МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
Факультет э***лектротехнический***
Кафедра ***автоматики***
Направление подготовки ***27.03.04 «Управление в технических системах»***

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____
_____ Голощапов С.С._____
_____2017 г.

НЕЙРОННАЯ СЕТЬ ДЛЯ ПОСТРОЕНИЯ ЛОКАЛЬНОЙ СИСТЕМЫ УПРАВЛЕНИЯ
(тема)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ (ПРОЕКТУ)
ЮУрГУ – 27.03.04.2017.217.00.00 ПЗ ВКР

Автор проекта
студент группы _МиЭт-428_____

_____/___Пахотин Ю. В._____
подпись ФИО
_____2017 г.

Руководитель проекта
_____Старший преподаватель_____
должность
_____/__ Елисеев В.П._____
подпись ФИО
_____2017 г.

Нормоконтроль

_____Старший преподаватель_____
должность
_____/__Елисеев В.П._____
подпись ФИО
_____2017 г.

Миасс 2017

Содержание

1 Вводная часть

В настоящие время для построения информационных систем применяются управляющие устройства, имеющие интеллект. Основой данных устройств составляет аппаратная составляющая основанная на архитектуре фон Неймана. Последние 10 ~ 15 лет идет интенсивное развитие нового направления в управляющих системах для решения неформализованных задач применяющая теорию нечетких множеств.

Цель данного дипломного проекта является исследование возможного построения нейронной сети с использованием стандартах утилит операционной системы в частности командной оболочки Bourne shell (BASH). Что же такое нейронные сети?

Нейронные сети, или, точнее, искусственный интеллект, возник на основе подражания процессам обработки информации и принятий решений, происходящих в живых организмах. Одним из наиболее важных свойств, является способность обучатся на основе представленных данных. Именно поэтому они находят свое применение в таких разнородных областях, как моделирование, анализ временных рядов, распознавание образов, обработки сигналов и управления. Тем самым они уходят корнями во множество дисциплин: нейрофизиологию, математику, статистику, физику, компьютерные науки и технику.

2 Основы искусственной нейронной сети

2.1 Основы нечеткой математики.

Аппарат нечеткой математики позволяет формализовать нечеткие понятия и знания, оперировать этими знаниями и делать нечеткие выводы. Основу этого аппарата составляет математическая теория нечетких множеств, предложенная американским ученным Л. Заде четверть века назад. Как следует из названия, эта теория предполагает неточные, неполные, приблизительные оценки объектов, ситуаций, явлений. Необходимость использования такого подхода вызвана следующими обстоятельствами:

- при решении некоторых проблем не нужна точная оценка параметров объектов и явлений;
- по утверждению Л. Заде с ростом сложности системы постепенно падает способность человека делать точные и в то же время значащие утверждения относительно ее поведения, пока не будет достигнут порог, за которым точность и значимость становятся взаимоисключающими характеристиками.

Наиболее существенные особенности моделей реальных систем, построенных с использованием аппарата нечеткой математики, состоят в следующем:

1) большая гибкость по сравнению с традиционными четкими, так как они позволяют описывать знания и опыт человека в привычной для него форме;

2) большая адекватность реальному миру, поскольку позволяют получить решения, по точности соотносимое с исходными данными;

3) возможность в ряде случаев более быстрого получения окончательного результата, чем на «точных» моделях, в силу специфического построения и простоты используемых нечетких операций.

Нечеткие методы, основанные на теории нечетких множеств, характеризуются тремя отличительными чертами:

- использованием так называемых лингвистических переменных вместо числовых переменных или в дополнение к ним;
- простые отношения между переменными описываются с помощью нечетких высказываний;
- сложные отношения описываются нечеткими алгоритмами.

2.2 Биологическая структура нейрона

Нейрон является элементом клеточной структуры головного мозга. В своем строении он имеет много общего с другими клетками. Несмотря на это, нервная клетка существенно отличается от иных по своему функциональному назначению. Нейрон выполняет прием, элементарное преобразование и дальнейшую передачу информации другим нейронам. Информация переносится в виде импульсов нервной активности, имеющих электрохимическую природу.

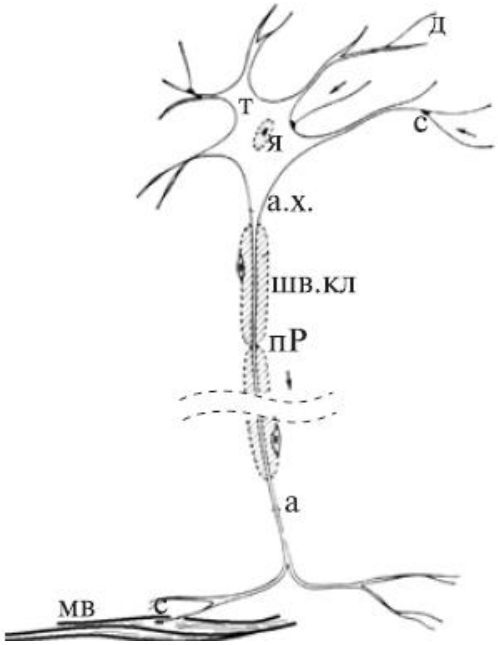

Рисунок 1 - Биологический нейрон

Биологический нейрон содержит следующие структурные единицы (Рисунок 1):

- Тело клетки (т) - содержит ядро (я).
- Дендриты (д) - собирают информацию от других нейронов.
- Мембрана - обеспечивает проведение нервных импульсов.
- Аксон (а) - выходное нервное волокно клетки, обеспечивающий проведение импульса и передачу воздействия на другие нейроны или мышечные волокна (мв).
- Синапс (с) - место контакта нервных волокон - передает возбуждение от клетки к клетке.

2.3 Электрические сигналы

В реальной биологической нейронной сети от входов сети к выходам передается электрический сигнал. В процессе жизнедеятельности нейронной сети сигналы, проходящие по нейрону, могут изменятся. Меняются величины этих электрических сигналов (становится сильнее или слабее). А значит любую величину всегда можно выразить числом (больше/меньше).

В нашей модели искусственной нейронной сети нам совершенно не нужно реализовывать поведение электрического сигнала, так как от его реализации все равно ничего зависеть не будет.

На входы сети мы будем подавать какие-то числа, символизирующие величины электрического сигнала, если бы он был. Эти числа будут продвигаться по сети и каким-то образом меняться. На выходе сети мы получим какое-то результирующее число, являющееся откликом сети.

Для удобства все равно будем называть наши числа, циркулирующие в сети, сигналами.

2.4 Синапсы - веса связей

Синапс, как упоминалось ранние место контакта нервных волокон (место связи нервных волокон). Он может усиливать или ослаблять проходящий по ним электрический сигнал.

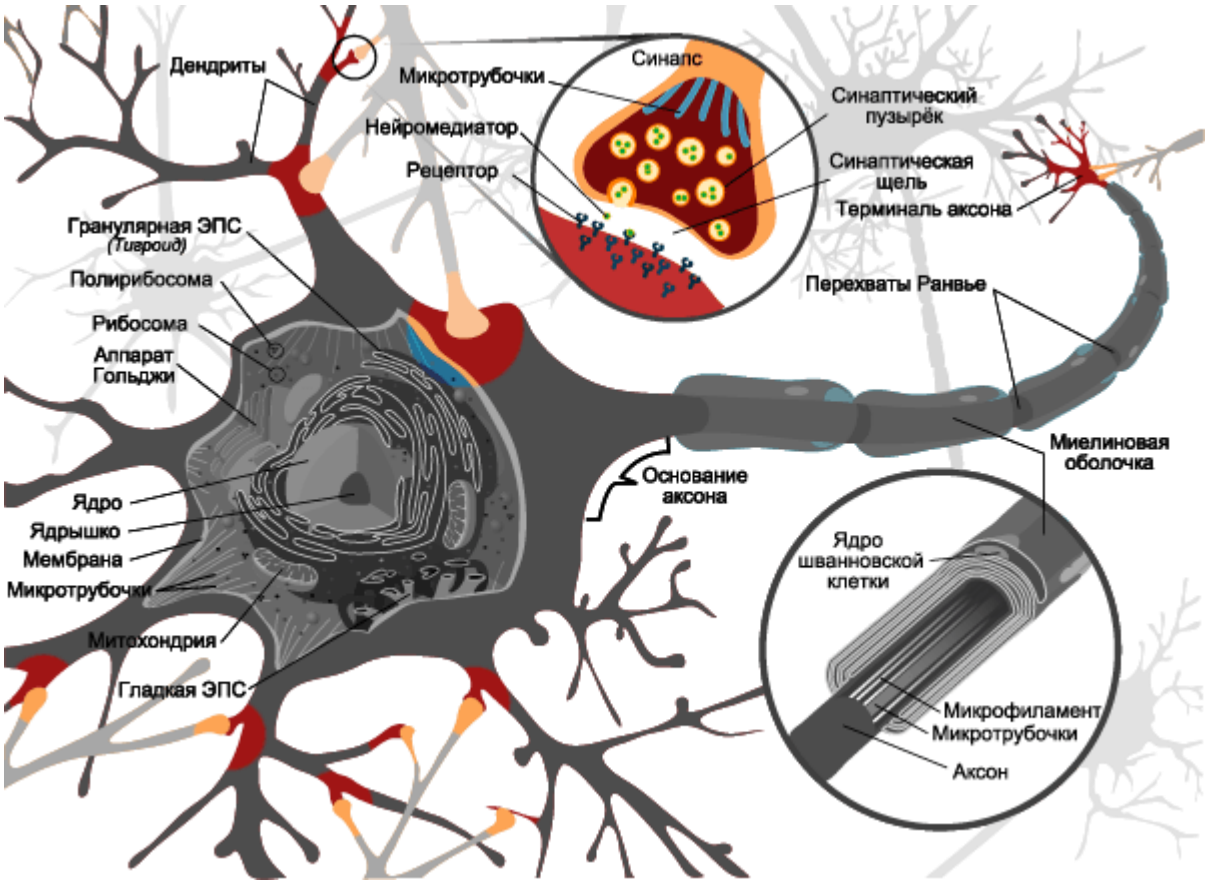

Рисунок 2 - Биологический нейрон

В дальнейшем мы будем характеризовать каждую такую связь неким числом, именуемым весом данной связи. Любой сигнал, прошедший через данную связь, умножается на вес соответствующей связи.

Это один из важнейших ключевых моментов в концепции построения (реализации) искусственных нейронных сетей. Рассмотрим рисунок 3. Каждой

связи на соответствует некоторое число w_i (вес связи). И когда сигнал проходит по этой связи, его величина умножается на вес этой связи.

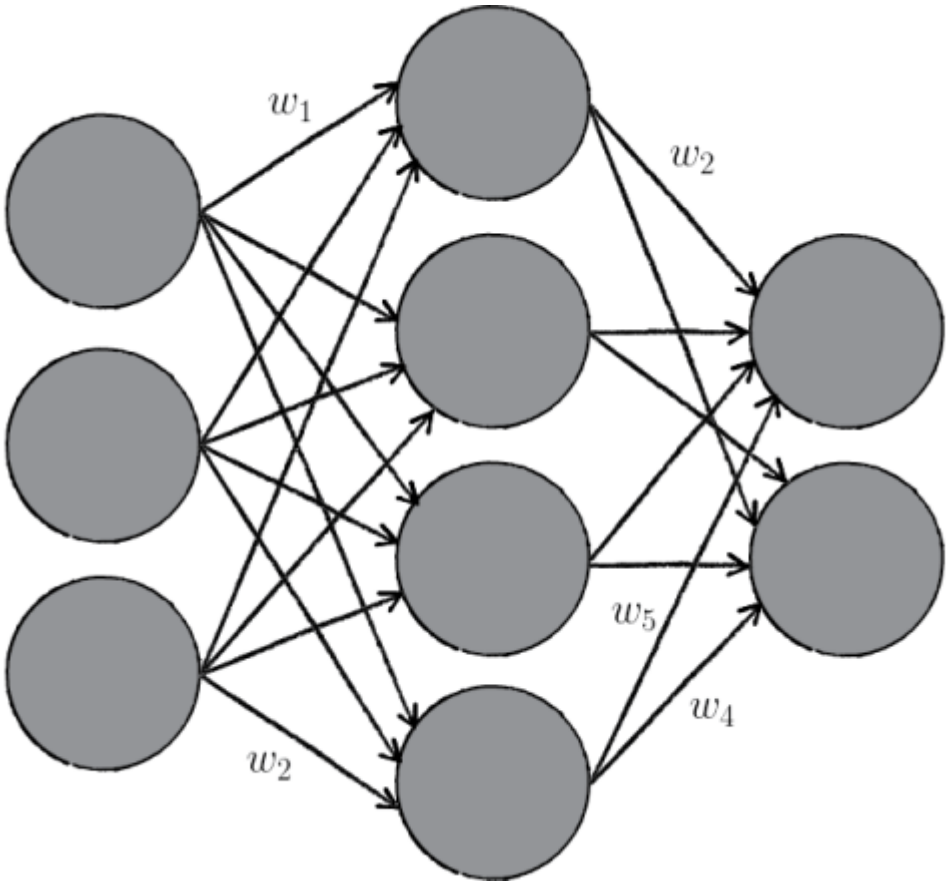

Рисунок 3 - Связи между нейровнами

2.5 Искусственный нейрон

Теперь мы переходим к рассмотрению внутренней структуры искусственного нейрона и того, как он преобразует поступающий на его входы сигнал.

На рисунке 4 представлена полная модель искусственного нейрона.

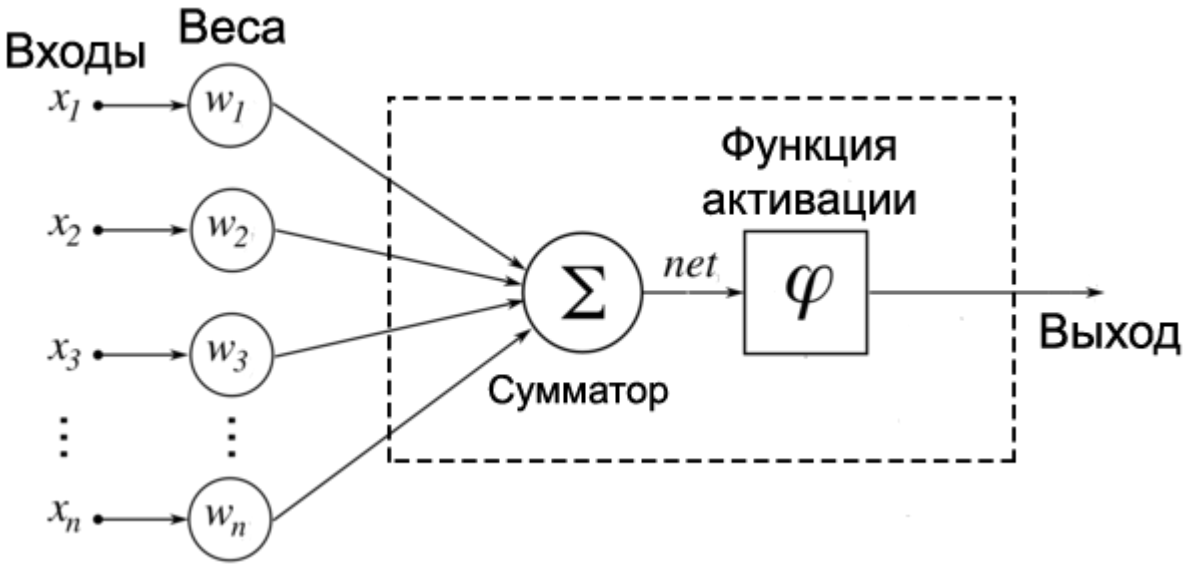

Рисунок 4 - Модель искусственного нейрона

У каждого нейрона, в том числе и у искусственного, должны быть какие-то входы, через которые он принимает сигнал. Мы уже вводили понятие весов, на которые умножаются сигналы, проходящие по связи. На картинке выше веса изображены кружками.

