунок 9. Сеть Кохонена

Ядра c^m являются весовыми коэффициентами нейронов. Каждый нейрон сети Кохонена запоминает один класс, то есть величина выхода тем выше, чем ближе предъявляемый образец к данному классу. Суть интерпретатора – выбрать номер нейрона с максимальным выходом (может не показываться). Если использовать функцию SOFTMAX, то выход можно трактовать как вероятность. Меняя количество нейронов, мы можем динамично менять количество классов.

Первым, а лучше сказать нулевым шагом при обучении сети Кохонена является инифиализация весов небольшими значениями с помощью генератора случайных чисел. Для равномерного распределения начальных значений весов коэффициентов синаптических связей применяется метод выпуклой комбинации.

Обучение сети Кохонена происходит следующим образом:

- подаем на вход один из векторов x^p;
- рассчитываем выход каждого нейрона слоя Кохонена и определяем номер выигравшего нейрона m_i, выход которого максимален (их может быть несколько – см. ниже);
- корректируем веса только выигравшего нейрона m_0 по формуле:

$$w_{m_0} = w_{m_0} + \alpha(x^p - w_{m_0})$$

α - скорость обучения, обычно используется монотонно убывающая функция $\alpha(t)$.

Обучение сети Кохонена продолжается пока не застабилизируются веса. Имеется в виду, что среднее суммарное изменение весов за несколько последних итераций (допустим 10) меняется менее, чем на заданную величину.

Примечание – если у нескольких нейронов одинаково максимальный выход, то обычно выигравшим считается первый по нумерации.

По сути, приведенный алгоритм есть т.н. векторное квантование.

Таким образом, сеть учится классифицировать входные векторы. При обучении сети Кохонена возникает проблема так называемых "мертвых" нейронов. Одно из ограничений всякого конкурирующего слоя состоит в том, что некоторые нейроны оказываются незадействованными, даже после прохождения всего этапа обучения. Это проявляется в том, что нейроны, имеющие начальные весовые векторы, значительно удаленные от векторов входа, никогда не выигрывают в процессе обучения, независимо от того, как долго продолжается обучение. Данный момент можно обойти, используя «жадные - алгоритмы», т.е. с небольшой вероятностью давать шанс «выигрывать» нейронам, у которых не самое большое значение на выходе. Этот принцип может также «выбивать» процесс обучения из локальных минимумов.

Сети Кохонена имеют разнообразные применения, в том числе:

- в задачах Data – mining (сегментация рынка, определение групп покупателей и т.д.);

- прогнозирование (косвенное). Например, мы соотносим объект к определенному кластеру и переносим на него свойства, характерные для других объектов данного кластера, т.е. по сути прогнозируем поведение объекта;

- обнаружение аномалий. Главный принцип применения сетей Кохонена для обнаружения аномалий – выявление кластеров, в которые попадает мало объектов. Другой вариант – попадает слишком много объектов, либо много «пограничных» объектов между двумя кластерами.

Отметим следующий момент – интерпретацию получившимся кластерам все равно дает аналитики, который смотрит результаты работы сети Кохонена для конкретной задачи.

Звезды Гроссберга

Стефан Гроссберг трудился на заре нейросетевой науки. Основным его достижением является создание звезд Гроссберга – входной и выходной. Это простые нейросетевые структуры, которые обычно, но незаслуженно, используются в учебных целях. Но они могут использоваться в более сложных нейросетевых топологиях, добавляя к ним новые возможности.

Нейрон в виде входной звезды имеет N входов $\{x_1, x_2, .., x_N\}$, которым соответствуют веса $\{w_1, w_2, .., w_N\}$ и один выход Y, который является взвешенной суммой его входов. Входная звезда Гроссберга учится выдавать сигнал на выходе всякий раз, когда на входы поступает определенный вектор. По сути, входная звезда является детектором совокупного состояния своих входов. Процесс обучения в итерационном виде по формуле:

$$w_i(t+1) = w_i(t) + \alpha \cdot (x_i - w_i(t))$$

где α - скорость обучения, которая обычно в начале ставится равной 0.1 и далее уменьшается. В процессе обучения (настройки) нейрон учится усредненным обучающим векторам.

Нейрон в виде выходной звезды Гроссберга выполняет противоположную функцию – командного нейрона, который выдает на выходах определенный вектор при поступлении нужного сигнала на вход.

Различают также карты Кохонена, которые предназначены для представления на двухмерной карте многомерных объектов. Часто используют либо шестиугольные ячейки, либо прямоугольные.

Для карт Кохонена наиболее распространен алгоритм пакетного обучения, при котором предъявляются все образцы, а затем происходит корректировка весов. Есть еще популярный алгоритм нейронного газа.

Сеть встречного распространения

Пример сети встречного распространения для подбора состава группы проекта (например, ИТ) рассмотрен в работе [2]. Входами сети Кохонена выбраны следующие параметры: количество выполненных подзадач, количество оставшихся подзадач (либо, как процент выполненных к общему), текущие задействованные ресурсы, запас по ресурсам, интегрированные показатели времени выполнения завершенных, текущих и будущих подзадач, приоритеты, наличие параллельных проектов и др. На выходе сети Кохонена мы получим номер класса, к которому относится текущее состояние дел проекта. Далее начинает работать механизм слоя звезд Гроссберга, который по номеру выхода выдает сценарий действий.

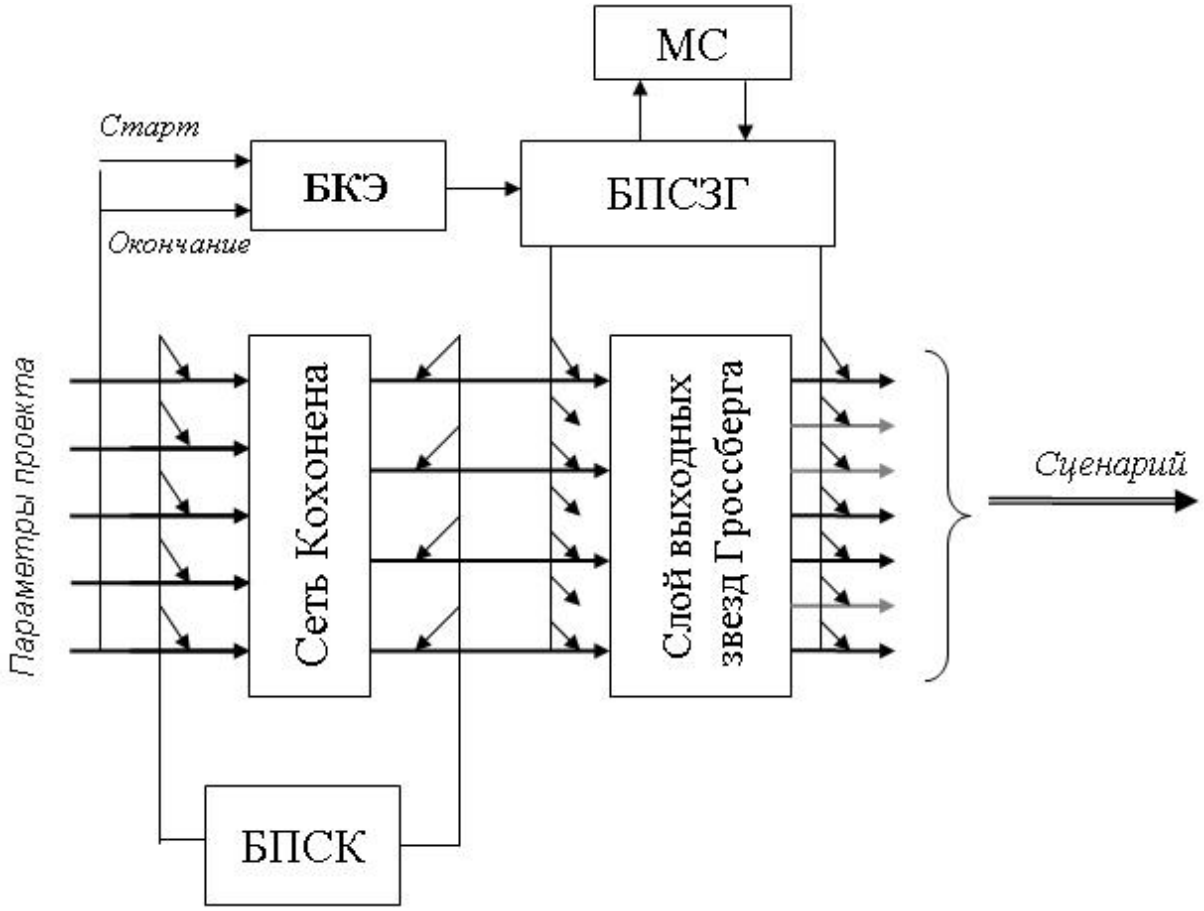


Рисунок 10. Сеть встречного распространения в топологии с подкреплением

Расшифровка к рисунку: БПСК – блок подстройки сети Кохонена, БКЭ – блок коэффициента эффективности, БПСЗГ – блок подстройки звезд Гроссберга, МС – матрица совместимости (для проекта).

Гибридная нейронная сеть

В работе [1] предложена модернизация стандартной гибридной нейронной сети, состоящая из сети Кохонена, слоя выходных звезд Гроссберга и стандартного многослойного персептрона.

Принцип работы данной топологии следующий:

- сеть Кохонена обучается классифицировать данные о залежи по способу МВ. Обучение сети Кохонена выполняется без учителя на основе самоорганизации;

- интерпретатор (или «функция конкуренции») – выполняет вспомогательную функцию по определению максимального выхода;

- веса выходных звезд Гроссберга выставляют в единицу те выходы, которые соответствуют параметрам, характерным для конкретного способа МВ (нехарактерные устанавливаются в ноль);

- многослойный персептрон обучается прогнозировать КИН (коэффициент извлечения нефти) на всех доступных примерах, учитывая только выходы, инициированные сработавшей звезды Гроссберга. Метод обучения – с учителем, например, с помощью алгоритма обратного распространения ошибки.

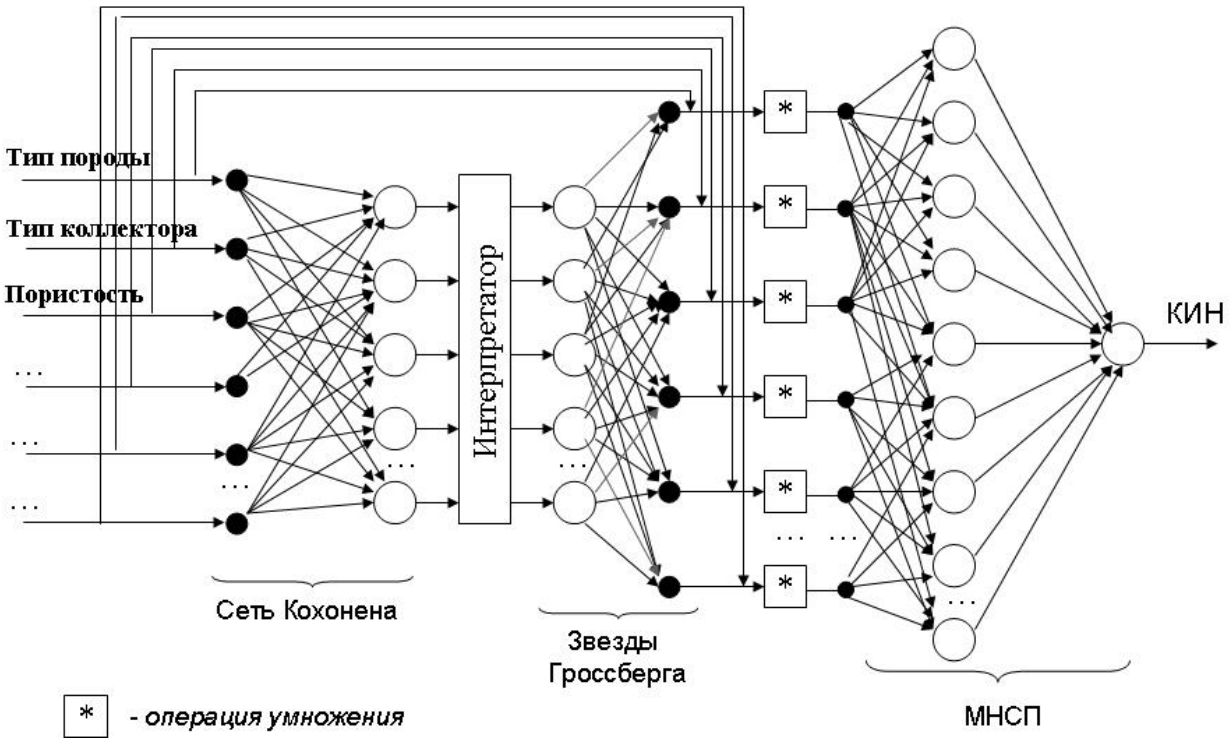


Рисунок 11. Гибридная нейронная сеть

Обучение сети Кохонена и многослойного персептрона производится раздельно стандартными для них алгоритмами обучения.

Пример применения гибридной нейронной сети

В работе [1] рассмотрен вопрос применения, как каскада нейронных сетей, так и гибридной нейросети для определения метода воздействия для извлечения нефти из нефтянных платов на поздней стадии их эксплуатации.

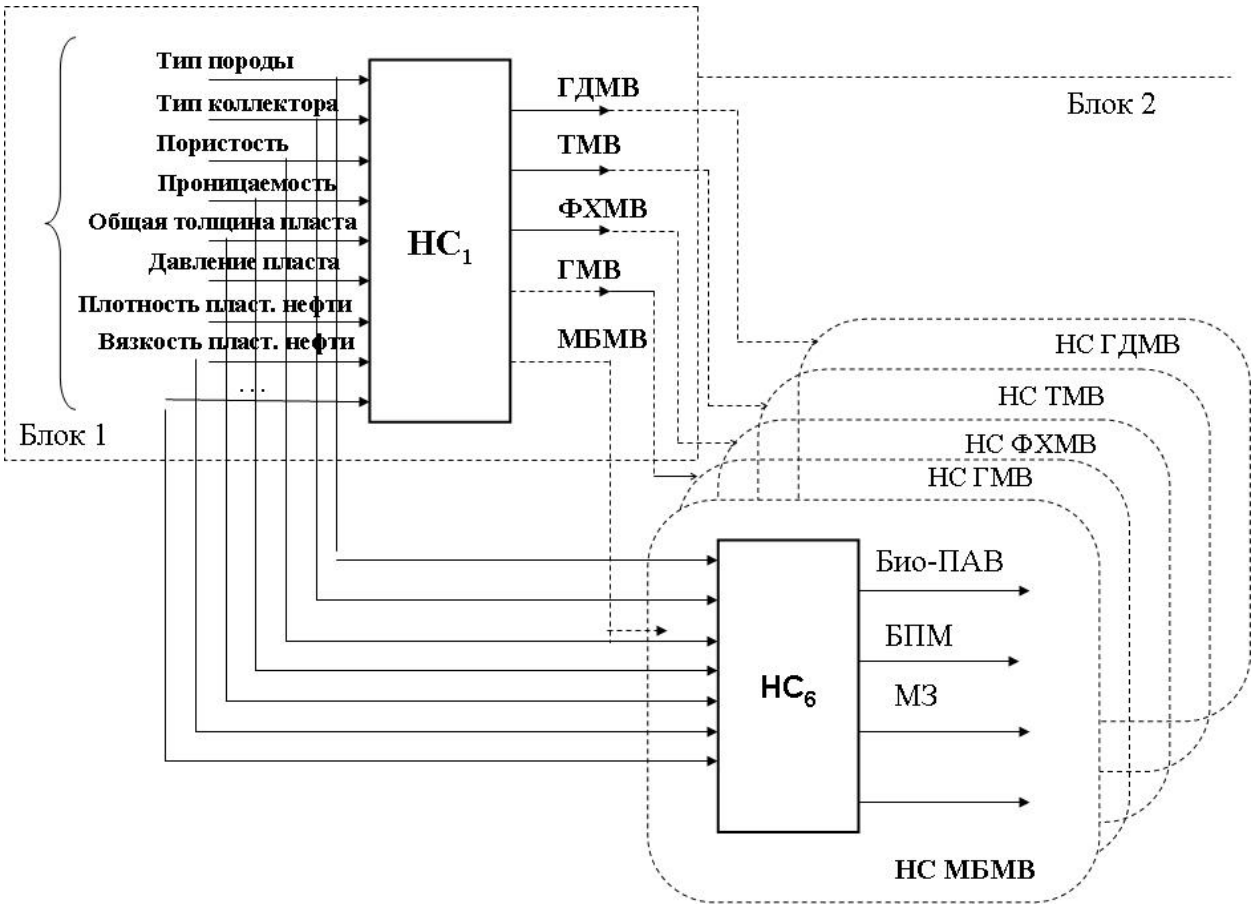


Рисунок 12. Каскад нейронных сетей для определения метода воздействия для извлечения нефти

Пример нейро – нечеткой нейронной сети

Учитывая, что большое количество месторождений нефти уже либо выработано, либо уже истощены, а новые открываются не так часто, то все актуальнее встает вопрос применения методов воздействия на нефтяные пласты с целью, как увеличения темпа отбора действующих месторождений, так и «реанимации» законсервированных. Существует целый ряд уже классических методов воздействия.

У каждого способа есть свои четко определенные достоинства и недостатки, но существует определенная проблема поиска оптимального метода воздействия для конкретного месторождения, как по цене, так и по результатам. Плюс немаловажный в данном случае – экологический аспект. При выборе того или иного вида воздействия требуется сбор большого количества данных, а также разработка физико – математической модели.

Отметим, что существует законтурное и внутриконтурное введение рабочего агента вытеснения. Выбор, в основном, зависит от уровня гидродинамической связи нефтеносной и водонасыщенной частей пласта. Иногда, в случае большой площади залежи, используют комбинированное введение – законтурное и приконтурное. Также существуют варианты по расположению скважин – очаговое, внутриконтурное кольцевое и осевое, и т.д.

Для определения способа вытеснения используются следующие параметры: количество рабочего агента (воды, газа,…) в сутки; давление нагнетания; количество реагентов – коагулянтов (например, сернокислый алюминий при закачке воды); предварительная обработка призабойной зоны пласта (гидравлический разрыв пласта, гидропескоструйная перфорация, разрыв пласта пороховым газом и др.); количество и расположение нагнетающих (инжекционных) скважин[8] и многие другие.

В последние два десятилетия в нефтегазовой области активно применяют искусственные нейронные сети, в т.ч. нейро – нечеткие сети. Это дает следующие преимущества – учет большого количества данных, как по количеству переменных, так и по «глубине» временных рядов, а также поиск скрытых закономерностей (*к применению нейросетей*). Другим немаловажным фактором является то, что способы могут применяться при пересекающихся характеристиках (*к применению нечеткой логики*). В данной работе предложим следующую гибридную нейро – нечеткую систему на базе сети ANFIS. Рассмотрим несколько упрощенный вариант по количеству входных и выходных переменных (данные согласно). Топология показана на Рисунке 1 (A_i, B_i, C_i – низкая, средняя, большая соответственно).

ЕСЛИ <Проницаемость> = *Низкая* И <Вязкость_нефти> = *Низкая* И <Глубина> = *Средняя* ТОГДА <Метод_вытеснения> = «Закачка горячей воды» // *на рис. обозначено, как ЗГВ*

ЕСЛИ <Проницаемость> = *Высокая* И <Вязкость_нефти> = *Высокая* и <Глубина> = *Средняя* ТОГДА <Метод_вытеснения> = «Паротепловое воздействие» // *ПВ*

ЕСЛИ <Проницаемость> = *Низкая* И <Вязкость_нефти> = *Низкая* И <Глубина> = *Большая* ТОГДА <Метод_вытеснения> = «Раствором полимеров» // *РП*

и т.д.

При этом подстройку функций принадлежности целесообразно делать с помощью дополнительной нейронной сети [3].

[8] В качестве инжекционных нередко используются старые эксплуатационные после промывки и очистки.

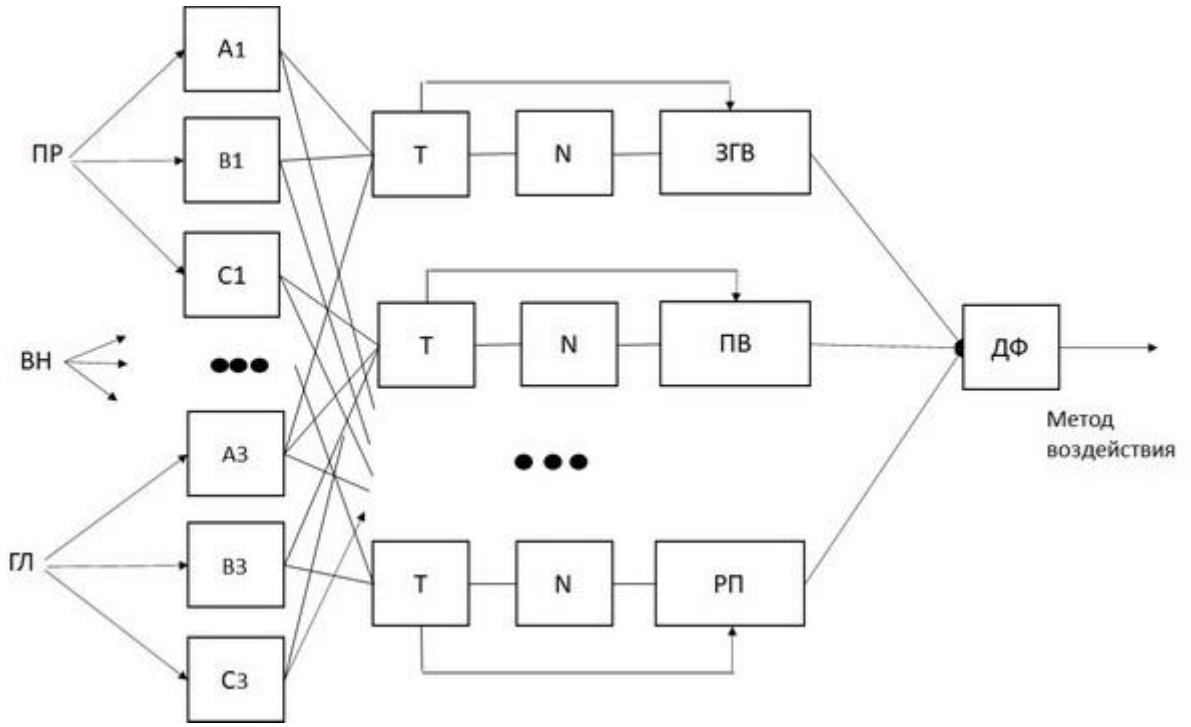


Рисунок 13. Нейро-нечеткая топология определения метода вытеснения

Литература:

1. Шумков Е.А. Применение нейронных сетей в задаче определения метода воздействия для разрабатываемого нефтяного месторождения // Инженерный вестник Дона, 2019, №9
2. Шумков Е.А. Применение нейронных сетей для подбора состава группы проекта // Электронный журнал КубГАУ, 2019, №153
3. Шумков Е.А. Определение способа вытеснения нефти с помощью нейро – нечеткой топологии // // Прогрессивные технологии в современном машиностроении. Материалы и технологии XXI века. XVI Международная научно – техническая конференция. Пенза, 2021 сс. 135 - 138.
4. Заенцев И. В. Нейронные сети: основные модели. Учебное пособие . Воронеж: ВГУ. 1998.- 76с.

Лекция 2-3. Рекуррентные нейронные сети

Краткая классификация рекуррентных сетей:

- сети, имеющие обратные связи локального типа;
- сети, имеющие связи глобального типа.

По большому, существует немного вариантов, как замкнуть обратную связь от нейрона (наверно, все они уже испробованы в том или ином виде.)

Рекуррентные нейронные сети

Рекуррентные нейронные сети (*англ. Recurrent neural network; RNN*) – семейство топологий искусственных нейронных сетей, в которых связи между элементами образуют направленную последовательность, благодаря которой появляется возможность обрабатывать серию событий (последовательную). Еще одна важная особенность – имеется использовать свою внутреннюю память для обработки последовательностей произвольной длины. Рекуррентные сети часто используют для решения задач, где целое можно разбить на части, например, распознавание машинописного и рукописного текста, распознавание речи, работа с видеопотоком и др.

Рекуррентные нейронные сети отличаются от нейросетей прямого распространения сигнала наличием хотя бы одной обратной связи. Таким образом, в этой сети может существовать один слой с обратными связями, а также может быть нейроны с обратной связью, где выход нейрона возвращается в себя как вход. Наличие обратной связи оказывает глубокое влияние на способность к обучению сеть и дает ей некоторые особые свойства (есть аналогия с обратной связью в ТАУ). Кроме того, в этих контурах обратной связи могут используются дополнительные ветви, состоящие из элементов единичной задержки, которые приводят к нелинейному динамическому поведению в силу нелинейного характера нейронов (точнее – их функций активации). Нелинейная динамика имеет ключевое значение роль в рекуррентных сетях.

В отличие от сетей с прямым распространением сигнала, рекуррентные сети могут быть чувствительны и адаптированы к прошлым данным (т.е. они хранят последовательность сигналов за несколько временных итераций). Среди нескольких Архитектуры рекуррентных нейронных сетей более подходящие для моделирования и управления нелинейными системами.

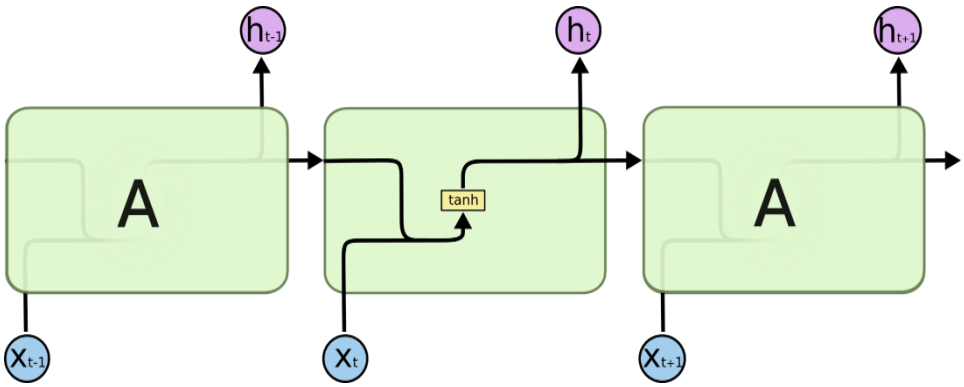


Рисунок 14. Схема рекуррентной сети

Количество типов рекуррентных нейронных сетей достаточно велико, но можно выделить несколько базовых вариантов:

- Сеть с долговременной и кратковременной памятью (LSTM);

- Управляемый рекуррентный блок (GRU).

Обычно топология рекуррентной нейронной сети имеет довольно простую структур, которая состоит из чередующихся нейронов, функция активации которых в большинстве случаев – гиперболический тангенс.

Но! Самой первой рекуррентной нейронной сетью по сути является сеть Хопфилда. С нее и начнем рассмотрение данного подвида топологий нейронных сетей.

Сеть Хопфилда предложил Джон Хопфилд еще в 1982 году.

Структурная схема сети Хопфилда приведена на рис.1. Все узлы в сети Хопфилда являются как входами, так и выходами, и они полностью взаимосвязаны. То есть каждый узел является входом для каждого другого узла в сети. Есть в принципе и обратная связь нейрона сама на себя, но она с нулевым весом (он не изменяется). В предыдущих нейронных моделях, которые вы рассматривали, обработка сигнала идет только в одном направлении: вы начинаете с входных узлов, вычисляете сумму и порог этих значений, чтобы получить выходные данные первого слоя и, возможно, передаете их значения на второй уровень обработки и так далее, но ничего не передается обратно из слоя 2 в слой 1 или даже не передается между узлами в слое 1 (здесь мы не рассматриваем алгоритм обучения! Только расчет состояния сети).

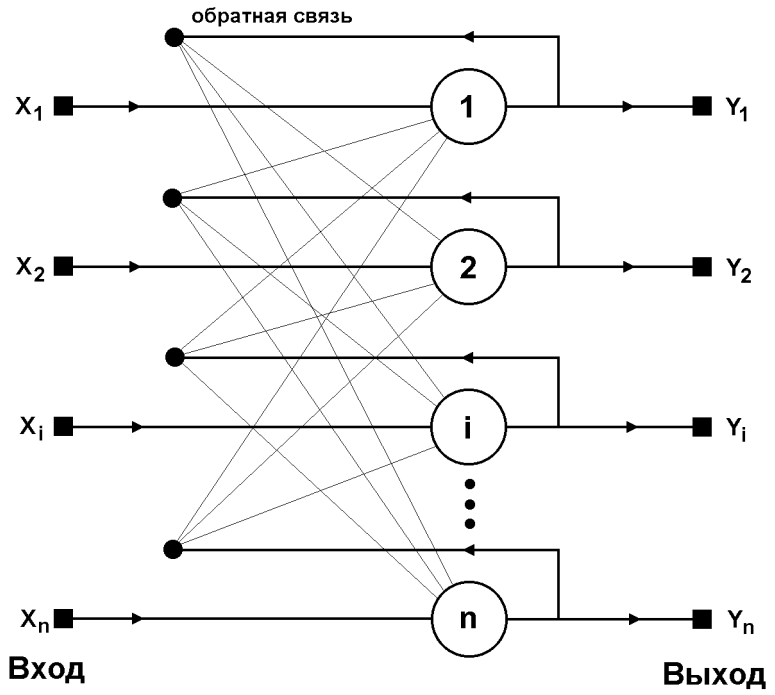


Рисунок 15. Структурная схема сети Хопфилда

Задача, решаемая сетью Хопфилда в качестве ассоциативной памяти, как правило, формулируется следующим образом. Известен некоторый набор двоичных сигналов (изображений, звуковых оцифровок, прочих данных, описывающих некие объекты или характеристики процессов), которые считаются образцовыми. Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить ("вспомнить" по частичной информации) соответствующий образец (если такой есть) или "дать заключение" о том, что входные данные не соответствуют ни одному из образцов. В общем случае, любой сигнал может быть описан вектором $\mathbf{X} = \{ x_i: i=0...n-1\}$, n – число нейронов в сети и размерность входных и выходных векторов. Каждый элемент x_i равен либо +1, либо -1. Обозначим вектор, описывающий k-ый образец, через $\mathbf{X^k}$, а его компоненты, соответственно, – x_i^k, k=0...m-1, m – число образцов. Когда сеть распознает (или "вспомнит") какой-либо образец на основе предъявленных ей данных, ее выходы будут содержать именно его, то есть $\mathbf{Y} = \mathbf{X^k}$, где $\mathbf{Y}$ – вектор выходных значений сети: $\mathbf{Y} = \{ y_i: i=0,...n-1\}$. В противном случае, выходной вектор не совпадет ни с одним образцовым и на выходе будет произвольный набор значений 0 и 1.

Если, например, сигналы представляют собой некие изображения, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, полностью совпадающую с одной из образцовых (в случае успеха) или же "вольную импровизацию" сети (в случае неудачи).

На стадии инициализации сети весовые коэффициенты синапсов устанавливаются следующим образом:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, \qquad i = j \end{cases}$$

Здесь i и j – индексы, соответственно, предсинаптического и постсинаптического нейронов; x_i^k, x_j^k – i-ый и j-ый элементы вектора k-ого образца.

Алгоритм функционирования сети следующий (p – номер итерации):

1. На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов:

$$y_i(0) = x_i, \; i = 0...n-1$$

поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер. Ноль в скобке справа от y_i означает нулевую итерацию в цикле работы сети.

2. Рассчитывается новое состояние нейронов

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p), \; j=0...n-1$$

и новые значения аксонов

$$y_j(p+1) = f\left[s_j(p+1)\right]$$

где f – стандартная активационная функция в виде скачка.

3. Проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да – переход к пункту 2, иначе (если выходы застабилизировались) – конец. При этом выходной вектор представляет собой образец, наилучшим образом сочетающийся с входными данными.

Как говорилось выше, иногда сеть не может провести распознавание и выдает на выходе несуществующий образ. Это связано с проблемой ограниченности возможностей сети. Для сети Хопфилда число запоминаемых образов m не должно превышать величины, примерно равной 0.15•n. Кроме того, если два образа А и Б сильно похожи, они, возможно, будут вызывать у сети перекрестные ассоциации, то есть предъявление на входы сети вектора А приведет к появлению на ее выходах вектора Б и наоборот.

Хотя сеть Хопфилда считается довольно простой нейросетью для «учебных» задач, но есть факты применения ее для серьезных задач глубокого обучения, в частности еще в 1993 году сеть Хопфилда из 1000 слоев решала такие задачи (см. Шмидхуберта).

Другой известной с 80-х годов рекуррентной сеть является сеть Хемминга.

Сеть LSTM

Сеть с долговременной и кратковременной памятью LSTM (другое название – *долгая краткосрочная память*). Данная сеть была предложена в 1997 году З. Хохрайтером и Ю. Шмидхубертом (Германия). Сеть LSTM хорошо приспособлена для решения задач классификации и прогнозирования, когда временные ряды разделены интервалами различной длительности. В частности, считается (и подтверждено экспериментально), что LSTM хорошо работает с распознаванием рукописного текста. Сеть LSTM состоит из цепочек повторяющихся блоков.

Рекуррентные сети LSTM и GRU имеют топологию «цепочка» из повторяющихся блоков (см. рисунок ниже). При этом базовая версия LSTM состоит из четырех взаимодействующих между собой слоев нейронов.