Поступившие на входы сигналы умножаются на свои веса. Сигнал первого входа x_1 умножается на соответствующий этому входу вес w_1. В итоге получаем $x_1 w_1$. И так до n-ого входа. В итоге на последнем входе получаем $x_n w_n$.

Теперь все произведения передаются в сумматор. Уже исходя из его названия можно понять, что он делает. Он суммирует все входные сигналы, умноженные на соответствующие веса:

$$x_1 w_1 + x_2 w_2 + \cdots + x_n w_n = \sum_{i=1}^{n} x_i w_i$$

Результатом работы сумматора является число, называемое взвешенной суммой.

Взвешенная сумма (*Weighted sum*) (net) - сумма входных сигналов, умноженных на соответствующие им веса.

$$net = \sum_{i=1}^{n} x_i w_i$$

Роль сумматора очевидна – он агрегирует все входные сигналы (которых может быть много) в какое-то одно число – взвешенную сумму, которая характеризует поступивший на нейрон сигнал в целом. Еще взвешенную сумму можно представить, как степень общего возбуждения нейрона.

Пример

Для понимания роли последнего компонента искусственного нейрона - функции активации - приведем аналогию.

Давайте рассмотрим один искусственный нейрон. Его задача - решить, ехать ли отдыхать на море. Для этого на его входы мы подаем различные данные. Пусть у нашего нейрона будет 4 входа:

1. Стоимость поездки
2. Какая на море погода
3. Текущая обстановка с работой
4. Будет ли на пляже закусочная

Все эти параметры будем характеризовать 0 или 1. Соответственно, если погода на море хорошая, то на этот вход подаем 1. И так со всеми остальными параметрами.

Если у нейрона есть четыре входа, то должно быть и четыре весовых коэффициента. В нашем примере весовые коэффициенты можно представить, как показатели важности каждого входа, влияющие на общее решение нейрона. Веса входов распределим следующим образом:

1. 5
2. 4
3. 1
4. 1

Очень большую роль играют факторы стоимости и погоды на море (первые два входа). Они же и будут играть решающую роль при принятии нейроном решения.

Пусть на входы нашего нейрона мы подаем следующие сигналы:

1. 1
2. 0
3. 0
4. 1

Умножаем веса входов на сигналы соответствующих входов:

1. 5
2. 0
3. 0
4. 1

Взвешенная сумма для такого набора входных сигналов равна 6:

$$net = \sum_{i=1}^{4} x_i w_i = 5 + 0 + 0 + 1 = 6$$

Все правильно, но что делать дальше? Как нейрон должен решить, ехать на море или нет? Очевидно, нам нужно как-то преобразовать нашу взвешенную сумму и получить ответ.

Здесь и появляется функция активации.

2.6 Функция активация

Просто так подавать взвешенную сумму на выход достаточно бессмысленно. Нейрон должен как-то обработать ее и сформировать адекватный выходной сигнал. Именно для этих целей и используют функцию активации.

Она преобразует взвешенную сумму в какое-то число, которое и является выходом нейрона (выход нейрона обозначим переменной out).

Для разных типов искусственных нейронов используют самые разные функции активации. В общем случае их обозначают символом $\phi(net)$. Указание взвешенного сигнала в скобках означает, что функция активации принимает взвешенную сумму как параметр.

Функция активации (*Activation function*) ($\phi(net)$) - функция, принимающая взвешенную сумму как аргумент. Значение этой функции и является выходом нейрона (out).

$$out = \phi(net)$$

Далее мы подробно рассмотрим самые известные функции активации.

2.7 Функция единичного скачка

Самый простой вид функции активации. Выход нейрона может быть равен только 0 или 1. Если взвешенная сумма больше определенного порога b , то выход нейрона равен 1. Если ниже, то 0.

Как ее можно использовать? Предположим, что мы поедем на море только тогда, когда взвешенная сумма больше или равна 5. Значит наш порог равен 5:

$$b = 5$$

В нашем примере взвешенная сумма равнялась 6, а значит выходной сигнал нашего нейрона равен 1, следовательно, мы едем на море.

Однако если бы погода на море была бы плохой, а также поездка была бы очень дорогой, но имелась бы закусочная и обстановка с работой нормальная (входы: 0011), то взвешенная сумма равнялась бы 2, а значит выход нейрона равнялся бы 0, следовательно, мы никуда не едем.

Из всего этого следует, что нейрон накапливает сигнал внутри себя, и, когда накопленный сигнал (взвешенная сумма) становится очень большим (больше порога нейрона), то нейрон выдает выходной сигнал, равный 1.

Графически эта функция изображена на рисунке 5.

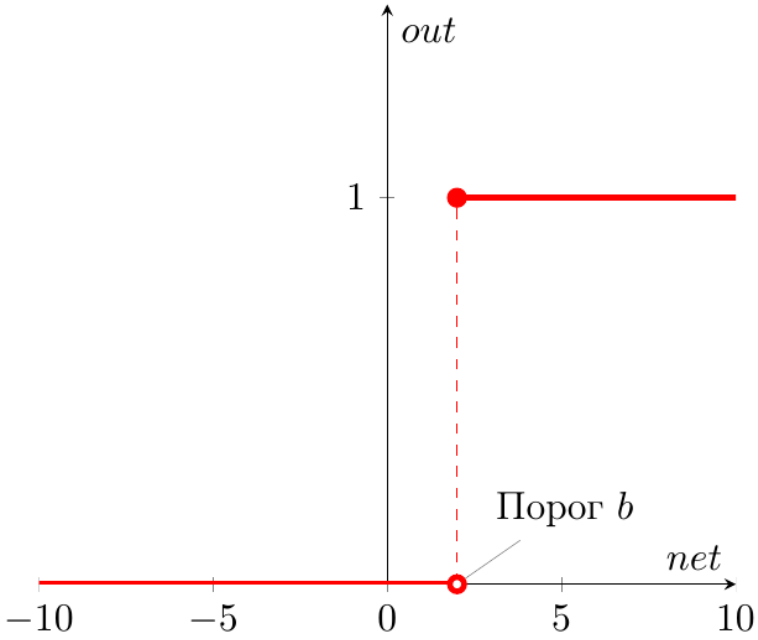

Рисунок 5 - Функция единичного скачка

На горизонтальной оси расположены величины взвешенной суммы. На вертикальной оси - значения выходного сигнала. Мы наблюдаем что, возможны только два значения выходного сигнала: 0 или 1. Причем 0 будет выдаваться всегда от минус бесконечности и вплоть до некоторого значения взвешенной суммы, называемого порогом. Если взвешенная сумма равна порогу или больше него, то функция выдает 1.

Следующем шагом будет, запись этой функции активации математически. В виде составной функции функция единичного скачка будет выглядеть следующим образом:

$$out(net) = \begin{cases} 0, net < b \\ 1, net \geqslant b \end{cases}$$

Выход нейрона (out) зависит от взвешенной суммы (net) следующим образом: если net (взвешенная сумма) меньше какого-то порога (b), то out (выход нейрона) равен 0. А если net больше или равен порогу b, то out равен 1.

2.8 Сигмоидальная функция (Сигмоида)

Существует целое семейство сигмоидальных функций, некоторые из которых применяют в качестве функции активации в искусственных нейронах.

Все эти функции обладают некоторыми очень полезными свойствами, ради которых их и применяют в нейронных сетях. Эти свойства станут очевидными после того, как вы увидите графики этих функций.

Самая часто используемая в нейронных сетях сигмоида - логистическая функция (Рисунок 6).

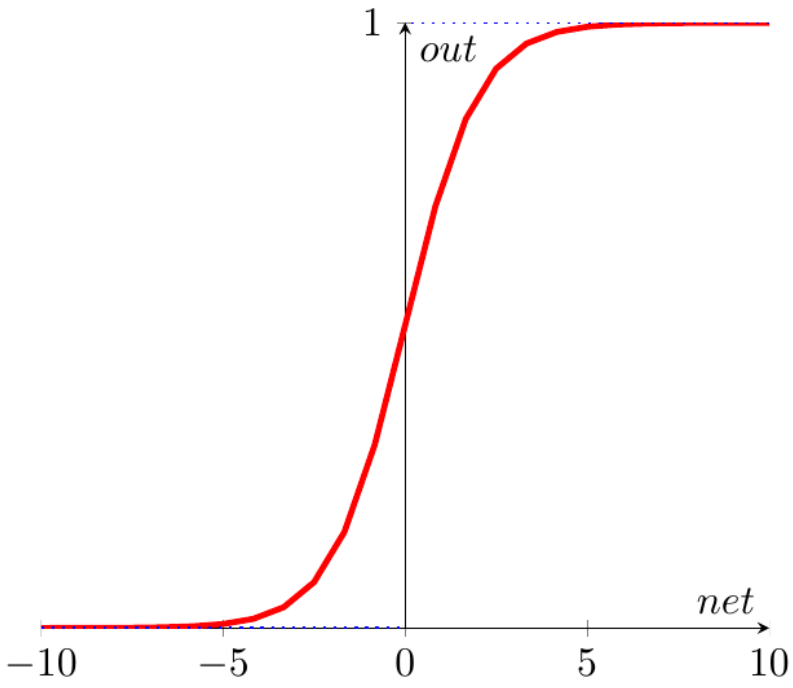

Рисунок 6 - Логическая функция

График этой функции выглядит достаточно просто. Если присмотреться, то можно увидеть некоторое подобие английской буквы S, откуда и пошло название семейства этих функций.

Аналитическая запись данной функции:

$$out(net) = \frac{1}{1+\exp(-a \cdot net)}$$

Параметр a - это какое-то число, которое характеризует степень крутизны функции. На рисунке 7 представлены логистические функции с разным параметром a.

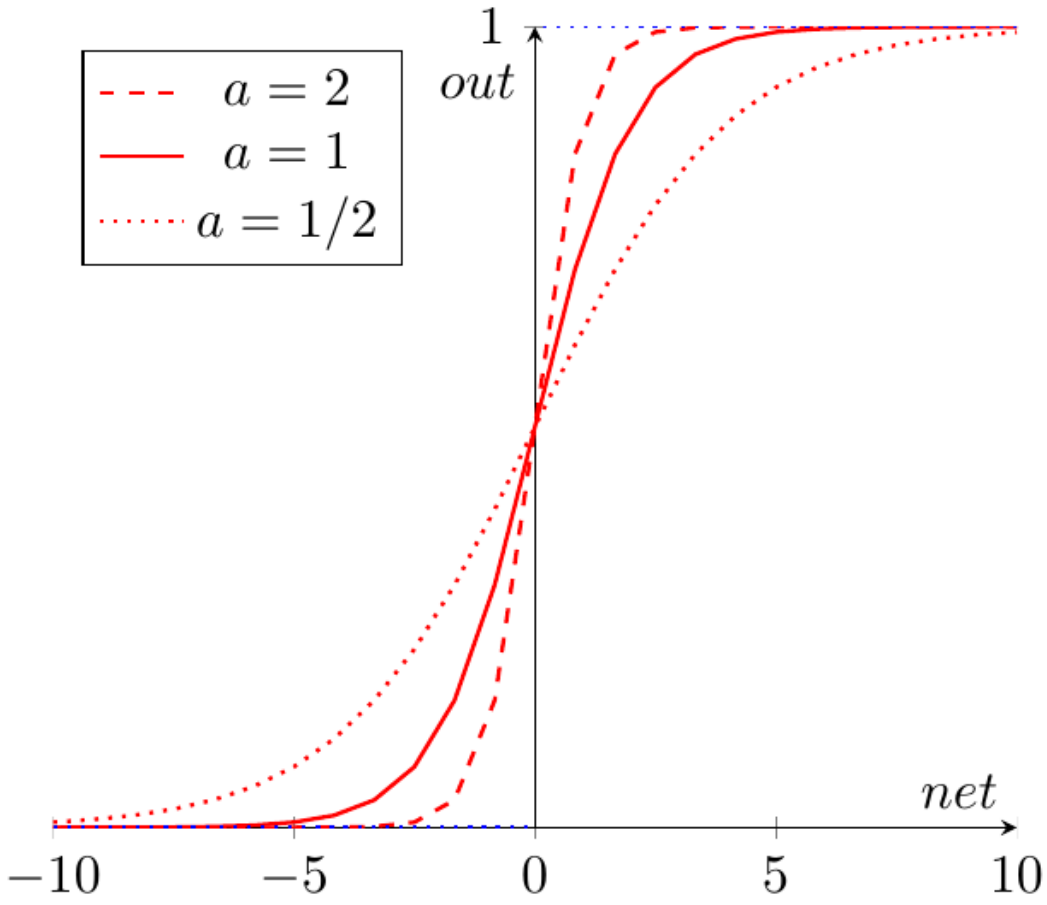

Рисунок 7 - Логическая функция (разные значения α)

Вспомним наш искусственный нейрон, определяющий, надо ли ехать на море. В случае с функцией единичного скачка все было очевидно. Мы либо едем на море (1), либо нет (0).

Здесь же случай, более приближенный к реальности. Мы до конца полностью не уверены - стоит ли ехать? Тогда использование логистической функции в качестве функции активации приведет к тому, что мы будем получать значения между 0 и 1. Причем чем больше взвешенная сумма, тем ближе выход будет к 1 (но никогда не будет точно ей равен). И наоборот, чем меньше взвешенная сумма, тем ближе выход нейрона будет к 0.

Например, выход нашего нейрона равен 0.8. Это значит, что он считает, что поехать на море все-таки стоит. Если бы его выход был бы равен 0.2, то это означает, что он почти наверняка против поездки на море.

Свойства логической функции:

- она является «сжимающей» функцией, то есть вне зависимости от аргумента (взвешенной суммы), выходной сигнал всегда будет в пределах от 0 до 1;
- она более гибкая, чем функция единичного скачка - ее результатом может быть не только 0 и 1, но и любое число между ними;
- во всех точках она имеет производную, и эта производная может быть выражена через эту же функцию.

Именно из-за этих свойств логистическая функция чаще всего используются в качестве функции активации в искусственных нейронах.

2.9 Гиперболический тангенс

Существует еще одна сигмоида - это гиперболический тангенс. Он применяется в качестве функции активации биологами для более реалистичной модели нервной клетки.

Такая функция позволяет получить на выходе значения разных знаков (например, от -1 до 1), что может быть полезным для ряда сетей.

Функция записывается следующим образом:

$$out(net) = \tanh(\frac{net}{a})$$

В данной выше формуле параметр a также определяет степень крутизны графика этой функции.

На рисунке 8 представлен график данной функции.

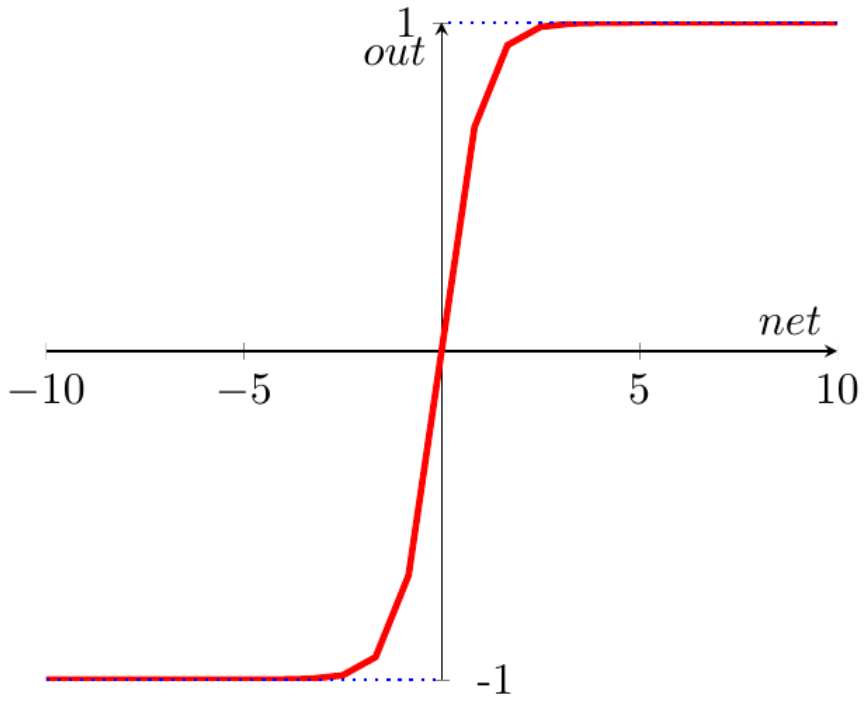

Рисунок 8 - Гиперболический тангенс

Как видно, он похож на график логистической функции. Гиперболический тангенс обладает всеми полезными свойствами, которые имеет и логистическая функция.

2.10 Краткое описание работы искусственного нейрона

У нейрона есть входы. На них подаются сигналы в виде чисел. Каждый вход имеет свой вес (тоже число). Сигналы на входе умножаются на соответствующие веса. Получаем набор «взвешенных» входных сигналов.

Далее этот набор попадает в сумматор, которой просто складывает все входные сигналы, помноженные на веса. Получившееся число называют **взвешенной суммой**.

Затем взвешенная сумма преобразуется **функцией активации,** и мы получаем **выход нейрона**.

Сформулируем теперь самое короткое описание работы нейрона – его математическую модель:

Математическая модель искусственного нейрона с n входами:

$out = \phi\left(\sum_{i=1}^{n} x_i w_i\right)$, где ϕ - функция активации

$\sum_{i=1}^{n} x_i w_i$ - взвешенная сумма, как сумма n произведений входных сигналов на соответствующие веса.

3 Виды искусственных нейронных сетей

Искусственные нейронные сети состоят из совокупности искусственных нейронов. Возникает логичный вопрос - а как располагать/соединять друг с другом эти самые искусственные нейроны?

Как правило, в большинстве нейронных сетей есть так называемый **входной слой**, который выполняет только одну задачу - распределение входных сигналов остальным нейронам. Нейроны этого слоя не производят никаких вычислений.

3.1 Однослойные нейронные сети

В однослойных нейронных сетях сигналы с входного слоя сразу подаются на выходной слой. Он производит необходимые вычисления, результаты которых сразу подаются на выходы.

На рисунке 9 представлена однослойная нейронная сеть.

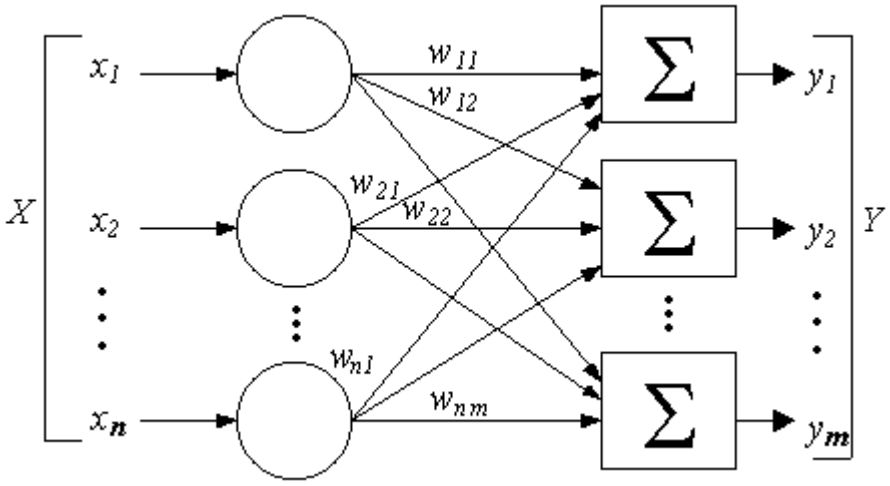

Рисунок 9 - Однослойная нейронная сеть

На этом рисунке входной слой обозначен кружками (он не считается за слой нейронной сети), а справа расположен слой обычных нейронов.

Нейроны соединены друг с другом стрелками. Над стрелками расположены веса соответствующих связей (весовые коэффициенты).

Однослойная нейронная сеть (*Single-layer neural network*) - сеть, в которой сигналы от входного слоя сразу подаются на выходной слой, который и преобразует сигнал и сразу же выдает ответ.

3.2 Многослойные нейронные сети

Многослойные нейронные сети, помимо входного и выходного слоев нейронов, характеризуются еще и скрытым слоем (слоями). Понять их расположение просто – эти слои находятся между входным и выходным слоями.

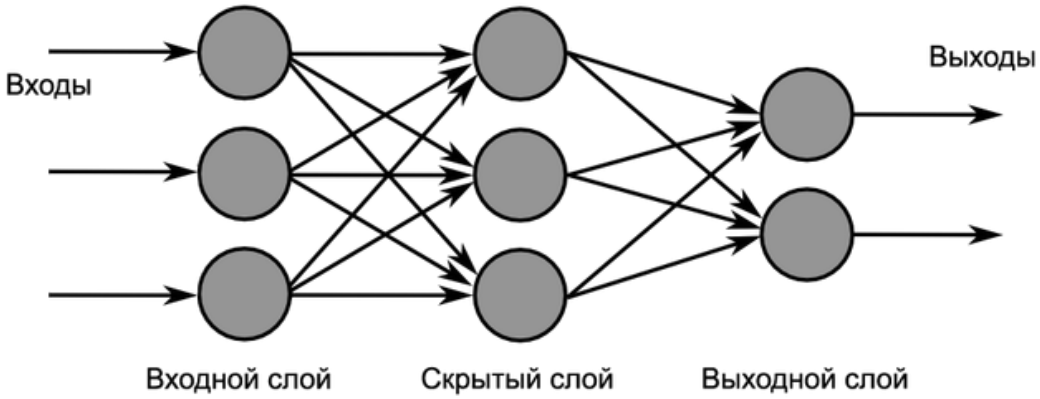

Рисунок 10 - Нейронная сеть

Такая структура нейронных сетей копирует многослойную структуру определенных отделов мозга.

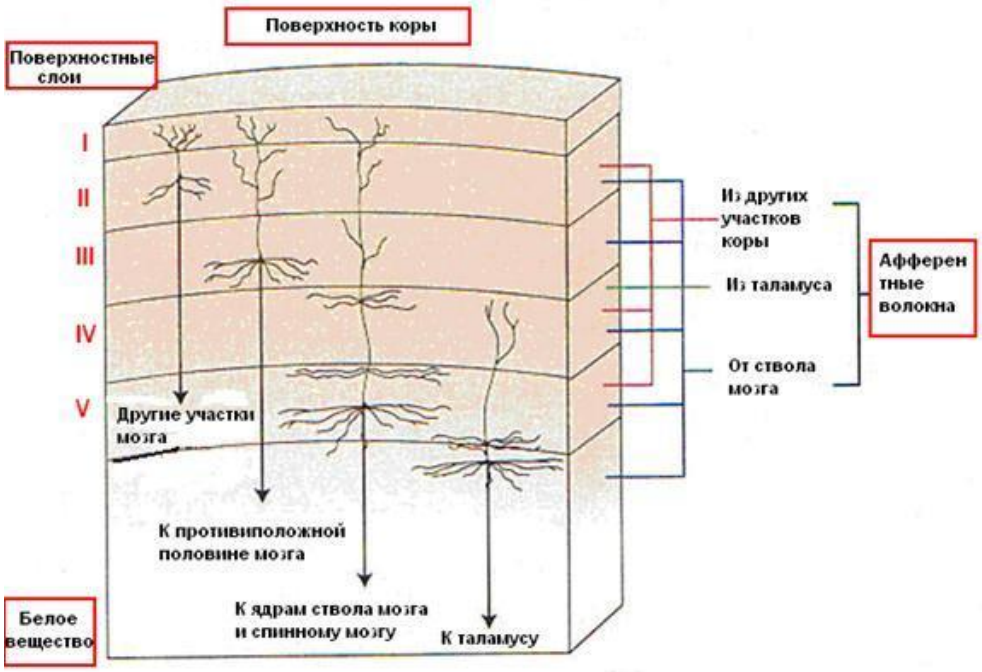

Рисунок 11 - Структура нейронных сетей отделов мозга

Название скрытый слой получил неслучайно. Дело в том, что только относительно недавно были разработаны методы обучения нейронов скрытого слоя. До этого обходились только однослойными нейросетями.

Многослойные нейронные сети обладают гораздо большими возможностями, чем однослойные.

Работу скрытых слоев нейронов можно сравнить с работой большого завода. Продукт (выходной сигнал) на заводе собирается по стадиям. После каждого станка получается какой-то промежуточный результат. Скрытые слои тоже преобразуют входные сигналы в некоторые промежуточные результаты.

Многослойная нейронная сеть (*Multilayer neural network*) - нейронная сеть, состоящая из входного, выходного и расположенного(ых) между ними одного (нескольких) скрытых слоев нейронов.

3.3 Сети прямого распространения (feedforward)

Можно заметить одну очень интересную деталь на картинках нейросетей в примерах выше.

Во всех примерах стрелки строго идут слева направо, то есть сигнал в таких сетях идет строго от входного слоя к выходному.

Сети прямого распространения (*Feedforward neural network*) (feedforward сети) - искусственные нейронные сети, в которых сигнал распространяется строго от входного слоя к выходному. В обратном направлении сигнал не распространяется.

Такие сети широко используются и вполне успешно решают определенный класс задач: прогнозирование, кластеризация и распознавание. Однако никто не запрещает сигналу идти и в обратную сторону.

3.4 Сети с обратными связями

В сетях такого типа сигнал может идти и в обратную сторону. В чем преимущество?

Дело в том, что в сетях прямого распространения выход сети определяется входным сигналом и весовыми коэффициентами при искусственных нейронах.

А в сетях с обратными связями выходы нейронов могут возвращаться на входы. Это означает, что выход какого-нибудь нейрона определяется не только его весами и входным сигналом, но еще и предыдущими выходами (так как они снова вернулись на входы).

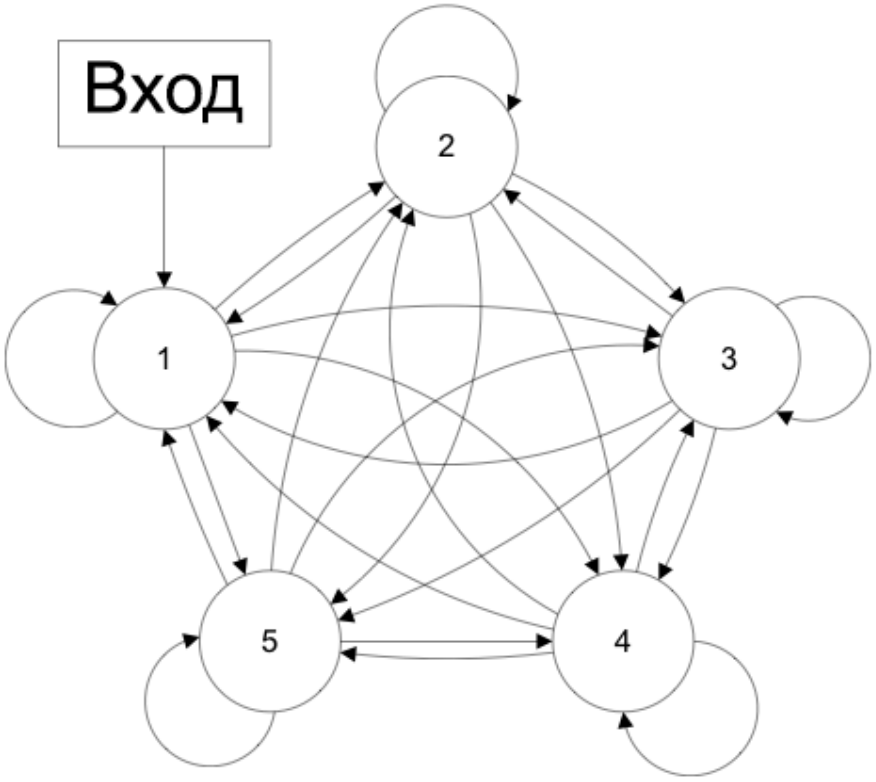

Рисунок 12 - Нейронная сеть с обратной связью

Возможность сигналов циркулировать в сети открывает новые, удивительные возможности нейронных сетей. С помощью таких сетей можно

создавать нейросети, восстанавливающие или дополняющие сигналы. Другими словами, такие нейросети имеют свойства кратковременной памяти.

Сети с обратными связями (*Recurrent neural network*) - искусственные нейронные сети, в которых выход нейрона может вновь подаваться на его вход. В более общем случае это означает возможность распространения сигнала от выходов к входам.

4 Обучение нейронной сети

Более подробно рассмотрим вопрос обучения нейронной сети. Что это такое? И каким образом это происходит?

Искусственная нейронная сеть - это совокупность искусственных нейронов. Теперь давайте возьмем, например, 100 нейронов и соединим их друг с другом. Ясно, что при подаче сигнала на вход, мы получим что-то бессмысленное на выходе.

Следовательно, нам надо менять какие-то параметры сети до тех пор, пока входной сигнал не преобразуется в нужный нам выходной.

Изменять общее количество искусственных нейронов бессмысленно по двум причинам. Во-первых, увеличение количества вычислительных элементов в целом лишь делает систему тяжеловеснее и избыточнее. Во-вторых, если вы соберете 1000 дураков вместо 100, то они все-равно не смогут правильно ответить на вопрос.

Сумматор изменить не получится, так как он выполняет одну жестко заданную функцию - складывать. Если мы его заменим на что-то или вообще уберем, то это вообще уже не будет искусственным нейроном.

Если менять у каждого нейрона функцию активации, то мы получим слишком разношерстную и неконтролируемую нейронную сеть. К тому же, в большинстве случаев нейроны в нейронных сетях одного типа. То есть они все имеют одну и ту же функцию активации.

Остается только один вариант - менять веса связей.

Обучение нейронной сети (*Training)* - поиск такого набора весовых коэффициентов, при котором входной сигнал после прохода по сети преобразуется в нужный нам выходной.

Такой подход к термину «обучение нейронной сети» соответствует и биологическим нейросетям. Наш мозг состоит из огромного количества связанных друг с другом нейросетей. Каждая из них в отдельности состоит из нейронов одного типа (функция активации одинаковая). Мы обучаемся благодаря изменению синапсов - элементов, которые усиливают/ослабляют входной сигнал.