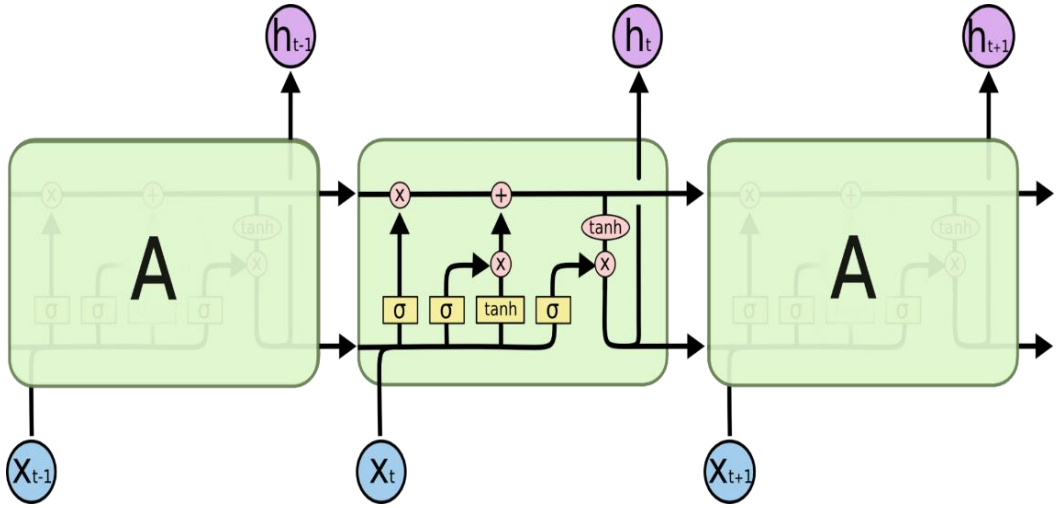


Рисунок 16. Схема LSTM сети (Википедия)

Алгоритм работы сети LSTM:

1. Определение ненужной (или избыточной) информации из входного сигнала. Это происходит в первом (сигмоидальном) слое. Сигмоидальный слой возвращает «0» (игнорировать) или «1» (сохранить) для каждого числа из состояния ячейки. Формула вычисления:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

2. Работа слоя входного фильтра
   2.1. Первый слой входного фильтра (функция активации - сигмоида). В данном слое определяется, какие значения необходимо обновить.
   2.2. Второй слой фильтра забвения формирует вектор новых значений (по определенным на первом слое входного фильтра). Формулы работы:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c * [h_{t-1}, x_t] + b_c)$$

3. Процедура замены старого состояния ячеек памяти на новые значения по формуле:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

4. Определение выходных значений сети LSTM
   4.1. Применение сигмоидального фильтра
   4.2. Применение фильтра с гиперболическим тангенсом – значения ячеек проходят через данный фильтр и умножаются на выходы сигмоидального слоя. Формулы:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

На последнем шаге необходимо определить, что нужно получить на выходе сети. Выходные данные будут основаны на состоянии ячейки с применением некоторых фильтров. Первым делом применяется сигмоидальный фильтр для определения информации, которая будет подаватьбся на выход. После этого все значения состояния ячейки проходят через слой гиперболического тангенса, для того чтобы на выходе получить значения от -1 до 1 и перемножаются с данными полученными на выходе сигмоидального слоя. Это позволяет выводить лишь необходимую информацию (источник – Википедия).

Нейронная Сеть управляемых рекуррентных блок GRU

По топологии сеть управляемых градиентных блоков GRU (англ. Gated Recurrent Unit) управляемых рекуррентных блоков является упрощенной сетью LSTM. Предложен в 2014 году (авторы: Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun & Bengio, Yoshua). В данной сети нет выходных вентилей, а также вентиль фильтра забвения и входной вентиль объединены в один вентиль, если сравнивать с LSTM, т.е. меньше параметров и вычислений. Сеть GRU обучается быстрее, чем LSTM, но по результатам экспериментов дает хуже результат. Также существуют ограничения на длину входной информации. Можно использовать сеть GRU для макетирования нейросетевого решения поставленной задачи и, в случае успешного результата, решать задачу на сети LSTM.

GRU направлен на решение проблемы исчезающего (уменьшающегося) градиента (*Vahishing gradient problem*), которая появляется при работе со стандартной рекуррентной нейронной сетью - в некоторых случаях градиент будет крайне малым, что фактически не позволит синаптическому весу изменить свое значение. В худшем случае это может полностью остановить дальнейшее обучение нейронной сети. GRU также можно рассматривать как разновидность LSTM, потому что оба они спроектированы одинаково и в некоторых случаях дают одинаково хорошие результаты. Чтобы решить проблему исчезающего градиента, GRU использует вентили обновления и сброса. По сути, это два вектора, которые определяют, какую информацию

нужно передать на выход. Их особенность заключается в том, что их можно обучить хранить информацию за несколько итераций (можно расширять до определенных размеров), не стирая ее во времени и не удаляя информацию, не имеющую отношения к текущей решаемой задаче.

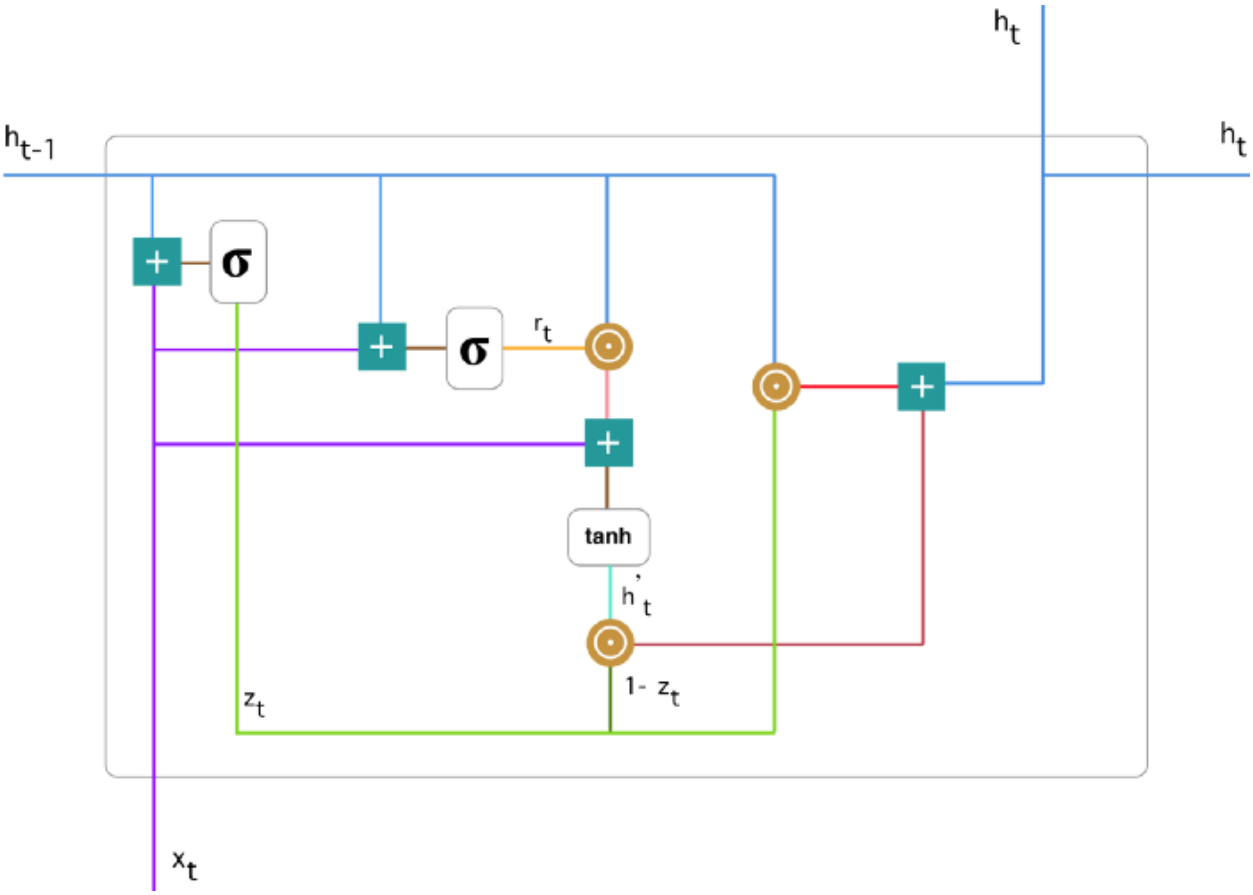


Рисунок 17. Структура сети GRU

Кратко остановимся на вопросе предварительной очистки текстовых данных для работы с ними рекуррентными нейронными сетями. Если использовать русский алфавит, то необходимо заменить буквы «ё» на «е», «й» на «и», а также убрать предлоги.

Введем обозначения:

X_t – входной вектор на шаге t

h_t – вектор скрытого слоя на шаге t

δ- нейросетевой слой с сигмоидальной функцией активации

tanh – нейросетевой слой с тангенсоидой в качестве функции активации.

Также на схеме есть элементы поэлементного умножения и сложения.

Алгоритм работы сети GRU:

1. Инициализация

2. Определение, что забыть, а что оставить по формуле

$$Z_t = \delta(W_z * [h_{t-1}, x_t] + b_z)$$

Далее Z_t, поэлементно вычитается из 1 и умножается на вектор состояния на предыдущем слое

$$h_t \otimes (1 - Z_t)$$

$$\widehat{h_t} = \tanh(W_{xh} * x_t + W_{hh} * (r_t \otimes h_{t-1}) + b_z)$$

Или в развернутом виде то, что получаем на выходе

$$\widehat{h_t} = h_{t-1} \otimes (1 - Z_t) + \widehat{h_t} \otimes Z_t$$

В принципе работы блока GRU есть некоторое сходство с принципов работы фильтра Калмана.

Исследователи отмечают, что управляемый рекуррентный блок дает очень хорошие результаты при работе с аудио форматами, а также в синтезе речи.

Сеть Элмана

Нейронная сеть Элмана (иногда пишут *Элмена*) относится к рекуррентным сетям (иногда их относят к частично – рекуррентным сетям [**Ошибка! Источник ссылки не найден.**]). Была предложена в 1990 году.

Элман предложил частично рекуррентную нейронную сеть, где прямые связи модифицируются, а повторяющиеся соединения фиксируются. Она имеют набор контекстных узлов для хранения внутренних состояний. Таким образом, сеть Элмана имеет уникальные возможности по хранению и использованию предыдущих значений.

Структура сети Элмана следующая – в дополнение к стандартным скрытому и выходному слоям нейронов добавлен специальный слой обратной связи (т.н. *контекстный слой* или *слой состояний*). Данный слой получает сигналы со скрытого слоя и через элементы задержки z^{-1} подает их на входной слой, тем самым сохраняя обрабатываемую информацию в течении одного временного такта (т.н. *эпизодная подача данных* или *limited memory effect*).

Элементы сети Элмана – стандартные нейроны, адаптивные линейные ассоциаторы и элементы задержки z^{-1}.

Решающим (или строительным) блоком сети Элмана являются классические нейроны с типом функции активации – гиперболический тангенс.

Выходной слой сети Элмана может быть нелинейным.

Алгоритм работы сети Элмана можно описать с помощью следующих уравнений:

$$u_j(k+1) = \sum_{j=1}^{n} w_{ji}^{[1]} x_i(k) + \sum_{i=1}^{n1} w_{ji}^{C} o_i(k) + \theta_j^{[1]}$$

$$o_j(k+1) = \phi(u_j(k+1)) = \tanh(u_j(k+1)), \quad j = 1,2,3...n1$$

$$y_j(k+1) = \sum_{i=1}^{n1} w_{ji}^{[2]} o_i(k+1) + \theta_j^{[2]}, \quad j = 1,2,3...m$$

Так же можно описать в матричной форме, см. [**Ошибка! Источник ссылки не найден.**]. Структурная схема показана на рисунке ниже (взято из [**Ошибка! Источник ссылки не найден.**]):

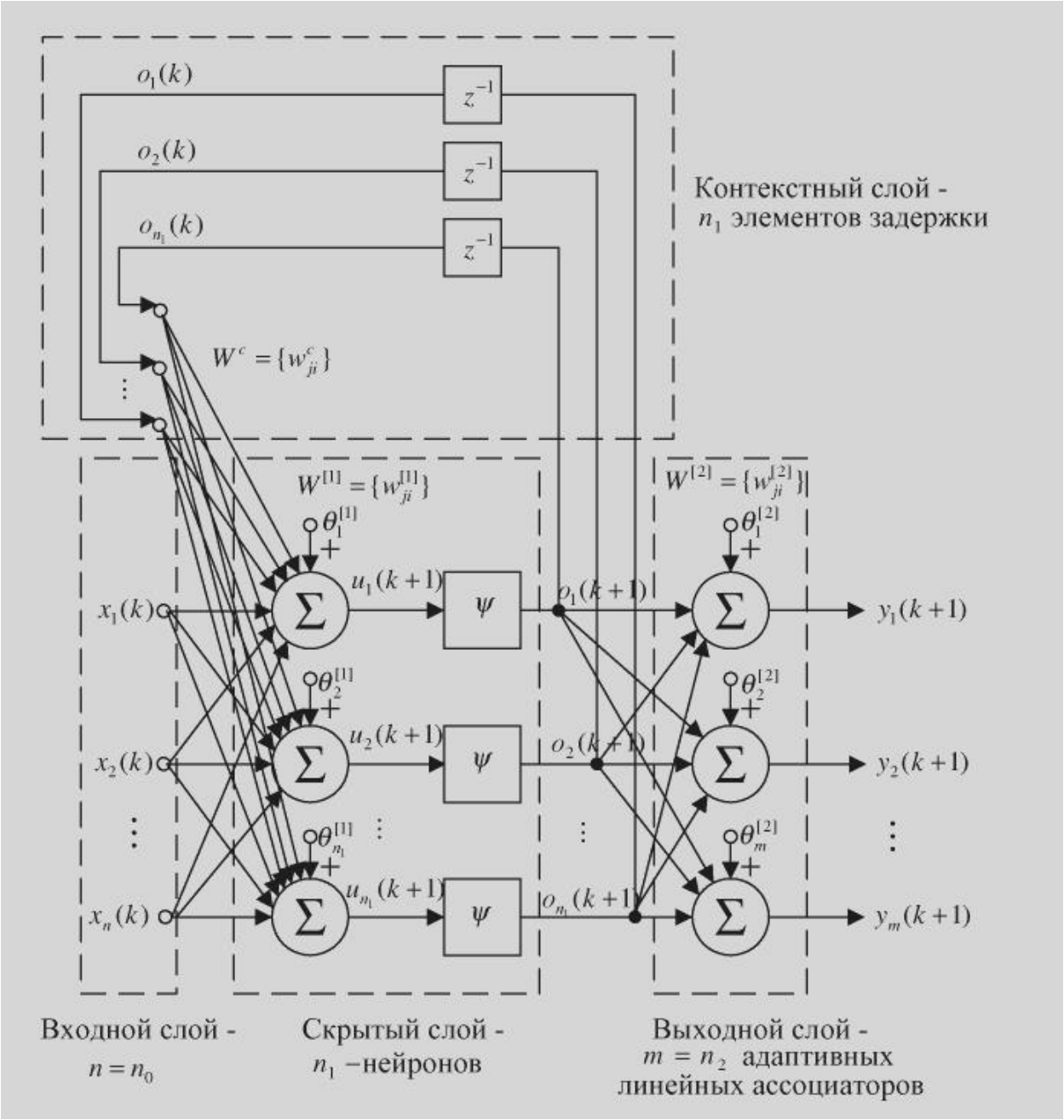


Рисунок 18. Нейронная сеть Элмана

Алгоритм обучения сети Элмана основан на градиентном методе наискорейшего спуска, основанного на работах Р.Вильямса и Д.Зипсера. По сути - это тот же самый алгоритм обратного распространения ошибки.

Существуют следующие возможные варианты работы сети Элмана (подробнее см. указание к практическим занятиям).

Литература:

6. Рутковская Д., Пилиньский М.. Рутковский Л. «Нейронные сети, генетические алгоритмы и нечеткие системы». Пер. с польск., И.Д. Рудинского. М.: Горячая линия – Телеком, 2006. 452 с.
7. Хайкин С. Нейронные сети: полный курс. 2-е изд.: Пер. с. англ. – М.: Издательский дом «Вильямс», 2006 – 1104 с.
8. Заенцев И. В. Нейронные сети: основные модели. Учебное пособие . Воронеж: ВГУ. 1998.- 76с.
9. Шумков Е.А. Система поддержки принятия решений предприятия на основе нейросетевых технологий. Дисс. канд. техн. наук. Краснодар: КубГТУ. 2004.

ЛЕКЦИЯ 3-1. Сверточные нейронные сети

Глубокое обучение

Вначале поговорим о «глубоком обучении». Считается, что глубокое обучение появилось в 2006 году, т.е. не так давно. Но, на самом деле, по мнению автора – это некий «ребрендинг» или «модное название» обычного нейросетевого обучения, на больших объемах информации. Да, объемы информации сильно возросли и требуются все большие вычислительные мощности для их обработки и новые алгоритмы, в т.ч. в области искусственных нейронных сетей.

Есть несколько мнений, что считать глубоким обучением. Одни говорят, что это нейронная сеть с большим (глубоким) числом слоев, но нейронные сети в тысячу слоев и даже больше, тестировались (и работали) еще в начале 90-х годов!

Другой мнение – «глубокое» значит глубокий, детальный поиск зависимостей, параметров и т.д. При этом не обязательно применять «глубокие» нейронные сети. Это скорее относится к машинному обучению или искусственному интеллекту в целом.

Еще один момент – при анализировании «в глубину» вначале выделяются общие, крупные признаки или детали, а затем они последовательно детализируются до самых маленьких, «глубинных» признаков и деталей (и так, кстати, работают сверточные нейронные сети).

Введение в сверточные нейронные сети

Преамбула: *«В 2012 году первая сверточная нейросеть AlexNet становиться победителем международного конкурса «ImageNet Large-Scale Visual Recognition Challenge» выиграв с большим отрывом (количество ошибок было равно около 15%, в отличие от второго места с более чем 26%)»*[9]

Сверточные нейронные сети (англ. - *convolutional neural network, CNN*) – это специальная архитектура нейронных сетей, которая была предложена Яном Лекуном в уже далеком 1988 году. Можно дать следующее определение: «*Сверточная нейронная сеть – тип глубокой нейронной сети, показывающий наилучшие результаты в области распознавания изображений*».

Немного из истории работы Лекуна. Суть заключалась в том, что необходимо было распознать в автоматическом режиме ZIP – коды на конвертах (по всей видимости это была сортировочная станция на одном из Парижских почтампов). Лекуну это удалось сделать с помощью многослойного персептрона (?) с добавлением нескольких предварительных слоев (о них позже). Хотя эксперимент был удачным, заказчиков, да и научную

[9] https://neurohive.io/ru/vidy-nejrosetej/alexnet-svjortochnaja-nejronnaja-set-dlja-raspoznavanija-izobrazhenij/

общественность насторожил тот факт, что сеть обучалась более 3-х суток! И это в 1988 году! (Сейчас сверточные сети обучаются неделями, правда на гигантских массивах информации, но никого это не останавливает). Поэтому работа была положена, что называется «в ящик» до … примерно начала 2000-х годов. Следует отметить, что есть параллели работы Лекуна с топологией неокогнитрона, предложенного еще в конце 70-х годов.

Также кратко представим биографию Я. Лекуна (*Yann LeCun*):

- родился в 1960 году в Париже (француз по национальности);

- докторская степень по информатике в Университете Пьера и Мари Кюри (Париж, Франция);

- работал в «AT&T Bell», сейчас один из руководителей в Facebook (Meta);

- преподавал в Нью-Йоркском университете;

- лауреат Премии Тьюринга в 2018 году совместно с Бенжио и Хинтоном за развитие глубокого обучения;

- кроме всего прочего – один из создателей технологии сжатия DjVu

Преимущества и недостатки сверточных нейронных сетей

1. Один из лучших алгоритмов по распознаванию и классификации изображений.
2. По сравнению с послойно - полносвязной нейронной сетью (типа персептрона) — гораздо меньшее количество настраиваемых весов. Нейросеть обобщает информацию, а не запоминает пиксели каждой показанной картинки в мириадах весовых коэффициентов, как это делает персептрон *(это стандартная формулировка из сети Интернет – на самом деле в СНС и используется многослойный персептрон! Т.е. существует некоторое недопонимание топологии).*
3. Удобное распараллеливание вычислений, а, следовательно, возможность реализации алгоритмов работы и обучения сети на графических процессорах.
4. Относительная устойчивость к повороту и сдвигу распознаваемого изображения.
5. Обучение при помощи классического метода обратного распространения ошибки.

К недостаткам можно отнести:

1. При работе с реальными данными в практических задачах учитывается большое количество гиперпараметров, т.е. на самом деле существует проблема сложности настройки.

2. При работе с большими данными сеть требует значительных аппаратных ресурсов.

Архитектура сверточной нейронной сети

Сверточная нейронная сеть состоит из следующего типа слоев:

- слой свертки (*convolutional layer*) – главный блок сверточной нейронной сети (также называют *сверточный слой*).

- слой подвыборки (subsampling layer)

- слой активации () – получает на вход скалярный результат каждой свертки. При этом можно считать, что функция (или слой) активации встроена в слой свертки.

- несколько слоев обычного многослойного персептрона.

Сверточные слои представляют собой слои на которых происходит операция свертки (отсюда и название данных слоев) – ядро свертки, представляющее из себя матрицу весов (обычно это матрица 3x3) «скользит» над двумерным изображением, поэлементно выполняя операцию умножения с той частью входных данных, над которой оно сейчас находится, и затем суммирует все полученные значения в один выходной пиксель

Ядро свертки

Ядро свертки есть матрица, обычно 3x3, «пробегает» над двумерным изображением (но лучше сказать – матрицей, ведь не только с изображениями работает сверточная нейронная сеть) и преобразует в двумерную карту признаков. При этом признаки являются взвешенными суммами входных признаков (*расположенных примерно в том же месте, что и выходной пиксель на входном слое*).

То, что было рассмотрено выше называется операцией 2-D свертки и в таком случае изображение имеет один входной канал. Однако на практике большинство изображений имеют минимум 3 выходных канала (по цветам RGB – Red, Green, Blue). И в таком случае ядро свертки будет разным для каждого входного канала, а сумма ядер будет называться фильтром. Таким образом, операция 3-D свертки будет проходить немного иначе[10].

Для того чтобы создать выходной канал – карту признаков, каждое ядро скользит по своему входному каналу, создавая свою версию этого канала. Затем все эти версии суммируются и создают новый общий канал, к которому

[10] https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/

добавляется базовое смещение, образуя при этом новую карту признаков (активации).

Каждый слой нейросети находит похожие на находящиеся в нем фильтры участи изображения, что похоже на операцию свертки двух функций. Математически эта операция показана формулой:

$$f[x,y]\cdot g[x,y]=\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty}f[n_1,n_2]\cdot g[x-n_1,y-n_2]$$

Проведя аналогию со сверткой изображения можно сказать, что эта операция является поэлементным умножением и суммой матрицы изображения $f[x,y]$ и фильтра $g[x,y]$.

Размерность выхода сверточного слоя, где вход имел размерность $w\times h$, а фильтр $k_x\times k_y$, считается по формуле:

$$n\times\left(w-k_x+1\right)\times\left(h-k_y+1\right)$$

где n – количество сверток слоя.

После одного или нескольких слоев свертки обычно идет слой подвыборки, где происходит уменьшение размерности изображения путем сжатия блоков признаков пикселей (обычно размером 2x2) до одного признака с использованием функции максимума. Можно так же использовать и другие функции, например, среднего значения, однако на практике преимущество пуллинга с функцией максимума неоспоримо.

После чередования операций сверток и подвыборки данные преобразуются из двумерного формата в одномерный и передаются в полносвязный слой обычной нейронной сети. Последний слой обеспечивает классификацию изображения.

Сам классификатор представлен слоями обычных нейронов, связанных друг с другом. Выход каждого нейрона рассчитывается стандартно по формуле:

$$OUT=f(w_0+\sum_{n=1}w_n x_n)$$

где w_n – вес синаптической связи нейрона;

x_n – входные сигналы;

w_0 – параметр смещения;

$f()$ – функция активации

Проблема сглаживания краев матрицы изображения

В процессе сверки края матрицы, по сути, обрезаются, т.к. крайние пиксели (элементы матрицы) никогда не оказываются в центре ядра (потому что ядру тогда не над чем будет скользить). Для того чтобы такого не происходило следует добавить к краям поддельные пиксели (обычно нулевого значения) – это операция называется **padding**. Таким образом, ядро при проскальзывании позволяет неподдельным пикселям оказываться в своем центре, а затем распространяется на поддельные пиксели за пределами края, создавая выходную матрицу того же размера, что и входная.

Функции активации

Изначально в сверточных нейронных сетях использовались тангенсоида и сигмоида (см. Лекция 1), но в начале 2000-х годов было предложено использовать функцию ReLU (*rectified linear unit*) и ее модификации Noisy ReLU, Leaky ReLU. Использование данной функции позволяет обойти проблему затухания или увеличения градиентов если используются сигмоида или тангесоида (которые нелийнены).

График функции ReLU показан на Рисунке .

Формула

$$f(s) = \max(0, s)$$

Некоторые преимущества использования ReLU[11]:

а) производная ReLU равна либо нулю, либо единице и поэтому не может произойти разрастание или затухание градиентов;

б) использование данной функции приводит к прореживанию весов;

в) меньше вычислительных затрат в сравнении с вычислением сигмоиды и гиперболического тангенса;
г) отсекает ненужные детали в канале при отрицательном выходе.

К недостаткам можно отнести следующее – при большом значении сигнала, проходящего через ReLU, нейрон может быть «загнан» в ноль и далее любой сигнал, идущий через данный нейрон всегда равен нулю. Данная проблема обходится за выбора надлежащей скорости обучения. Другой вариант - подобрать другие весовые коэффициенты, но основное решение заключается в том, чтобы изменить горизонтальную линию графика на небольшой наклон

$$f(x) = \max(0.01x, x)$$

[11] https://habr.com/ru/post/348000/

Эта функция называется Leaky ReLU. Она идентична обычной ReLU, однако, при отрицательных данных градиент не будет равным нулю, при этом она по-прежнему легко вычислима

Принцип работы сверточной нейронной сети

Суть работы сверточной нейронной сети заключается в переходе от конкретных особенностей изображения к более абстрактным деталям за несколько итераций к выделению понятий более высокого уровня. Считается, что сеть самонастраивается и сама формирует необходимую иерархию абстрактных признаков, отбрасывая при этом незначительные детали и выделяя существенные.

Примеры применения сверточной нейронной сети

А) Распознавание изображений

Б) Работа с видеопотоком

Программное обеспечение для использования сверточных нейронных сетей: сети SqueezeNet, Inception-v3, ResNet-101, GoogLeNet, VGG-19 и т.д. или импортировать сеть из TensorFlow, Keras и Caffe.

Литература:

1. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

ЛЕКЦИЯ 3-2. Алгоритм работы сверточной нейронной сети (обхождение некоторых проблем)

Предварительная подготовка данных

После того как была выбрана функция активации, нужно подготовить данные. Если количества данных не хватает, например, недостаточно большой объем фотографий, и существуют опасения того, что это повлияет на корректность обучающего процесса, то можно провести процедуру, называемую аугментацией.

Аугментация данных – метод создания дополнительных данных из уже имеющихся. Есть множество вариантов аугментации, но самыми популярными являются отражение по горизонтали и вертикали, случайное кадрирование, то есть обрезание фотографии со случайным центром и изменение цвета. Так же можно применять различные комбинации этих способов, например, поворот и затем сразу масштабирование и изменять насыщенность пикселей как целого, так и отдельных частей изображения.

После того как количество данных стало приемлемым, нужно приступить к их предобработке. Одним из самых популярных способов является центрирование данных к нулю. Это нужно для того, чтобы сделать их симметричными относительно нуля и избежать их смещения, а также предотвратить ситуацию, когда данные оказываются только положительными или отрицательными. Чтобы воспользоваться данным методом нужно вычесть среднее значение из каждого элемента. При обработке изображения мы можем вычесть одно значение из каждого пикселя.

Далее следует нормализовать значения данных так чтобы они были в диапазоне [-1, 1], или же разделить каждое измерение на его стандартное отклонение. Нормализацию стоит применять, только тогда, когда разные входные признаки имеют разный масштаб или единицы измерения, но имеют одинаковую ценность для модели. В случае изображений, пиксели находятся в диапазоне [0, 255] – таким образом, не имеет смысла проводить нормализацию.

Работа слоя подвыборки

После одного или нескольких слоев свертки обычно идет слой подвыборки, где происходит уменьшение размерности изображения путем сжатия блоков признаков пикселей (обычно размером 2x2) до одного признака с использованием функции максимума. Можно так же использовать и другие функции, например, среднего значения, однако на практике преимущество пуллинга с функцией максимума показывает лучшие результаты [1].

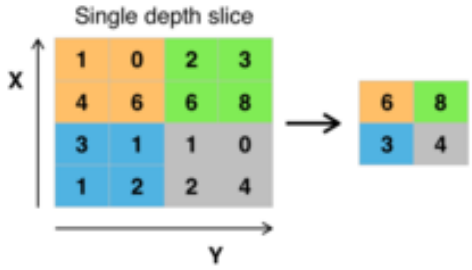


Рисунок 19. Пример операции подвыборки

После чередования операций сверток и подвыборки данные преобразуются из двумерного формата в одномерный и передаются в полносвязный слой обычной нейронной сети типа многослойный персептрон прямого распространения сигнала. Он и отвечает за классификацию изображения.

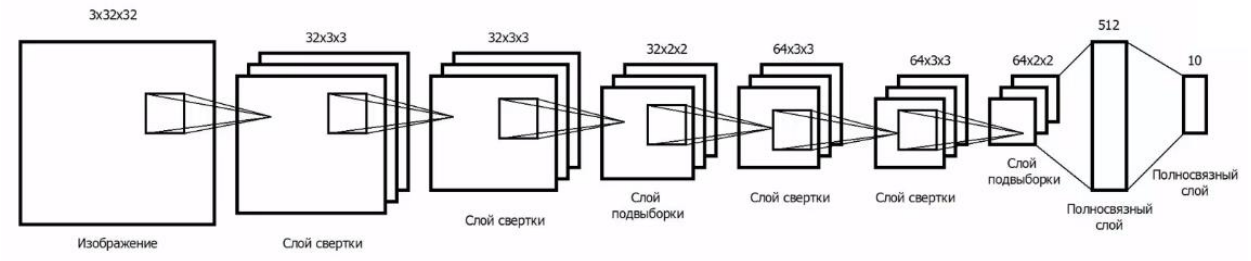


Рисунок 20. Пример сверточной нейронной сети (Википедия)

Проблема построения архитектуры сверточной нейронной сети для прикладной задачи

Сверточные нейронные сети состоят из 3-х видов слоев: слои свертки, слои подвыборки, а также полносвязный слой или другими словами – классификатор. Слои свертки осуществляют создание карт признаков, которые на первых слоях содержат низкоуровневые признаки изображения, такие как края, границы, а на последних более абстрактные части уже объектов на изображении. Слои подвыборки или пуллинга необходимы для уменьшения размерности изображения путем группировки ряда пикселей в один. Их целью считается уменьшения количества параметров в сети для облегчения вычислений и контроля над переобучением. Полносвязный слой выполняет роль инструмента для классификации изображения. Построение архитектуры сверточной нейронной сети творческая задача, ведь в зависимости от конкретной задачи, быстродействия или даже размера датасета оптимальная модель сети может меняться.

Поэтому важно обратить свое внимание не сколько на саму архитектуру, а сколько на обучение самого классификатора. Ведь он определяет предсказание для изображений, и от точности его работы будет зависеть эффективность нашей архитектуры.

Чтобы правильно обучить классификатор, нужно выяснить оптимальную настройку весов для модели. Для этого существует такое понятие как функция потерь. Процесс поиска оптимальных весов называется оптимизацией.

Алгоритм обучения сверточной нейронной сети

По сути – это стандартный алгоритм обратного распространения ошибки (см. Лекция 2-1). Но, можно немного его видоизменить.

Для ряда задач используют рекуррентные сети типа LSTM и GRU – для них выбирается соответствующий алгоритм обучения (опять же стандартный!).

Функция потерь для сверточной нейронной сети

Функция потерь вычисляет эффективность нейронной сети по отношению обучающей выборки к необходимым результатам. Эта функция одномерна. Варианты расчета функции потерь для СНС – можно выделить следующие:

- перекрестная энтропия;
- с.к.о.;
- экспоненциальная;
- расстояние Кульбака – Лейблера и некоторые другие.

Формулы можно посмотреть на слайдах презентации к лекции. Некоторые комментарии:

- основной момент – необходимо рассчитать ошибку между априорными и полученными данными;

- по ряду мнений – перекрестная энтропия является более динамичной и предпочтительна для использования в качестве оценки функции потерь.

- у функции потерь не должно быть каких либо зависимостей от активационных значений кроме выходных[12].

Приведем пример функции потерь по перекрестной энтропии для двух несвязанных переменных p и q:

[12] https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri/

$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

где q – заданное распределение вероятностей, p – истинное распределение.

Другая трактовка перекрестной энтропии для распознавания изображения:

$$-\sum_{c=1}^{M} (y_c \cdot \log \hat{y}_c)$$

где M - количество классов (т.е. 1000 в ImageNet), а y_C - прогноз модели для этого класса (т.е. результат softmax для класса C). Из-за того, что метки закодированы, а y представляет собой вектор единиц и нулей (1000 × 1), y_C равно либо 1, либо 0.

Еще один пример функции потерь:

$$L = \frac{1}{N} \sum_i L_i\left(f\left(x_i, W\right), y_i\right)$$

где L_i – это функция потерь, x_i – входные данные, W – весовая матрица, y_i – прогноз для класса, L – общие потери всех обучающих данных.

Методы оптимизации нейронных сетей

Кратко рассмотрим вопрос оптимизации нейронных сетей и сверточных в частности. Данный вопрос крайне важен, т.к. нейронная сеть избыточного размера (для определенной задачи) дольше обучается, т.к. размерность матриц выше и соответственно дольше рассчитывает выходное значение по входам. Нейронная сеть меньше необходимо размера для определенной задачи будет дольше обучаться с непредсказуемым результатом в виде плохого итогового качества обучения.