Однако есть еще один важный момент. Если обучать сеть, используя только один входной сигнал, то сеть просто «запомнит правильный ответ». Со стороны будет казаться, что она очень быстро «обучилась». И как только вы подадите немного измененный сигнал, ожидая увидеть правильный ответ, то сеть выдаст бессмыслицу.

В самом деле, зачем нам сеть, определяющая лицо только на одном фото. Мы ждем от сети способности обобщать какие-то признаки и узнавать лица и на других фотографиях тоже.

Именно с этой целью и создаются обучающие выборки.

Обучающая выборка (*Training set*) - конечный набор входных сигналов (иногда вместе с правильными выходными сигналами), по которым происходит обучение сети.

После обучения сети, то есть когда сеть выдает корректные результаты для всех входных сигналов из обучающей выборки, ее можно использовать на практике.

Однако прежде чем пускать свежеиспеченную нейросеть в работу, часто производят оценку качества ее работы на так называемой **тестовой выборке**.

Тестовая выборка (*Testing set*) - конечный набор входных сигналов (иногда вместе с правильными выходными сигналами), по которым происходит оценка качества работы сети.

Теперь возникает вопрос - а как можно обучать сеть? Существует два подхода, приводящие к разным результатам: обучение с учителем и обучение без учителя.

4.1 Обучение с учителем

Суть данного подхода заключается в том, что вы даете на вход сигнал, смотрите на ответ сети, а затем сравниваете его с уже готовым, правильным ответом.

Важный момент. Не путайте правильные ответы и известный алгоритм решения! Мы можете обвести пальцем лицо на фото (правильный ответ), но не сможете сказать, как это сделали (известный алгоритм). Тут такая же ситуация.

Затем, с помощью специальных алгоритмов, вы меняете веса связей нейронной сети и снова падаете ей входной сигнал. Сравниваете ее ответ с правильным и повторяете этот процесс до тех пор, пока сеть не начнет отвечать с приемлемой точностью (однозначно точных ответов сеть давать не может).

Обучение с учителем (*Supervised learning*) - вид обучения сети, при котором ее веса меняются так, чтобы ответы сети минимально отличались от уже готовых правильных ответов.

Если мы хотим, чтобы сеть узнавала лица, мы можем создать обучающую выборку на 1000 фотографий (входные сигналы) и самостоятельно выделить на ней лица (правильные ответы).

Если мы хотим, чтобы сеть прогнозировала рост/падение цен, то обучающую выборку надо делать, основываясь на прошлых данных. В качестве входных сигналов можно брать определенные дни, общее состояние рынка и другие параметры. А в качестве правильных ответов - рост и падение цены в те дни.

Стоит отметить, что учитель, не обязательно человек. Дело в том, что порой сеть приходится тренировать часами и днями, совершая тысячи и десятки тысяч попыток. В 99% случаев эту роль выполняет компьютер, а точнее, специальная компьютерная программа.

4.2 Обучение без учителя

Обучение без учителя применяют тогда, когда у нас нет правильных ответов на входные сигналы. В этом случае вся обучающая выборка состоит из набора входных сигналов.

В результате при таком «обучении» сеть начинает выделять классы подаваемых на вход сигналов (кластеризацию).

Например, вы демонстрируете сети конфеты, пирожные и торты. Вы никак не регулируете работу сети. Мы просто подаем на ее входы данные о данном объекте. Со временем сеть начнет выдавать сигналы трех разных типов, которые и отвечают за объекты на входе.

Обучение без учителя (*Unsupervised learning*) - вид обучения сети, при котором сеть самостоятельно классифицирует входные сигналы. Правильные (эталонные) выходные сигналы не демонстрируются.

5 Персептрон

В основе персептрона лежит математическая модель восприятия информации мозгом. Разные исследователи по-разному его определяют. В самом общем своем виде (как его описывал Розенблатт) он представляет систему из элементов трех разных типов: сенсоров, ассоциативных элементов и реагирующих элементов (рисунок 13, 14).

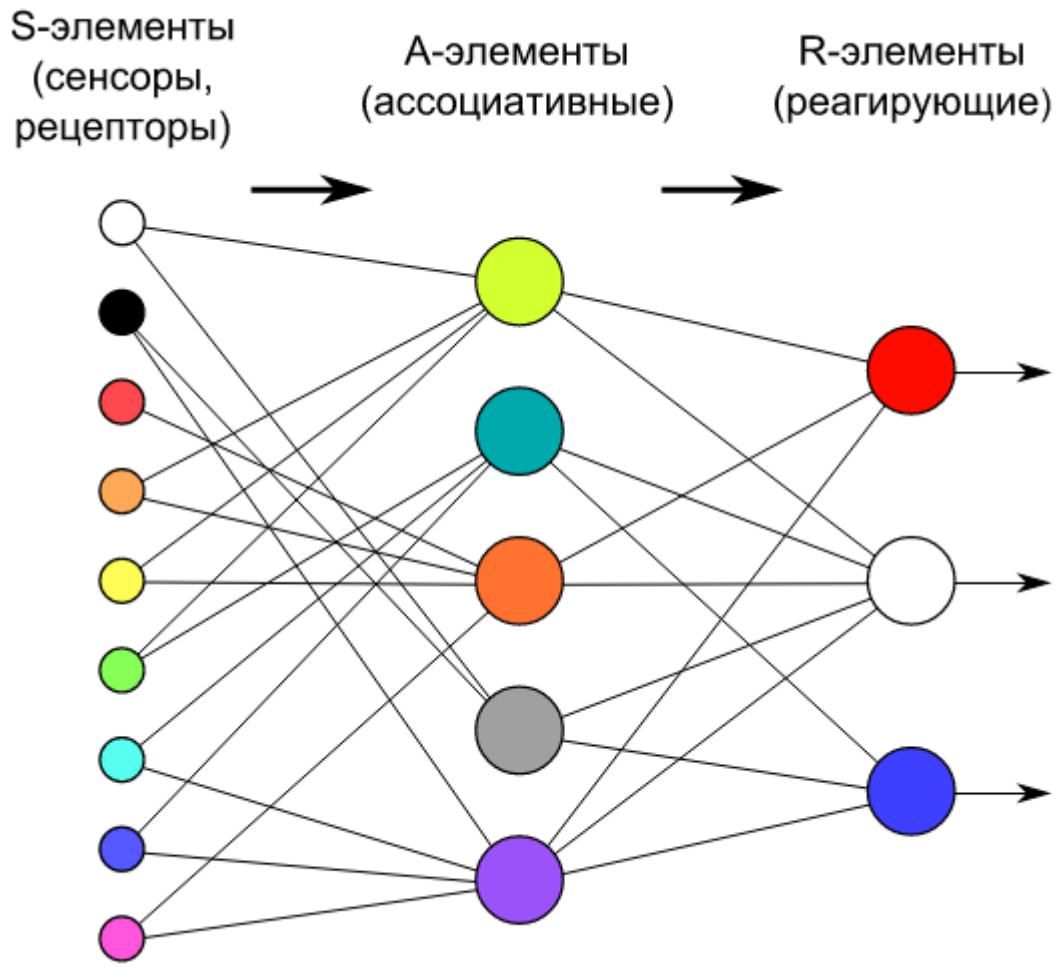

Рисунок 13 - Общий вид персептрона

Рассмотрим принцип работы персептрона.

Первыми в работу включаются S-элементы. Они могут находиться либо в состоянии покоя (сигнал равен 0), либо в состоянии возбуждения (сигнал равен 1).

Далее сигналы от S-элементов передаются A-элементам по так называемым S-A связям. Эти связи могут иметь веса, равные только -1, 0 или 1.

Затем сигналы от сенсорных элементов, прошедших по S-A связям попадают в A-элементы, которые еще называют ассоциативными элементами. Стоит заметить, что одному A-элементу может соответствовать несколько S-элементов. Если сигналы, поступившие на A-элемент, в совокупности превышают некоторый его порог θ, то этот A-элемент возбуждается и выдает сигнал, равный 1. В противном случае (сигнал от S-элементов не превысил порога A-элемента), генерируется нулевой сигнал.

Почему A-элементы назвали ассоциативными? Дело в том, что A-элементы являются агрегаторами сигналов от сенсорных элементов. Например, у нас есть группа сенсоров, каждый из которых распознает кусок буквы «Д» на исследуемой картинке. Однако только их совокупность (то есть когда несколько сенсоров выдали сигнал, равный 1) может возбудить A-элемент целиком. На другие буквы A-элемент не реагирует, только на букву «Д». То есть он ассоциируется с буквой «Д». Отсюда и такое название.

Можно привести и другой пример. На самом деле наши глаза состоят из невероятного количества S-элементов (сенсоров), улавливающих падающий свет (около 140 000 000). Некий A-элемент, который распознает конкретную часть лица. И вот вы увидели на улице человека. Некоторые A-элементы, которые распознали конкретные части лица, возбуждаются.

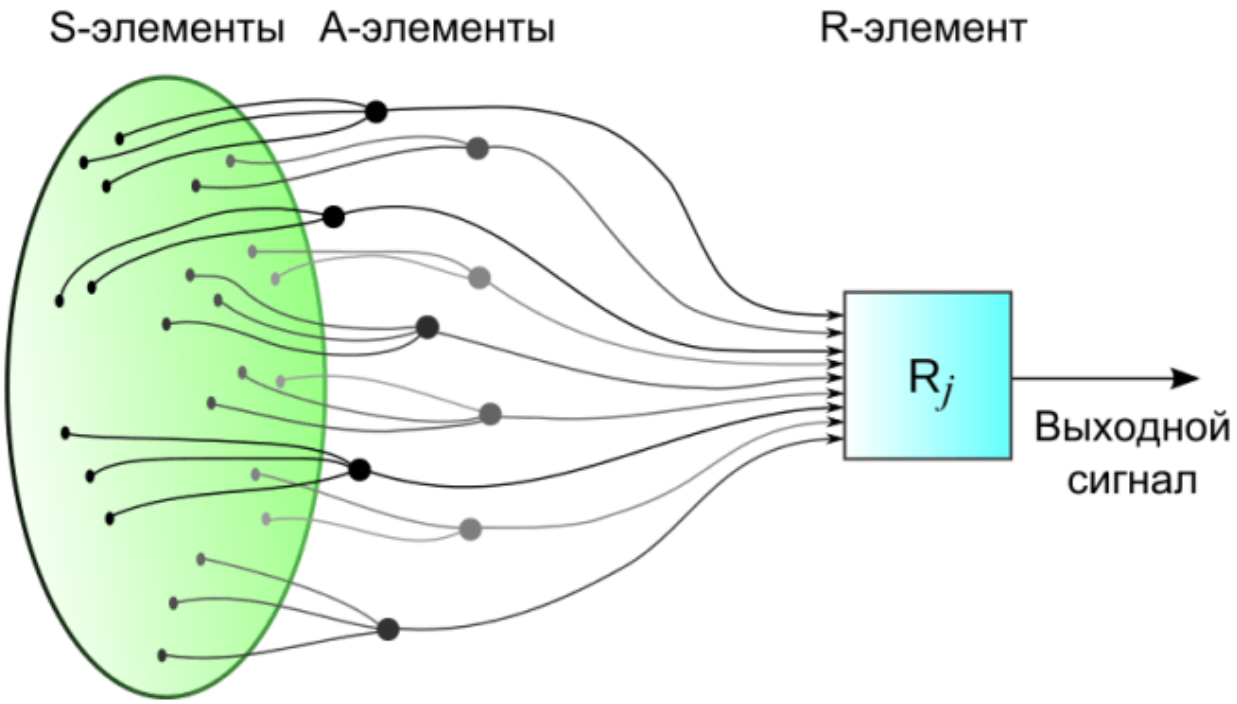

Рисунок 14 - Персептрон

Далее сигналы, которые произвели возбужденные A-элементы, направляются к сумматору (R-элемент), действие которого вам уже известно. Однако, чтобы добраться до R-элемента, они проходят по A-R связям, у которых тоже есть веса. Однако здесь они уже могут принимать любые значения (в отличие от S-A связей).

В конце пути R-элемент складывает друг с другом взвешенные сигналы от A-элементов и, если превышен определенный порог, генерирует выходной сигнал, равный 1. Это означает, что в общем потоке информации от глаз мы распознали лицо человека.

Если порог не превышен, то выход персептрона равен -1. То есть мы не выделили лицо из общего потока информации.

Так как R-элемент определяет выход персептрона в целом, его назвали реагирующим.

Персептрон (*Perceptron*) - простейший вид нейронных сетей. В основе лежит математическая модель восприятия информации мозгом, состоящая из сенсоров, ассоциативных и реагирующих элементов.

5.1 Классификация персептронов

Исторически сложилось несколько типов персептронов. Их часто путают, так как разные авторы, писавшие статьи на эту тему, зачастую понимали под персептроном похожие, но все же разные математические модели. Именно поэтому человеку, который только начинает изучение ИНС бывает трудно разобраться, что такое персептрон, ведь в разных книгах на эту тему его определяют по-разному.

5.2 Персептрон с одним скрытым слоем

Общее определение персептрона было приведено выше. Теперь рассмотрим его подвид - персептрон с одним скрытым слоем (или элементарный персептрон). Именно этот тип персептронов часто подразумевают, когда говорят о персептронах вообще.

Почему слой именно скрытый? Потому что слой A-элементов расположен между слоями S-элементов и R-элементов.

Персептрон с одним скрытым слоем - персептрон, у которого имеется только по одному слою S, A и R элементов.

Обе картинки выше изображают именно персептрон с одним скрытым слоем.

5.3 Однослойный персептрон

Названия похожие. Но это не тоже самое, что и персептрон с одним скрытым слоем, хотя так может показаться. Именно этот тип персептронов мы и будем изучать подробнее всего. Этот тип персептронов, как и элементарные персептроны, тоже часто подразумевают, когда говорят о персептронах вообще.

Его ключевая особенность состоит в том, что каждый S-элемент однозначно соответствует одному A-элементу, все S-A связи имеют вес, равный +1, а порог A элементов равен 1.

Объясню подробнее. Возьмем рисунок 15 (персептрона в общем смысле) и преобразуем его в однослойный персептрон.

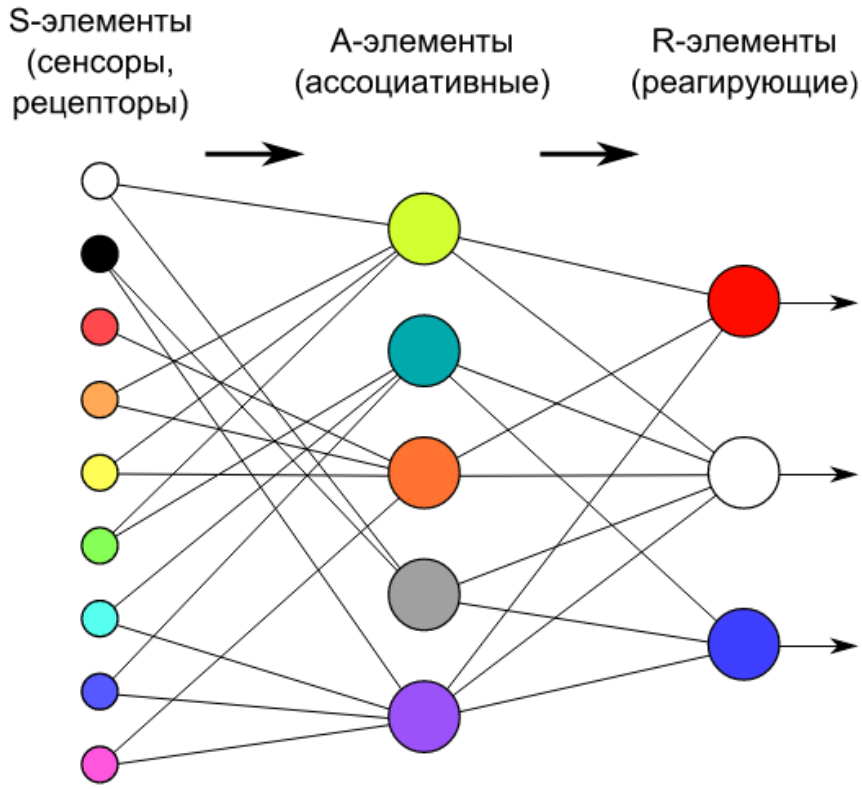

Рисунок 15 - Персептрон в общем смысле

Исходя из ключевой особенности однослойного персептрона сенсор может быть однозначно связан только с одним ассоциативным элементом. Посмотрим на белый сенсор на картинке (левый верхний угол). Он передает сигнал салатовому (первому) и серому (четвертому) ассоциативным элементам. Непорядок. Сенсор может передавать сигнал только одному А-элементу. Убираем лишнюю связь. Ту же операцию проводим и с другими сенсорами.