Другой момент данной проблемы – мы обучили нейросеть, получили необходимую ошибку на выходе и хотим снизить количество нейронов, а то и слоев в нейросети, чтобы сеть быстрее работала. Особенно это касается аппаратной реализации нейросетей.

Еще одна сторона проблемы – а можно ли улучшить результаты работы уже обученной нейронной сети??

При всей важности вопроса, до сих не существует гарантированно работающих методов оптимизации нейронных сетей. Можно выделить следующие:

- оптимизатор импульса (Momentum);

- адаптивная оценка моментов (Adam), которая сочетает в себе среднеквадратичное распространение и оптимизацию импульса, а также использует среднее значение вторых моментов градиентов (про Adam на следующей лекции).

Еще трудным одним моментом для сверточных нейронных сетей является необходимость наличия большого (можно даже сказать - огромного) количества обучающих примеров. Датасеты для современных прикладных задач могут достигать сотен мегабайт! Но это не всегда возможно (правда, в некоторых задачах можно обойтись относительно скромным числом примеров). Как можно обойти проблему недостаточного количества примеров? Есть достаточно хорошо зарекомендовавшие себя способы:

- принцип селекции – здесь работает подход ансамбля нейронных сетей. Кратко алгоритм: входная выборка случайным бьется на две части (обучающую и тестовую), далее генерируется некоторое количество нейронных сетей, тестируются и из них выбираются несколько лучших. Здесь возможно несколько проходов (а это уже каскадный подход);

- принцип консилиума – на одной и той же обучающей выборке тренируются несколько нейронных сетей с разными архитектурами и методами обучения (+ в рамках одной топологии с разным количеством входов), далее выбираются лучшие или, как вариант, берется среднее значения выходов по решаемой задаче у лучших;

Есть приблизительная формула, позволяющая оценить количество примеров в выборке для адекватного обучения нейронной сети поставленной задаче:

$$p = \frac{d}{\varepsilon^2}$$

где d – количество входов нейросети, а ε – требуемая точность нейросетевой модели.

Есть и более строгая формула (взято из справочной системы программы ATREE):

$$P = l * (m + 1) * (\frac{\delta}{\varepsilon})^2$$

где m – количество входных переменных, l – желаемое количество дискретных значений модели, δ – значение СКО шума в данных.

Некоторые моменты предварительной обработки изображений для сверточной нейронной сети

Данные шаги в основном для использования сверточных нейронных сетей на персональных компьютерах (в «облаках» и серверах это можно не делать).

1. Необходимо уменьшать размер изображений.
2. Для ряда задач можно преобразовать цвета в градации серого (как пример – обработка чертежей и их спецификаций).
3. Использовать хэш изображения

Пример свертки

Кратко напомним пример свертки входных данных в сверточную нейронную сеть (и это не обязательно изображение).

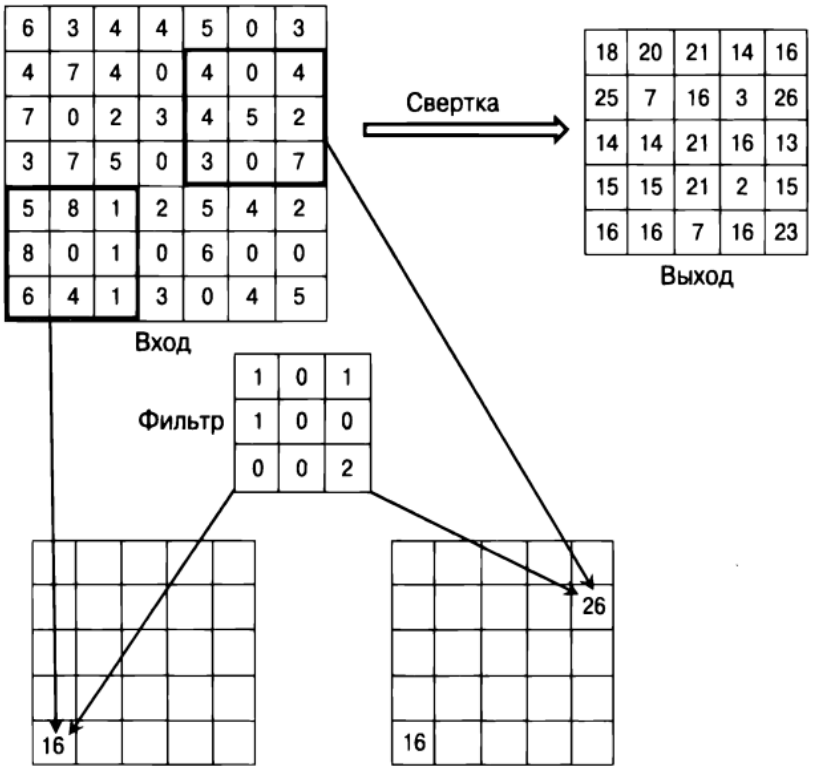


Рисунок 21. Пример сверки

Библиотеки для реализации сверточных нейронных сетей

В конце лекции рассмотрим кратко библиотеки для реализации сверточных нейронных сетей. Нет, конечно, лучше сеть написать самому – чтобы «все встало на свои места», но для промышленного применения, конечно, лучше использовать готовые библиотеки с большим функционалом.

Keras

Keras – это библиотека для Python с открытым исходным кодом, которая позволяет легко создавать нейронные сети. Она совместима с Microsoft

Cognitive Toolkit, Theano и MXNet. Tensorflow является одной из самых популярных и распространённых платформ на Python для разработки алгоритмов глубокого обучения.

TensorFlow

«TensorFlow – открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации жестов, достигая качества человеческого восприятия. Применяется как для исследований, так и для разработки собственных продуктов Google.»

Модуль **Scikit Learn**

Является одним из самых популярных библиотек машинного обучения. Эта библиотека включает в себя различные алгоритмы классификации, регрессии и кластеризации и позволяет взаимодействовать с другими библиотеками численного моделирования.

Также при построении нейронных сетей и построения графики будет полезна библиотека Matplotlib, Pandas (библиотека для работы с массивами), pyplot (библиотека визуализации) и конечно NumPy (Python библиотеки).

Anaconda

Для установки всех необходимых библиотек, которые нам понадобятся при создании сверточных нейронных сетей, необходим менеджер пакетов. Одним из лучших является Anaconda. Anaconda – дистрибутив для языка программирования Python, включающий набор популярных свободных библиотек, объединенных проблематиками науки о данных и машинного обучения. Основная цель – поставка единым согласованным комплектом наиболее востребованных соответствующим кругом пользователей тематических модулей (таких как NumPy, SciPy, Astropy и других – по состоянию на 2020 год содержит более 1,5 тысяч) с разрешением возникающих зависимостей и конфликтов, которые неизбежны при одиночной установке. А также системой управления изолированными виртуальными средами.

Основная особенность дистрибутива – оригинальный менеджер разрешения зависимостей Conda с графическим интерфейсом Anaconda Navigator, что позволяет отказаться от стандартных менеджеров пакетов (таких, как pip для Python). Дистрибутив скачивается единожды, и вся последующая конфигурация, в том числе установка дополнительных модулей, может проводиться в offline-режиме.

Данный дистрибутив имеет поддержку платформ Linux, Windows и macOS. Распространяется по лицензии BSD, существует также коммерческая версия (Anaconda Enterprise).

Использование данной платформы очень удобно при разработке искусственных нейронных сетей. Проект можно разбивать на блоки, запуская каждый блок с кодом по отдельности. При этом связь между блоками не теряется. Так же в ней при установке уже имеются в комплектации необходимые библиотеки для разработки машинного обучения.

Рассмотрим пример реализации сверточной нейронной сети на TensorFlow (смотрим слайды к лекции).

Построение нейронной сети требует настройки слоёв, а затем сборки модели с функциями оптимизации и потерь. Базовым элементом при построении нейронной сети является слой. Слой извлекает представление из данных, которые поступили ему на вход.

Пусть сеть состоит из шести слоёв (согласно исходного кода ниже):

1. Входного tf.keras.layers.Flatten – этот слой преобразует изображения размером 224x224 пикселей в 1D-массив. На этом слое у нас нет никаких параметров для обучения, так как этот слой занимается только преобразованием входных данных.
2. Скрытый слой «fc1» – полносвязный слой из 128 нейронов. Каждый нейрон принимает на вход все значения с предыдущего слоя, изменяет входные значения согласно внутренним весам и смещениям во время тренировки и возвращает единственное значение на следующий слой.
3. Под этим пунктом описано обобщение скрытых слоёв «fc2», «fc3», «fc4». Все эти слои полносвязные и выполняют те же функции, что слой «fc1». Они реализованы для того, чтобы иметь большую возможность для обучения. Единственное отличие слоя «fc4» от других слоёв – он содержит 64 нейрона.
3. Между слоями указан параметр Dropout – метод регуляризации искусственных нейронных сетей, предназначен для уменьшения переобучения сети за счет предотвращения сложных коадаптаций отдельных нейронов на тренировочных данных во время обучения.
4. Выходной слой ts.keras.layers.Dense – softmax-слой состоит из пяти нейронов, каждый из которых представляет определенный класс элемента одежды. Как и в предыдущем слое, каждый нейрон принимает на вход значения всех 64 нейронов предыдущего слоя. Веса и смещения каждого нейрона на этом слое изменяются при обучении таким образом, чтобы результирующее значение было в интервале от нуля до единицы, и представляло собой вероятность того, что изображение относится к этому классу. Сумма всех выходных значений пяти нейронов равна одному. Пример инициализации слоев приведён в листинге ниже.

Листинг – Инициализация слоёв

```
base_model = model1  # Топлесс
# Добавление верхнего слоя
```

```
x = base_model.output
x = Flatten()(x)
x = Dense(128, activation='relu', name='fc1')(x)
x = Dense(128, activation='relu', name='fc2')(x)
x = Dense(128, activation='relu', name='fc3')(x)
x = Dropout(0.5)(x)
x = Dense(64, activation='relu', name='fc4')(x)
predictions = Dense(5, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
// выполнение настроек
model.compile(optimizer=tf.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

После реализации необходимо обучить модель. За выполнение этой операции отвечает функция model.fit. Код данной функции представлен в листинге ниже.

Листинг – Функция запуска обучения нейросети

```
    model.fit(X_train_rgb,   y_train_rgb,   epochs=200,   batch_size=16,
validation_data=(X_test_rgb,     y_test_rgb),     verbose=1,     callbacks
=[early_stopping, model_checkpoint])
```

и т.д.

Литература:

1. Сверточная нейронная сеть [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Сверточная_нейронная_сеть (дата обращения 16.04.2020)
2. Аггарвал Чару Нейронные сети и глубокое обучение.: учебный курс.: Пер. с англ. – СПБ.: ООО “Диалектика”, 2020 – 752 с.
3. Рашид Тарик Создаем нейронную сеть.: Пер. с англ. – СПБ. : ООО “Альфа-книга”, 2017. – 272 с.
4. Хайкин Саймон Нейронные сети: полный курс, 2-е издание.: Пер. с. англ. – М.: Издательский дом “Вильямс”, 2019. – 1104 с.
5. Джоши Пратик. Искусственный интеллект с примерами на Python.: Пер. с англ. – СПБ.: ООО “Диалектика”.
6. Продвинутое использование библиотеки Pytorch: от подготовки данных до визуализации [Электронный ресурс] – https://habr.com/ru/post/553716/ (дата обращения: 16.11.2021)

ЛЕКЦИЯ 3-3. Сверточные нейронные сети (часть 3)

Некоторые определения

Функция потерь – алгоритм измерения того, насколько далеко находится желаемое значение от спрогнозированного.

Функция оптимизации – алгоритм «подгонки» внутренних параметров (весов и смещений) модели для минимизации функции потерь.

Метрики – используются для мониторинга процесса тренировки и тестирования.

Метрики обучения сверточной нейронной сети

Часто вводятся метрики precision – точность, recall – полнота. Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. Для этого существует параметр f1-score, являющийся F-мерой – средним гармоническим между параметрами precision и recall, имеющий представление, описанное формулой

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

где β – вес точности в метрике. И здесь, конечно, необходимо учитывать количество экземпляров изображений данного класса.

Алгоритм обучения сверточной нейронной сети

Можно сказать, что свертка - это простое применение фильтра к входным данным на сверточную нейронную сеть. Повторное применение одного и того же фильтра к входу приводит к созданию карты активаций, называемой картой функций, с указанием местоположения и силы обнаруженной функции на входе, такой как изображение. Новшество сверточных нейронных сетей заключается в способности автоматически применять большое количество фильтров параллельно, специфичных для обучающего набора данных, в условиях конкретной задачи моделирования, такой как классификация изображений. Результатом являются очень специфические особенности, которые можно обнаружить во входных изображениях. Фильтр предназначен для обнаружения особого типа во входных данных и применение этого фильтра систематически по всему входному изображению дает фильтру возможность

обнаружить особенности в изображениях. Эту возможность обычно называют трансляционной инвариантностью.

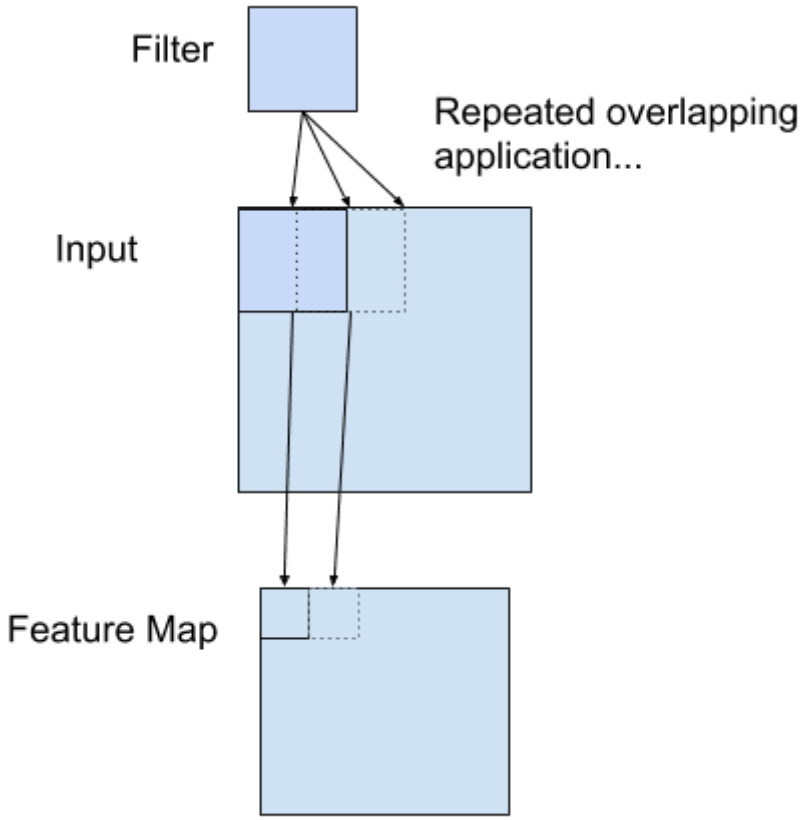


Рисунок 22. Работа фильтра (machinelearningmastery.com)

Приведем пример алгоритма обучения Adam для сверточной нейронной сети. Точнее сказать, что это – метод оптимизации сверточной нейронной сети (по сути – то же самое, имеется в виду подстройка весовых коэффициентов под входные данные). Данный алгоритм сочетает в себе идеи как среднеквадратичного распространения, так и оптимизатора импульса. Вместо того, чтобы адаптировать скорость обучения параметров на основе среднего значения, как в методе среднеквадратичное распространение, Adam также использует среднее значение вторых моментов градиентов. Данный метод оптимизации выражается формулами, представленными ниже:

$$m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^{(t)},$$

$$v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1-\beta_2)\left(\nabla_w L^{(t)}\right)^2$$

$$\widehat{m}_w = \frac{m_w^{(t+1)}}{1-\beta_1^{t+1}}$$

$$\hat{v}_w = \frac{v_w^{(t+1)}}{1 - \beta_2^{t+1}}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}$$

Расшифровка обозначений - $w^{(t)}$ – некоторые параметры, $L^{(t)}$ – функция потерь, t – индекс текущей итерации, ϵ – малая надбавка, используемая для предотвращения деления на ноль, β_1 и β_2 – коэффициенты забывания градиентов и вторых моментов градиентов соответственно. Процедуры возведения в квадрат и квадратный корень вычисляются поэлементно.

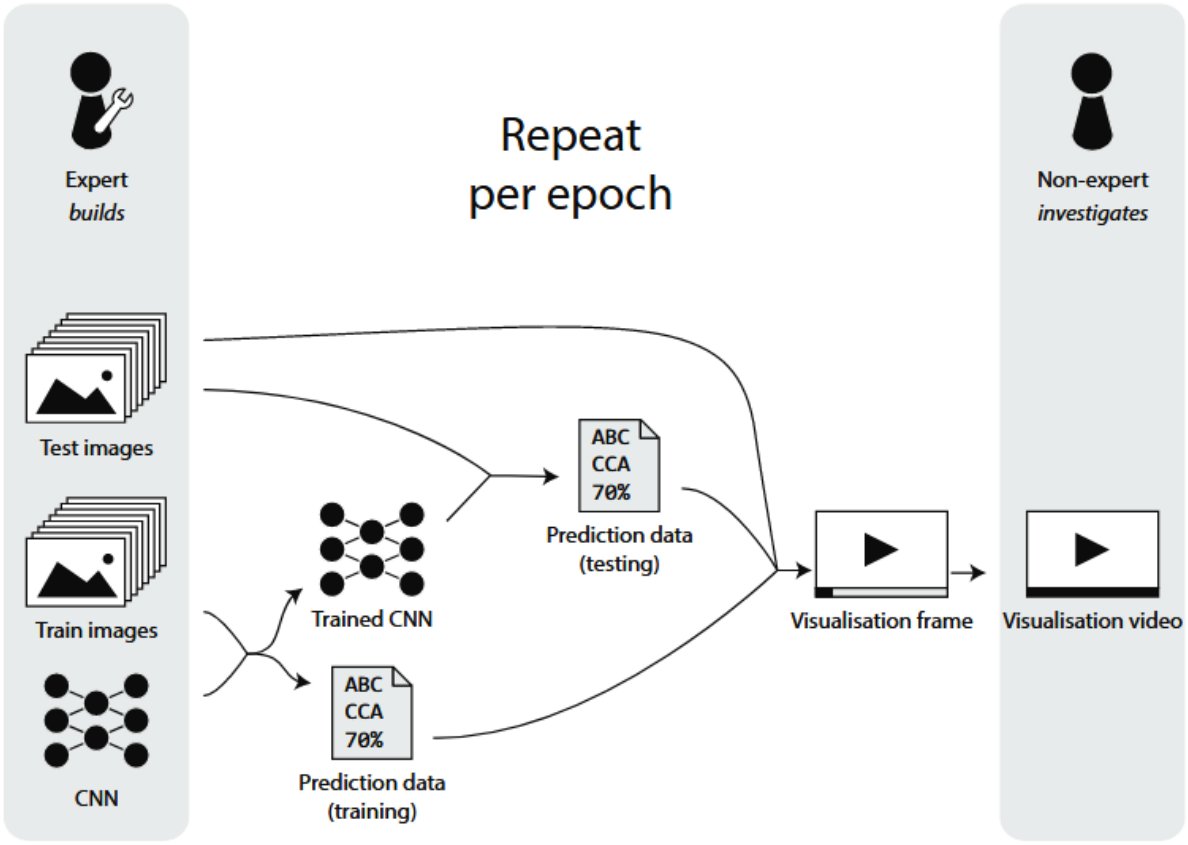


Рисунок 23. Общий процесс работы по использованию сверточной нейронной сети в обработке видеоизображений (M.Peters, L.Kempen and etc.)

Небольшой комментарий к обучению МСП в СНС

В backprop вычисляя градиент ∇_w на каждом шаге, учитываются все данные из обучающей выборки. Так, общий градиент ∇_w – сумма всех градиентов каждого элемента обучения. Такие вычисления могут проходить очень долго, поэтому для ускорения этого процесса на практике используется метод стохастического градиентного спуска показанный в формуле

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=0}^{N} \nabla_W L_i\left(x_i, y_i, W\right) + \lambda \nabla_W R(W)$$

Принцип – работаем на каждом шаге с одним примером (либо на небольшой его части). Считается, что это ускоряет процесс обучения. Опять же вспомним процедуру Робинсона – Монро…

Общий принцип построения и обучения СНС

Общий порядок операций показан на рисунке ниже

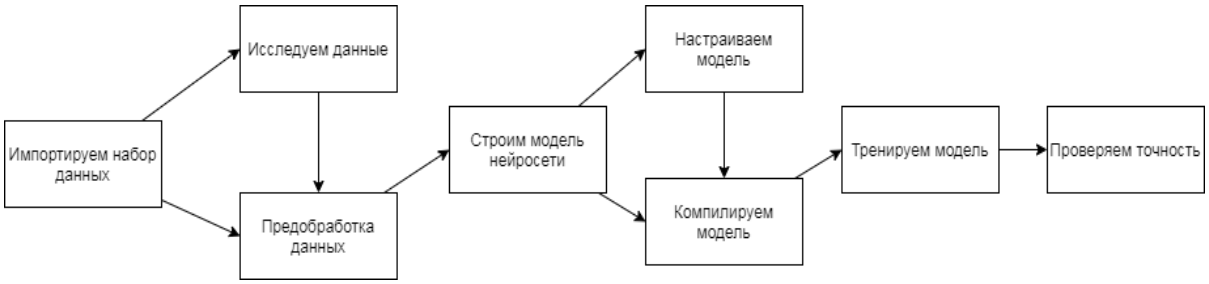


Рисунок 24. Структурная схема работы над (автор - Харций М. КубГТУ)

Оптимизация гиперпараметров СНС

Вообще говоря, под гиперпараметрами подразумевается начальная скорость обучения, коэффициент скорости обучения и регуляризация. Т.е. это то, что в 1986 году предложили в алгоритме обратного распространения ошибки (опять же ребрендинг…). Но, тут, конечно, с большими датасетами – это очень важно! При высокой скорости обучения уменьшение процента потерь в первых эпохах больше чем даже при оптимальной скорости обучения, к последним эпохам низкая скорость обучения является гораздо более эффективной чем высокая, так как спустя время алгоритм высокой скорости начинает переобучаться из-за чего замедляет и останавливает изменение функции потерь. Слишком высокая скорость обучения наоборот увеличивает свои потери, из-за чего крайне не рекомендуется при большом количестве эпох.

Сейчас используют два способа регуляризации – ограничение нормы вектора весов и т.н. «дропаут». Первое – это «старый» способ ограничения роста весов (как в плюс, так и в минус). Второй метод состоит в том, что при обучении в сети с какой-либо долей вероятности отключаются определенное количество нейронов, и обновление весов происходит только в рамках работающих нейронов. По сути – тоже «старый» способ динамического удаления нейронов в процессе обучения (но здесь массовое отключение нейронов). Данный подход можно найти у Заенцева.

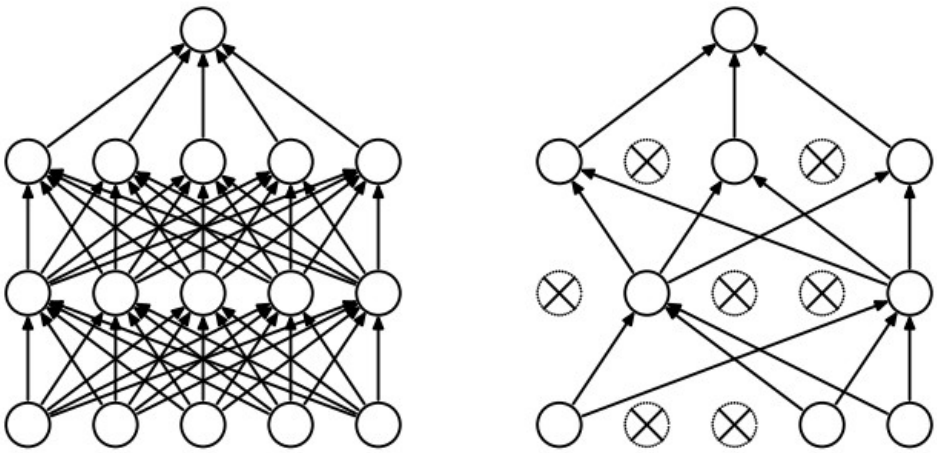


Рисунок 25. Принцип dropout [2]

Проблема переобучения нейронных сетей

Основной смысл переобучения (overfitting) заключается в том, что модель хорошо классифицирует только те данные, которые принадлежат обучающей выборке, при этом не в состоянии интерпретировать примеры, не принимавшие участие в обучающем процессе. Таким образом, теряется одно из главных характеристик – обобщение. При этом, данная проблема естественным образом распространяется и на сверточные нейронные сети (т.к. базовым решающе – обучающемся модулем которых является многослойный персептрон).

Обобщающая способность искусственной нейронной сети – это такая характеристика модели, способная отражать входные данные в требуемые результаты на всём множестве исходных данных, учитывая все сценарии. Величина обобщения можно оценить с помощью отклонения или ошибки – математического значения разности между результатом обучающей и тренировочной выборки. Иными словами, обобщением является некоторое численно выраженное соответствие, благодаря которому можно выявить взаимосвязь между входными и выходными данными. Способность к обобщению достаточно тесно связана с переобучением искусственной нейронной сети.

Проблема переобучения проявляется в значительно меньшей вероятности ошибки обобщения на тренировочных данных, в отличии от обучающих. Это возникает в результате использования моделей со очень сложной, либо со схожей структурой.

Ниже перечислены основные способы борьбы с переобучением:

1. Батч-нормализация – метод ускорения для глубокого обучения. Основная суть метода состоит в том, что не смотря на изначальную нормализацию сигнала во входе, он может исказиться после прохождения через внутренние слои по математическому ожиданию и дисперсии. Данный метод

решает эту проблему путём нормализации входных данных и приведению их математического ожидание в нулевое, а дисперсию – в единичную.

2. Метод ансамблей (или метод конгломерата нейронных сетей). У некоторых нейронных сетей возникает проблема распределения данных на более, чем два класса. К примеру, одна и та же нейронная сеть может качественно отличать единицу от нуля, но при этом не научится корректно отличить двойку от тройки, в то время, как другая нейронная сеть – наоборот. Т.е. две одинаковые по параметрам нейронные сети обучаются на одних и тех же параметрах по разному – у них будет различный набор весовых коэффициентов за счет того, что обучение происходит с использованием генератора случайных чисел. Данный метод подразумевает построение нескольких копий с разными изначальными значениями и вычисление их среднего результата на одних и тех же входных данных.

3. Ранняя остановка. Обычно обучение нейросетей начинается с малых случайных значений весов. Пока значения весов малы по сравнением с характерным масштабом нелинейной функции активации (обычно принимаемом равным единице), вся сеть представляет из себя суперпозицию линейных преобразований, т.е. является также линейным преобразованием с эффективным числом параметров равным числу входов, умноженному на число выходов. По мере возрастания весов и степень нелинейности, а вместе с ней и эффективное число параметров возрастает, пока не сравняется с общим числом весов в сети.

В методе ранней остановки обучение прекращается тогда, когда сложность сети достигает оптимального значения. Этот момент оценивается по поведению во времени ошибки валидации [1].

Проблемы безопасности нейронных сетей

Да, и нейронные сети тоже подвергаются хакерским атакам! Точнее сами системы управления на базе нейронных сетей. В частности, есть такие адверсальные атаки.

Адверсальные возмущения – это Тьюринг-подобные паттерны, которые, при подаче их на вход искусственной нейронной сети, могут существенно нарушить её работу. Понятие «паттерн» впервые было сформулировано английским математиком Аланом Тьюрингом. В основе этого термина лежит обоснование различных узоров, встречающихся в природе, таких как полосы и пятна в окраске животных.

Такие возмущения способствуют серьёзному нарушению работоспособности нейросетей. В 2018 году вышла публикация исследования, проведённая группой авторов. В этом исследовании говорится о том, как обмануть автомобиль, работающий на автопилоте. Если в обзор такого автомобиля попадёт обычная уличная реклама, отображающая логотип в виде дорожного знака, то это может нарушить его автономное управление. Также

способность ввода в заблуждение имеют естественные Тьюринг-подобные паттерны, например полосы морских обитателей.

В информационных технологиях разновидностью таких возмущений являются adversarial-атаки. В основе этого вида атаки лежит преобразование изображения, поступающего на вход нейросети таким образом, чтобы она интерпретировала его, как ошибочный класс. К примеру, если нейронная сеть умеет распознавать числа от нуля до одного и имеется некоторое изображение, похожее на число ноль, то в результате такого преобразования сеть распознает его, как число один.

Как бороться с адверсальными атаками? Есть, по сути, один реально действующий способ – при их обнаружении, добавлять новый паттерн в обучающую выборку.

Примеры применения сверточных нейронных сетей

Классические задачи сверточных нейронных сетей — такие задачи, как идентификация объекта, семантическая сегментация, распознавание лиц, распознавание частей тела человека, семантическое определение границ, выделение объектов внимания на изображении и выделение нормалей к поверхности.

Приведем еще примеры применения сверточных нейронных сетей (или глубокого обучения в целом):

- алгоритм SSPDH (Supervised Semantics – Preserving Deep Hashing) - распознавание местоположения по изображению. Часто теперь применяется в мобильных приложениях (точность, по разным данным, может достигать 70%);

- Распознавание лиц разбивается сверточной нейронной сетью на следующие основные компоненты: выявление каждого лица на картинке; сосредоточение внимания на каждом лице независимо от внешних факторов, таких как свет, ракурс, поза и т. д.; выявление уникальных особенностей; сравнение всех собранных данных с уже существующими данными в базе данных для сопоставления лица с именем (это можно сделать и другими способами;

- Сверточные нейронные сети также можно использовать для анализа документов. Это не только полезно для анализа почерка, но также имеет большое значение для систем автоматического распознавания документов, удостоверяющих личность;

- Сверточные нейронные сеи могут быть использованы для того, чтобы сыграть важную роль в борьбе с изменением климата, особенно для понимания причин, по которым мы наблюдаем такие резкие изменения климата (в России, практически везде, наблюдаются аномально теплые зимы и жара летом!), и

того, как человечество могло бы бы экспериментировать, чтобы снизить этот эффект;

- сверточные сети уже (!) существенно изменили рекламу, внедрив программную закупку и персонализированную рекламу на основе своих результатов.

Необходимо также отметить и возможность работы сверточных в области обработки аудиоданных. Вот некоторые «современные» примеры применения:

– индексирование музыкальных коллекций согласно их аудио признакам (Youtube Music);

– рекомендация музыки для радио миксов (Spotify).

– поиск сходства аудиоданных в звуке (Shazam, TrackId).

– обработка и синтез речи – генерирование искусственного голоса для диалоговых агентов (Siri, Алиса).

Литература:

1. Методы борьбы с переобучением искусственных нейронных сетей [Электронный ресурс]. URL: https://na-journal.ru/2-2019-tehnicheskie-nauki/1703-metody-borby-s-pereobucheniem-iskusstvennyh-neironnyh-setei
2. Стэндфордский курс: лекция 7. Обучение нейросетей, часть 2 [Электронный ресурс]. URL: https://www.reg.ru/blog/stehnfordskij-kurs-lekciya-7-obuchenie-nejrosetej-chast-2/

ЛЕКЦИЯ 4. Задача распознавания изображений

Общие сведения о графических файлах (изображениях)

Определение – «цифровое изображение – это последовательность кадров (или один кадр) в виде дискретного массива точек (т.н. ‘пикселей’), расположенных в памяти либо устройства ввода, либо непосредственно системы технического зрения (СТЗ)» [2].

По большому счету формат изображений можно разделить на монохромные и цветные. При этом монохромные изображения (как и черно – белые фотографии) не теряют актуальность и часто используются при решении прикладных задач, т.к. 1) они меньше занимают памяти, 2) быстрые

обрабатываются, 3) … для большого круга задач можно обойтись монохромными изображениями.

Но, если углубиться, то выделяют 4 типа изображений: монохромные, полутоновые, в естественных цветах и палитровые. Кратко разберем эти 4 типа изображений:

1) Монохромные (другое название - двухградационные) до сих пор часто используются при решении простых прикладных задач, в частности, если требуется определить только наличие какого – либо объекта в поле зрения. Это самое компактное изображение, один пиксель в нем кодируется одним битом! Лучше сказать – кодируется яркость пикселя. Но нередко используют байт.
2) Полутоновые изображения. Здесь яркость пикселя кодируется одним байтом, т.е. можно закодировать 256 оттенков (2^8 = 256 или от 0 до 255). В ряде источников [2] отмечается, что это самое распространенный тип изображений в промышленных и бытовых задачах. Как пример – система видеонаблюдения, работающая ночью.
3) Цветное изображение в естественных тонах. Здесь цвет каждого пикселя сохраняется в так называем формате RGB – тройки, т.е. требуется 3 байта, с помощью которых можно закодировать примерно 16,8 миллионов различных цветов! Это называется True Color. Этот формат, по сути, наиболее близок к тому, что воспринимают глаза человека – он естественен. Данный тип часто применяется в таких областях, как металлургия, медицина, безопасность и т.д. Но для данного формата уже требуются значительные массивы хранения.
4) Палитровое представление изображения. Можно рассматривать, как упрощенный True Color – он более компактен. Здесь палитра передается с помощью 16 или 256 RGB – троек и цвета здесь определяются косвенно. Цвет пикселя кодируется либо 4, либо 8 битами, при этом используется ссылка на цветовую палитру, а не прямое представление цвета. Это позволяет уменьшить размер изображения, но возникает проблема с подменой цветов – часто встречаются цвета, которых не было в исходном изображении. Как с этим бороться? Поступают просто – преобразовывают в полутоновый формат.

Векторное изображение.

Векторное изображение – это совокупность независимых математических объектов (контуров), при этом каждый из них можно перемещать и масштабировать, что очень удобно (Corel Draw!). В программах, обрабатывающих векторные изображения есть соглашения, как обрабатывать эти математические объекты (+атрибуты: цвет и толщина линий). Данный формат крайне компактен, но не подходит для работы с реальными

изображениями (нет, когда мы выделяем, допустим, контур лица и с ним работаем, то это как раз допустимо, а вот, когда обрабатываем пейзаж – там многое теряется в данном представлении, можно сказать, что изображение в векторном формате - нереалистично). Наиболее распространенные векторные форматы в таблице ниже.

Формат	**Разработчик**
DXF	AutoDesk
SGO	Silicon Graphics
POV	Фирма POV-Team

Растровое изображение.

Растровое изображение есть набор отдельных пикселей, записанных в единицу объема памяти в виде матрицы (т.н. «битовая карта»). При этом физический размер пикселя напрямую связан с разрешением по полю устройства получения изображения [2], т.е. число пикселей на дюйм (dot per inch – dpi).

Устройство представления разрешения	**Разрешение (обычное)**
Монитор компьютера	100 dpi
Принтер	600 dpi
Фотонаборный аппарат	3500 dpi

Главное достоинство растрового изображения – реалистичность. Вот допустим, туман с помощью векторной графики представить невозможно (… можно, но размер файла гигантский получиться), а в векторном – без труда. Но для реальных изображений (в растровом формате), конечно, требуется значительный объем информации. Поэтому, во всех уважающих себя растровых форматах предусмотрено сжатие изображения. Но, здесь, конечно, момент отметим – сжатие изображения в естественных тонах малоэффективно. Растровые изображения невозможно подвергать реальной трансформации. Наиболее распространенные, на взгляд автора, форматы растровых изображений в таблице ниже.

Формат	**Разработчик**
PCX	ZSoft Inc.
BMP	Microsoft
TIFF	Aldus Corporation

GIF	CompuServe Inc.
TGA	Treuvision Inc.

Отметим существование комбинированного формата изображения, в котором закодирована и векторная и растровая информация – это так называемые метафайлы, среди которых наибольшее распространение получили форматы CGM (фирма ISO) и WMF (фирма Microsoft).

Структура графического файла

Графический файл – это заголовок и данные.

Структура заголовка:

- Магическое число – стоит вначале файла и указывает формат, допустим, GIF.

- содержание заголовка: формат, тип (монохромный, полутоновое, в естественных цветах);

- использовано ли сжатие;

- позиция видеоданных (если есть).

Далее идет таблица цветов, которое при обработке трансформируется в RGB- тройки. Также в файлах присутствуют метки и теги, которые позволяют графическому редактору или программе, использующей изображение, получить быстрый доступ к тому или иному месту на изображении. Здесь еще есть различие в форматах – в некоторых есть теги и фиксированные поля, а в некоторых есть фиксированные поля и потоки. Подробнее см. [2].

Сжатие изображений обычно происходит с помощью «заслуженных» алгоритмов RLE (патент на изобретение от 1996 года, но корни уходят в 60-е годы прошлого столетия) и LZW (Лемпеля – Зива – Велча, датирован далеким 1984 годом).

Форматы графических изображений

Изначально каждая уважающая себя компьютерная фирма или производитель цифровой фото-видео техники, разрабатывали свой собственный графический формат. К чему это привело? Появилось и используются более сотни графических форматов! Но.. к настоящему времени в «обиходе» осталось около десятка (нет, остальные разработанные используются, но в ограниченных, специфических областях). Приведем наиболее распространенные форматы графических файлов:

- JPG – возможно самый популярный формат (есть несколько разновидностей, например, JPEG);
- GIF – популярный формат во многом благодаря возможности создания анимированной картинки;
- BMP – «красочный формат», но занимает много места и др.

Вопрос обработки изображения при зашумлении

Часто на полученных изображениях или видеопотоке заметно какое-либо искажение, из-за которого может теряться качество картинки. Искажения может вызывать зашумление. Искажение зависит от многих параметров и иметь достаточно сложный характер. Зашумления условно можно разделить на естественные и искусственные. К естественным можно отнести такие критерии как:

освещённость;
затуманенность;
природные осадки, дождь, снег, туман, морось;
блики, тени.

К искусственным обычно относят такие критерии как:

характеристики оптических систем (расфокусировка, замутнённость зеркал и линз, дисторсия);
характеристики устройств оцифровки;
характеристики электронной регистрирующей аппаратуры;
характеристики каналов передачи данных.

Пример изображения, искажённого зашумлением показан на рисунке ниже.

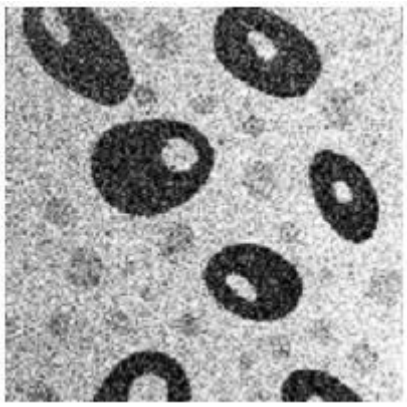


Рисунок – Изображение, искажённое зашумлением

При всём этом процесс формирования изображения зависит от многих достаточно сложных нелинейных уравнений, различных математических моделей. К примеру - только параметры яркости имеют зависимость от множества геометрических формул и преобразований. Исходя из этого следует,

что к таким изображениям необходимо применять фильтрацию для более качественного распознавания.

Фильтрация изображения – это процедура обработки некоторого растрового изображения для улучшения его качества. Иначе задачу можно назвать фильтрацией помех. Подразумевается, что изначально существовало оригинальное изображение, к которому было применён какой-либо вид искажения, иными словами зашумления.

Существует достаточно большое множество различных фильтров. В зависимости от типа шума применяется какой-либо тип фильтра. Можно выделить четыре самых распространённых из них:

сглаживающие фильтры;
фильтры Винера;
медианные фильтры;
ранжирующие фильтры.

Также существуют линейные и нелинейные фильтры. Так как одним из самых распространённых шумов является белый шум Гаусса, соответственно для шумов такого вида часто применяются линейные фильтры. Среди линейных фильтров можно выделить следующие:

линейная оконная фильтрация;
скользящее среднее в окне;
фильтрация Гаусса.

«Линейные операции состоят из умножения каждого пикселя окрестности на соответствующий коэффициент и суммирование этих произведений для получения результирующего отклика процесса в каждой точке (x, y). Если окрестность имеет размерность $m \times n$, то потребуется $m * n$ коэффициентов. Эти коэффициенты сгруппированы в виде матрицы, которая называется фильтром, маской, фильтрующей маской, ядром, шаблоном или окном» [1].

Под процессом линейной пространственной фильтрации понимается смещение центра фильтрующей маски от точки к точке изображения. В каждой точке (x, y) откликом фильтра является сумма произведений коэффициентов фильтра и соответствующих пикселей окрестности, которые накрываются фильтрующей маской. Оптимальным размером маски считается размер 3×3.

В ходе исследований было принято решение использовать именно фильтрацию Гаусса, так как она эффективно борется с различного рода белыми шумами, в том числе и зашумлением Гаусса.

Фильтрация Гаусса

Для того, чтобы устранить помехи, появившиеся на изображении, как было описано в предыдущем подразделе, выбор пал на фильтрацию Гаусса.

Фильтрация Гаусса – вид электронного фильтра, применяемого для шумоподавления в изображениях или видеопотоках. Данный фильтр, как и многие другие электронные фильтры, имеет импульсную переходную функцию. В данном случае этой функцией является функция Гаусса, описываемая формулой.

$$\mathrm{g}(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

где a, b, c – произвольные вещественные числа. Данная функция введена Гауссом еще в 1809 году, как функция плотности нормального распределения. В этом случае параметры выражаются через среднеквадратичное отклонение σ и математическое ожидание μ. Выражения вещественных чисел представлены в формуле:

$$a = \frac{1}{\sigma\sqrt{2\pi}}\,, b = \mu\,, c = \sigma$$

Ниже представлен график функции Гаусса, изображённый на рисунке

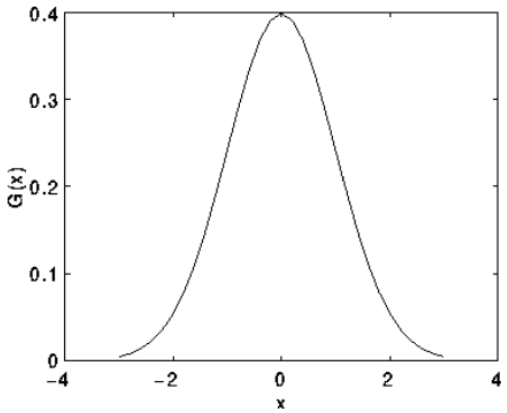


Рисунок 26. Функция Гаусса

Данный фильтр использует Гауссово распределение для вычисления преобразований и применяет его к изображению попиксельно. Как правило, шум распределяется равномерно по всему изображению, при этом не зависит от пикселей. Главное условие – математическое ожидание значения шума должно быть равным нулю. При этом важно учесть, что слишком большое окно фильтрации будет уменьшать интенсивность шума, но побочным эффектом этого фильтра будет являться размытие некоторых деталей на картинке. Пример фильтрации методом Гаусса изображён на рисунке ниже.

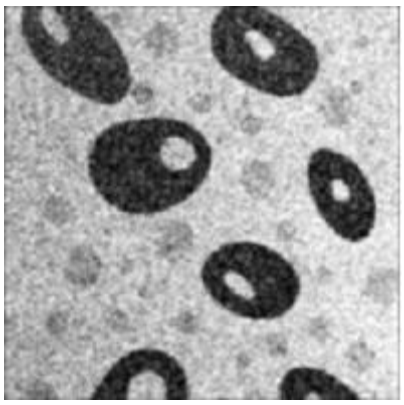


Рисунок 27. Фильтрация изображения фильтром Гаусса

Процесс бинаризации изображения

Нередко в прикладных задачах по обработке изображений необходимо провести процесс бинаризации.

«Бинаризация – это один из простых методов препарирования изображения. В общем понимании, после необходимой обработки изображение бесцветным чёрно-белым, у которого убраны лишние детали. Самый важный параметр данного преобразования – это порог. Порогом является критерий проверки интенсивности точки изображения» [1]. Общую схему бинаризации можно посмотреть на слайдах к презентации.

Самыми распространёнными методами бинаризации изображения являются:

глобальный фиксированный порог;
локальный адаптивный порог;
метод Оцу.

«В отличии от методов глобального и адаптивного порогов, значение порога которых задаётся вручную, в методе Оцу значение порога вычисляется автоматически». Благодаря этой особенности и был выбран данный метод.

Метод Оцу использует гистограмму распределения значений яркости пикселей растрового изображения. Математическое описание гистограммы представлено в нижеприведенных формулах.

$$\omega_0(k) = \sum_{i=1}^{k} p_i \,,$$

$$\omega_1(k) = \sum_{i=k+1}^{L} p_i = 1 - \omega_0(k) \,,$$

$$\mu_0(k) = \sum_{i=1}^{k} \frac{ip_i}{\omega_0} \,,$$

$$\mu_1(k) = \sum_{i=k+1}^{L} \frac{ip_i}{\omega_1}$$

Далее строится гистограмма по значениям $p_i = n_i/N$, где N – это общее количество пикселей на изображении, n_i – количество пикселей с уровнем яркости i. Диапазон яркостей делится на два класса с помощью порогового значения уровня яркости k, где k – целое значение от нуля до L. Каждому классу соответствуют относительные частоты $\omega_0\omega_1$. Среднее арифметическое классов выражается через μ_0 и μ_1.

После этого происходит вычисление максимального значения оценки качества разделения изображения на две части. Вычисление в математическом виде представлено в формуле:

$$\eta(k) = \max_{k=1}\left(\frac{\sigma^2_{\text{кл}}(k)}{\sigma^2_{\text{общ}}}\right)$$

где $\sigma^2_{\text{кл}} = \omega_0\omega_1(\mu_1 - \mu_0)^2$ – межклассовая дисперсия, а $\sigma^2_{\text{общ}}$ – это общая дисперсия для всего изображения целиком.

Обработка видеопотока

Есть несколько подходов к формированию видеопотоков, точнее сказать, как их организовать**.** Допустим старый проверенный способ – с помощью фреймов, где под фреймом понимается последовательность изображений, либо в векторном, либо в растровом формате. Кстати, так до сих пор работают многие охранные видеосистемы – записывается фотографии с камеры, созданные через определенное количество миллисекунд (или реже).

Некоторые подходы к обработке изображений

Выделим следующие подходы:

- Bayesian – belief propagation (2003 год) // первое применение можно описать так – человек может узнать другого (знакомого) человека, наблюдая только половину лица и данный алгоритм восстанавливает изображение по некоторому фрагменту;

- классификация с помощью стандартного байессовского алгоритма с использованием байесовской нейронной сети, в которой значение синаптической связи интерпретируется, как вероятность. С помощью данного алгоритма можно достаточно точно рассчитать вероятность, используя как ранее известную информацию, так и данные новых наблюдений (т.е. пополнение датасета). Но данный подход требует большого количества вычислений;