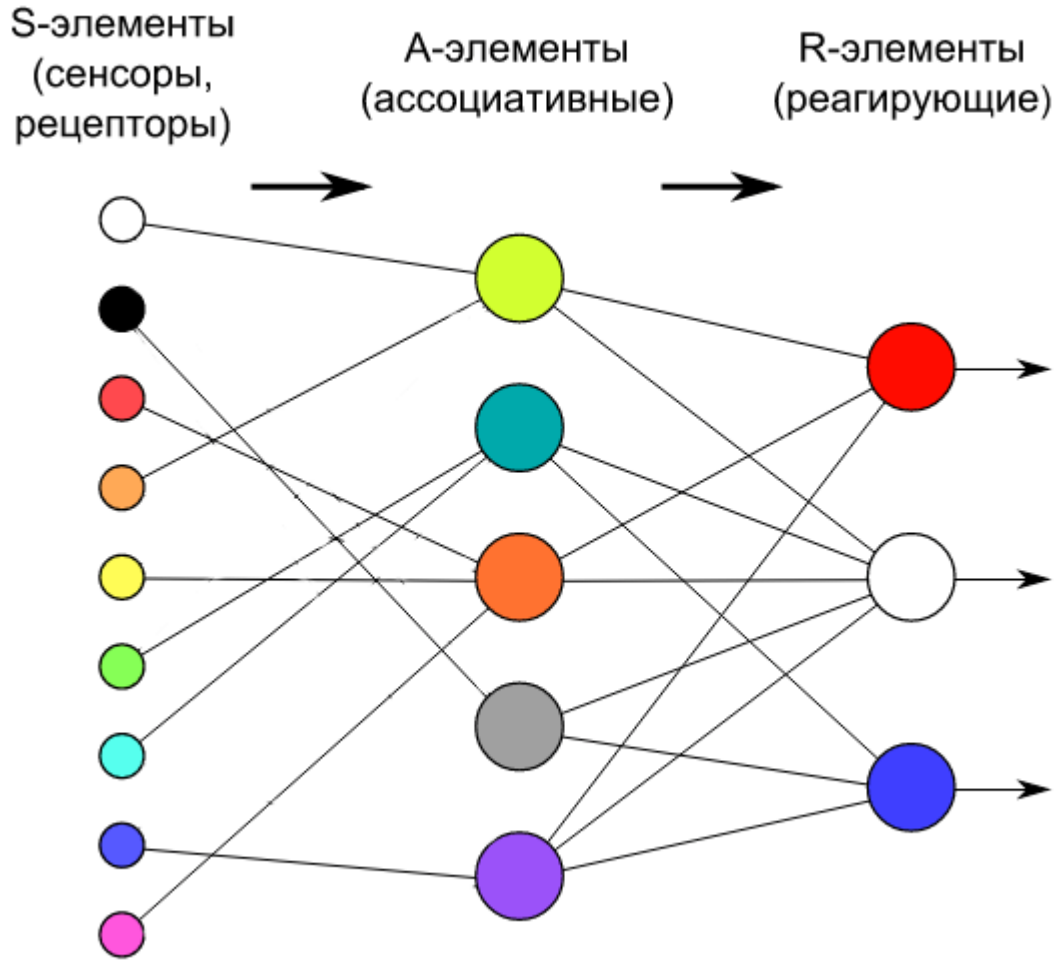

Рисунок 16 - Персептрон

Обязательно убедитесь, что поняли фразу «каждый S-элемент однозначно соответствует одному А-элементу». Это означает, что каждый сенсор может передавать сигнал только одному А-элементу. Однако это утверждение вовсе не запрещает ситуации, когда несколько сенсоров

передают сигнал на один A-элемент, что и продемонстрировано на рисунке 16 (1, 2 и 3 A-элементы).

Далее, S-A связи всегда имеют вес, равный единице, а порог A-элементов всегда равен +1. С другой стороны, нам известно, что сенсоры могут подавать сигнал равный только 0 или 1.

Рассмотрим первый S-элемент на последней картинке. Пусть он генерирует сигнал, равный единице. Сигнал проходит по S-A связи и не изменяется, так как любое число, умноженное на 1 равно самому себе. Порог любого A-элемента равен 1. Так как сенсор произвел сигнал, равный 1, то A-элемент однозначно возбудился. Это означает, что он выдал сигнал, равный 1 (так как он тоже может генерировать только 1 или 0 на своем выходе). Далее этот единичный сигнал умножается на произвольный вес A-R связи и попадает в соответствующий R-элемент, который суммирует все поступившие на него взвешенные сигналы, и, если они превышают его порог, выдает +1. В противном случае выход данного R-элемента равен -1.

Не считая сенсорных элементов и S-A связей, мы только что описали схему работы искусственного нейрона. И это неслучайно. Однослойный персептрон действительно представляет собой искусственный нейрон с небольшим отличием. В отличие от искусственного нейрона, у однослойного персептрона входные сигналы могут принимать фиксированные значения: 0 или 1. У искусственного нейрона на вход можно подавать любые значения.

В персептроне R-элементы суммируют взвешенные входные сигналы и, если взвешенная сумма выше некоторого порога, выдают 1. Иначе выходы R-элементов были бы равны -1.

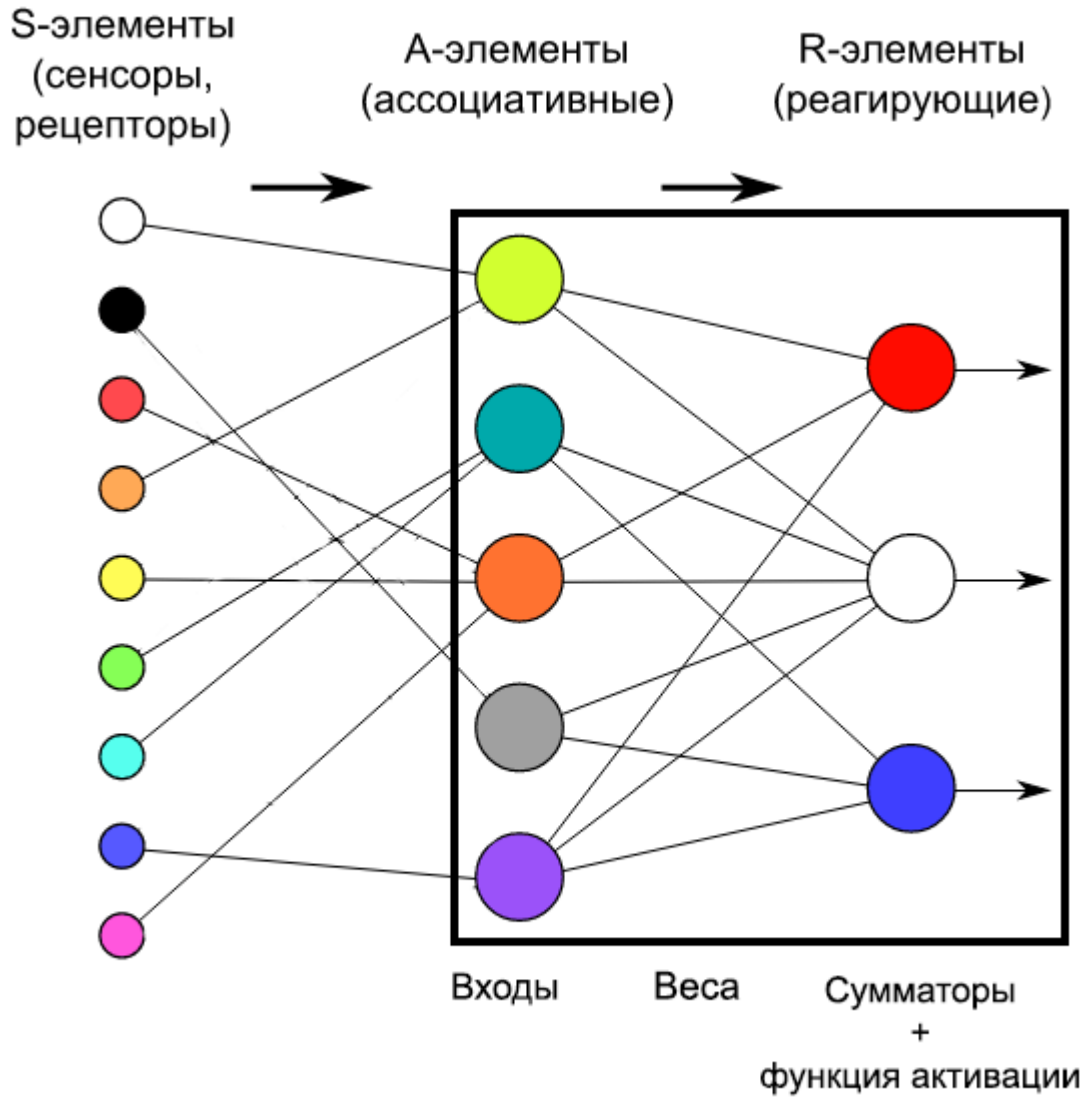

Рисунок 17 - Искусственный персептрон

Данное поведение легко задается функцией активации под названием функция единичного скачка, которую мы уже рассматривали. Отличие заключается в том, что функция единичного скачка выдает 0, если порог не превышен, а здесь выдает -1, но это не существенно.

Таким образом становится ясно, что часть однослойного персептрона (выделена черным прямоугольником на рисунке 17) можно представить в виде искусственного нейрона, но ни в коем случае не следует путайте два этих

понятия. Во-первых, никто не отменял S-элементы, которых в искусственном нейроне просто нет. Во-вторых, в однослойном персептроне S-элементы и A-элементы могут принимать только фиксированные значения 0 и 1, тогда как в искусственном нейроне таких ограничений нет.

Однослойный персептрон - персептрон, каждый S-элемент которого однозначно соответствует одному A-элементу, S-A связи всегда равны 1, а порог любого A-элемента равен 1.

Часть однослойного персептрона соответствует модели искусственного нейрона.

Однослойный персептрон может быть и элементарным персептроном, у которого только по одному слою S, A, R-элементов.

5.4 Многослойный персептрон

Под многослойным персептроном понимают два разных вида: многослойный персептрон по Розенблатту и многослойный персептрон по Румельхарту.

Многослойный персептрон по Розенблатту содержит более 1 слоя A-элементов.

Многослойный персептрон по Румельхарту является частным случаем многослойного персептрона по Розенблатту, с двумя особенностями:

1. S-A связи могут иметь произвольные веса и обучаться наравне с A-R связями.
2. Обучение производится по специальному алгоритму, который называется обучением по методу обратного распространения ошибки.

Этот метод является краеугольным камнем обучения всех многослойных ИНС. Во многом благодаря ему возобновился интерес к нейронным сетям.

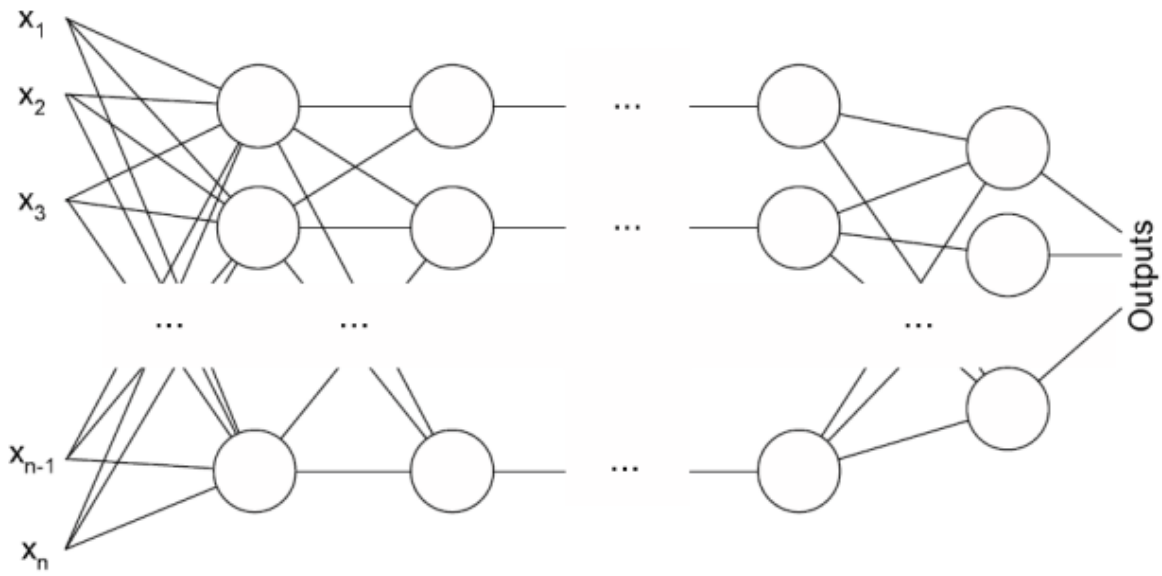

Рисунок 18 - Многослойная искусственная нейронная сеть

Многослойный персептрон по Розенблатту - персептрон, у которого имеется более 1 слоя А-элементов.

Многослойный персептрон по Румельхарту - многослойный персептрон по Розенблатту, у которого обучению подлежат еще и S-A связи, а также само обучение производится по методу обратного распространения ошибки.

5.5 Опорная схема

Для облегчения запоминания классификации (а ее обязательно надо помнить, чтобы ориентироваться в материале) приведу опорную схему изображённая на рисунке 19.

Рисунок 19 - Классификация нейронной сети

6 Задачи, решающие персептрон

Подробнее о том, какие задачи персептрон способен решать, а какие нет. Также я приведу несколько удобных аналогий, позволяющих понять, возможно ли использовать персептрон в данной задаче или нет. Это крайне важно, так как персептрон, как простейший вид нейросети, не так уж много и умеет.

Персептроны очень хорошо решают задачи классификации. Если у вас есть группы объектов (например, кошки и собаки), то персептрон после обучения сможет указывать к какой группе относится объект (к кошкам или собакам).

«Очень хорошо» - понятие растяжимое. Насколько хорошо? Розенблатт доказал несколько теорем:

1. Если имеется поле сенсоров (матрица) и какая-то классификация, зависящая от него, то множество элементарных персептронов, проводящих успешную классификацию не является пустым.

Под полем сенсоров понимается множество всех S-элементов. Под классификацией - созданные нами классы (те же кошки и собаки). Под «непустым множеством элементарных персептронов, проводящих успешную классификацию» понимается, что найдется хотя бы один персептрон, справившийся с классификацией объектов.

Рассмотрим на примере.

Я хочу, чтобы персептрон научился различать кошек и собак.

У нас будет 3 сенсора: длина лап, окрас и форма морды. Так как S-элементы могут принимать значения 0 или 1, то условимся, что значения 1 будут соответствовать коротким лапам, смешанному окрасу и округлая морда соответственно. Значения 0 будут означать признак собаки на данном S-элементе (длинные лапы, однотонный окрас и вытянутая морда). Вот мы и получили сенсорное поле. Его можно представить в виде множества возможных значений 0 и 1 у каждого S-элемента. Например, абсолютная кошка должна вызвать срабатывание всех S-элементов {1,1,1}.

Рисунок 20 – Идеальное представление кота

Идеальной же собаке соответствует следующий набор выходов S-элементов: {0, 0, 0}.

Рисунок 21 - Идеальное представление собаки

Сами по себе сенсоры не играют роли. Но добавив к набору выходов сенсоров смысл: кошка или собака, мы тем самым задали некоторую классификацию. Математически это означает, что мы задали некоторую функцию, которая принимает набор выходов S-элементов, а ее значением является 0 или 1 (собака или кот).

Из приведенной выше теоремы следует, что множество персептронов, правильно проводящих нашу классификацию не является пустым. То есть такие персептроны есть!

Но ведь можно выбрать любой набор S-элементов и любую классификацию. И множество «решений» все равно не будет пустым!

Это означает, что теоретически персептроны способны решать любую задачу на классификацию.

Важное замечание!

1. Речь идет об элементарных персептронах.

2. Объекты классификации должны обладать свойством линейной разделимости.

Но есть и вторая теорема, доказанная Розенблаттом:

2. Если имеется поле сенсоров (матрица) и какая-то классификация, зависящая от него, то процесс обучения с коррекцией ошибок, начинающийся с произвольного исходного состояния, всегда приведёт к достижению решения в течение конечного промежутка времени.

Под произвольным исходным состоянием тут понимается персептрон с произвольными S-A и A-R весами связей. Под решением в теореме понимается персептрон с определенными весами, успешно решающий нашу задачу на классификацию.

Эта теорема не оставляет задачам на классификацию никаких шансов. Теперь известно, что мы всегда сможем решить нашу задачу за конечный промежуток времени. Единственный нюанс заключается в том, что никто не говорит о длительности «конечного промежутка времени». Секунда, минута, час, год, 1 000 лет?

Итак, элементарные персептроны гарантировано решают задачи на классификацию линейно разделимых объектов. Может показаться, что это мало. Но это не так. Существует очень много задач на классификацию, а многие можно к ним свести.

7 Линейная разделимость

Для понятности, визуализируем классификацию линейно разделимых объектов на примере кошек и собак. Чтобы их различать (классифицировать) введем два параметра: размер и приручённость. По сочетанию размера и приручённости нейронная сеть должна будет принять решение: кошка это или собака. Не может быть одновременно кошка и собака для одинаковых показателей размера и приручённости.

В этом случае потребуется всего два S-элемента, определяющие размер и приручённость. Так как размер животного и степень его приручённости могут быть разными и иметь промежуточные значения, то давайте немного отойдем от принятого определения S-элемента. Представим, что он может выдавать не только 0 (маленький размер, совсем дикий) или 1 (большой, полностью ручной), но и другие значения.

Так как есть два сенсорных элемента, то их значения можно расположить вдоль двух координатных осей. На получившейся координатной плоскости можно размещать точки, каждая из которых характеризует какой-то вид кошки или собаки. Как вы знаете, у каждой точки есть координаты. В нашем случае их две: размер и приручённость. Так вот, задачей персептрона в данном случае (основанного на двух S-элементах) – провести некоторую прямую, которая максимально точно разделит два множества точек (кошек и собак). На рисунке 22 видно 4 этапа обучения сети на все более большой обучающей выборке.

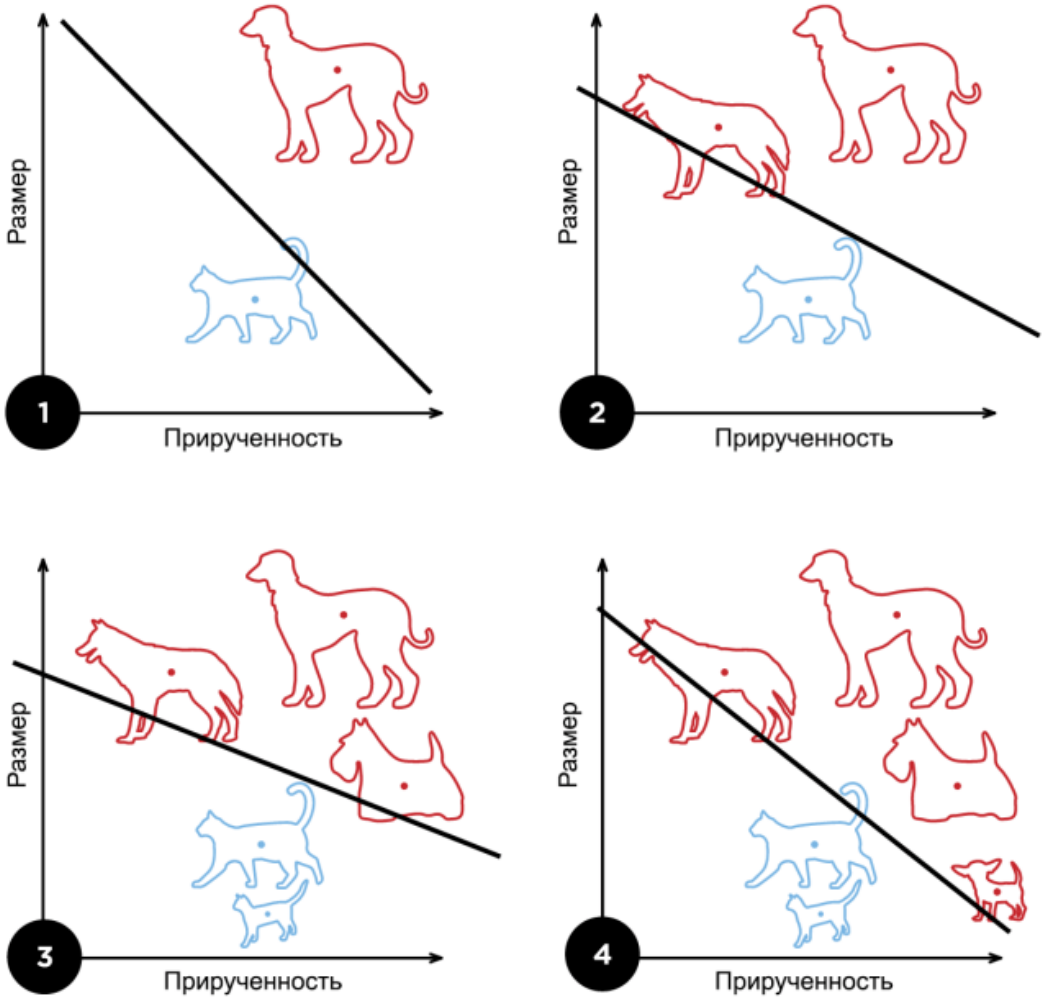

Рисунок 22 - Этапы обучения сети

Естественно, может быть больше признаков, а значит, и больше сенсорных элементов. В случае трех признаков будет три S-элемента, то есть имеем уже трехмерное пространство. В таком случае, между точками, каждая из которых соответствует определенным значениям всех трех S-элементов, проводилась бы плоскость. И так далее. В общем случае для n S-элементов в n-мерном пространстве строится так называемая гиперплоскость с размерностью n-1.

Вы заметили, что в картинках выше в качестве разделителя используется прямая? А ведь мы могли бы в качестве разделителя использовать и любую другую кривую. Но прямая - проще. Именно поэтому рассматриваем задачи на классификацию линейно разделимых объектов. Именно такие задачи способны решать элементарные персептроны.

Бывают и такие множества объектов, которые невозможно разделить линией (плоскостью/гиперплоскостью). В общем случае, такие множества называют линейно неразделимыми. Теоремы о сходимости элементарных персептронов на них не распространяются.

7.1 Задача на классификацию

Можно ли классифицировать логические функции? Да, и к тому же эта задача отлично проиллюстрирует такую классификацию.

Что такое логические функции? Это функции от какого-то числа переменных. Причем как сами переменные, так и значения логических функций могут принимать только фиксированные (дискретные) значения: 0 или 1.

Начнем с логического «И». Вы отправили Сашу в магазин за продуктами. Ему надо купить хлеб и квас. Если он ничего не купил, вы не пускаете его домой. Если он купил только хлеб или только квас, вы не пускаете его домой. Другими словами, Саша может войти в дом только когда он купил хлеб **и** квас. Также работает и логическое «И». У нас есть две

бинарные переменные (то есть они могут быть равны только 0 или 1). Значением функции логического «И» будет 1 только тогда, когда значения обеих переменных тоже равны 1. Во всех остальных случаях значение этой логической функции равно 0.

Для того чтобы лучше понимать принцип работы логической функции, часто используют таблицы истинности, где в первых двух столбцах располагают возможные комбинации переменных, а в третьем значение функции в данном случае.

Конъюнкция

A	B	A & B
0	0	0
1	0	0
0	1	0
1	1	1

Рисунок 23 - Таблица истинности для логического "И"

А есть еще логическое «ИЛИ». Снова посылаем Сашу в магазин за продуктами. Ему надо купить хлеб и квас. Если он ничего не купил, вы не пускаете его домой. Если он купил только хлеб или только квас, или оба продукта – вы пускаете его домой. Также работает и логическое «ИЛИ». Значением функции логического «ИЛИ» будет 0 только тогда, когда значения обеих переменных тоже равны 0. Во всех остальных случаях значение этой логической функции равно 1.

Таблица истинности для логического ИЛИ выглядит следующим образом.

Дизъюнкция

A	B	A || B
0	0	0
1	0	1
0	1	1
1	1	1

Рисунок 24- Таблица истинности для логического "ИЛИ"

Логические функции очень хорошо иллюстрируют идею классификации. Любая такая функция принимает на вход два аргумента. Но логические функции могут принимать только дискретные аргументы (0 или 1). В итоге получаем, что для изображения любой логической функции на плоскости достаточно 4 точки (с координатами (0,0) (1,0) (0,1) (1,1)), это изображено на рисунке 25:

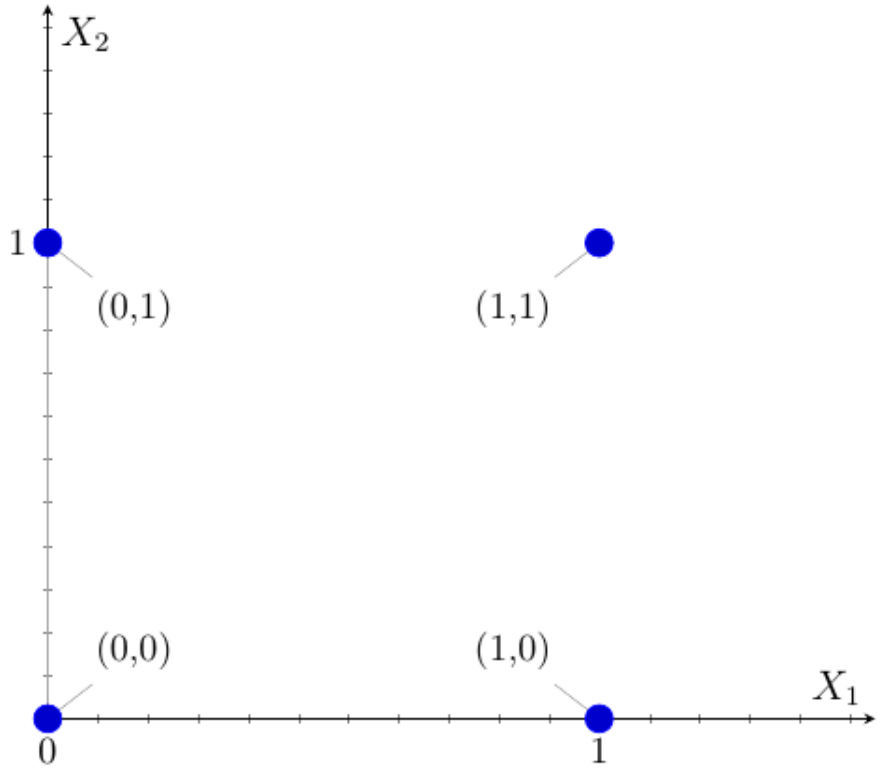

Рисунок 25 - Логическая функция на плоскости

Рассмотрим логическую функцию И. Она равна нулю для любого набора входных аргументов, кроме набора (1,1).

Налицо задача классификации: у нас есть 4 точки. Мы должны провести прямую так, чтобы по одну сторону у нас оказались точки, для которых значения логического «И» равно 1, а по другую, для которых это значение равно 0.

В случае с логическим «И» эту прямую, например, можно провести так, как показано на рисунке 26. Все точки, находящиеся под этой прямой, приводят к 0 значению этой функции. Единственная точка над этой прямой приводит к значению логического «И», равному 1.

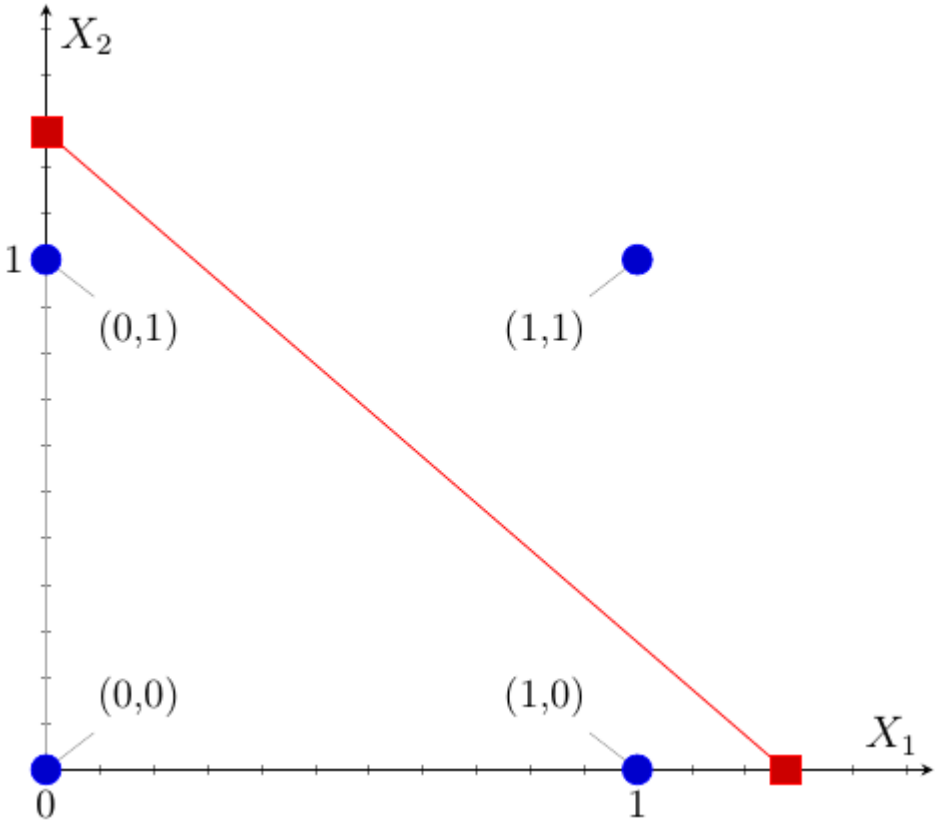

Рисунок 26 - Графическое значение функции в случаи логического "И"

Похожим образом ведет себя логическое ИЛИ.