- алгоритм Виолы – Джонса. Датирован 2001 годом, считается «прорывным» в области распознавания (детектирования лиц). Используется технология скользящего окна. Использует примитивы Хоара.

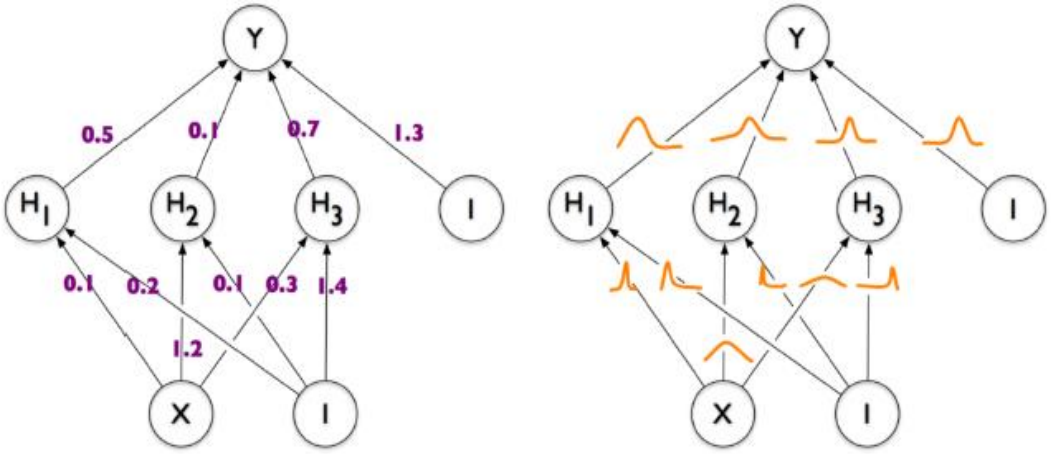


Рисунок 28. Схема байесовской нейронной сети

Распознавание изображений с помощью искусственных нейронных сетей.

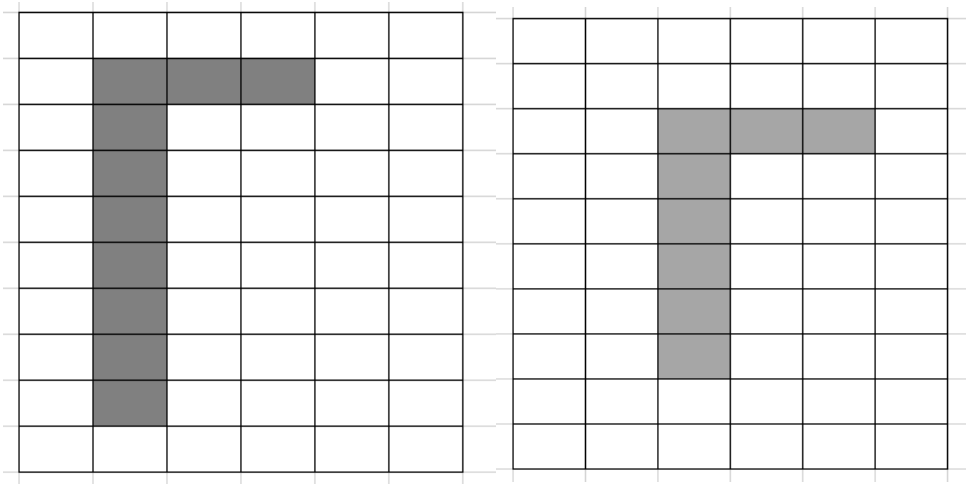
Рассмотрим классический пример, как использовать нейронную сеть типа многослойный персептрон для распознавания изображений. По большому счету – это подход Ф. Розенблатта конца 50-х годов прошлого столетия.

Пусть у нас имеется фотоматрица, на которой фиксируются буквы русского алфавита (или цифры – не важно) в монохромном формате (см. лекция 4-1). В данном случае матрица 6 х 10. На изображениях ниже представлена буква «Г» - она может быть смещена, увеличена, уменьшена, могут быть некоторые искажения или шум. Человек без труда в большинстве случаев распознает букву (ну если она не совсем затерта, или не «почерк врача»). Да, здесь речь идет о машинописном тексте, об рукописном речь будет идти дальше. А вот распознать букву для технических устройств, точнее сказать, для алгоритмов распознавания изображений – задача посложнее. Почему? Ответ такой – человек (ребенком) тоже не сразу начинает понимать, где какая буква написана! Он долго учится! Очень долго! Если сравнивать с методами автоматического распознавания текста. В принципе, взрослому человеку, допустим россиянину, дать выучить арабский алфавит за пару дней и потом спрашивать, где какая буква, не говоря уже о словах – думаю, это вызовет крайнее затруднение. При этом, арабу дать русский алфавит и потом спрашивать – результат, думаю, также не будет отличаться. Так вот вернемся к фотоматрице. Как ее обработать с помощью нейронной сети? Понятно, что

входами на нейронную сеть будут именно эти ячейки фотоматрицы и их (входов) будет:

$$6*10 = 60$$

Т.е. у нейронной сети типа многослойный персептрон будет 60 входов! Много это или мало? Для старых компьютеров, как у Ф. Розенблатта – это конечно много. Даже для «Intel Pentium – II», по опыту и памяти автора, тоже много. А вот для современных «Intel Core Coffee Lake i7» с 6-ю ядрами, тактовой частотой в 4 ГГц и трехуровневым кэшем - это уже очень немного. Далее – вопрос «Сколько выходов необходимо сделать в нейронной сети при распознавании букв русского алфавита?». Ответ очевиден – 33 буквы, значит 33 выхода. Далее, немного поразмыслив – буквы «й», «ъ», «ь», «ё», «ы» можно удалить, т.к. они в реальных, практических задачах практически не используются (например, их нет в автомобильных номерах). При этом – буквы «ё» и «й», если необходимо, можно заменить на «е» и «и», без особых потерь (ну в ряде случаев, может, конечно, потеряться суть слова или предложения, но это редко).



Далее, возникают некоторые трудности, в частности, как закодировать выходы нейросети, чтобы они отвечали каждый за свою букву? Можно, конечно, пойти «напрямую» - первый выход за букву «А», второй за «Б» и т.д., но по опыту автора – это не самый лучший вариант. Т.е. делаем разметку входных образцов:

Входы «образец буквы» - «номер выхода по порядку равен 1, остальные 0», «0,0,0,1,1,0,0,0,0,0,1,0,0,1,…» => »1,0,0,0,0,0….» и тренируем нейронную сеть, чтобы она правильно «зажигала» выходы на подаваемую букву. При этом в выходном слое, чтобы на выходах четко были 0 и 1 необходимо использовать в нейронах выходного слоя линейные функции активации, а это требует некоторой, возможно не совсем корректной, модернизации стандартного алгоритма обратного распространения ошибки!

Можно пойти несколько иначе – использовать в выходах нелинейные функции активации и получать не только 0 и 1, а и промежуточные значения,

далее ставить блок интерпретатора (но программно – это просто цикл с перебором), который бы выбирал нейрон с наибольшим значением выхода. Т.е. здесь трактовка в вероятностном смысле.

Можно обойти многие проблемы, если использовать обучение без учителя (допустим по правилу Хебба), т.е. сеть сама будет распределять выходы по входам.

Можно использовать сеть Кохонена, опять же обучающуюся без учителя, но она потребует больше время на обучение и возможно будет работать с низкой точностью.

Если с буквами и цифрами задача, в принципе, не такая сложная, то с распознаванием более трудных изображений, например, лица человека, задача становится более трудной.

Немного из истории теории распознавания изображений

Хотя распознаванием изображений, а точнее символов, занимался еще Френк Розенблатт в конце 50-х годов, считается, что история компьютерного зрения началась в начале 60-х годов, когда ученый Ларри Робертс опубликовал набор работ «Block World» (рисунок ниже) в которых говорилось о восстановлении трехмерной геометрии, состоящей из простых сцен.

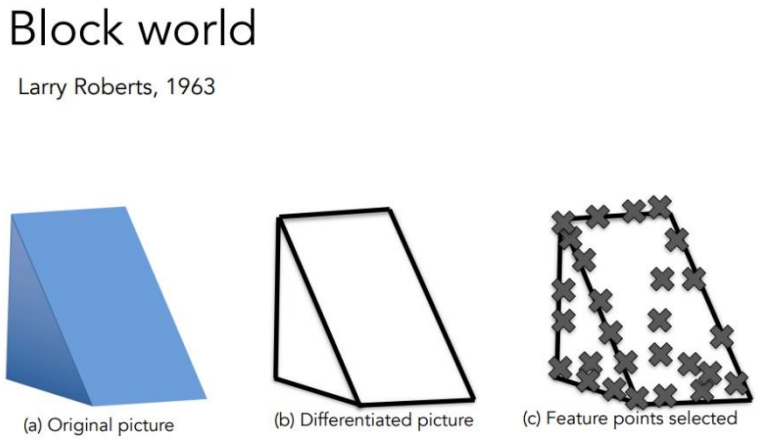


Рисунок 29. Пример фигурок из Block World

Затем вышла книга «Vision: A computational investigation into the human representation and processing of visual information» Дэвида Мара (MTI, США), в которой описаны процессы мыслительной обработки визуальной информации. Здесь же отметим труд Марвина Мински про фреймы (это, правда, про экспертные системы), но они также стали «кирпичиком» в фундаменте решения задач компьютерного зрения.

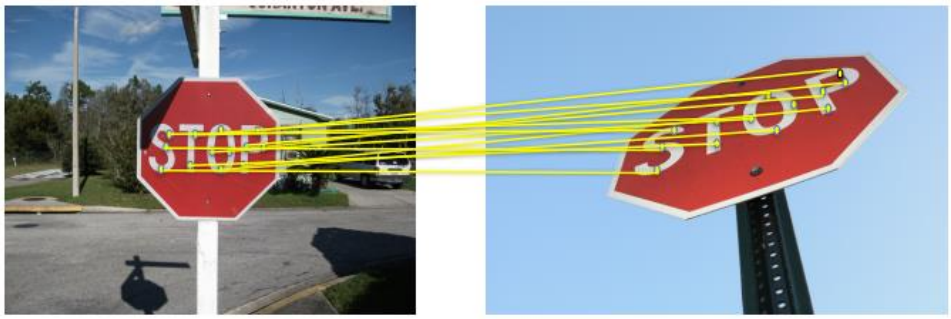
В начале 90-х годов стали сегментировать изображения – «задача распознавания объектов свелась к группировке пикселей со схожими свойствами для определения типа объекта, к которому они относятся» [4].

Главными достижениями этого века в области распознавания изображений стали практические работы:

- Пауля Виолы и Майка Джонса по распознаванию лиц;

- алгоритм SIFT Д.Лоу (*масштабно инвариативная трансформация признаков*). Данный алгоритм позволяет находить локальные признаки изображения, которые не зависят от различных изменений (угла съемки, масштаба, освещения) и сопоставлять их с похожими объектами$

- студенты Стэндфордского университета в 2009 году представили проект ImageNet – огромнейший, на тот момент времени, датасет, состоящий из более чем 22-х тысяч классов изображений и 14 миллионов фотографий!



"SIFT" & Object Recognition, David Lowe, 1999

Рисунок 30. Работа алгоритма SIFT

Основные подходы к распознаванию изображений

«Распознаванием называется процесс, при котором на основании набора признаков некоторого изображения объекта определяется его принадлежность к определенному классу» [2]. Из определения понятно, что изображение «в практическом применении» сегментируется на зоны и на объекты (образы). При этом должна быть какая - то база объектов или предметов, которые мы хотим распознать. По большому счету необходима база знаний или даже экспертная система.

Вообще говоря, теория распознавания образов – это отдельный раздел науки (сейчас это называется машинным обучением), который активно развивался в 70-х годах XX века – труды Вапника, Червоненкиса, Фукунаги.

Все методы распознавания изображений можно разделить на две категории:

- теоретические – используют принципы теории принятия решений;

- структурные – возникли из теории формальных языков и основаны на модели Хомского;

Кратко про теоретический подход – пусть есть группа из M классов некоторых объектов $\{m_1, m_2, \dots m_M\}$. У этих объектов есть n – мерное пространство признаков и у каждого объекта есть свой вектор признаков (обычно это геометрические признаки объектов, раз мы рассматриваем распознавание изображений). Распознавание образа тогда есть процедура отнесения рассматриваемого объекта к одному из M классов (с помощью нейронных сетей это проще сделать с помощью сети Кохонена). С точки зрения теории принятия решений необходимо найти M дискриминирующих функций [2]. И таким образом, рассматриваемый объект относится (распознается) к одному из i-классов.

Т.к. часто определить реальное значение признаков объекта невозможно (ошибка измерения), то ищут вероятность принадлежности к тому или иному классу.

Свои системы распознавания лиц методами глубокого обучения имеют ИТ – гиганты: Microsoft, Facebook, Google.

Продукт фирмы Sighthound Inc. может распознавать даже эмоции на лице человека.

Задача распознавания рукописного текста

Распознавание рукописного текста является более сложной задачей, чем распознавание машинописного текста. Сложность построения такой модели заключается в корректном распознавание написанных слов. Трудности могут возникнуть из-за почерка автора текста, либо же не четко прописанное чернилом слове. При этом различают рукописное написание онлайновое и оффлановое. Первое – имеется в виду, что текст написан на экране (здесь лучше сказать – на устройстве ввода) электронного устройства – стилусом или пальцем. В данном случае системе сразу понятно, где пробел, где символ. Второй – имеется в виду, что написано от руки на бумаге.

Считается, что первым КПК, который смог распознать рукописный текст – является Apple Newton в 1993 году.

В случае написания на бумаге, кроме «борьбы» с самим почерком, появляются проблемы с дефектами бумаги, пятнами и прочим «шумом».

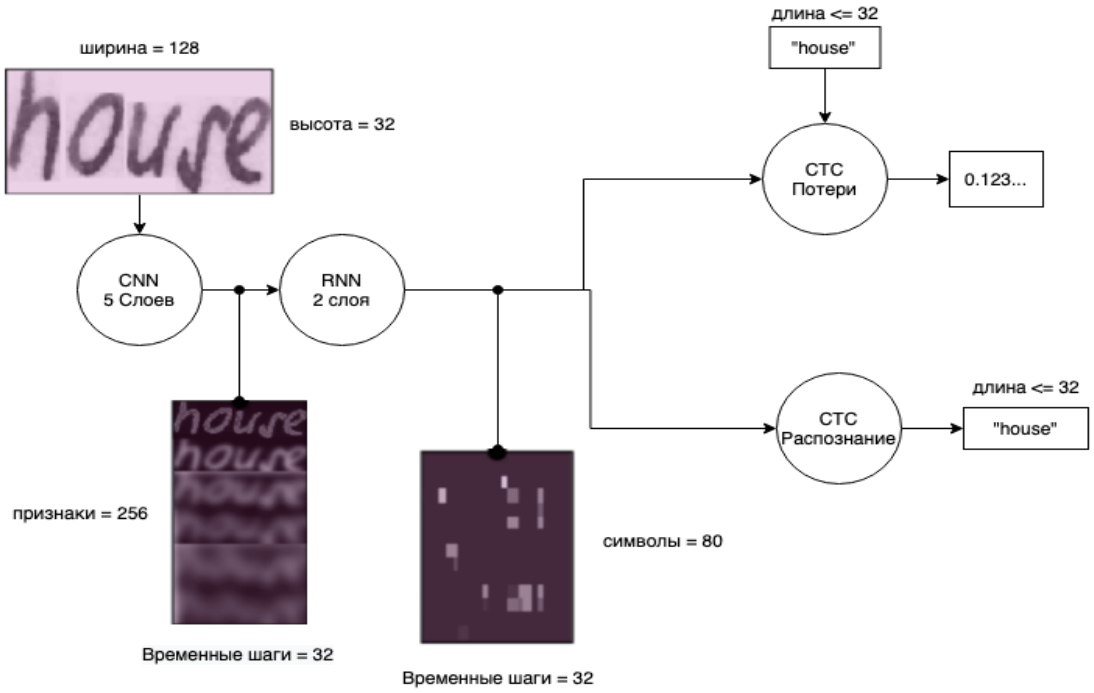


Рисунок 31. Пример системы распознавания рукописного текста

Алгоритм распознавания рукописного текста состоит из нескольких этапов:

1) Подготовка: выпрямление (снижение угла наклона) + бинаризация + удаление шумов + сегментация строк и символов;
2)
3) пост – обработка, в т.ч. «ручным» способом.

Да, кстати, сегментация часто проводится с помощью метода опорных векторов.

При распознавании рукописного текста очень помогает языковая модель с поиском по префиксному дереву.

В последнее время часто применяют для данной задачи сверточные нейронные сети в следующей топологии:

- стандартные слои свертки и пулинга;

- рекуррентная нейронная сеть одного из типов: LSTM, mdlstm, IDCN,

Современная точность распознавания рукописных текстов на представительных выборках обычно не превышает 75%.

Работа с видеопотоком

Для работы с свёрточными нейронными сетями видеоданные сначала необходимо преобразовать в векторный набор числовых значений. Преобразовать всё видео целиком – непростая задача, требующая больших

затрат времени и ресурсов, поэтому для работы с нейронными сетями можно использовать лишь отдельные кадры (фреймы) изображения. Чем меньше интервал между отбираемыми фреймами, тем больше деталей будет доступно нейронной сети и тем больше ресурсов будет затрачено.

В конечном счёте нейронная сеть будет решать проблему классификации с определённым набором изображений, полученных из видео. Каждое изображение может быть отнесено к тому или иному классу, характеризующему увиденное, например: море, лес, город, горы, марки автомобилей, предметы мебели и т.д. Полученные таким образом данные можно использовать для более точного поиска того или иного видео на различных веб-ресурсах.

Проверка корректности документов по изображениям

Еще одна практически значимая задача – т.к. в настоящее время очень многие перешли на удаленный формат работы, а также учитывая все возрастающую роль интернет – сервисов наподобие «Госуслуг». Кратко разберем пример.

Ниже представлена таблица, на которой указаны метки и классы электронных документов. Каждому входному изображению соответствует одна из перечисленных меток.

Таблица – Метки и классы изображений

Метка	Класс
0	Паспорт
1	Водительское удостоверение
2	ПТС
3	Тех. паспорт
4	Страховой полис

В качестве входного значения нейронной сети служит одномерный массив длиной 784. Массив именно такой длины по той причине, что каждое изображение сжимается и адаптируется под стандарт входных значений. В результате преобразований электронный документ представляет собой изображение размером 100×100 пикселей (10000 пикселей всего в изображении), которое мы преобразуем в одномерный массив. Процесс преобразования 2D-изображения в вектор называется сглаживанием (flattening) и реализуется посредством сглаживающего слоя – flatten-слоя.

Применение нейросети в данной задаче схематично показано на рисунке ниже.

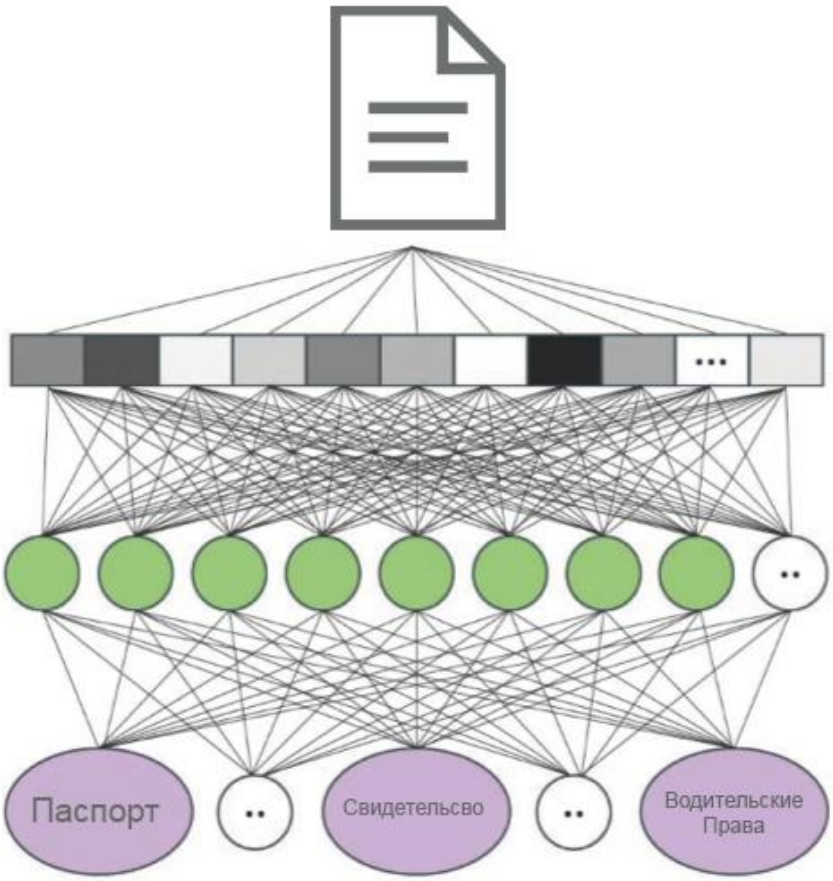


Рисунок 32. Применение нейронной сети для идентификации документа

Рисунок 33. Результат предсказания модели

Литература:

1. Исрафилов Х.С. Исследование методов бинаризации изображений // Вестник науки и образования. – 2017. – №6. – С.44-51.
2. Воротников С.А. Информационные устройства робототехнических систем. М.: МГТУ им. Н.Э.Баумана. 2005.
3. Прохоренок Н. А. OpenCV и Java. Обработка изображений и компьютерное зрение. — СПб.: БХВ-Петербург, 2018. — 320 с.
4. Стэндфордский курс: лекция 1. Введение [Электронный ресурс]. URL: https://www.reg.ru/blog/stenfordskij-kurs-lekciya-1-vvedenie/ (дата обращения 10.12.2021)

ЛЕКЦИЯ 5-1. Задача автоматической обработки текстов: задача обработки естественных языков, системы автоматической обработки текстов

Задача обработки естественных языков

Проблемой извлечения важных данных из неструктурированного текста занимается особое направление искусственного интеллекта: обработка естественного языка, или NLP (*англ. Natural Language Processing*), которая лежит на стыке нескольких направлений науки – лингвистика, математика и искусственный интеллект. Компьютерная лингвистика тесно связана с искусственным интеллектом, который, на взгляд автора, должен в итоге выполнять следующие функции на уровне человека – зрение и распознавание образов, понимание и генерация речи. И вторая функция, как раз и реализуется с помощью компьютерной лингвистики (если брать в целом).

Выделим основные области лингвистики:

1) Фонология;
2) Морфология;
3) Синтаксис
4) Семантика и прагматика
5) Лексикография

Понимание естественного языка иногда считают AI-полной задачей, потому как распознавание живого языка требует огромных знаний системы об окружающем мире и возможности с ним взаимодействовать (*Википедия*).

NLP – подраздел информатики и искусственного интеллекта, посвященный тому, как компьютеры анализируют естественные языки. NLP позволяет применять алгоритмы машинного обучения для текста и речи.

NLP изучает проблемы компьютерного анализа и синтеза естественных языков. Применительно к искусственному интеллекту анализ означает понимание языка, а синтез — генерацию грамотного текста.

Распространенные задачи обработки естественного языка:

- распознавание речи, перевод текста в текстовом документе;
- синтез речи;
- информационный поиск: извлечение информации из текстов, путем выявления сущности некоторых типов и установления отношения между ними;
- классификация текстовых данных;
- кластеризация текстовых данных;
- анализ социальных сетей: извлечение мнений пользователей социальных сетей о заданных объектах;
- создание вопросно-ответных систем, диалоговых систем и задач машинного перевода и др..

Решение этих задач будет означать создание более удобной формы взаимодействия компьютера и человека.

Есть также раздел в искусственном интеллекте – **компьютерная лингвистика**, в рамках которого также занимаются обработкой текстовых документов. При этом, обработку естественного языка, нередко включают, как раздел компьютерной лингвистики. Кстати, компьютерная лингвистика, практически одновременно зародилась и в СССР и в США. В СССР это датировано 1958 годов – вышел сборник «Машинный перевод и прикладная лингвистика» (интересовал перевод между языками народов СССР). В СССР прикладной лингвистикой занимались такие знаменитые математики, как Соболев С.Л. и Канторович Л.В. В США основоположником считается лингвист Н. Хомский, который работал над формализацией структур естественных языков.

Но официально днем рождения компьютерной лингвистики считается Джоржтаунский эксперимент, датированный январем 1954 году, в котором было машинно переведено 60 простых текстов с русского на английский (русский язык тогда очень ценился!).

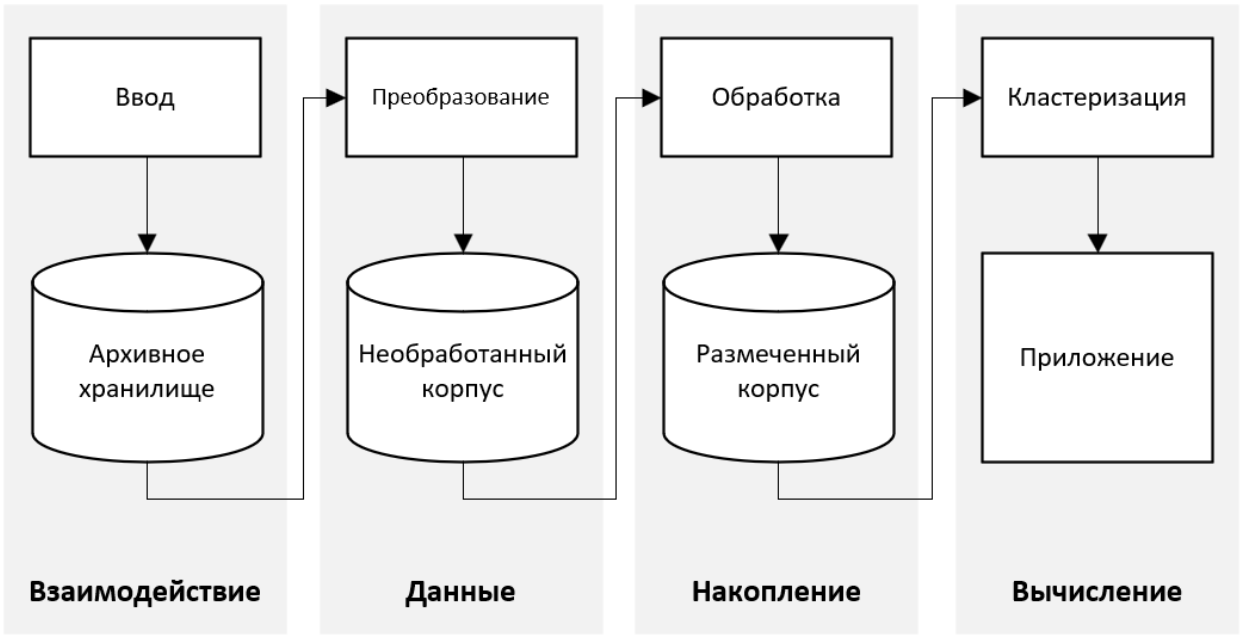


Рисунок 34. Общий принцип работы лингвистической системы

Выделим некоторые проблемы NLP:

- структурная или синтаксическая неоднозначность;

- смысловая неоднозначность (стандартный пример – «за’мок» и «замо’к»);

- падежная неоднозначность;

- референциальная неоднозначность;

- литерация.

С каждой из этих проблем можно бороться различными способами.

Некоторые термины:

Конвейер (pipeline) — это метод объединения последовательности преобразований. Смысл этого подхода в том, чтобы разбить проблему на части и решать их отдельно.

Корпус (corpus/corpora) — это коллекция родственных документов или высказываний на естественном языке. Выделяют даже направление компьютерной лингвистики – **корпусная лингвистика**.

Мешок слов (или Bag of Words) — это модель текстов на натуральном языке, в которой каждый документ или текст выглядит как неупорядоченный набор слов без сведений о связях между ними. Его можно представить в виде матрицы, каждая строка в которой соответствует отдельному документу или тексту, а каждый столбец — определенному слову. Ячейка на пересечении строки и столбца содержит количество вхождений слова в соответствующий документ.

Стемминг (stemming) — это процесс нахождения основы слова (неизменяемой части слова, которая выражает его лексическое значение) для заданного исходного слова.

Стеммер — программная реализация алгоритм *стемминга.*

Стоп-слова (stopwords) – слова в тексте, которые не несут смысловой нагрузки.

Токенизация (tokenization) – это разбиение текста на более мелкие части (токены).

Частотность (frequency) — термин, предназначенный для определения наиболее употребительных слов. Это отношение числа вхождений слова к общему числу слов документа.

Центроид — точка, которая является центром кластера. Каждый центроид является вектором, элементы которого представляют собой средние значения соответствующих признаков, вычисленные по всем записям кластера.

Классически модуль лингвистической модуль обработки текста действует по цепочке:

- определение языка (если это многоязычная лингвистическая система);

- отфильтровывание нечитаемых символов;

- нормализация текста (разделение введенного текста на слова и остальные последовательности символов);

- исправление орфографических и синтаксических ошибок (т.н. спелчекеры);

- удаление предлогов;

- лексический анализ (анализ слов);

- синтаксический анализ (анализ порядка слов в предложении с учётом правил грамматики);

- семантический анализ (анализ значения предложения самого по себе и в его связи с другими предложениями).

Небольшой пример семантического анализа для русского языка (на базе продукционных правил):

Правило 1: ЕСЛИ определение стоит на первом месте и за ним идет существительное, ТО существительное является подлежащим.

Правило 2: ЕСЛИ за подлежащим идет глагол, ТО этот глагол является сказуемым и поясняет, что делает подлежащее.

Правило 3: ЕСЛИ за подлежащим идет сказуемое, а за ним следует существительное, ТО это существительное является дополнением.

Правило 4: ЕСЛИ предложение имеет следующий порядок слов: подлежащее, глагол, дополнение, ТО вся фраза говорит о том, что подлежащее делает (действие, выраженное сказуемым) по отношению к дополнению.

Современный подход к решению задачи с помощью NLP в общем случае можно разделить на следующие этапы:

- нахождение источников данных;

- сбор данных;

- «очистка» данных;

- выбор представления данных (например, «мешок слов»);

- классификация;

- инспектирование и интерпретация модели;

- применение семантики.

Проблема очистки данных обычно содержит следующие шаги:

- удаление нерелевантных символов (но до этого необходимо составить словарь нерелевантных символов);

- токенизация текста;

- удаление нерелевантных слов;

- приведение всех символов в нижний регистр;

- совмещение слов, написанных с ошибками или имеющих альтернативное описание (например, «Апшеронск / Апширонск» или «хорошо / круто»);

- проведение лемматизации, если это необходимо.

Современные примеры применения NLP:

- идентификация различных категорий пользователей или клиентов для прогнозирования их оттока, прибыли от клиентов, продуктовых предпочтений и т.д.;

- детектирование отзывов пользователей о том или ином товаре, услуги или событии (при этом могут быть «вложенные задачи» - например, упоминание в отзывах размера одежды или ее цвета);

- классификация текста в соответствии с его смыслом по таким разделам, как запрос элементарной помощи или подсказки, решение определенной проблемы.

Далее рассмотрим сугубо прикладной пример работы с текстом – определение значимости того или иного события политического или экономического плана (более подробно см. [2, 3]).

События, происходящие в мире, сразу же опубликовываются в ресурсах сети Интернет и задача состоит в том, что бы за заданное время обнаружить новое сообщение, автоматически рейтингировать его и оценить возможные последствия. С другой стороны необходимо на исторических данных исследовать корреляцию событий и их влияние на экономико – политическое поле. При этом данная задача очень сложна, так как данные в ней неструктурированны и их огромное количество. Но нас в данном случае интересует только работа с распознаванием к какой предметной области принадлежит вышедшее событие.

Для решения данных задач необходимо создать информационный ресурс, который бы искал события и сохранял их. Также, для классификации необходима мера, по которой определялась бы «сила» события. В качестве меры можно использовать финансовые временные ряды, как валютного, так и фондового рынках, а также ряды макроэкономических показателей.

В качестве поставщика событий для таких «настольных» систем анализа целесообразно использовать RSS – каналы, а не разрабатывать обработчик поисковых запросов или робота поиска информации в сети. Информационный ресурс в минимальном составе состоит из БД, скрипта загрузки данных и скриптов (макросов, в зависимости от выбранного типа СУБД) обработки данных.

Табличный состав БД следующий (перечислены главные таблицы и основные поля таблиц):

- Страны (Имя, Регион, Население, Площадь, ВВП, первые лица государства и т. д.);
- Корпорации (Название, Страна, выручка, доход, численность рабочих и т.д.);
- События (Страна1, Страна2, Корпорация, Персона, текст, численные показатели,);
- Контракты;
- Макроэкономические показатели и индексы;
- Вспомогательные таблицы.

Использование RSS – каналов выгодно тем, что при использовании адекватного канала можно получить «полный» срез новостей в режиме реального времени по определенной тематике. Также RSS – канал можно подключить к Outlook Express и воспользоваться простым языком VBA для предварительной обработки данных и закачивании их в БД. Но здесь проблема в нахождении адекватных RSS – каналов и в дублировании одного события несколькими каналами. В частности, можно посоветовать использовать канал с

сайта http://www.bfm.ru по российским политическим и экономическим событиям. Приведем простой вариант анализа события (новости) при определении к чему оно относится. Пословно читается заголовок RSS – сообщения (слова длинной меньше 3 символов не учитываются, также не учитываются запрещенные слова длинной больше 2 (предлоги и др.)); затем последовательно делается запрос к разным таблицам в БД, к полям с ключевыми словами записей и в случае совпадения новость идентифицируется найденной записью в таблице. При этом, при поиске отсекается последний символ искомого слова – т. е. окончание. Параллельно ведется поиск численной информации. Также ключевые слова заданны, как на русском языке, так и на английском. Данный способ не исключает последующего ручного способа обработки информации, но сводит ручной труд к минимуму.

Практический пример системы распознавания текста

В качестве «полезного» примера применения нейронных сетей и автоматической обработки текста вообще, можно привести вариант использования байесовой нейронной сети для антиспамового фильтра [4]. Полезность такого фильтра в современных реалиях не вызывает сомнения и данный фильтр хорошо себя зарекомендовал, он используется еще с начала 2000-х годов.

Кратко принцип работы. Данный вариант работает на уровне приложения, а не на уровне сети (т.е. это не система обнаружения атак – письмо же ведь может быть «вредоносное») – на входе нейронной сети параметры электронного письма (сигнатуры): адрес отправителя, заголовок и текст, а на выходе нейронной сети – собственно признак «0/1 – спам / не спам».

Как вариант – на вход нейросети можно подавать статистические признаки сообщения, например, процент числа слов и фраз с подозрением на спам (т.е. работает предварительный фильтр на базе специально подобранного, периодически пополняемого словаря), а также не статистические признаки электронного письма: семантические признаки, направленность текста; морфологические признаки и, возможно, орфографические признаки. Данный подход неплохо работает и без нейронной сети, но применение нейронной сети предпочтительнее, т.к. т.н. «спамность» динамически меняется во времени и необходимо перенастраивать, как словарь, так и модель распознавания, а нейронная сеть умеет обобщать накопленный опыт. В работе [4] предлагается для реализации антиспамового фильтра использовать либо сеть Кохонена, либо гибридную нейронную сеть.

Как замечание – данный подход работает только с текстом, т.е. если в электронное письмо вставлена картинка с рекламным содержимым, то данный подход не будет работать.

Другими прикладными моментами компьютерной лингвистики являются:

А) задача автоматического перевода текста с одного языка на другой;

Б) задача понимания речи и ведение диалога (т.н. чат - боты)

Кратко рассмотрим задачу автоматического перевода текста, а о «чат - ботах» поговорим в одной из следующих лекций.

Задача автоматического перевода текста

Машинный перевод текста с одного языка на другой – это одно из первых направлений искусственного интеллекта. Первые труды (именно по автоматическому переводу) датируются практически в то же время, когда компьютеры стали выходить на промышленный уровень, т.е. 50-е года прошлого века. В настоящее время данное направление продвинулось далеко вперед, достаточно посмотреть уровень перевода таких систем, как «Google Translate» и «Яндекс.Переводчик», но не со всеми специализированными текстами справляются данные системы и далеко не все языки мира они охватили. При этом, точно известно, что в “Google.Translate” (GNMT – поддерживает перевод 30 языковых пар) полномасштабно используются нейронные сети (но основой все – таки является статистические методы). Также выделим системы машинного перевода: «Microsoft Translator», «Amazon Translate», «DeepL».

Обычно выделяют следующие направления машинного перевода:

- Rule-based machine translation (RBMT);

- статистическая трансляция (Statistical machine translation, SMT) – базируется на статистику переводных пар слов и словосочетаний;

- Нейронный машинный перевод (Neural machine translation, NMT) – считается подразделом глубокого обучения.

Нейронный машинный перевод кардинально отличается от предыдущих подходов к машинному переводу. С одной стороны, NMT использует непрерывные представления вместо дискретных символьных представлений в SMT. С другой стороны, NMT использует одну большую нейронную сеть для моделирования всего процесса перевода. Обучение NMT является сквозным, в отличие от отдельно настраиваемых компонентов в статистическом переводе (SMT).

Отметим систему нейросетевого машинного перевода с открытым кодом – OpenNMT от Гарвардского университета.

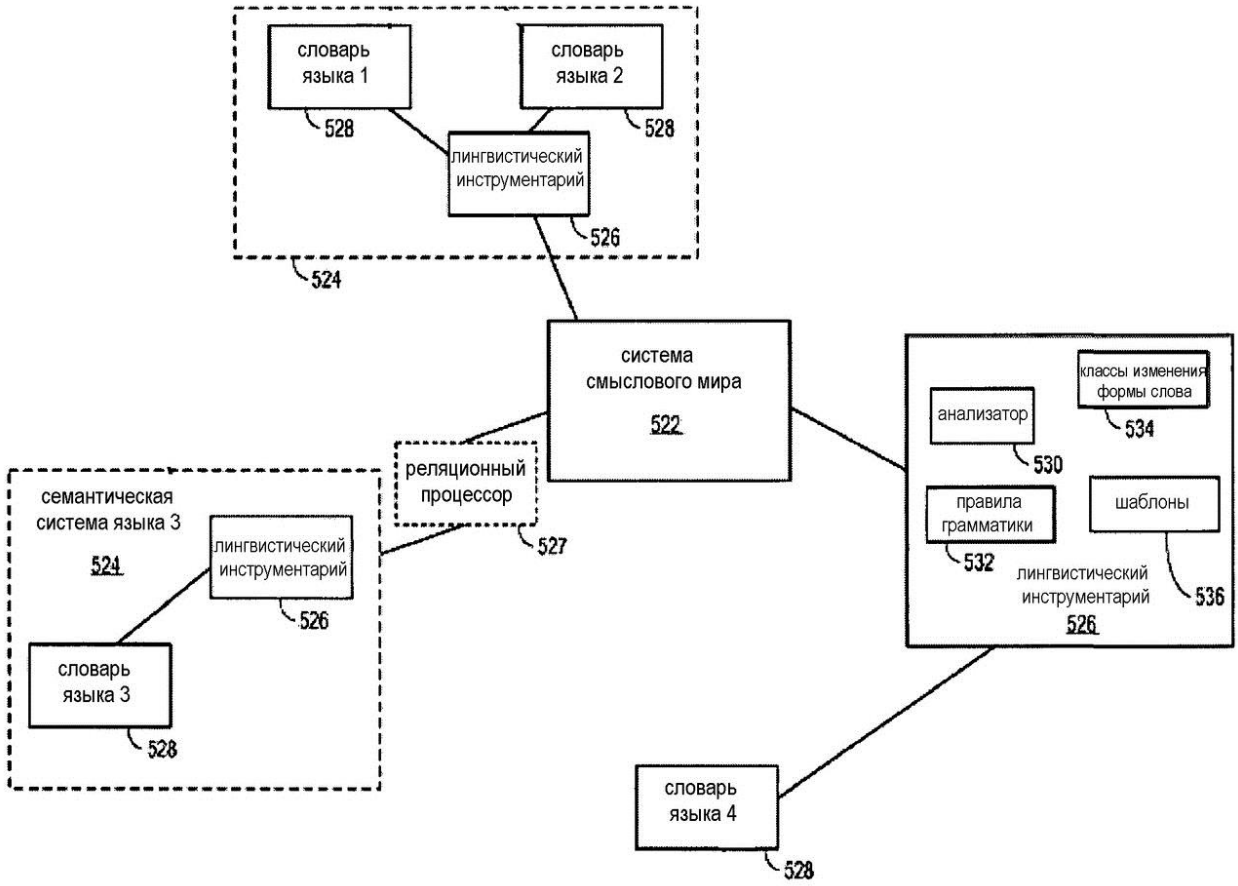


Рисунок 35. Схема обработки в NLTK

Еще приложения компьютерной лингвистики:

- информационный поиск (Informational Retrieval), с которым связаны задачи реферирования, классификации и категоризации, а также рубрицирования документов и текстов. В настоящее время этим занимается т.н. “Text Mining”, который в свою очередь относится к “Data – mining”.

Здесь интересным видится направление реферерирования текста – «выжимки» главного из входного текста, его сути. Можно сказать, что краткое содержание документа. Основной подход к решению проблемы – статистический анализ. Здесь есть еще одна схожая проблема – аннотирование текста, т.е. автоматическое составление аннотации к тексту.

- Задача формирования ответов на вопросы (т.н. FAQ);
- Задача выделение мнений (популярная современна задача. Пример – выявление мнений пользователей определенной соцсети на или иное событие или оценка качества товаров);
- Задача анализа тональности текста – оценка общей тональности высказываний по тому или иному вопросу. Примеры тональных оценок: позитивная, негативная, нейтральная.

Направление распознавания (процесс преобразования речевого сигнала в цифровую информацию) и автоматического синтеза речи будет рассмотрено далее.

Литература:

1. Бенгфорт Б., Билбро Р., Охеда Т. Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка. — СПб.: Питер, 2019. — 368 с.
2. Шумков Е.А. Расчет силы влияния макроэкономических новостей // Сетевой электронный научный журнал «Труды КубГТУ». №1, 2016
3. Шумков Е.А., Карлов Д.Н. Автоматическая система отслеживания политических и экономических новостей / XXV Международная научно-техническая конференция (летняя сессия) "Математические методы и информационные технологии в экономике, социологии и образовании". Пенза. 2010. сс. 29-31.
4. Ларионова А.В., Хорев П.Б. Метод фильтрации спама на основе искусственной нейронной сети // Интернет-журнал «НАУКОВЕДЕНИЕ» Том 8, No3 (2016) http://naukovedenie.ru/PDF/04TVN316.pdf (доступ свободный).