Для такой функции графическое представление изображено на рисунке 27:

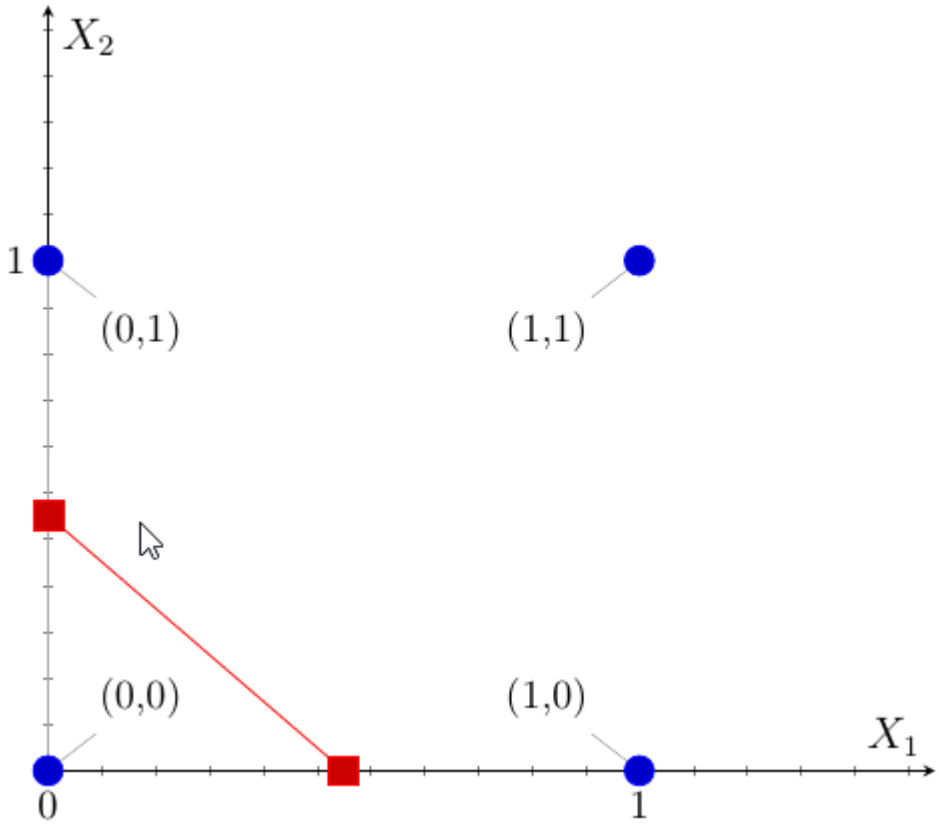

Рисунок 27 - Графическое значение функции в случаи логического "ИЛИ"

Данный рисунок представляет собой графическое представление логического «И», но наоборот (тоже одна точка, но для которой значение функции равно 0 и уже под прямой).

Переходим к краеугольному камню нейронных сетей — их обучению. Ведь без этого свойства они не имеют никакого смысла.

8 Обучение персептронов

Под обучением нейронной сети имеется ввиду процесс корректировки весовых коэффициентов связи таким образом, чтобы в результате при

поступлении на вход сети определенного сигнала она выдавала нам правильный ответ.

8.1 Упрощаем до предела

Начнем обучение наших нейронных сетей с самого простого случая. Для этого мы сильно упростим и без того простой однослойный персептрон с одним скрытым слоем:

1. Будем считать, что его A-R связи могут принимать только целые значения (…, -2, -1, 0, 1, 2, …).

2. У каждого A-элемента может быть только один S-элемент.

3. Будет только 1 R-элемент.

На словах такое большое количество упрощений может выглядеть сложно. Поясним сказанное на схеме. Рассмотрим рисунок 28 однослойного персептрона с одним скрытым слоем и преобразуем ее.

Изначально мы имеем следующий персептрон.

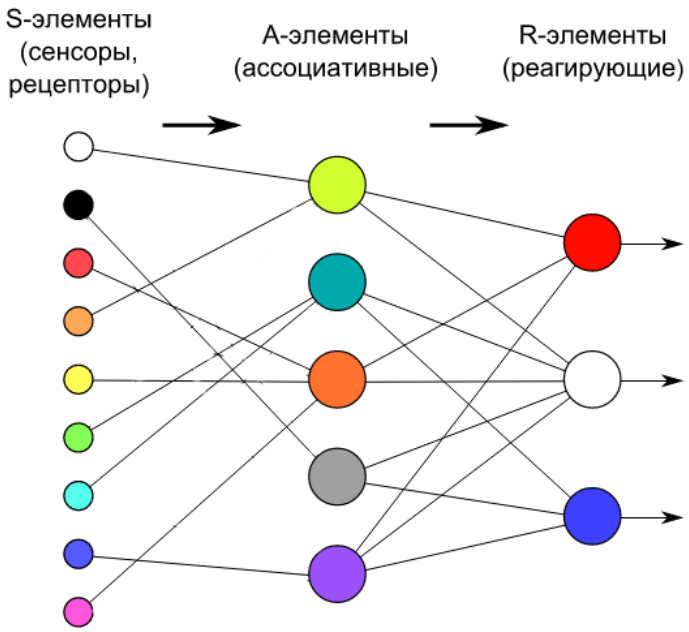

Рисунок 28 - Однослойный персептрон

Мы должны упростить его. Теперь А-элементы могут быть соединены только с одним S-элементом. Убираем все лишние связи.

На рисунке выше 3 R-элемента. Оставляем только один.

S-A веса и пороги А элементов у нас теперь равны +1. Отмечаем это на рисунке 29.

В итоге получаем следующий рисунок.

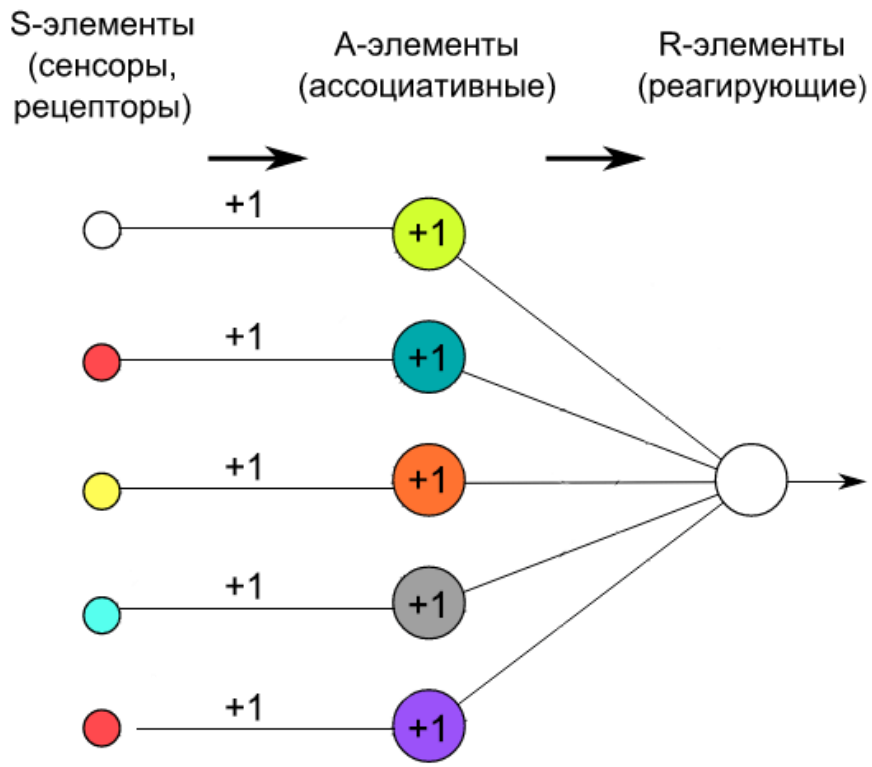

Рисунок 29 – Этап упрощения однослойного персептрона

Однако получается, что слой A-элементов не выполняет никакие функции. Он эквивалентен S-слою. Поэтому проводим следующее упрощение. Выбрасываем слой сенсоров. Теперь роль сенсоров у нас будут выполнять ассоциативные элементы .

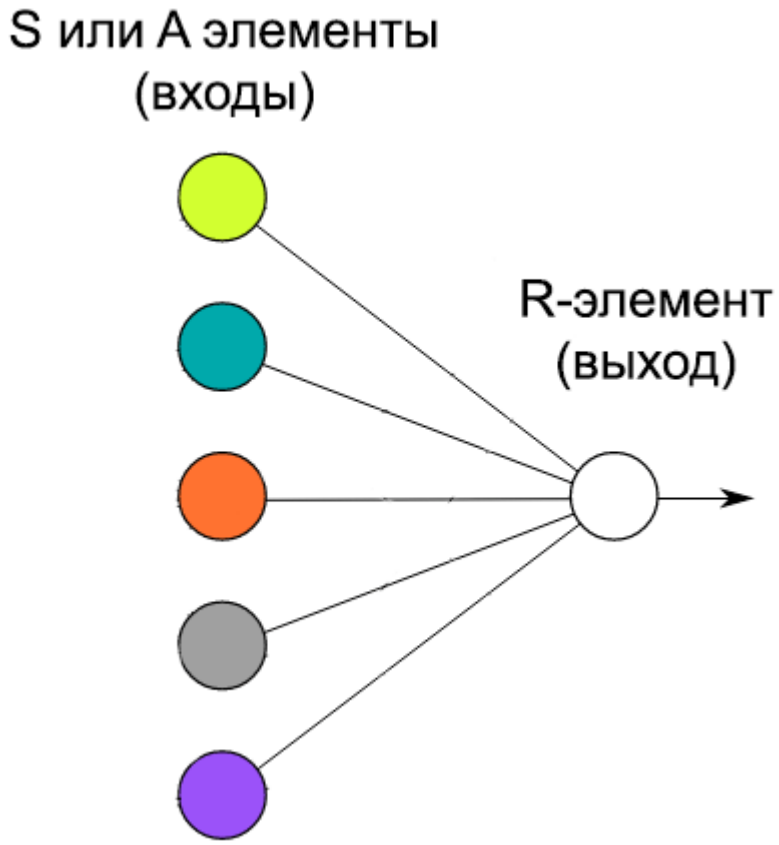

Рисунок 30 - Упрощённый однослойный персептрон с одним скрытым слоем

В итоги, мы только что очень упростили однослойный персептрон с одним скрытым слоем (рисунок 30).

Даже в таком кастрированном виде нейронная сеть будет работать и даже решать задачи на классификацию.

Сначала попробуем научить такую простую модель решать простейшую задачу на классификацию. Например, распознавание цифр.

9 Интерпретатор shell

Программирование на языке интерпретатора shell приобретает все большую популярность по мере утверждения Linux в качестве удобной в работе и отказоустойчивой операционной системы. Трудно оценить, какое количество пользователей работают с Linux. Эта операционная система распространяется бесплатно, хотя многие компании разрабатывают ее коммерческие варианты. Кроме того, несмотря на сделанные несколько лет назад неутешительные прогнозы специалистов относительно будущего UNIX, данная ОС также не теряет популярности, и число ее приверженцев продолжает расти.

Shell - это командная оболочка. Но это не просто промежуточное звено между пользователем и операционной системой, это еще и мощный язык программирования. Программы на языке shell называют сценариями, или скриптами. Фактически, из скриптов доступен полный набор команд, утилит и программ UNIX. Если этого недостаточно, то к вашим услугам внутренние команды shell - условные операторы, операторы циклов и т.д., которые увеличивают мощь и гибкость сценариев. Shell-скрипты исключительно хороши при программировании задач администрирования системы и др., которые не требуют для своего создания полновесных языков программирования.

Знание языка командной оболочки является залогом успешного решения задач администрирования системы. Даже если вы не предполагаете заниматься написанием своих сценариев. Во время загрузки Linux выполняется целый ряд сценариев из /etc/rc.d, которые настраивают конфигурацию операционной системы и запускают различные сервисы, поэтому очень важно четко понимать эти скрипты и иметь достаточно знаний, чтобы вносить в них какие либо изменения.

Язык сценариев легок в изучении, в нем не так много специфических операторов и конструкций. Синтаксис языка достаточно прост и прямолинеен, он очень напоминает команды, которые приходится вводить в командной строке. Короткие скрипты практически не нуждаются в отладке, и даже отладка больших скриптов отнимает весьма незначительное время.

Shell-скрипты очень хорошо подходят для быстрого создания прототипов сложных приложений, даже не смотря на ограниченный набор языковых конструкций и определенную "медлительность". Такая метода позволяет детально проработать структуру будущего приложения, обнаружить возможные "ловушки" и лишь затем приступить к кодированию на C, C++, Java, или Perl.

Скрипты возвращают нас к классической философии UNIX - "разделяй и властвуй" т.е. разделение сложного проекта на ряд простых подзадач. Многие считают такой подход наилучшим или, по меньшей мере, наиболее эстетичным способом решения возникающих проблем, нежели использование нового поколения языков "все в одном".

Название BASH - это аббревиатура от "Bourne-Again Shell" и игра слов от, ставшего уже классикой, "Bourne Shell" Стефена Бурна (Stephen Bourne). В последние годы BASH достиг такой популярности, что стал стандартной командной оболочкой de facto для многих разновидностей UNIX. Большинство принципов программирования на BASH одинаково хорошо применимы и в других командных оболочках, таких как Korn Shell (ksh), от которой BASH позаимствовал некоторые особенности, и C Shell и его производных.

10 Реализация нейронной сети: распознавание цифр

Картинка, как всем известно, состоит из пикселей. Следовательно, очень удобно на каждый пиксель выделять один сенсор (для его распознавания).

Запрограммируем простейшую нейросеть, которая будет распознавать цифры. На самом деле это трудная задача, поэтому мы ее упростим:

- Будем распознавать только черно-белые цифры от 0 до 9.
- Цифры будут состоять из черных квадратиков в табличке 3x5 квадратов.
- Распознать нейросеть должна будет научиться только одну цифру.

На рисунке 31 изображены цифры для нашей программы.

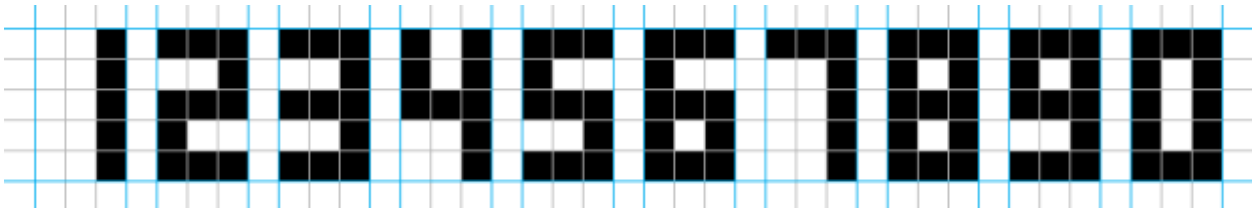

Рисунок 31 - Изображение цифр в формате 3x5

В сети будет по 1 S-элементу (он же A-элемент) на каждый квадратик из таблицы. Поэтому для распознавания цифры нам потребуется 15 сенсоров. Черный цвет квадрата соответствует возбуждению S-элемента (значение передаваемого сигнала равно 1). Белый цвет – выход соответствующего S-элемента равен 0.