5. Большакова Е.И., Воронцов К.В. и др. Автоматическая обработка текстов на естественном языке и анализ данных. М.: Изд-во НИУ ВШЭ. 2017. 269 с.
6. Леонтьева Н.Н. Автоматическое понимание текстов: системы, модели, ресурсы. – М.: Академия, 2006.

ЛЕКЦИЯ 5-2. Составление семантических словарей. Машинное обучение в задаче обработки текстов

Темы:

1. Составление семантических словарей
2. Машинное обучение в задаче обработки текстов

Составление семантических словарей

Задача создания семантических словарей для всех языков мира огромна и требует новых, эффективных методов. Наиболее распространенный метод текстового корпуса (корпуса текстов) не может использоваться для всех языков малых народностей, так как у них отсутствуют тексты. Как минимум, критерий «жанр» не может быть для них использован. Т.е. лексические отношения просто невозможно построить (можно, конечно, записывать разговоры, но это не выход).

Определение:

«семантический частотный словарь - словарь, в котором вся семантика входных единиц. Такие словари могут быть язычными (например, словари языка писателя, где толкуются значения и указываются частоты слов), двуязычными и язычными (например, учебные словари минимумы). Семантические частотные словари используются в системах машинного перевода, перевода с помощью машины, в обучающих лингвистических автоматах и в качестве учебных пособий: учебных словарей-минимумов, лексических справочников и пр., сохраняя при этом все достоинства обычного частотного словаря, но существенно расширяя сферу своего применения и круг пользователей»[13].

Для составления семантических словарей используется частотный анализ текста и составление корпуса слов языка. При этом опираться на ретроспективную выборку нельзя, т.к. корпус, что называется «плывет» в течении времени (это особенно заметно по общению в соцсетях – прим. автора).

Семантический словарь должен содержать толкование единиц языка, обладать динамичностью и гибкостью, ограничивать сверху размножение формализаций единиц языка, уметь выбирать адекватный ситуации смысл единицы языка.

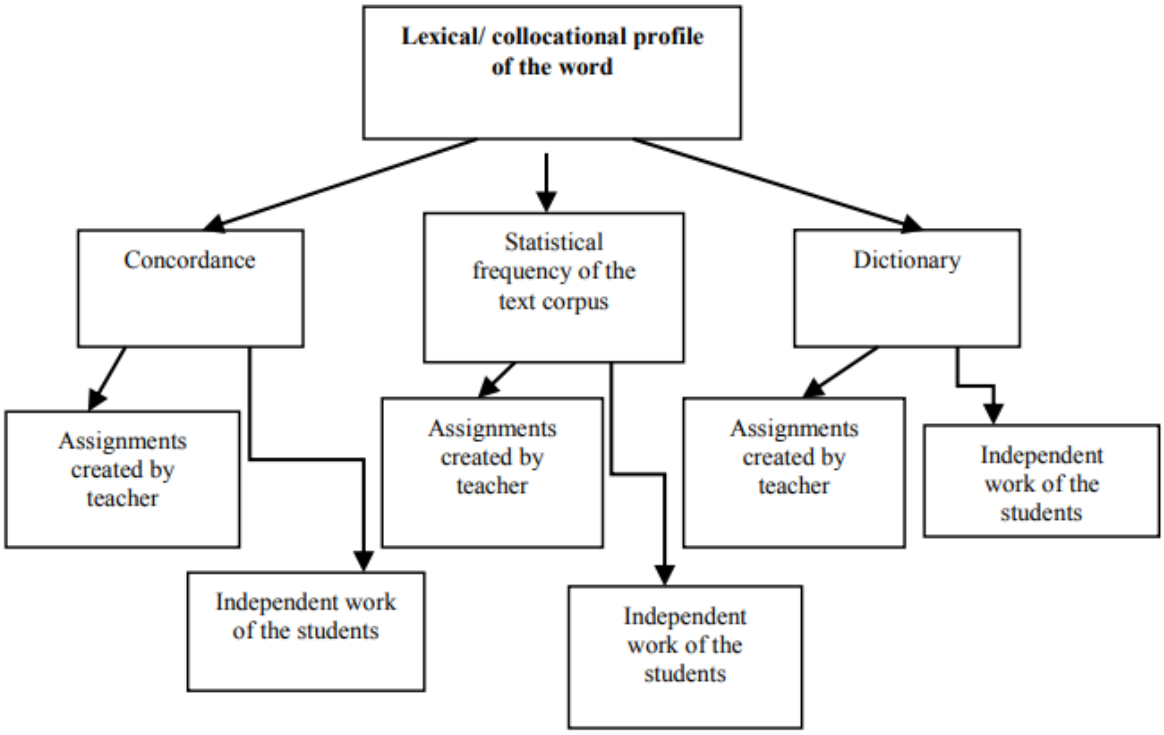


Рисунок 36. Создание корпуса слов и другие проблемы анализа текста. https://www.redalyc.org/

[13] Источник - https://sanstv.ru

Машинное обучение в задаче обработки текстов

Методы машинного обучения, применяемые в обработке текстов:

- метод Байеса;

- метод k-ближайших соседей;

- линейная регрессия;

- нейронные сети;

- классификатор Роше;

- деревья решений;

- машина опорных векторов.

Можно сказать, что используется практически весь спектр методов машинного обучения.

Да, кстати, данные методы помогают в машинном переводе…

Диалоговые системы (чат - боты)

Определение: «*ДИАЛОГОВАЯ СИСТЕМА, или интерактивная система, или система вопрос-ответ — автоматизированная человекомашинная система, работающая в режиме диалога, при котором она отвечает на каждую команду пользователя и по мере надобности обращается к нему за информацией*»[14].

Диалоговые системы относятся к одному из семи важнейших направлений искусственного интеллекта и имеют уже солидную историю.

Краткая история диалоговых систем: одной из первых (и удачных) диалоговых систем была программа – психоаналитик ELIZA (разработчик В. Вейценбаум), которая была создана и успешно работа в 60-х годах прошлого века. Другим пионером диалоговых систем была программа PARRY 1972 года , которая также работа в области психоанализа! Автор – психиатр Кеннет Колбай (Стенфордский университет). Есть интересный эксперимент – в 1972 году по сети ARPANET (предшественник современного Интернета), в котором ELIZA и PARRY общались друг с другом! При этом – данный диалог прошел тест Тьюринга! Опытные психиатры не смогли отличить диалог от диалога с настоящим больных шизофренией (это документально подтверждено).

Диалоговые системы по большому счету классифицируются на две категории – задаче-ориентированные (*General — Task-oriented*) и общего назначения (*Open Domain — Closed Domain*). Вышеупомянутая ELIZA была узкоспециализированной, она могла говорить только на тему психоанализа (в

[14] https://publishing_dictionary.academic.ru

то же время у нее не было четкой задачи, о чем говорить), но по классификации – общего назначения закрытого домена. Есть также другая классификация:

- открытого домена;

- закрытого домена.

Первые – системы, способные говорить на любую тематику, вторые – на строго определенные.

Архитектура современных диалоговых систем (а также голосовых помощников) обычно строится по модульной архитектуре, состоящей из [2]:

- модуль распознавания человеческой речи;

- модуля понимания естественного языка;

- модуля набора разговорных навыков;

- модуля составления и ведения диалога;

- модуль генерации естественной речи.

В современных диалоговых системах первый модуль обычно состоит из нейросетевой топологии с возможностью понимания текста, разметки, последовательности и моделей извлечения информации. По сути – это компонент, который определяет направления разговора, предварительно определив тематику (здесь работает функция классификации). В то же время, если разговор общего плана, то возможны различные ньансы. Отдельно выделяют цельные диалоговые системы.

В диалоговых системах сейчас активно применяют нейронные сети, но изначально (и это сохраняется) они использовали экспертные системы различных типов: семантические, фреймовые и продукционные.

Общий принцип работы диалоговой системы, следующий:

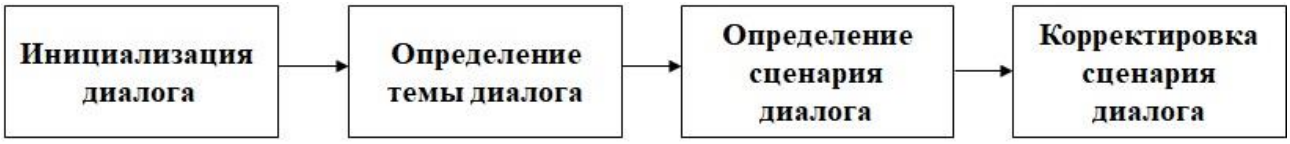


Рисунок 37. Общим принцип функционирования диалоговой системы

Здесь необходимо упомянуть о тесте Тьюринга, который должна пройти каждая уважающая себя диалоговая система: «*Человек взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы — ввести человека в заблуждение, заставив сделать неверный выбор*»[15].

[15] https://ru.wikipedia.org/

Примеры современных диалоговых систем:

- «Алиса» от Яндекса;

- Siri от Apple;

- Cortana от Microsoft;

- Amazon Echo / Alexa;

- Google Now / Assistant.

Здесь мы видим «большую пятерку» ИТ – гигантов – все они развивают голосовых помощников (а также системы автоматического перевода) для создания т.н. «эко – ИТ - системы». Все системы (по неподтвержденным данным) используют нейросетевые «движки». У Microsoft есть также система Xiaoice, которая обладает некоторыми «творческими» возможностями – пишет стихи и исполняет песни. Система Siri также использует принципы коллаборативной фильтрации и т.н. персональный подход, когда изучаются в процессе длительного общения с пользователя его предпочтения, т.е. система приспосабливается к пользователю.

Упомянем о международном соревновании разговорного искусственного интеллекта - Conversational Intelligence Challenge (ConvAI) и конкурсе “ Alexa Prize Socialbot Grand Challenge”.

Вопрос определения категории диалога

Рассмотрим один из этапов ведения диалога – определение категории, к которой относится диалог. Нередко для этих целей используется технология т. н. «Мешка слов». Под мешком слов понимается - модель текстов на натуральном языке, в которой каждый документ или текст выглядит как неупорядоченный набор слов без сведений о связях между ними. Его можно представить в виде матрицы, каждая строка в которой соответствует отдельному документу или тексту, а каждый столбец — определенному слову. Ячейка на пересечении строки и столбца содержит количество вхождений слова в соответствующий документ. Одним из основных и сложных моментов данного подхода является определение словаря. При работе с реальными текстовыми массивами получается большой объем словаря, поэтому для сокращения словаря необходимо игнорировать регистр слов и пунктуацию, удалять стоп – слова, предлоги и … Мешок слов естественно формируется на существующей выборке (очевидно, что чем больше выборка, тем качественней сформирован мешок слов).

Далее необходимо ввести кластеры, которые отвечают за категории диалога. Это можно сделать различными способами, в частности: k –means, mini – batch k – means, agglomerative clustering и др.

Далее введем такое понятие, как *сценарий диалога*. Понятно, что диалог может быть совершенно различный, в частности «пустой». Но, будем ориентироваться в сторону профессионального диалога. Также введем понятие соответствия диалога и точность ответов машины. Настройка сценария диалога может проходить, как в режиме реального времена, так и в офф-лайн режиме на наработанной истории. Необходима также т.н. разметка диалога, т.е. режим, в котором эксперт (или пользователь) проставляет коэффициенты точности ответов и соответствия диалога. Одной из перспективных технологий для создания подобных систем с корректировкой являются нейросетевые топологии с подкреплением.

Библиотеки для построения диалоговых систем

Выделим следующие распространенные библиотеки обработки естественного языка:

- NLTK - поддерживает такие задачи, как классификация, стемминг, маркировка, синтаксический анализ и семантическое рассуждение (ЯП – Python, разработчик – Пенсильванский университет, США. Авторы С. Берд и Э. Лопер).

- spaCy – «spaCy разработан, чтобы помочь вам выполнять настоящую работу - создавать настоящие продукты или собирать реальные идеи. Библиотека уважает ваше время и старается не тратить его зря. Его легко установить, а его API прост и продуктивен» - пример машинного перевода translate.google.com . Считается, что spaCy обладает самым быстрым парсером из всех существующих NLP систем, по крайней мере с открытым кодом. Написан на Cython. Поддерживает 7 языков (на 2020 год).

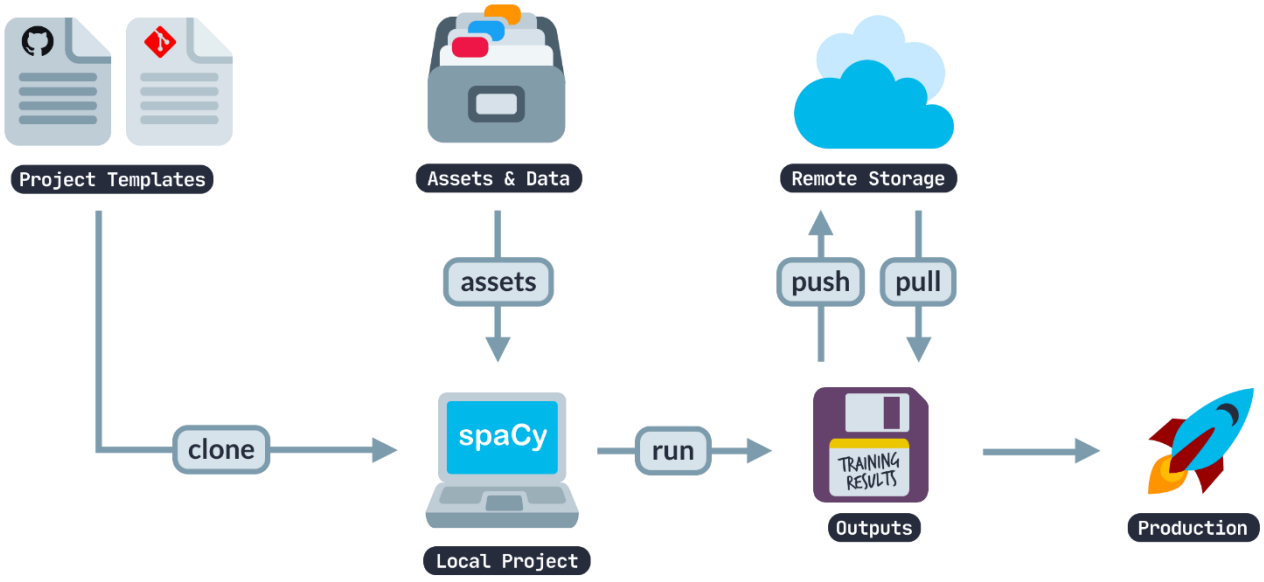


Рисунок 38. Принцип работы spyCy (схема с сайта https://spacy.io/)

«несколько компонентов могут использовать общую модель «токен-вектор», и лемматизатор легко заменить или отключить. Трубопроводы спроектированы так, чтобы быть эффективными с точки зрения скорости и размера и хорошо работать, когда трубопровод запущен на полную мощность» - еще вариант современного машинного перевода.

- fastText - для изучения семантики и классификации текста, созданная Facebook AI Research.

- CoreNLP - ЯП Java, но есть оболочки на Python, первая «промышленная» версия вышла в 2010 году. Создана в Стенфордском университете (США). Работает по принципу «pipeline» (трубопровод). Краткий принцип работы – «текст помещается в объект аннотации, а затем аннотаторы добавляют информацию в аналитический трубопровод. Результирующая аннотация, содержащая всю аналитическую информацию, может выводиться в виде XML или обычного текста».

CoreNLP проводит морфологический и семантический анализ. Результаты пока оставляют желать лучшего…

- Pattern. Еще одна библиотека NLP на Python. Предоставляет инструменты для частеречной разметки (*part-of-speech tagging*), анализа настроений, векторных пространств, моделирования (SVM), классификации, кластеризации, n-граммы поиска и WordNet

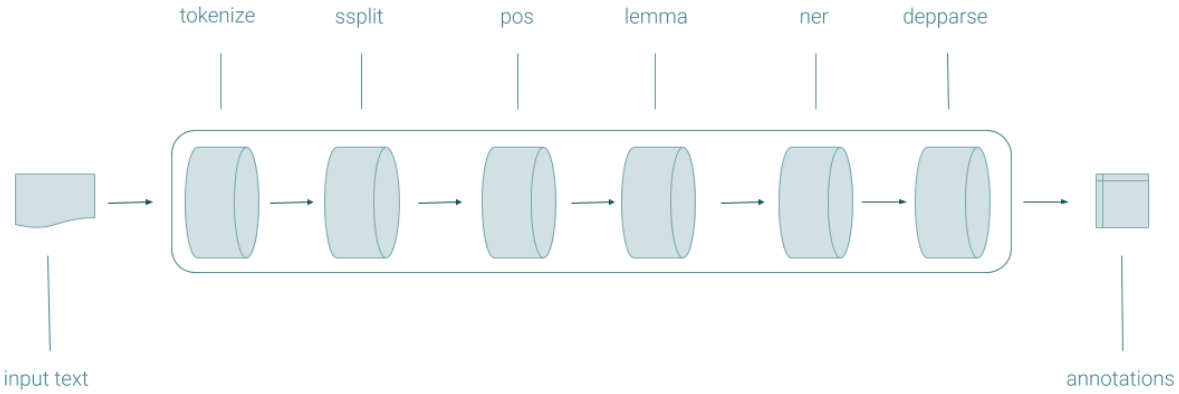


Рисунок 39. Принцип работы CoreNLP

Изображения с https://stanfordnlp.github.io .

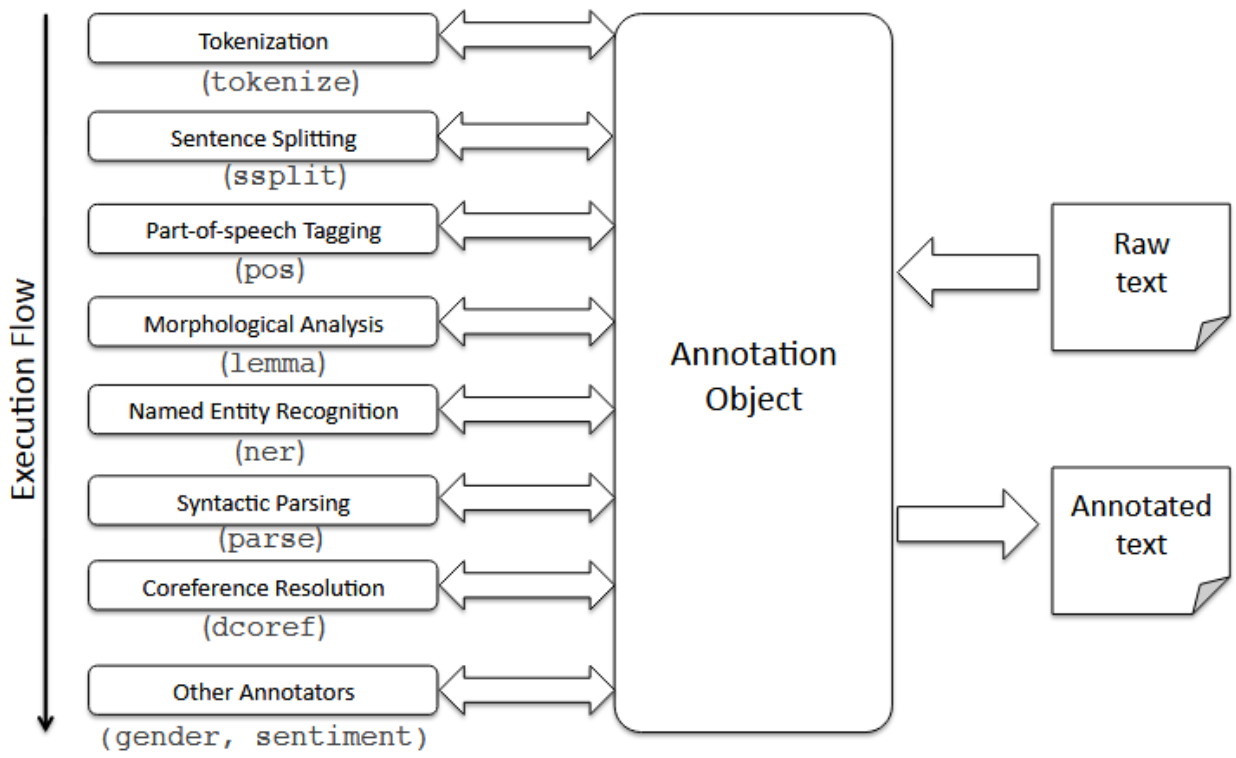


Рисунок 40. Архитектура CoreNLP

- TextBlob – библиотека для обработки текстовых данных на естественном языке. Хорошо работает в задачах анализа настроений (да, есть и такая задача…), POS – маркировка (маркирование семантических предложений в предложениях), извлечение именных фраз. ЯП – Python.

Из нейросетевых инструментов наибольшую популярность получили: TensorFlow, Keras, PyTorch, а также некоторые другие, в основном на языке программирования Python.

Отметим, что среди вышеперечисленных, нет ни одной системы (библиотеки) российского производства (хотя российские программисты наверняка трудились над ними).

Варианты применения нейронных сетей для обработки текстов

Пример применения нейронной сети Хемминга для распознавания слов

Одним из примеров работы с текстом (а также для диалоговых систем) является применение сети Хемминга для определения ошибок в словах или, можно сказать, автоматического исправления орфографии, по принципу автоассоциативной памяти. Сеть Хемминга, напомним, как раз и служит для

такого рода задач. Данный пример применения хорошо разобран на сайте www.basegroup.ru вместе с исходных кодом.

Суть – пользователь, допустим, в строке поиска расписания маршрутов движения поездов, вводит название города и он может ошибиться (возможно, просто, спешит), набрав, для примера, вместо «Москва» - «Масква». Соответственно, если в системе нет возможности исправления подобных ошибок в автоматическом режиме, то пользователь получит ошибку и будет терять время, чтобы понять в чем причина отказа. Дело в том, что в итоге запрос идет в СУБД, где в таблице населенных пунктов есть только запись «Москва». А если есть система автоматической корректировки, то пользователь получит нужный ответ, даже не догадываясь, что он ошибся при вводе.

Чтобы решить данную задачу с помощью сети Хемминга, необходимо придумать систему кодирования символьной информации в вектора. Учитывая, что сеть Хемминга работает только с -1 и 1 – это сделать проще так:

Буква «А» - «-1,-1,-1,-1,1»

Буква «Б» - «-1,-1,-1,1,-1»

Буква «В» - «-1,-1,-1,1,1»

И так далее.

Т.е. мы задаем битовую маску для каждого символа.

Соответственно, составим таким образом словарь населенных пунктов в нашей системе расписания движения поездов, мы обучаем на них сеть Хемминга, которая за счет свое принципа работы, будет автоматически исправлять ошибки! Единственный момент – сеть плохо распознает ошибки, если длина слова с ошибкой и без ошибки различна. Но это можно обойти, подавая в цикле по очереди по одному символу в каждой позиции и добавляя по одной букве в каждую позицию, если на первой итерации на выходе не было распознан н.п.

Ряд авторов отмечает, что по экспериментальным данным рекуррентные нейронные сети при работе с текстом показывают результаты лучшие в сравнении со сверточными сетями, в частности двунаправленные модели с долгой – короткой памятью (BiLSTM).

Для работы с текстом применяются также т.н. «неглубокая широкая сверточная нейросеть» (SWCNN – swallow and wide convolutional neural network).

Направление Text – mining

Первые публикации на тему «Text - mining» (с этим названием), датированы еще 1992 годом. Сейчас это актуальное и довольно развитое

направление искусственного интеллекта. Но, на взгляд автора, это скорее «модное название» одного совокупности направлений компьютерной лингвистики. При этом оно в рамках Data – Mining (интеллектуальный анализ данных). По идее Text Mining – это семантический анализ текста. Интеллектуальный анализ текста используется для прогнозирования принадлежности строк, предложений, абзацев или документов к набору категорий. Поскольку он прогнозирует категорию текста или документа на основе изучения аналогичных шаблонов из предыдущих текстов, он квалифицируется как метод прогнозной аналитики. Хотя интеллектуальный анализ текста не использует какие-либо методы классификации или регрессии, он концептуально идентичен прогнозированию, когда он используется для изучения категорий текста из предварительно классифицированного набора текстов, а затем использует обученную модель для прогнозирования новых входящих документов, новостей элементы, абзацы и т. д. Интересно, что другая форма интеллектуального анализа текста может использовать кластеризацию, чтобы увидеть, какие новости, сообщения в твиттере (твиты), жалобы клиентов и документы «похожи», поэтому интеллектуальный анализ текста может относиться как к описательной, так и к прогнозной аналитике.

Ну и в окончании лекции немного о современном состоянии машинного перевода:

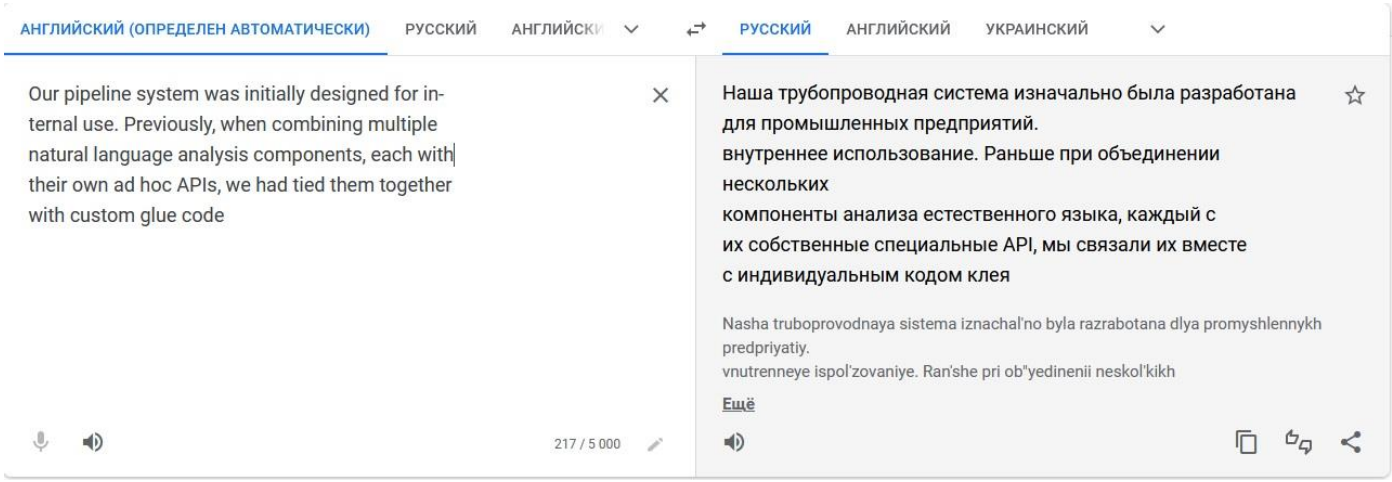


Рисунок 41. Перевод Google Translate. Фраза с
https://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf

Приведен пример современного технического перевода…

Литература:

1. Тузов В.А. Компьютерная семантика русского языка. – СПб: Изд-во СПбГУ, 2004
2. Баймурзина Д.Р. Нейросетевые модели и диалоговая система для ведения разговора на общие темы. Дисс. канд. техн. наук. МФТИ. 2021. 136 с.
3. Бонгард М.М. Проблема узнавания. — М.: Наука, 1967. — 320 с.
4. Куторба А.Ю. Обработка англоязычных текстов на основе семантического словаря // Вестник Санкт – Петербургского университета. 2005.
5. Добров Б.В., Лукашевич Н.В., Автоматическая рубрикация полнотекстовых документов по классификаторам сложной структуры // Восьмая национальная конференция по искусственному интеллекту — Коломна, 2002.
6. Аношкина Ж.Г. Морфологический процессор русского языка. // Бюллетень машинного фонда русского языка / отв. редактор В.М. Андрющенко — М., 1996. — Вып.3, с.53-57.

ЛЕКЦИЯ 5-3. Системы автоматической обработки текстов

Вместо введения

Понятно, что самыми крупными и проработанными системами автоматической обработки текста[16] являются «Google» (США), “Байду” (Китай), «Яндекс» (Россия). В Японии, как ни странно, до сих пор в лидерах “Yahoo”, который в остальном сдал свои лидирующие позиции. Но, мы эти системы, в рамках лекции рассматривать не будет, а сосредоточимся на других.

ЛА – лингвистический автомат (подробно см. [6].)

АОТ – автоматическая обработка текста.

Часть 1.

Основные проблемы обработки текстов:

1. Тексты обычно неструктурированы или слабо структурированы;
2. Большое количество орфографических ошибок в текстах из сети Интернет;

[16] Часто используется название «автоматическая переработка текста» (АПТ).

3. В существующих подходах обычно происходит увеличение информации, передающейся по цепочке автоматической обработки текста (см. ниже)
4. Проблема редких языков.

И др.

Часть 1. Применение систем автоматической обработки текстов

Если не брать в расчет тексты в сети Интернет (в т.ч. социальные сети), то существуют такие важные предметные области: здравоохранение (истории болезней), научные труды, патентное дело, ….

В сети Интернет есть такие интересные задачи: проверка орфографии, фильтрация спама, автоматический перевод, нахождение релевантных ответов на вопросы, анализ + градация мнений и отзывов, создание рекомендательных систем, анализ тональности высказываний, генерация речи, определение жанровой принадлежности, автоматическая проверка правописания и др. Выделим, возможно, самое сложное применение – извлечение смысла из текста.

Извлечение информации (знаний) из текстов – отдельное направление в компьютерной лингвистике (Information Extraction)[17], которое в свою очередь идет как часть Информационного поиска (Informational Retrieval). Из текстов обычно извлекаются:

- значимый объект;

- атрибуты объекта;

- отношения между объектами;

- событие или факт.

Очень интересная задача на автоматическую обработку текстов встречается в рекрутинговых агентствах – обработка резюме и поиск подходящих кандидатов на вакансию.

Все перечисленные задачи, по большому счету, отличаются отсутствием формализации и для них нет готовых, гарантированно работающих алгоритмов.

(не только Интернетом едины системы АОТ). Сейчас, в связи с оцифровкой старых документов, журналов, книг и т.д. возникает задача, кроме машинного распознавания написанного, допустим, задача автоматической спецификации для чертежей, блок – схем. Или, например, для патентов. Для последних необходимо определить: номер патента, страна патентования,

[17] Есть интересное применение – извлечение знаний из финансовой аналитики. Или допустим в правоохранительной деятельности.

заявитель, название, дата принятия в патентный отдел, дата приоритета, организация (авторы) и т.д.

Часть 2. Существующие методы

Методы выделения ключевых слов, автоматическое реферирование, классификация текстов,

Предобработка текстов, когда большой целостный текст заменяется несколькими меньшими по размеру.

Фактически любая задача автоматической обработки текстов состоит из следующих шагов:

- графематический анализ – выделение токенов;

- морфологический анализ – выделение грамматической основы слова и определение частей речи (самое сложно, наверно, это обработка «неграмотного» общения в соцсетях…);

- синтаксический анализ – определение синтаксических связей между словами;

- семантический анализ – будем говорить, что это – понимание сути.

Понятно, что каждая такая задача имеет собственный набор алгоритмов.

Общая последовательность: на вход АОТ-системы поступает текст в виде последовательности символов, далее первый этап – лексический анализ (разбиваем на слова и выделяем спецзнаки) + деобфускация (поиск ошибочного или преднамеренно ошибочного написания текста – в рекламных рассылках); на втором этапе идет обработка отдельных слов – морфологический анализ[18] (поиск граммем) и основной словоформы, часто проходит с помощью лемматизации или стемминга (см. определения в предыдущей лекции).

Родственные слова ищутся с помощью специальных словарей – тезаурусов. Словарь – тезаурус обычно представляет собой семантическую сеть, у которой вершины соответствуют словам, а дуги – связи между словами (обычно это раскрашенная семантическая сеть). При этом – близость двух слов определяется, как ближайшая вершина в графе.

Существуют специальные алгоритмы нахождения связности слов на базе их совместной встречаемости (также с использованием нечеткой логики).

Синтаксический анализ часто проводится с помощью теории графов.

[18] Есть два подхода: а) точный морфологический анализ (точная морфология) – проходит долго, б) неточная морфология – проходит быстро, но не точно. Для морфологического анализа до сих пор трудно определять родственные слова, как пример, «безопасность» и «защита».

В ряде задач можно обходится без синтаксического анализа (допустим в классификации текстов).

По итогам семантического анализа должен получиться локальный тезаурус и здесь важен момент автоматического его пополнения.

Часто в системах автоматической обработки текстов анализ проходит в два этапа – предварительное сканирование и выявление необходимых участков текста для анализа, а далее вторая фаза – с детальным анализом. При этом возможны рекурсивные возвраты на первую фазу для уточнения.

В задачах извлечения информации, на выходе системы генерируется структурированная информация, извлеченная из коллекции текстов и решается задача автоматического извлечения данных, которые релевантны заданной теме (проблеме или вопросу). На выходе часто визуальная информация в виде семантических сетей или когнитивных карт.

Задача распознавания именованных сущностей (Named Entity Recognition - NER) – здесь есть определенные успехи, но до «идеального» варианта еще далеко. Часто решается методами машинного обучения (как вариант с размеченными или неразмеченными текстами), но есть, допустим подход на продукционных правилах:

ЕСЛИ за словом господин, президент, товарищ …, следует слово с большой буквы

ТО извлечь из этого и последующих слов фамилию, имя и отчество

ЕСЛИ за словом господин, президент, товарищ…, следует комбинация из одиночных букв с большой точки

ТО извлечь из данных символов инициалы персоны

{вложено} ЕСЛИ за одиночными буквами с большой буквы, следует слово с большой буквы

ТО извлечь из этого слова Фамилию

И т.д. Понятно, что таких правил в реальной системе много. Или допустим, могут быть сокращения, например, для населенных пунктов: «гор.», «г.», «пос.», «н.п.», «нас. пункт», «ст.» и т.д. и это тоже необходимо учитывать.

Конечно, должен быть, словарь имен и отчеств (еще момент – для разных стран могут разные варианты, где то есть отчество, где нет, или допустим в арабских странах полное Ф.И.О. человека может состоять из 5 или более слов..).

Приведем формулу оценки качества извлечения данных из текста – т.н. F – мера:

$$F = \frac{2RP}{R + P}$$

где R – полнота, P – точность.

Считается, что система качественно выполнила извлечение данных, если F больше 90%.

Часть 3. Программные системы компьютерной лингвистики

Консольные приложения (из 90-х, начало 2000-х): Mystem (морфологический анализатор), АОТ[19], PyMorphy 1 и 2, «Томита парсер» (от «Яндекса»), «ЭТАП3» (синтаксический парсер), KPLM[20], OpenXerox, Snowball, которые предназначены для различных задач: токенизации, морфологического анализа (пример - NLTK), создания синтаксической структуры и др. Первая IE – система – AutoSlog, также известность получила OntosMiner (извлечение данных из деловых текстов), OpenNLP.

Системы можно классифицировать по следующим признакам:

- по виду лицензирования (отметим, что есть довольно много открытых систем от западных университетов и сообществ сети Интернет - Apache OpenNLP, StanfordNLP, NLTK, GATE);

- открытости;

- мультиязычности (возможно самые дорогие);

- характеру (либо готовое «коробочное» решение, либо библиотеки для конструирования);

А также по используемым данным, математическим моделям и др.

Кратко обзор существующего программного обеспечения по данному направлению:

- Microsoft Bing Translator;

- Google Translator;

- ОРФО;

- ABBYY Compreno (парсер);

[19] Да, кстати, с официального сайта АОТ: «Наши технологии базируются на многоуровневом представлении естественного языка, которое, в свою очередь, было заимствовано у системы ФРАП (Система французско-русского автоматического перевода была разработана коллективом лаборатории машинного перевода Всесоюзного центра переводов совместно с коллективом лаборатории машинного перевода МГПИИЯ им М. Тореза. 1976-1986 гг.)», т.е. в СССР этим довольно серьезно занимались…

[20] KPML (Komet Project MultiLinual) – Тактический генератор - специальная среда для генерации текстов на различных языках. Имеются ресурсы для английского, французского, немецкого, голландского, японского, чешского, русского и др.: http://purl.org/net/kpml

- система IBM Watson – предназначена для медицины (также для генерации текстов);

- рекомендательная система новостных сообщений News360;

- популярная программа переключения раскладки клавиатуры Punto Switcher также относится к системам компьютерной обработки текста!

Другие примеры: IBM Content Analytics, SAS Text Miner.

Отметим крупные словари текстов организаций, которые занимаются их сбором (выборки текстов превышают сотни миллионов слов, а некоторые перевалили за миллиард!): Association for Computational Linguistics' Data Collection Initiative (ACL/DCI), European Corpus Initiative (ECI), ICAME, British National Corpus (BNC), Linguistic Data Consortium(LDC), Consortium for Lexical Research (CLR), Electronic Dictionary Research (EDR).

Часть 4. Структура систем Автоматической обработки текста (АОТ)

Структура систем автоматической обработки текстов по решающему ядру сильно зависит от подхода. Выделяют ИИ – подход – основан на методах искусственного интеллекта, второй ИЛ – подход (инженерно - лингвистический). Первый считается традиционным алгебраическим, второй – эмпирическим.

Большинство систем автоматической обработки текстов используют морфологические словари (пример такого словаря – «АОТ», http://www.aot.ru/ - по сути, словарь Зализняка в цифровом формате).

Практически всегда используется Тезаурус – семантическая сеть, которая показывает взаимосвязи семантических отношений. Как пример – WordNet (русского аналога WordNet пока не существует).

В таких системах также обязательно наличие пополняемой (или корректируемой) базы знаний. По сути это динамическая экспертная система с дополнительными модулями, например, нейросетевым.

При обработке документов часто выделяют две задачи [2]:

- обработка отдельных документов[21] - на входе и выходе будет текстовых документ. Даная задача обычно решается с помощью экспертных правил, методами машинного обучения;

- обработка массивов документов. Решается обычно на базе системы правил извлечения информации (шаблонов)[22].

[21] Здесь еще возникает задача корректировки и реферирования документов.
[22] При обработке документа просматриваются результаты синтаксического анализа и ищутся фрагменты, структура которых отвечает шаблонам из правил извлечения информации. Далее в соответствии со

Во второй задаче происходит извлечение смысла, фактов, выделение ключевых слов документа и др. Плюс, возможно, генерация каких – либо обобщающих документов. Еще одна подзадача из второй группы – собственно антиплагиат (или задача выявления задвоенных документов).

И здесь, конечно, необходимо отметить, что данные задачи решаются либо для локальных текстов, либо для текстов из сети Интернет. И подходы различаются. Ряд задач, например, «спам - фильтрация» может решаться и без алгоритмов компьютерной лингвистики. И, конечно, в сети Интернет просто гигантские массивы новой, ежедневной текстовой информации (даже не рассматривая видео, фото и аудио).

Существенным является вопрос обучения нахождению знаний (информации в текстах). Здесь выделяют контролируемые методы и неконтролируемые. Контролируемые работают на аннотированных текстах, которые предварительно обработал эксперт – дал смысл семантический каждому тексту. Неконтролируемые методы соответственно на неаннотированных текстах.

Часть 5. Фреймовый подход к автоматической обработке текстов

Это популярный подход к задачам автоматической обработке текстов. Теория фреймов была предложена в середине 70-х годов прошлого века. Автором является Марвин Мински (мы его фамилию встречали в лекциях по нейронным сетям – правда там он выступал, как критик нейросетей). Так вот, Мински предложил когнитивный подход описания мира или предметной области в виде фреймов и считается, что так мыслит человек. «Фрейм любого вида — это та минимально необходимая структурированная информация, которая однозначно определяет данный класс объектов» (М.Мински). Соответственно, встречаясь с каким либо объектом или событием или .. в сознании всплывает фрейм. Например, когда говорят «комната…», то всплывает фрейм – описание «у комнаты обычно есть 4 стены, минимум одна дверь и обычно есть окно или несколько». Есть фреймы верхнего уровня, есть подфреймы. Допустим, когда говорят «гостиничный номер», то всплывает фрейм «гостиничный номер», который наследуется от фрейма «комната» и т.д.

Различают фреймы – структуры, фреймы – сценарии, фреймы – роли, фреймы – ситуации и т.д.

Принцип применения фреймов в автоматической обработке текстов можно описать следующим образом [6]:

«В системах АОТ фреймы строятся по традиционной схеме, в которой заранее заполненная тематическая строка сопровождаются пустым полем —

«сработавшими» правилами часть слов извлекается из текста и преобразуется в формализованную структуру [2].

12«дырой»-слотом. В него ЛА должен вставить обнаруженные им в тексте тематические комментарии. Задача алгоритмизатора состоит в том, чтобы поместить в базу знаний ЛА индикаторы, выявляющие с достаточно большой вероятностью и отправляющие в слот тот рематический фрагмент текста, который комментируют соответствующую им топиковую строку. Алгоритм должен достаточно полно учитывать вероятности коммуникативно-семантических связей между заранее за данными во фрейме топиками и попадающими в слоты текстовыми фрагментами или их переводами. В этом случае он не только даёт возможность автомату организовывать содержание текста. Он обеспечивает также пользователю определённый психологический комфорт при восприятии машинной аннотации, перевода и т.п. Уязвимой стороной фреймовой методики является то, что выбранный системой рематический фрагмент может попасть в «чужой» слот.»

Про фреймы можно почитать здесь [7, 8].

Часть 6. Задача генерации текстов

Довольно интересная задача по генерации новых текстов. Был даже нашумевший случай, когда ученые одного из университетов США создали нейросетевую программу по генерации научных статей и отправили один из результатов в один из научных журналов. И эта сгенерированная «научная» статья прошла рецензирование и была опубликована! Но потом все вскрылось…

Генерация текстов состоит в построении корректных документов, содержащих описание формально заданной информации [2]. Часто применяются в чат – ботах, см. предыдущую лекцию.

Данная задача довольно актуальна для владельцев веб – ресурсов – для рейтинга в поисковых системах необходим уникальный материал (но, правда, он потом быстро расходится по сети, обычно без указания авторства). Еще одна задача – как на основе имеющегося текста (или набора текстов) сгенерировать уникальный текст? Решение данной задачи довольно сильно экономит бюджет веб – сайтов, если написание уникального текста человеком стоит около 2 долларов за 1000 знаков (цена примерная), то для автоматических систем – на порядок ниже (на основании опыта автора). Часто это задача возникает для Интернет – магазинов, когда необходимо под ассортимент в тысячи товаров сделать их описание (а заодно и сгенерировать отзывы владельцев). И есть даже такое направление как «SEO анализ текста». Но! Есть такой момент, что необходимо написание текста, оптимизированного и под читателя и под поисковую машину! А это две разные вещи…

Основной подход для систем генерации текстов – на базе шаблонов (паттернов). Самый простой способ – есть набор шаблонных фраз (допустим для описания бытовой техники), связывая которые, используя семантические

отношения и ключевые слова, получается текст. Примеры подобных систем - Employee Appraiser (компания Austin-Haynes), Performance Now (компания KnowledgePoint). Но (почти) полная классификация подходов:

- Canned-based methods – на базе неизменяющихся шаблонов;

- Template-based methods – с изменяющимся шаблоном;

- Phrase-based methods – на базе контекстной вставки;

- Feature-based methods – синтез на базе грамматических признаков;

При этом для инженеров поисковых систем встает вопрос разработки алгоритмов выявления таких «сгенерированных» текстов (в этих текстах обычно не соответствие падежей и прочие ляпы, наподобие «заказать телевизора за 15500 рублей»).

Рисунок 42. Система генерации и проверки текста (источник habr.com)

Ну и интересный факт из истории – еще в 1983 году существовала система ANA автоматического создания бюллетеня о биржевых торгах на базе результатов торгов за день. В 1994 году работала система FoG – автоматическое создание метеорологических сводок. Пример таблицы для данной системы на рисунке ниже

0) Исходные данные			1) Понятия ПО		2) Элементы текста
время	Напр. ветра	Скорость ветра	Скорость ветра	Напр. ветра	
7 a.m.	235	17	15-20	southwest	wind 15-20 southwest
9 a.m.	231	21	15-20	southwest	
…	…	…	…	…	diminish to
9 p.m.	280	12	light	(west)	
10p.m.	307	11	light	(northwest)	wind light southwest
11p.m.	182	8	light	(south)	
12p.m.	246	10	light	(southwest)	

Рисунок 43. Таблица для формирования метеорологических сводок системы FoG (источник – Соколова Е.Г.)

Здесь возможно применение фреймов макроструктуры текста (фреймы – см. М.Мински), а также другие способы [4].

Литература:

1. Ильвовский Д., Черняк Е. Системы автоматической обработки текстов // Открытые системы. 2010, №1.
2. Селезнев К., Владимиров А. Лингвистика и обработка текстов // Открытые системы. 2013, №4.
3. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск. М.: Вильямс, 2011. — 528 с.
4. Соколова Е.Г. Автоматическая генерация текстов на ЕЯ. Российский НИИ искусственного интеллекта [https://www.dialog-21.ru/media/2570/sokolova.pdf]
5. Маккьюин К. Дискурсивные стратегии для синтеза текста на естественном языке // НЗЛ. Вып. XXIV.М., 1989. С. 311-356.
6. Луканин А.В. Автоматическая обработка естественного языка: учебное пособие. Челябинск: ЮУрГУ. 2011. 70 с.
7. Minsky M. A. Framework for representing knowledge. Trans. from engl. M.: Energiya. 1979. 152 p.
8. Шумков Е.А. Фреймовые экспертные системы с использованием нейронных сетей // Научные труды КубГТУ. 2019, №154. сс. 226 – 232.

ЛЕКЦИЯ 6-1. Глубокое обучение с подкреплением: обучение с подкреплением, Q-обучение

Метод обучения с подкреплением – это самостоятельное и уже вполне сформировавшееся направление кибернетических исследований. Обучение с подкреплением используется в различных областях науки: нейронных сетях, психологии, искусственном интеллекте, управлении, исследовании операций и т. д. Главное достоинство этого метода – его сравнительная простота, но не реализация: наблюдаются действия обучаемого объекта и в зависимости от результата поощряют, либо наказывают данный объект, не объясняя обучаемому объекту, как именно нужно действовать. Роль учителя может играть внешняя среда. В данном методе большое внимание уделяется поощрению/наказанию не только текущих действий, которые непосредственно привели к положительному/отрицательному результату, но и тех действий, которые предшествовали текущим. Основные принципы обучения с подкреплением [7, 8]:

- обучение через взаимодействие;
- целенаправленное обучение;
- обучение через взаимодействие с окружающей средой.

Основные термины и определения

Объект управления (или агент) – то, чем управляем. Например, мобильный робот, механическая торговая система, чат – бот и т.д.

Характеристики объекта управления. Под характеристиками объекта управления будем понимать состояния его параметров, например, запас топлива, заряд аккумуляторной батареи, состояние корпуса, быстродействие и т.д.

Стратегия – это отображение воспринимаемых состояний среды в действия, отвечающие этим состояниям. *Или* – определение характера обучающегося агента в каждый момент времени.

TD – временная разность (от англ. Time – Difference). См. ниже Ошибка временной разности.

ОИС – обобщенная итерация по стратегиям.

Функция оценки – показывает, что есть хорошо в продолжительный период, тогда как функция подкрепления показывает, что есть хорошо в текущий момент. Оценка состояния - это итоговое подкрепление агента, которое предположительно может быть накоплено при последующих стартах из этого состояния. В то время как подкрепление определяет прямую, характерную желательность состояния окружения, оценки показывают долгосрочную желательность состояний после принятия во внимания состояний, которые последуют за текущим, и подкреплений, соответствующих этим состояниям. Например, состояние может повлечь низкое непосредственное подкрепление, но иметь высокую оценку, потому как за ним

регулярно следуют другие состояние, которые приносят высокие подкрепления [7, 9].

Функция подкрепления - определяет цель в процессе обучения с подкреплением. Это соответствие между воспринимаемыми состояниями среды и числом, подкреплением, показывающим присущую желательность состояния. Единственная цель агента состоит в максимизации итогового подкрепления, которое тот получает в процессе длительной работы. Функция отражает и определяет существо проблемы управления для агента. Она может быть использована как базис для изменения правил. Например, если выбранное действие повлекло за собой низкое подкрепление, правила могут быть изменены для того, чтобы в следующий раз выбрать другое действие. В общем случае, функция подкрепления может быть стохастической [7].

Краткая история обучения с подкреплением

В рамках искусственного интеллекта обучение с подкреплением начало развиваться с начала 80-х годов прошлого века и было связано с задачами оптимального управления и динамического программирования для построения систем управления. Но можно также утверждать, что обучение с подкреплением было в той или иной мере использовано в работах Ф. Розенблатта, а также Уидроу и Хоффа в начале 60-х годов прошлого столетия, а также в диссертационной и последующих работах М.Мински. В современном виде обучение с подкреплением сформировалось в конце 1980-х годов и связно с именами Барто и Саттона. Но основная идея «подкрепления» идет от экспериментальных работ обучения животных с точки зрения психологии [7]. Есть классический закон влияния, выраженный Эдвардом Торндайком еще в 1911 году[23], который сводится к следующему: «Среди нескольких различных откликов на одну и ту же ситуацию при прочих равных условиях те реакции, которые сопровождались поощрением животного, будут более тесно связаны с этой ситуацией и, следовательно, при ее повторении будут воспроизведены с наибольшей вероятностью. Реакции, которые связаны с дискомфортом, утрачивают свою связь с данной ситуацией и вряд ли будут возобновлены в ответ на ту же ситуацию. Чем выше уровень удовлетворения или дискомфорта, тем значительнее будет усиление или ослабление». Другими словами – чем чаще через нервную клетку проходит определенный сигнал, тем устойчивее реакция на него. Еще можно привести пример школьного обучения и обучения вообще, чтобы что-то запомнить, нужно повторить это несколько раз, что бы хорошо что - то делать, надо повторить данную работу несколько раз. Но это их «западная» точка зрения, наша же – все восходит к работе великого русского физиолога Павлова!

[23] Т.н. «Закон влияния» Торндайка.

Интересно использование термина «подкрепление» в работе небезызвестного Ф.Розенблата: «.. перцептрон всегда исследуется не изолированно, а как часть замкнутой экспериментальной системы, в которую помимо самого перцептрона, включается определенная окружающая среда и автоматическая система или экспериментатор, способные применить четко формулируемые алгоритмы преобразования или выработать «подкрепление» состояний памяти перцетрона … ». Т.е. еще в начале 60-х были попытки использовать принципы обучения с подкреплением с искусственными нейронными сетями.

Обычно выделяют два подхода к обучению с подкреплением [7, 10]:

1) Классический подход. В данном подходе процесс обучения происходит за счет поощрения и наказания («кнута и пряника») для достижения *высококлассного поведения*. Данный подход уходит корнями в давние работы по психологии, в частности Павлова – по условному рефлексу и Торндайка– обучение животных. Также к классическому подходу относят использование генетических алгоритмов для задач с подкреплением.

2) *Современный подход*. Данный подход основан на методе динамического программирования, используемом для формирования последовательности действий с учетом будущих состояний без фактического их осуществления. Также используются различные статистические техники. Данный подход будет подробно рассмотрен ниже. Учитывая, что методы динамического программирования требуют полного знания об объекте управления, то выглядит немного странным использование их в качестве ядра для построения алгоритмов обучения с подкреплением [7], но данные методы многошаговые и итерационные приводят к ответу постепенно, что и используется в алгоритмах обучения с подкреплением.

Но фундаментально можно выделить три класса методов обучения с подкреплением:

1). с использованием принципов динамического программирования (в т.ч. асинхронного);

2). на базе методов Монте – Карло;

3). на базе метода временных различий.

С другой стороны выделяют две основных принципа построения алгоритмов обучения с подкреплением – с использованием модели (*model - based*) без использования модели (*model - free*). Второй тип оценивает оптимальную политику без использования динамики окружающей среды, первый наоборот ее использует для оценки оптимальной политики.

Для задач, решаемых с помощью обучения с подкреплением характерно следующее:

- разные действия приводят к разным выигрышам;

- агент получает выигрыш с задержкой времени;

- выигрыш зависит от текущего состояния среды.

Одним из ключевых понятий, при построении систем, описанных ниже, является *окружающая среда* (внешняя среда). Окружающая среда может быть агрессивно настроенной по отношению к объекту управления, может быть «дружественной», но в общем случае она нейтральна. Обычно влияние окружающей среды на объект управления велико, а влияние объекта управления на окружающую среду крайне незначительно. В общем случае, для моделирования поведения внешней среды можно использовать методы Монте – Карло

При обучении с подкреплением запоминается соответствие между ситуациями и действиями, которые объект управления должен выполнить в той или иной ситуации. При обучении с учителем необходима выборка для обучения, в случае обучения с подкреплением начальная выборка зачастую не нужна – она появляется в ходе работы объекта. В результате проб и ошибок накапливается база знаний объекта об окружающей среде. Обучающая выборка генерируются автоматически в фазе, называемой «исследование» (разведка). Обычно - это случайный поиск в пространстве состояний (естественно, если пространство большое, то покрыть полностью его не возможно, но для этого обычно используют универсальный аппроксиматор – нейронную сеть). Обычно генерация обучающей выборки идет параллельно исследованию, поэтому обучение – возрастающее. Общая схема обучения с подкреплением показана на рисунке 4 [9, 10].

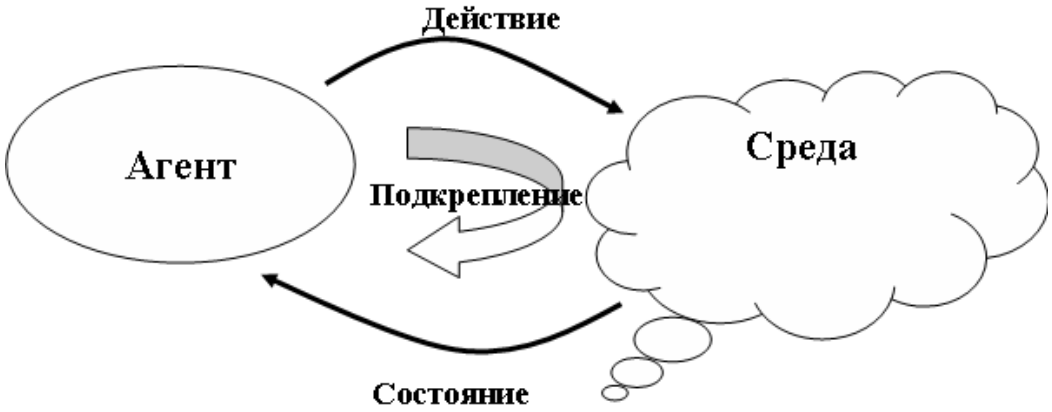


Рисунок 44. Обобщенная схема обучения с подкреплением

Агент и среда взаимодействуют на каждом из последовательности дискретных временных шагов, t = 0, 1, 2, 3, … . На каждом временном шаге, t, агент получает некоторое представление о *состоянии* среды, $s_t \in S$, где S это множество всех возможных состояний. На основе состояния агент выбирает

действие, $a_t \in A(s_t)$, где $A(s_t)$ это множество действий, возможных в состоянии s_t. Во время следующего шага, как часть ответа на действие, агент получает числовое *подкрепление,* $r_{t+1} \in R$, и переводит себя в состояние s_{t+1}.

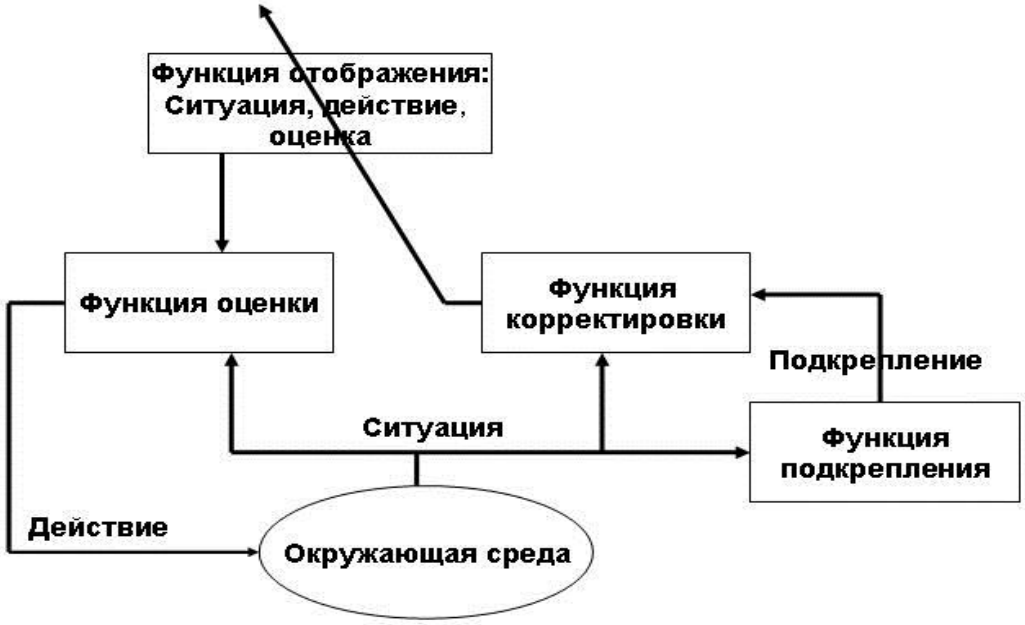


Рисунок 45. Принцип работы систем, использующих обучение с подкреплением

На каждом временном шаге, агент осуществляет отображение состояний на вероятности выбора каждого действия. Такое отображение называется *правилами* агента и обозначается π_t, где $\pi_t(s,a)$ вероятность, что $a_t = a$, если $s_t = s$. Методы обучения с подкреплением указывают, как агент изменяет свои правила в результате получения нового опыта. Цель агента максимизировать итоговую сумму подкрепления, полученную в течение долговременной работы.

Одним из ключевых моментов в обучении с подкреплением является собственно само *подкрепление* (англ. – reinforcement: *подъем, укрепление, подкрепление и др.*). *Подкрепление* – это некоторая оценка действий агента или объекта управления (обычно безразмерная). Иногда подкреплением называют *выгодой, вознаграждением или поощрением* [7]. В работах [8, 9, 10] вводится коэффициент эффективности, являющийся модернизированным подкреплением.

Приведем определение подкрепления из области психологии – «*Подкрепление – любое событие, стимул, действие, реакция или информация, которые, если следуют за реакцией, служат увеличению относительной частоты или вероятности возникновения этой реакции*» [3]. Классически – это

ассоциация, образующаяся, когда за операндной реакцией следует подкрепляющий стимул. В психологии также различают положительное и отрицательное подкрепление. В первом случае – это возникновение стимула, во втором, наоборот, исчезновение некоторого неприятного стимула или события[24]. Также различают прямое и косвенное, осознанное и неосознанное, намеренное и стихийное, вторичное и частичное, развое и системное и т.д.. В психологии и психиатрии существует несколько десятков видов работы с подкреплением.

Формально задачу расчета подкрепления (*вознаграждения*) можно поставить следующим образом – «*пусть приходится многократно выбирать одно из возможных действий (альтернатив), каждый выбор влечет за собой получение определенного вознаграждение. Целью последовательности действий является максимизация ожидаемого полного вознаграждения за заданный период времени. Например, за 1000 попыток выбора варианта действия*».

Задача объекта управления (или *агента*) – максимизировать суммарное вознаграждение по итогам своей работы. При этом согласно [7], оценка вознаграждения должна проходить максимально формально и ее расчет должен проходить вне[25] и без участия объекта управления. Т.е. объект управления, ни каким образом не должен влиять на расчет своего подкрепления.

Цель объекта управления может варьироваться от задачи к задаче, но стратегией объекта управления должна быть максимизация подкрепления (поощрения), при этом выбор стратегии может осуществляться различными способами. Обычно [7] выгода объекта управления (т.е. возможное будущее подкрепление) определяется как простая сумма вознаграждений:

$$R_t = r_{t+1} + r_{t+2} + ... + r_T$$

где T - завершающий временной шаг (т.н. *терминальный шаг*). Также, если после определенного шага идут только нулевые вознаграждения, то вводят т.н. *поглощающее состояние*, которое переходит только само в себя, при этом возможно, что $T = \infty$ или $\gamma = 1$ (но не одновременно!). Здесь важно отметить, что вообще говоря, задачи могут быть и без завершающего состояния, т.е. вышеприведенная формула пригодна лишь для задач с конечным состоянием. Следует отметить, что обычно процесс взаимодействия объекта управления со средой, может быть многократно повторен, т.е. для каждого состояния необходимо рассчитывать усредненное подкрепление – т.н. задача с

[24] Например, поднять зонт, чтобы перестать намокать под дождем.
[25] Понятно, что блок расчета, допустим, физически может находиться на борту мобильного робота и т.д.

эпизодами[26]. В тоже время взаимодействие может быть непрерывно и бесконечно, т.е. $T=\infty$ и получается, что возможное вознаграждение по первоначальной формуле может стремится к бесконечности. Для «обхода» данного момента вводится понятие *приведенной величины выгоды* (одна из трактовок *приведения*), которая рассчитывается как:

$$R_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + .. = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1}$$

где γ - *коэффициент приведения* $(0 \le \gamma \le 1)$. Суть применения приведенной величины – объект управления должен стараться выбирать такие действия, которые приводят к максимальному суммарному вознаграждении. Коэффициент приведения, иногда называемый *коэффициентом забывания*, имеет следующее свойство: если $\gamma = 0$, то объект управления старается максимизировать только текущее поступление подкрепления («близорукий» объект управления), т.е. выбирать те действия a_t, которые максимизируют только ближайшее вознаграждение r_{t+1}; если коэффициент равен единице, то мы возвращаемся к формуле ()[27]. При этом, если $\gamma < 1$, то сумма ряда будет ограниченной. Также отметим, что вознаграждение, которое будет получено через k итераций, будет отличаться от вознаграждения получаемого сейчас в γ^{k-1} раз [7] (при условии, что мы оперируем подкреплением равным 1 на каждом шаге). В общем случае, выбор текущего действия с максимальным вознаграждением не всегда приводит к максимальному итоговому вознаграждению.

Существуют две основные стратегии для реализации обучения с подкреплением с использованием нейронных сетей – Q-обучение и сети адаптивной критики. Об адаптивных критиках ниже, а сейчас рассмотрим Q – обучение в его базовой вариации.

Q – обучение

Основной принцип работы Q- обучения – это ассоциация функции подкрепления с каждой парой ситуация – действие, так называемая «таблица Q-обучения».

Основной принцип работы Q- обучения [5, 10] – это ассоциация функции подкрепления с каждой парой ситуация – действие. Схема Q – обучения, такая же, как в стандартном представлении обучения с подкреплением, единственное

[26] Эпизод – это либо разбивка на *подпоследовательности* взаимодействия объекта управления с внешней средой, либо полностью «проигранная» комбинация взаимодействия до терминального состояния. Обычно под эпизодом понимается первое. Примеры эпизода: партия в игре, однократное прохождение лабиринта и другие типы повторяющихся испытаний. Эпизоды также часто в литературе называют *испытаниями*.

[27] Также есть трактовка «чем агент дальше заглядывает в будущее, тем меньше у него уверенность в оценке награды (рубль сегодня стоит больше, чем рубль завтра)» [**Ошибка! Источник ссылки не найден.**].

отличие – в функции отображения (в данном случае она называется Q – функция), которая зависит от пары ситуация - действие. В Q – обучении присутствует 3 различных функции: память, исследование и корректировка. При ответе на существующую ситуацию, действие выбирается с помощью функции оценки с использованием памяти. После выполнения действия в реальной среде, функция подкрепления выдает значение подкрепления. Это значение $(\in[-1;1])$ используется функцией корректировки для приспосабливания значения оценки Q в соответствии с последними действиями объекта.

Алгоритм $Q-$ обучения строит Q – функцию, которая отображает пары ситуация – действие (i,a) в ожидаемое значение r. $Q(i,a)$ – это оценка системы. Алгоритм использует таблицы для хранения накопленной оценки Q. Непустые ячейки таблиц отражают уже испытанные значения пар.

Общий алгоритм Q – обучения следующий:

- инициализируется память: для всех значений пар ситуация – действие присваивается значение $Q(i,a)=0$, либо небольшое значение, чуть больше 0;
- цикл
  1. пусть i – есть ситуация среды;
  2. функция оценки выбирает действие a для выполнения: $a = Max(Q(i,a'))$, где a' – представляет любое возможное действие[28];
  3. объект отрабатывает действие a в среде на данной итерации, r – подкрепление (r может быть равно нулю) ассоциированное с выполнением действия a в среде;
  4. обновляется память по формуле (1.2):

$$Q_{t+1}(i,a) = Q_t(i,a) + \beta(r + g * Max(Q_t(i'a')) - Q_t(i,a))$$

где i' - новая ситуация после выполнения действия a в ситуации i, a' - любое возможное действие.

Существуют следующие варианты Q – обучения без применения искусственных нейронных сетей [5, 9, 10]:

- Q – обучение с использованием расстояния Хемминга;
- Q – обучение со статической кластеризацией.

Также приведем Q – обучение с применением нейронных сетей:

- Q – обучение с использованием многослойного персептрона;
- Q – обучение на базе нейронной сети Кохонена или самоорганизующихся карт Кохонена (SOM);
- Dyna – Q;

[28] Процесс выбора а’ может быть стохастическим, это позволяет исследовать новые области

- Competitive MLP - CMLP (на основе конкуренции между нейронными сетями).

У каждого варианта есть свой набор хорошо решаемых задач, но в большинстве случаев наилучшие результаты показывает CMLP (и это не удивительно, т.к. используется каскад нейронных сетей). Существует много плюсов при использовании нейронных сетей в Q – обучении, выделим две из них: качество обобщения (аппроксимации) и сравнительно небольшой размер памяти для хранения наборов «ситуация – действие – накопленное подкрепление», но, конечно, возникает вопрос по времени обучения нейронных сетей.

Часто для аппроксимации Q – таблицы используют нейронные сети – схематично показано на нижеприведенном рисунке.

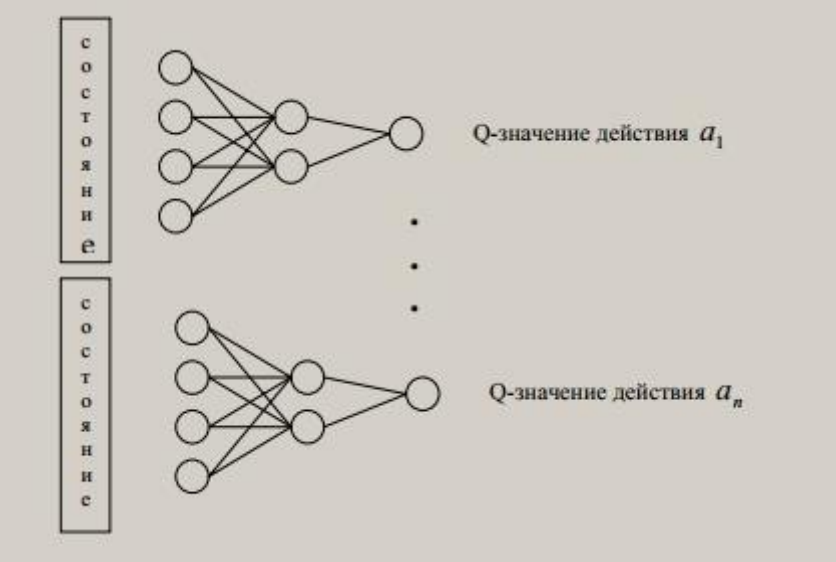


Рисунок 46. Нейросети для Q – обучения (автор – Кузьмин В.)

Литература

1. Prokhorov D., Santiago R., and Wunsch D. "Adaptive Critic Designs: A Case Study For Neurocontrol". Neural Networks, vol. 8, no. 9, pp. 1367-1372, 1995.
2. Sutton R.S., "Learning to Predict by the Methods of Temporal Differences" Machine Learning, vol. 3, pp. 9-44, 1988.
3. Thorndike E.L. Animal intelligence. Darien, CT: Hafner, 1911.
4. Watkins C. Learning from delayed rewards. Ph. D. thesis, Cambridge Univ., Cambridge, England, 1989.
5. Watkins C., Dayan P. Q – learning. // Machine Learning, vol. 8, pp. 279 – 292. 1992.

6. Беллман Р. Динамическое программирование. – М.: Иностранная литература, 1968. – 261 с.
7. Саттон, Барто. Обучение с подкреплением
8. Стасевич В. П., Шумков Е. А., Ключко В. И., Воротников С. А. Адаптивные системы на основе самообучающихся нейросетей // Труды КубГТУ. - 2002. - Вып.2. - С. 192 - 198.
9. Стасевич В.П. Анализ и адаптивное управление в недетерминированных средах на основе самообучения: дис. канд. техн. наук. – Краснодар, КубГТУ, 2007. – 170 с.
10. Шумков Е.А. Система поддержки принятия решений предприятия // дисс. канд. техн. наук. Краснодар: КубГТУ. 2004.
11. Хайкин С. Нейронные сети: полный курс. 2-е изд.: Пер. с. англ. – М.: Издательский дом «Вильямс», 2006 – 1104 с.
12. Бодянский Е.В., Руденко О.Г. Искусственные нейронные сети: архитектуры, обучение, применения. Харьков: ТЕЛЕТЕХ. 2004. 369 с.
13. Редько В.Г., Прохоров Д.В. Нейросетевые адаптивные критики //

ЛЕКЦИЯ 6.2. Q- обучение и Адаптивные критики

Q – Обучение (продолжение)

Существует несколько вариантов расчета подкрепления для Q – обучения. Рассмотрим их подробнее. Основной принцип Q – обучения (см.) – это накопление значений ценностей действий в различных ситуациях и последующий обоснованный их выбор. Базовым вариантом накопления ценностей действий является *средний выборочный* [19]. Пусть к моменту времени t действие a_j в ситуации s_i было выбрано k_{ij} раз и это привело к получению соответствующих вознаграждений $r_1, r_2, .., r_{k_{ij}}$. Тогда ценность этого действия в ситуации s_i можно определить как простое среднее:

$$Q_t(a_i) = \frac{r_1 + r_2 + .. + r_{k_{iji}}}{k_{ij}}$$

В случае, если $k_{ij} = 0$, т.е. данное действие еще не выбиралось, устанавливается значение по умолчанию, обычно равное нулю. В ходе работы системы с Q – обучением с расчетом по среднему выборочному логично выбирать, то действие в сложившейся ситуации, у которого накопленная ценность выше. Таким образом, всегда выбираются те действия, которые приводят к максимальной текущей ценности. Но, при этом выбираются только, те действия, которые уже были пройдены системой, т.е. нет реального

«исследовательского режима». У данного метода есть и плюсы – достаточно простой с минимумом вычислений, и минусы, нет реального поиска в пространстве состояние – действие.

Усложненным вариантом расчета ценности предстоящего действия в данном случае является использование т.н. «жадных»[29] стратегий.

Для понимания Q – обучения важным является матричное и графическое его представление. Обычно Q – матрицу, т.е. матрицу отображающую ценность пар (s_t, a_t) представляют следующим образом (Таблица 1). В таблице n - возможное количество состояний, а m - возможное количество действий (обычно $n \succ m$).

Таблица 1

Представление Q – таблицы (ситуация - действие)

	s_1	s_2	s_3	…	s_n
a_1	Q_{11}	Q_{12}	Q_{13}	…	Q_{1n}
a_2	Q_{21}	Q_{22}	Q_{23}	…	Q_{2n}
a_3	Q_{31}	Q_{32}	Q_{33}	…	Q_{3n}
				…	
a_m	Q_{m1}	Q_{m2}	Q_{m3}	…	Q_{mn}

Но не менее важной является таблица вероятностей перехода из одного состояния в другое (Таблица 2). Отметим, что объект управления может остаться в том же состоянии, что и был.

Таблица 2

Вероятности перехода системы из одного состояния в другое

	s_1	s_2	s_3	…	s_n
s_1	V_{11}	V_{12}	V_{13}	…	V_{1n}
s_2	V_{21}	…	…	…	…
s_3	V_{31}	…	…	…	…
…	…	…	…	…	…

[29] Англ. – greedy.

s_n	V_{n1}	…	…	…	…

Вероятности перехода из одного состояния в другое можно рассчитывать несколькими способами, …

Кроме того, в ряде задач возникает необходимость рассчитывать вероятности перехода из одного состояния в зависимости от выполняемого действия (см. Таблица 3).

Таблица 3

	s_1				s_2				…
	a_1	a_2	…	a_m	a_1	a_2	…	a_m	…
s_1	$V(S_1)_{11}$	$V(S_1)_{11}$		$V(S_1)_{1m}$	$V(S_2)_{11}$	$V(S_2)_{12}$		V_{1m}	…
s_2	$V(S_1)_{21}$								…
s_3	$V(S_1)_{31}$								…
…	…	…	…	…	…	…	…	…	…

Еще один вариант представления Q – таблицы: ведется таблица 4, в которой отражается накопленное подкрепление (значение функции ценности) при переходе из одного состояния в другое, но при этом есть еще одна таблица, в которой прописаны действия переводящие агента из одного состояния в другое. В некоторых случаях, если $S(t) = S(t+1)$ также может поступать подкрепление.

Таблица 4

Накопление подкрепления при переходе из одного состояния в другое

	S_1	S_2	…	S_n
S_1	**	R_{12}	…	R_{1n}
S_2	R_{21}	**	…	…
…	…	…	**	…
S_n	R_{n1}	R_{n2}	…	**

Таблица 5

Действия, переводящие из одного состояния в другое

$S(t)$	$a(t)$	$S(t+1)$
S_1	a_1	S_2
S_2	a_2	S_4
S_2	a_3	S_1
S_3	a_1	S_4
…	…	…

Понятно, что все таблицы могут иметь высокую размерность.

Достоинства Q – обучения (общие):

- нет необходимости строить внутреннюю модель среды;

- простой механизм выбора действия;

- малая требовательность к вычислительным ресурсам [**Ошибка! Источник ссылки не найден.**];

Общие недостатки Q – обучения:

- Q – таблицы при решении реальных практических задач могут иметь высокую размерность.

- тенденция к завышению оценок функции награды.

Кратко поясним отличие стандартного Q – Learning от модифицированного – в последнем используется оценка Q_{t+1}, которая связана с выбранным действием, а не максимальная, как в стандартном алгоритма. Это гарантирует, что ошибки временной разности будут рассчитываться правильно и не будут зависеть от того, как выбираются действия с «жадной» политикой и нет необходимости обнулять λ. Также здесь важным является то, что постоянный выбор действия с максимальной текущей оценкой часто приводит к тому, что итоговое значение подкрепления будет не максимальной, т.к. возможны пропуски дальнейших действий с более высокой оценкой.

В алгоритме $Q(\lambda)$ прогноз Q_t не присутствует в выражении корректировки Δw_t в явном виде, за исключением случая, когда действия выбираются в соответствии с «жадной» политикой[30]. Другая трактовка алгоритма – «данный метод базируется на выполнении обычного одношагового правила обновления

[30] Данный алгоритм можно интерпретировать, как взвешенную сумму урезанных возвратов [**Ошибка! Источник ссылки не найден.**].

для улучшения текущего прогноза Q_t и последующем использовании временной разности между следующими друг за другом `жадными` прогнозами». Т.е. в данном методе действие не выбирается с помощью `жадного` правила, но `жадное` правило присутствует на более высоком уровне.

Сети адаптивной критики

Адаптивные критики являются частью нескольких научных направлений, в частности: обучения с подкреплением и нейроуправления. Часто используется название «*исполнитель - критик*». Также адаптивные критики известны как *приближенное динамическое программирование* (англ. - *Approximated Dynamic Programming, ADP*).

Понятие критика (*critic*) было введено в работе [**Ошибка! Источник ссылки не найден.**]. *Критик* – это блок системы управления, который оценивает качество ее работы. Обучение в адаптивных критиках всегда строится по TD – методу. Название *критик* введено в связи с тем, что данный блок, ответственный за функцию ценности, критикует действия, осуществляемые исполнителем. Критика обычно принимает форму TD – ошибки и этот скалярный сигнал, являющийся единственным сигналом от критика, регулирует все обучение, как критика, так и исполнителя [**Ошибка! Источник ссылки не найден.**]. В основном критик реализует функцию ценности состояния и после каждого выбора действия критик производит оценку нового состояния на предмет улучшения и ухудшения по сравнению с тем, что ожидалось. Оценка считается, как TD – ошибка:

$$\delta_t = r_{t+1} + \gamma \cdot V(s_{t+1}) - V(s_t)$$

где V - текущая функция ценности, реализованная критиком.

Агент задает действия системы управления. Цель агента – максимизировать суммарную награду (подкрепление), которую можно получить в будущем в течение длительного периода времени. Агент оценивает суммарную награду с учетом коэффициента забывания:

$$U(t) = \sum_{k=0}^{\infty} \gamma^k \cdot r(t+k)$$

где $U(t)$ - оценка суммарной награды, γ - коэффициента забывания, $0 < \gamma < 1$. Коэффициент забывания означает следующее – чем дальше в будущее заглядывает агент, тем меньше у него уверенности в оценке награды.

Адаптивных критиков целесообразно использовать в тех случаях, когда не работает итеративная схема формирования матрицы $Q(S(t),a(t))$.

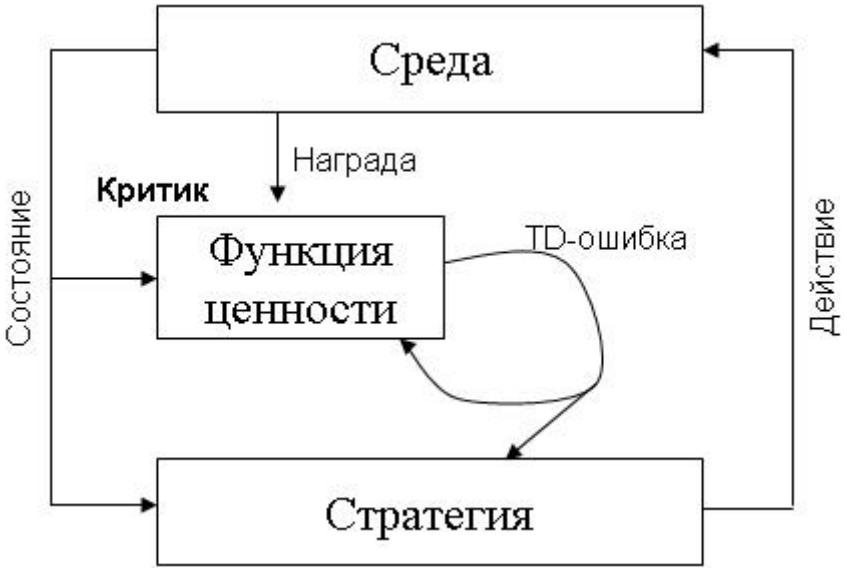


Рисунок 47. Обобщенная схема адаптивного критика

Для подробного рассмотрения современного подхода к обучению с подкреплением необходимо вначале рассмотреть принцип динамического программирования, который является ключевым (отправным моментом) при построении таких систем.

Отметим, что современный подход к обучению с подкреплением также называется *нейродинамическим программированием* [11], в связи с тем, что используется принцип динамического программирования с реализацией на нейронных сетях.

Принцип динамического программирования основан на идее Р. Беллмана [], известной, как *принцип оптимальности Беллмана* и представляет собой поэтапный метод принятия решений[31]. При этом перед принятием следующего решения последствия текущего решения предсказываются на некоторый интервал будущего. Напомним формулировку принципа оптимальности Р. Беллмана:

Оптимальная стратегия имеет следующее свойство - какими бы ни были начальные состояние и решение, остальные решения должны составлять оптимальную стратегию по отношению к состоянию, вытекающему из первого решения.

Напомним рекомендованную последовательность шагов в алгоритме динамического программирования [6, **Ошибка! Источник ссылки не найден.**]:

[31] В то же время, согласно ретроспективе, данной проблеме уделял много внимания русский ученый Марков, а Беллман предложил конкретное уравнение и удачную формулировку. Также, уравнение Беллмана является следствием теоремы Якоби – Гамильтона.

1. Выбрать параметры, характеризующие состояние S управляемой системы перед каждой итерацией.

2. Разбить операцию на этапы.

3. Определить набор шаговых управлений x_i для каждого шага и налагаемые на них ограничения.

4. Оценить, какой выигрыш приносит на i-м шаге управление x_i, если система до него была в состоянии S_{i-1}. Таким образом необходимо записать функции выигрыша:

$$w_i = f_i(S_{i-1}, x_i)$$

5. Определить, как изменяется состояние S системы под влиянием управления x_i на i - м шаге, то есть рассчитать:

$$S_{i+1} = \varphi_i(S, x_i)$$

6. Записать основное рекуррентное уравнение динамического программирования, выражающее условный оптимальный выигрыш $W_i(S)$ (с i-го шага и до конца), через уже известную функцию $W_{i+1}(S)$ [32]:

$$W_i(S) = \max\{ f_i(S, x_i) + W_{i+1}(\varphi_i(S, x_i))\}$$

7. Произвести условную оптимизацию последнего m-го шага, задаваясь возможными состояниями S, из которых можно за один шаг перейти в конечное состояние. При этом для каждого возможного состояния S вычисляется условный оптимальный выигрыш по формуле:

$$W_t(S) = \max\{ f_t(S, x_t)\}$$

и определить условное оптимальное управление $x_t(S)$, при котором этот максимум определяется.

8. Произвести условную оптимизацию $(t-1)$ - го шага, $(t-2)$ - го и других шагов по формуле (1.29), при этом полагая в ней $i = (t-1),(t-2),...$ и для каждого из шагов указать условное оптимальное управление $x_i(S)$, при котором максимум достигается.

9. Произвести безусловную оптимизацию управления на каждом шаге, то есть взять найденное управление на первом шаге[33] x_1^*, затем на втором x_2^* и так далее.

Задача динамического программирования может иметь конечный и бесконечный горизонт. Бесконечный горизонт может быть на самом деле конечным, но с очень большим количеством шагов. Общие ожидаемые затраты в задачах с бесконечным горизонтом определяются по формуле:

---

[32] Этому выигрышу соответствует условное оптимальное управление на i - м шаге.

[33] Обычно состояние системы в начальный момент известно и на первом шаге варьировать состояния системы не нужно – необходимо сразу взять оптимальный выигрыш на первом шаге.

$$J^{\pi}(i) = E[\sum_{n=0}^{\infty} \gamma^n \cdot g(X_n, \mu_n(X_n), X_{n+1}) \mid X_0 = i]$$

где $X_0 = i$ - начальное состояние, $\pi = \{\mu_n\}$ - стратегия поведения. Ожидаемое значение затрат вычисляется по цепи Маркова $\{X_1, X_2, X_3, ...\}$. Функция $J^{\pi}(i)$ называется функцией стоимости перехода для стратегии π, начиная с шага i (англ. *cost – to – go - function*). Оптимальное значение функции затрат (в смысле минимума) $J^*(i)$ определяется соотношением:

$$J^* = \min_{\pi} J^{\pi}(i)$$

Решение уравнения оптимальности Беллмана является одним из подходов к поиску оптимальной стратегии, в частности для задачи обучения с подкреплением. Но для задач с подкреплением принцип оптимальности Беллмана применяется не часто по следующим причинам:

- метод динамического программирования требует точной и полной модели окружающей среды, что накладывает определенные ограничения для его использования для большого круга задач, решаемых с помощью обучения с подкреплением
- необходим достаточный вычислительный ресурс;
- сигнал состояния должен обладать марковским свойством (окружающая среда должна описываться как конечный МППР).

Задачи редко удовлетворяют трем вышеперечисленным условиям и поэтому приходится работать с приближенными решениями.

Нейродинамическое программирование позволяет системе обучаться принятию правильных решений с помощью наблюдения за собственным поведением и совершенствовать свое поведение путем использования встроенного механизма усиления.

В задачах нейродинамического (и динамического) программирования существуют две серьезные проблемы:

1) *Проклятие размерности*. В большинстве реальных задач количество возможных состояний и действий в них является настолько большим, что их во – первых трудно все учесть, а во – вторых возникают серьезные требования к вычислительным мощностям. Например, при игре в нарды имеется 10^{20} возможных состояний. Вообще говоря, в задаче динамического программирования содержащей K возможных состояний и M возможных действий в них, каждая итерация алгоритма требует $K^2 \cdot M$ операций для стационарной стратегии [11].

2) *Неполнота информации*. Алгоритмы итераций по стратегиям и значениям требуют знания вероятностей переходов между состояниями p_{ij} и стоимости этих переходов. Но такие данные зачастую в реальных задачах априори неизвестны.

Поэтому в реальных задачах приходится отказываться от оптимальных стратегий и искать субоптимальные.

Для поиска субоптимальных стратегий обычно применяются нейронные сети, так как они являются одними из лучших аппроксиматоров. При этом, например, в вводят обучение с подкреплением, как подраздел обучения без учителя. В случае нейродинамического программирования нейронная сеть аппроксимирует функцию стоимости перехода $J^*(i)$ для $\forall i \in X$. В частности, для конкретного i функция затрат $J^*(i)$ заменяется подходящей аппроксимацией $\hat{J}(i,w)$, где w - вектор параметров. Функция $\hat{J}(\cdot,\cdot)$ называется приближенной функцией стоимости перехода (см. рисунок).

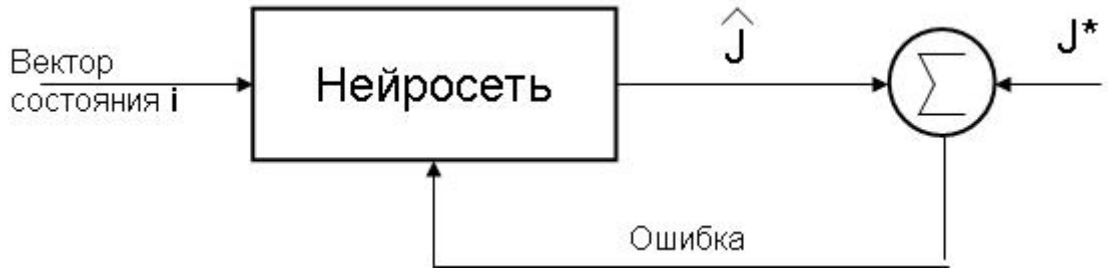


Рисунок 48. Нейросеть в качестве аппроксиматора функции затрат

Заметим, что в ряде работ, например, в [] на вход нейросети подается только номер состояния i, что на взгляд автора в реальных задачах встречается редко, в основном такое описание задач встречается в учебных примерах. Необходимо подавать вектор, описывающий текущее состояние i.

В реальных задачах также зачастую необходимо идентифицировать текущее состояние и для этого, как раз необходим вектор, описывающий текущее состояние объекта управления. Для данной задачи, в случае конечного количества состояний, прекрасно подходит сеть Кохонена.

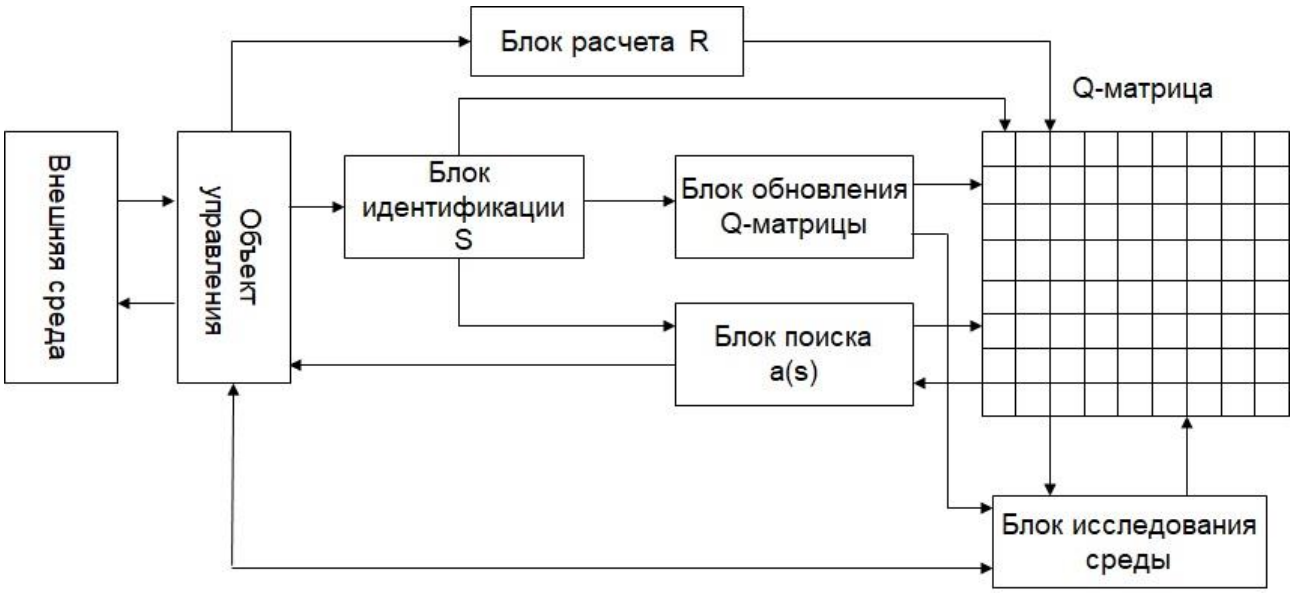


Рисунок 49. Один из вариантов схемы с Q-обучением

Литература:

14. Prokhorov D., Santiago R., and Wunsch D. "Adaptive Critic Designs: A Case Study For Neurocontrol". Neural Networks, vol. 8, no. 9, pp. 1367-1372, 1995.
15. Sutton R.S., "Learning to Predict by the Methods of Temporal Differences" Machine Learning, vol. 3, pp. 9-44, 1988.
16. Watkins C. Learning from delayed rewards. Ph. D. thesis, Cambridge Univ., Cambridge, England, 1989.
17. Watkins C., Dayan P. Q – learning. // Machine Learning, vol. 8, pp. 279 – 292. 1992.
18. Беллман Р. Динамическое программирование. – М.: Иностранная литература, 1968. – 261 с.
19. Саттон, Барто. Обучение с подкреплением
20. Стасевич В. П., Шумков Е. А., Ключко В. И., Воротников С. А. Адаптивные системы на основе самообучающихся нейросетей // Труды КубГТУ. - 2002. - Вып.2. - С. 192 - 198.
21. Стасевич В.П. Анализ и адаптивное управление в недетерминированных средах на основе самообучения: дис. канд. техн. наук. – Краснодар, КубГТУ, 2007. – 170 с.
22. Шумков Е.А. Система поддержки принятия решений предприятия // дисс. канд. техн. наук. Краснодар: КубГТУ. 2004.
23. Хайкин С. Нейронные сети: полный курс. 2-е изд.: Пер. с. англ. – М.: Издательский дом «Вильямс», 2006 – 1104 с.

24. Бодянский Е.В., Руденко О.Г. Искусственные нейронные сети: архитектуры, обучение, применения. Харьков: ТЕЛЕТЕХ. 2004. 369 с.
25. Редько В.Г., Прохоров Д.В. Нейросетевые адаптивные критики //

ЛЕКЦИЯ 6-3. Адаптивные критики (продолжение) и глубокое обучение с подкреплением

Адаптивные критики (продолжение)

Алгоритм SARSA

Рассмотрим простой алгоритм адаптивного критика, и один из первых по публикации – SARSA[34]. Данный алгоритм применяется в том случае если множество возможных ситуаций и действий конечно. В алгоритме SARSA существует следующая цепочка:

$$S(t) \to a(t) \to r(t) \to S(t+1) \to a(t+1)$$

В данном алгоритме пошагово (итеративно) вычисляется оценка величины суммарной награды $Q(S(t), a(t))$, которую получит Агент, если в ситуации $S(t)$ он выполнит действие $a(t)$. Математическое ожидание награды вычисляется по формуле [**Ошибка! Источник ссылки не найден.**]:

$$Q(S(t), a(t)) = E\{r(t) + \gamma \cdot r(t+1) + \gamma^2 \cdot r(t+1) + ...\} \mid S = S(t), a = a(t)$$

Из (3.2) и (3.3) следует:

$$Q(S(t), a(t)) = E\{r(t) + \gamma \cdot Q(S(t+1), a(t+1))\}$$

Ошибка определяется как ОВР:

$$\delta(t) = r(t) + \gamma \div Q(S(t+1), a(t+1)) - Q(S(t), a(t))$$

где δ - есть ошибка временной разности.

На каждой итерации работы системы происходит как выбор действия, так и обучение агента (вернее сказать – переобучение или, в зависимости, от типа нейронной сети - дообучение). Действие в момент времени t выбирается с максимальным значением $Q(S(t), a_j)$ с вероятностью $(1-\xi)$:

$$a(t) = \arg(\max_a \{Q(S(t), a_j)\})$$

[34] Название SARSA собственно и произошло от сокращения данной цепочки событий (s → a → r → s → a)

где $0 < \xi << 1$. То есть выбор действия происходит с помощью «ξ - жадного правила» .

Переоценка $Q(S,a)$ производится в соответствии с оценкой ошибки $\delta(t)$ - к значению $Q(S(t),a(t))$ добавляется величина, пропорциональная ошибке временной разности:

$$\Delta Q(S(t),a(t)) = \alpha \cdot \delta(t) = \alpha \cdot [r(t) + \gamma \cdot Q(S(t+1),a(t+1)) - Q(S(t),a(t))]$$

где α - параметр скорости обучения.

Приведем полную последовательность действий алгоритма SARSA:

1. Инициализировать произвольно $Q(s,a)$

2. Для каждого эпизода повторять:

   2.1. Инициализировать s

   2.2. Выбрать a по s, используя стратегию, полученную из Q (можно ε-жадную)

   2.3. Повторять для каждого шага эпизода, пока s не станет завершающим состоянием:

      2.3.1. Выполнить действие a, найти r, s'.

      2.3.2. Найти a' по s', на базе стратегии, полученной из Q (можно ε-жадную)

      2.3.3. Рассчитать и обновить значения по формулам:

      $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot Q(s',a' - Q(s,a))]$

      $s \leftarrow s'$, $a \leftarrow a'$

Так как количество ситуаций и возможных в них действий конечно, то в алгоритме SARSA происходит формирование матрицы $Q(S_j, a_i)$ со всеми возможными ситуациями и действиями. Оценку суммарной награды $Q(S,a)$ можно рассматривать как оценку качества действия $a(t)$ в ситуации $S(t)$

Пример реализации алгоритма SARSA на нейронных сетях показан на слайдах к лекции.

Другими популярными вариантами реализации адаптивных критиков являются:

Q – критик;

V – критик;

В схемах Q – критиков блок критики делает оценку величины суммарной награды $Q(S(t),a(t))$, которую агент ожидает получить в будущем, если он в

данной ситуации $S(t)$ выполнит действие определенное действие $a(t)$. Т. е. происходит оценка качества того или иного действия в известной ситуации (аналогично SARSA) []. Функционирование Q – критика происходит следующим образом. В момент времени t решатель по вектору текущего состояния $S(t)$ определяет вектор действия $A(t)$, далее выполняется действие $A(t)$, объект управления получает награду $r(t)$. При этом параллельно на Критик подаются векторы $A(t)$ и $S(t)$, по которым Критик производит оценку качества $Q(t) = Q(S(t), A(t))$ действия $A(t)$ в текущей ситуации $S(t)$. После этого объект управления переходит в состояние $S(t+1)$. Момент времени системы инкрементируется $t+1$. Далее все операции повторяются и вычисляется оценка значения $Q(t+1)$, после вычисления которой происходит расчет ошибки временной разности:

$$\delta(t) = r(t) + \gamma \cdot Q(t+1) - Q(t)$$

После накопления определенного количества примеров выполняется обучение нейронных сетей Критика и Решателя, обычно методом обратного распространения ошибки. Корректировку весов синаптических связей Критика и Решателя можно записать следующим образом, согласно:

$$\Delta W_C = \alpha_1 \cdot \nabla_{WC}(Q(t))\delta(t)$$

$$\Delta W_A = \alpha_2 \sum_K \{\frac{\partial Q(t)}{\partial A_K(t)} \cdot \nabla_{WA} A_K(t)\}$$

В топологиях V – критиков блок Критики делает оценку качества ситуации $V(S(t))$, то есть оценку ожидаемой суммарной награды в данной ситуации. При этом топология дополняется блоком прогноза (см. Рисунок в презентации) и система управления стремится выполнять те действия, которые, согласно прогнозу, приведут к ситуациям $S(t+1)$ с наибольшими оценками $V(S(t+1))$.

Также существуют более сложные топологии адаптивных критиков.

Стоит отметить, что Модель и Критик обычно реализуют на нейронных сетях. В случае нейросетевой реализации параметрами аппроксимирующих функций являются веса синапсов нейронной сети, а оптимизация обычно производится путем подстройки весов, например методом обратного распространения ошибки.

В случае V - критика оценивается ошибка следующей ошибки временной разности:

$$\delta(t) = r(t) + \gamma \cdot V(S^{PR}(t+1)) - V(S(t))$$

Критик обучается за счет подстройки весов:

$$\Delta W_C = \alpha \cdot grad_{WC}(V(t)) \cdot \delta(t)$$

В случае реализации Модели на нейронной сети задача прогнозирования решается стандартным образом.

Сети адаптивной критики обладают несомненными достоинствами, такими как:

- старт работы системы, как без знания истории, так и без знания окружающей среды;
- последовательное, пошаговое исследование среды;
- ядро на базе нейронной сети способно аппроксимировать большое пространство состояний – действий.

Но кроме достоинств, сети адаптивной критики обладают и недостатками:

- переобучение нейронных сетей Критика и Агента требуют достаточно продолжительного времени;
- сложность разработки математической модели Критика для реальных задач;

негарантированное поступление подкрепления.

Примеры применения адаптивных критиков

Задача об автотрейдере

Рассмотрим, как вариант, структуру построения МТС, работающую на базе прогнозирования будущего курса пары валют (или акции) с использованием обучения с подкреплением [1, 2, 3]. Автотрейдер должен отслеживать изменение ряда котировок и либо выдавать сигнал на совершение торговой операции, либо непосредственно совершать ее. Отметим, что торговые операции могут быть составными с использованием отложенных ордеров, стоп – приказов, слежением и т.д. Одним из основных инструментов анализа финансовых рынков являются технические индикаторы (далее ТИ), также на них часто строят автотрейдеры. Покажем, как для Q-OLAP (технология изпользующая Q-обучение и OLAP таблицы) использовать ТИ. Состоянием в данном случае логично выбрать показания ТИ, например, для MACD – это будут расстояние между сигнальной и основной линией, а также количество временных итераций с момента пересечения линий. Действиями, в простейшем случае, будут операции продажи и покупки, а также закрытие открытых позиций. Целесообразно также ввести еще одно измерение – это параметры ТИ, которые довольно сложно подобрать. Фактами в ячейках будут показатели автотрейдера, например, отношение прибыль к убыткам, количество прибыльных сделок, количество убыточных сделок, относительная просадка, матожидание выигрыша и т.д.. Таким образом, мы получаем новый механизм, который использует в качестве Q - таблицы современный механизм OLAP, а

также на эту почву прекрасно ложится вариант создания автотрейдера на техническом индикаторе.

Рисунок 50. Обобщенная структура МТС

В механической торговой системе должно быть три параллельно работающих контура: отслеживания новостей и макроиндикаторов, контур поиска и тестирования, а также контур рабочей торговой системы. При этом контур поиска и тестирования может быть разделен на два отдельных контура [1].

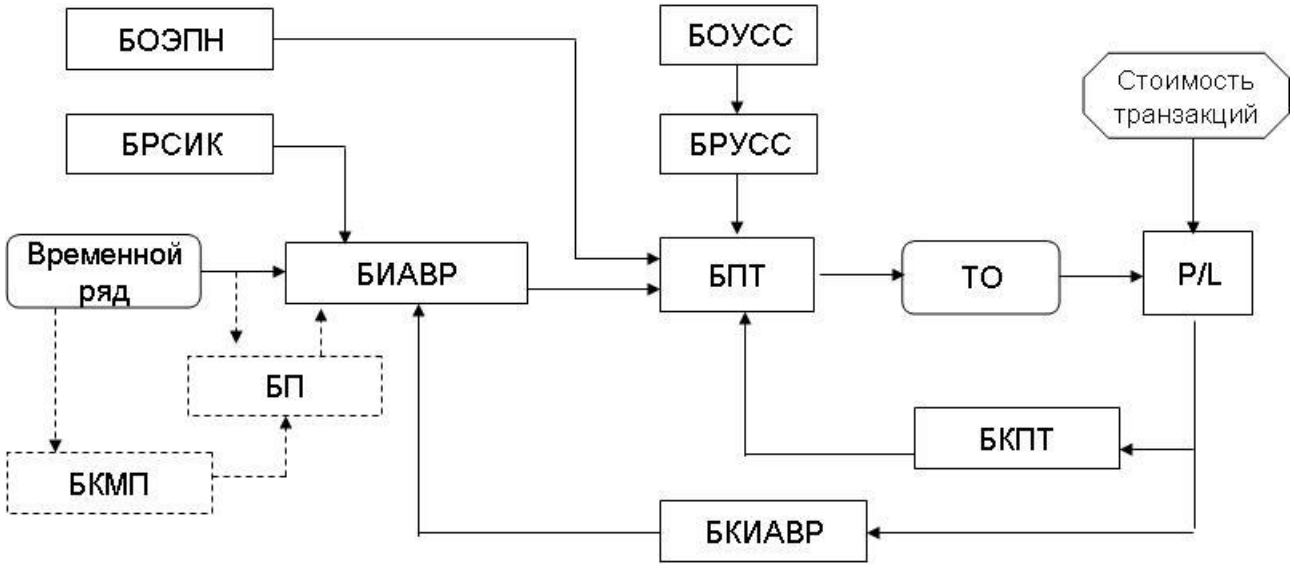
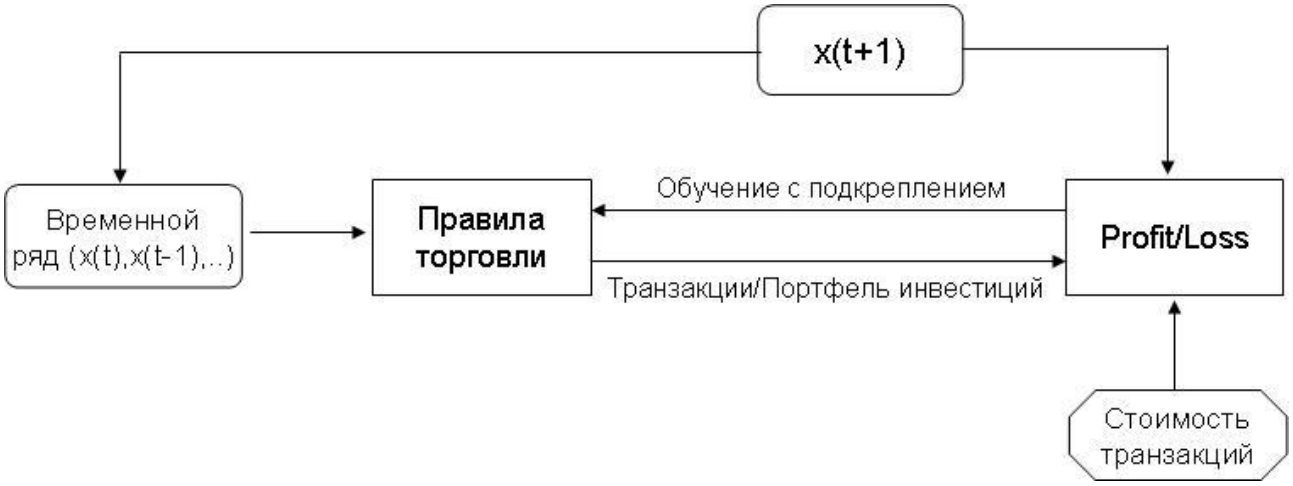


Рисунок 51. Расширенная структура МТС

Теперь покажем, как выглядит структура МТС с использованием обучения с подкреплением.



Применение в сфере здравоохранения

В сфере здравоохранения пациенты могут получать лечение в соответствии с политиками, полученными из систем с подкреплением (но здесь, конечно, важна ретроспективная выборка). С помощью обучения с подкреплением можно найти оптимальную политику, например, лечения пациента в зависимости от диагноза, используя предыдущий опыт, но не используя информацию о математической модели самого процесса лечения и физиологии человека (хотя … их и так трудно построить). Выделяют даже процесс режима динамического лечения (DTR) при хронических заболеваниях или интенсивной терапии, автоматизированной медицинской диагностике и других общих областях.

Применение глубокого обучения с подкреплением в коллаборативных системах

Пользовательские предпочтения могут часто меняться, поэтому рекомендации пользователям новостей на основе обзоров и т.н. лайков могут быстро устареть. С помощью обучения с подкреплением система может отслеживать ответное поведение читателя и в режиме он-лайн корректировать выдачу результатов. Построение такой системы будет включать получение функций новостей, функций чтения, функций контекста и функций чтения новостей. На основе поведения пользователя определяется вознаграждение. При этом, конечно, здесь конкретный пользователь не важен – он просто «один из коллаборативной выборки».

Обучение с подкреплением в маркетинговых задачах

Есть работы по моделированию торгов в реальном времени с мультиагентным обучением с подкреплением (multiagent reinforcement learning). Работа с большим количеством рекламодателей решается с использованием метода кластеризации и назначения каждому кластеру т.н. «шаблонного» агента по торгам. Чтобы уравновесить компромисс между конкуренцией и сотрудничеством между рекламодателями (а это уже из теории игр), предлагается распределенное согласованное назначение ставок с участием нескольких агентов (англ. DCMAB). В маркетинге очень важно умение точно ориентироваться на отдельного человека. Исследование в этой статье было проведено на Taobao - крупнейшей платформе электронной коммерции в Китае (после Alibaba). Предлагаемый метод по данным из публикаций превосходит классические подходы к данной проблеме.

Обучение с подкреплением в робототехнике

Возможно одно из самых первых и популярных задач на применение обучения с подкреплением. Использование глубокого обучения и обучения с подкреплением позволяет обучать роботов, способных захватывать различные объекты - даже те, которые не известны во время обучения (здесь, правда, важен момент идентификации опасности). Это можно, например, использовать при сборке узлов агрегатов на сборочной линии РТК. Это достигается за счет объединения крупномасштабной распределенной оптимизации и варианта глубокого Q-обучения (QT-Opt). Поддержка QT-Opt для пространств непрерывного действия делает его подходящим для задач робототехники. Модель сначала обучается в автономном режиме, а затем развертывается и настраивается на реальном производстве.

Глубокое обучение с подкреплением

Есть несколько подходов к пониманию «глубокого обучения с подкреплением», в частности: «Глубокое обучение и обучение с подкреплением - это системы, которые обучаются автономно. Разница между ними в том, что глубокое обучение - это обучение на основе обучающего набора с последующим применением этого обучения к новому набору данных, в то время как обучение с подкреплением - это динамическое обучение путем корректировки действий на основе непрерывной обратной связи для максимизации вознаграждения», т. е. «глубокое обучение с подкреплением» есть «что – то» на пересечении этих двух направлений искусственного интеллекта.

Но на самом деле под «глубоким обучением с подкреплением» необходимо понимать использование сверточных нейронных сетей в топологии с подкреплением. Здесь, чтобы соответствовать названию, главное наличие сверточной нейронной сети. Неважно какой блок и какой алгоритм будет с ее помощью реализован: критик, решатель, аппроксиматор Q-таблицы или что-то другое. Но, если серьезно, то для серьезных задач, допустим, для автопилота автомобиля, где требуется а) распознавание дорожной ситуации и б) принятие решения по управлению автомобилей, действительно необходимо использование сложных нейросетевых структур и сверточных («глубоких») нейронных сетей, в частности, потому - что другие методы и алгоритмы либо просто не работают на таких данных, либо дают результат ниже необходимого. В то же время и работа с глубокими нейронными сетями в данных задач находится еще в начале пути.

Можно также дать такую формулировку глубокого обучения с подкреплением: «Глубокое обучение с подкреплением сочетает в себе искусственные нейронные сети со структурой обучения с подкреплением, которая помогает программным агентам узнать, как достичь своих целей. То есть он объединяет аппроксимацию функций и оптимизацию целей, сопоставляя состояния и действия с вознаграждениями, к которым они приводят».

В обучении с подкреплением сверточные сети могут использоваться для распознавания состояния агента, когда ввод является изображением или видеопотоком.

Также в обучении с подкреплением, учитывая изображение, которое представляет состояние, сверточная сеть может ранжировать действия, которые можно выполнить в этом состоянии. Для большей конкретики, Q - функция сопоставляет пары состояние-действие с наибольшим значением (немедленного) вознаграждения со всеми будущими вознаграждениями, которые могут быть получены более поздними действиями по выбранной политике действий объекта управления.

В качестве примера приведем возможные варианты реализации (но лучше сказать - трактовки) глубокого Q-обучения[35].

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Рисунок 52. Пример, расчета Q-функции

Глубокое Q – обучение

Основная идея глубокого Q – обучения (Deep Q-Learning) состоит в том, чтобы адаптировать алгоритм ошибки временной разности таким образом, чтобы обновление для формулы стандартного обновления Q-таблицы будет эквивалентна шагу градиентного спуска при обучении нейронной сети.

На взгляд автора, что такое «deep Q-learning» можно ответить следующей схемой из сети Интернет

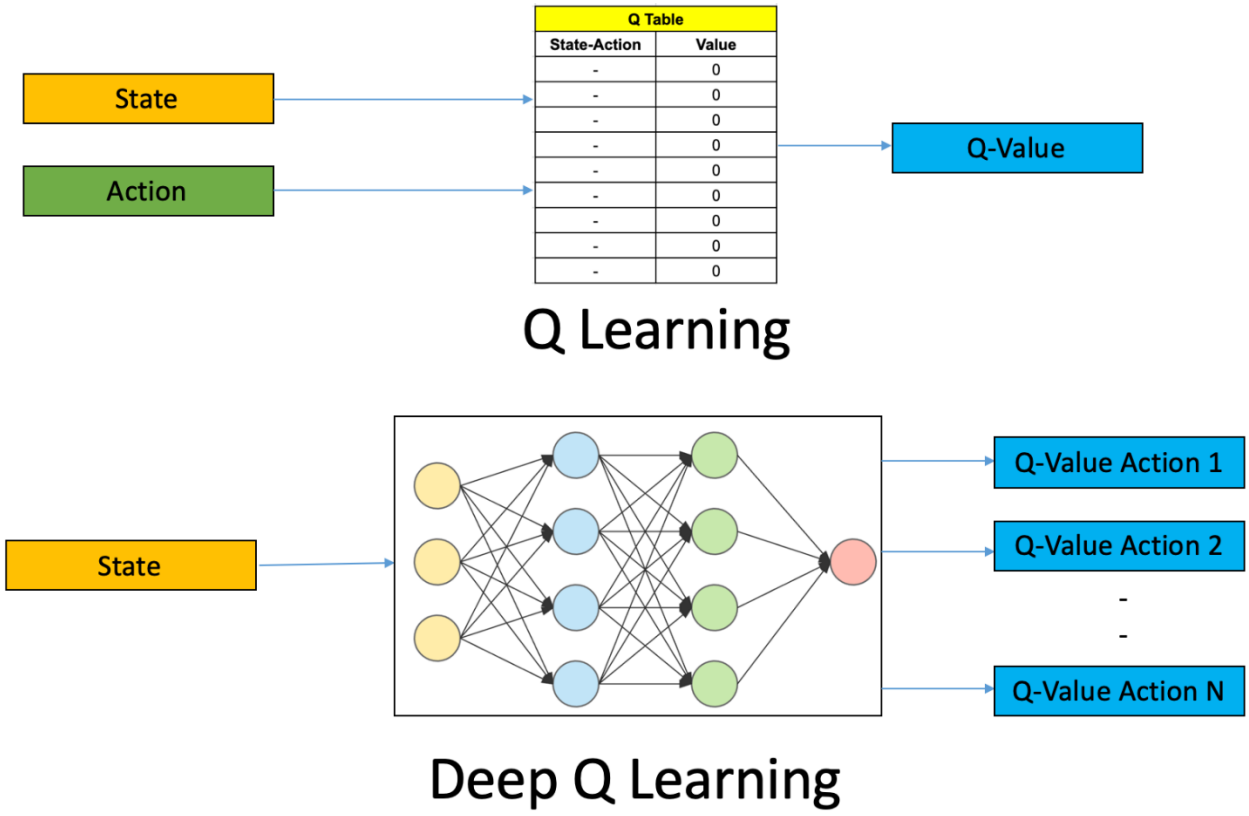


Рисунок 53. Глубокое Q-обучение

[35] Здесь исследования находятся в самом начале пути

Опять же можно сказать, что это «ребрендинг» (но можно сказать, что это не понимание…). Данную схему можно найти, например, в работе В.Кузьмина из Латвии, датированную 2003 годом…

Здесь интересным выглядит момент именно использования сверточных нейронных сетей для «понимания» (обработки) Q – таблиц для равных уровнях детализации задачи. Здесь хорошие результаты показывает алгоритм Double Q-Learning и Noisy DQN (см. слайды).

И в окончании – на данный момент, наилучшие результаты т.н. глубокое обучение с подкреплением показывает в эмуляции видеоигр...

Литература:

1. Шумков Е.А. "Механические торговые системы" // Сетевой электронный научный журнал КубГТУ. №4, 2017
2. Шумков Е.А., Чистик И.К. Автотрейдер с использованием Q-обучения // "Математические методы и информационные технологии в экономике, социологии и образовании". Сборник статей XXX Международной научно - технической конференции. - Пенза: Приволжский Дом знаний. 2012. С. 126-127.
3. Шумков Е.А. Структуры механических торговых систем // "Прикладная информатика". М: 2012, №3