10.1 Цифры в строковом формате

Чтобы работать с нейросетью, необходимо на ее входы подавать сигналы в виде чисел (0 или 1). Таким образом изображение цифры должны перевести в последовательность сигналов в виде чисел. Это легко сделать, если представить цифры в строковом формате.

Каждая цифра представляет собой всего пятнадцать квадратиков, причем только двух возможных цветов. Как оговаривалось ранее, за белый квадратик отвечает 0, а черный квадратик – 1. Поэтому наши десять цифр от 0 до 9 в строковом формате будут выглядеть, как представлено на рисунке 32:

Рисунок 32 - Представление цифр в строковом формате

Для записи каждой цифры у нас используется по 5 строк с 3 символами в каждой. Необходимо убрать все переносы строк, чтобы получить для каждой цифры от 0 до 9 одну длинную строку длиной в 15 символов.

Цифры в таком строковом формате уже можно использовать для работы с нейросетью.

Почему мы не используем картинки, а перешли к строчкам символов? Взгляните на рисунок 33.

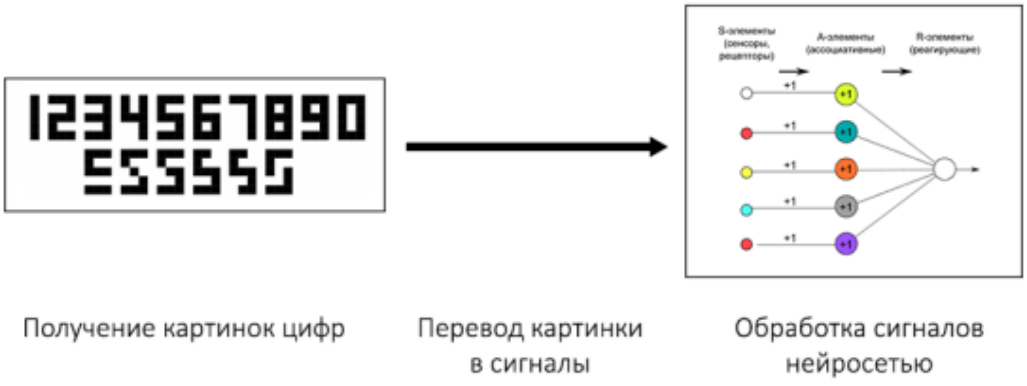

Рисунок 33 - Распознавание цифр

Если бы намеревались сделать готовый продукт, принимающий и узнающий картинки цифр, то нам пришлось бы сначала написать отдельную подпрограмму, анализирующую по квадратам полученную на входе картинку, переводящую эти квадраты и их цвет в набор символов, а только потом уже передавать этот набор в нейросеть. Плюс ко всему понадобилось бы создать еще и красивый интерфейс программы.

Поэтому мы пропустим первые два этапа, указанные на картинке, и будем сразу подавать на вход нейросети цифры в строковом формате.

10.2 Постановка задачи

Необходимо создать программу, которая из всех 10 цифр будет распознавать нужную нам цифру. Например, пусть это будет цифра 5 (можно и любую другую).

Нашей обучающей выборкой будут все цифры от 0 до 9. Когда нейросеть обучиться безошибочно распознавать нужную нам цифру (5), тогда мы проверим ее «интеллект» уже на тестовой выборке. Она будет уже похитрее: на вход будут подаваться уже искаженные изображения пятерки.

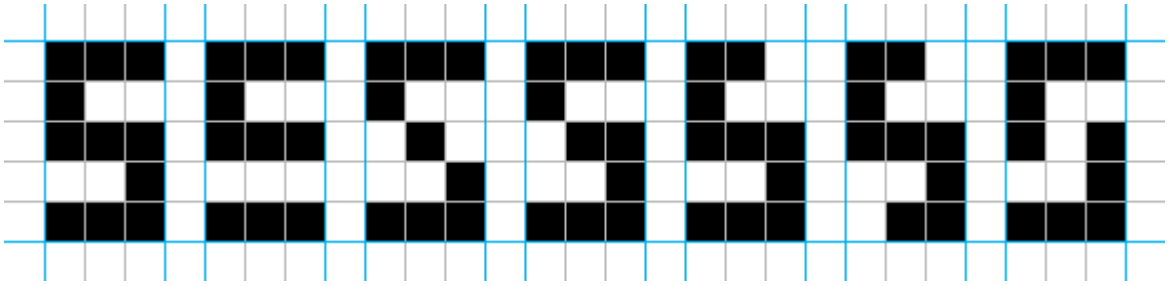

Рисунок 34 - Искажённое изображения цифры пять

Посмотрим, сможет ли обученная нейросеть с высокой точностью распознавать пятерку?

По аналогии, искаженные изображения пятерки тоже должны преобразовать в строковый формат.

10.3 Алгоритм обучения

Дошли до главного: как обучать сеть. Необходимо случайным образом выбирать цифру и прогонять ее через сеть, модифицируя ее веса. Но как их модифицировать?

Знаем, что важность тем или иным входам (в нашем случае - S-элементам) придают веса, связывающие их с R-элементом. Таким образом, чем сильнее повлиял какой-то вес связи на результат, тем сильнее надо его изменить.

Следовательно, мы должны учесть следующие важные моменты:

- Если наша нейросеть правильно распознала/отвергла цифру 5, то мы ничего не предпринимаем.
- Если нейросеть ошиблась и распознала неверную цифру как 5, то мы должны ее наказать – мы уменьшаем веса тех связей, через которые прошел сигнал. Другими словами, веса, связанные с возбудившимися входами, уменьшаются.
- Если нейросеть ошиблась и не распознала цифру 5, то мы должны увеличить все веса, через которые прошел сигнал. Таким образом мы как бы говорим сети, что такие связи, а значит и связанные с ними входы - правильные.

Алгоритм обучения, который будет реализовываться в программе:

1. Подать на входы нейросети цифру в строковом формате.
2. Если цифра распознана/отвергнута верно, то перейти к шагу 1.

3. Если сеть ошиблась и распознала неверную цифру как 5, то вычесть из всех связей, связанных с возбудившимися S-элементами единицу.
4. Если сеть ошиблась и отвергла цифру 5, то добавить единицу ко всем связям, связанным с возбудившимися S-элементами.

Почему в алгоритме мы прибавляем или отнимаем именно 1? На самом деле эту величину можно задать любой. Понятно, что эта величина влияет на эффективность обучения.

10.4 Программа

Программа для реализация нашей нейронной сети, будет написана на языке программирования BASH.

Совершенно не обязательно писать нейросети на BASH. Искусственные нейросети – математическая модель, и их можно запрограммировать с помощью любого языка. Так что если вам больше по душе Java, C, C#, Python… то можете реализовывать сети на них. Никаких принципиальных различий нет. Алгоритм один и тот же. Реализация на разных языках разная.

Для начала запишем все цифры от 0 до 9. Просто записывайте цифру в 5 строк по 3 символа, а затем удаляйте переносы строк

```
area_num_0 = ( 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 )

area_num_1 = ( 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 )

area_num_2 = ( 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 )

area_num_3 = ( 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 )

area_num_4 = ( 1 0 1 1 0 1 1 1 1 0 0 1 0 0 1 )

area_num_5 = ( 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 )

area_num_6 = ( 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 )
```

area_num_7 = (1 1 1 0 0 1 0 0 1 0 0 1 0 0 1)

area_num_8 = (1 1 1 1 0 1 1 1 1 1 0 1 1 1 1)

area_num_9 = (1 1 1 1 0 1 1 1 1 0 0 1 1 1 1)

Функция area_name позволяет нам создать список (массив), состоящий из отдельных символов, на которые разбивается длинная строка.

Теперь запишем 6 видов искаженной пятерки в строковом формате.

area_num_51 = (1 1 1 1 0 0 1 1 1 0 0 0 1 1 1)

area_num_52 = (1 1 1 1 0 0 0 1 0 0 0 1 1 1 1)

area_num_53 = (1 1 1 1 0 0 0 1 1 0 0 1 1 1 1)

area_num_54 = (1 1 0 1 0 0 1 1 1 0 0 1 1 1 1)

area_num_55 = (1 1 0 1 0 0 1 1 1 0 0 1 0 1 1)

area_num_56 = (1 1 1 1 0 0 1 0 1 0 0 1 1 1 1)

Теперь нам необходимо создать список весов. В теореме об обучении персептронов сказано, что сеть из любого состояния может обучиться правильно отвечать. Так что, чтобы не загадывать, пусть все веса вначале у нас будут равны 0. Так как у нас 15 входов и все они сразу соединены с одним R-элементом, то нам потребуется 15 связей.

area_weights = (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)

Так как мы используем пороговую функцию активации, то нам необходимо установить какое-то число (порог). Если взвешенная сумма будет больше или равна ему, то сеть выдаст 1 (что означает, что она считает предоставленную цифру пятеркой). Порог можно выбрать любой. Пусть будет равен 7.

bias = 7

Теперь создадим простейшую функцию, которая будет считать взвешенную сумму и сравнивать ее с порогом. Фактически эта функция представляет собой единичный шаг работы нашей нейронной сети.

```
net = 0
LIMIT_1=14
for ((a=0; a <= LIMIT_1 ; a++))
{
        net = $(expr ${area_num[i]} * ${area_weights[i]})
}
net >= bias
```

Результатом работы выражения net >= bias этой функции может быть True (Правда/Да), что означает 1 или False (Ложь/Нет), что означает 0.

Теперь определим еще две вспомогательные функции.

Первая функция вызывается, когда сеть считает за 5 неверную цифру (выход равен 1 при демонстрации не 5) и уменьшает на единицу все веса, связанные с возбужденным входами. Напомню, что мы считаем вход возбужденным, если он получил черный пиксель (то есть 1).

```
decrease()
{
LIMIT_1 = 15
for ((a=0; a <= LIMIT_1 ; a++))
        {
        if ( {area_num[a]} == 1 )
```

```
                {
                        area_weights[a] = $(area_weights[a]  --)
        }
        }
}
```

Вторая функция вызывается, если сеть не смогла распознать цифру пять (выход равен 0 при демонстрации 5) и увеличивает на единицу все веса, связанные с возбужденными входами.

```
increase()
        {
                LIMIT_1 = 15
                for ((a=0; a <= LIMIT_1 ; a++))
                {
                 # Возбужденный ли вход
                if ( {area_num[a]} == 1 )
                        {
                                area_weights[a] = $(area_weights[a] ++)
                        }
                }
}
```

Теперь начнем обучать нашу нейронную сеть. Делать мы это будем в цикле, который будет повторятся достаточно большое количество раз. Для примера возьмем 10 000 раз. Действия за каждый этап повторения мы описали выше в алгоритме обучения.

```
LIMIT_2=10000
for ((a=1; a <= LIMIT_2 ; a++))
{
        i = $( shuf -i 0-9 )
        if ( i -ne 5 )
        {
                while [ "$i" -le bias ]
                do
                decrease
        }
        else
        {
                until [ "$i" -le bias ]
                do
                increase
        }
}
```

Осталось только вывести результаты обучения. Выводить их мы будем следующим образом. Сначала выведем результирующие значения весов. Героев надо знать в лицо. После этого еще раз прогоним сеть по всем цифрам от 0 до 9, но уже без обучения. Просто чтобы удостоверится, что она обучилась. А в конце опробуем сеть на нашей тестовой выборке.

```
# Вывод значений весов
echo ' Значений весов $(area_weights)'
# Прогон по обучающей выборке
        echo ' 0 это 5? $(proceed area_num_0)'
        echo ' 1 это 5? $(proceed area_num_1)'
        echo ' 2 это 5? $(proceed area_num_2)'
        echo ' 3 это 5? $(proceed area_num_3)'
        echo ' 4 это 5? $(proceed area_num_4)'
        echo ' 6 это 5? $(proceed area_num_6)'
        echo ' 7 это 5? $(proceed area_num_7)'
        echo ' 8 это 5? $(proceed area_num_8)'
        echo ' 9 это 5? $(proceed area_num_9)'
# Прогон по тестовой выборке
        echo ' Узнал 5? $(proceed area_num_5)'
        echo ' Узнал 5 - 1? $(proceed area_num_51)'
        echo ' Узнал 5 - 2? $(proceed area_num_52)'
        echo ' Узнал 5 - 3? $(proceed area_num_53)'
        echo ' Узнал 5 - 4? $(proceed area_num_54)'
        echo ' Узнал 5 - 5? $(proceed area_num_55)'
        echo ' Узнал 5 - 6? $(proceed area_num_56)'
```

Результаты - сеть стала распознавать пятерку во все случаях только с 3 попытки (рисунок 35).

```
[1, 1, 1, 3, 0, -8, 1, 2, 1, -8, 0, 1, 1, 1, 1]
0 это 5?  False
1 это 5?  False
2 это 5?  False
3 это 5?  False
4 это 5?  False
6 это 5?  False
7 это 5?  False
8 это 5?  False
9 это 5?  False

Узнал 5?  True
Узнал 5 - 1?  True
Узнал 5 - 2?  True
Узнал 5 - 3?  True
Узнал 5 - 4?  True
Узнал 5 - 5?  True
Узнал 5 - 6?  True

Process finished with exit code 0
```

Рисунок 35 – Результаты программы

Не обязательно сеть научится распознавать пятерки из тестовой выборки с первой попытки. Если результат программы не устраивает (не вся тестовая выборка успешно прошла проверку), то просто запустите программу снова. Все будет сброшено и начнется с самого начала.

Но почему сеть не обучается с первого раза?

При любом запуске программы 10 тысяч обучающих цифр генерируются случайным образом. Таким образом при каждом запуске программы процесс обучения сети уникален.

Из-за случайности генерации обучающих цифр они могут подаваться на вход неравномерно, а значит обучение будет идти однобоко. Действительно, если показывать только одну или две цифры, то сеть ничему не научиться.

Чтобы сеть гарантированно научилась распознавать нужную нам цифру надо соблюсти два условия:

1. Добиться равномерности показа всех обучающих цифр.
2. Увеличить общее количество шагов обучения (50 тысяч или 100 тысяч).

Рассмотрим мой результат. Начнем с первой строчки (это веса сети):

[1, 1, 1, 3, 0, -8, 1, 2, 1, -8, 0, 1, 1, 1, 1]

Теперь расположим эти цифры в виде «цифры».

1 1 1

3 0 -8

1 2 1

-8 0 1

1 1 1

Если приглядеться, то можно заметить цифру пять. Важно то, что все квадраты, составляющие силуэт пятерки положительные. Некоторые даже больше 1 (3 и 2). В результате получили слепок весов нашей сети, наложенной на цифру пять.

Предположим, что цифра указывает яркость конкретного квадрата (рисунок 36).

Рисунок 36 - Значение весов с использование цвета

Более темные квадраты уверяют нашу нейросеть, что предлагаемая цифра несомненно является пятеркой и вносят большой вклад в взвешенную сумму (3 и 2). Соответственно если в предлагаемой сети картинке эти квадраты черные, то взвешенная сумма наверняка будет больше порога. Серые области вносят обычный вклад (по 1) в взвешенную сумму. Красные квадратики вызывают отвращение у сети и их появление на входах приводит к отрицательным значениям (-8), что сильно снизит взвешенную сумму и она будет гарантированно ниже порога.

Дальше идет 9 строк обучающей выборки.

0 это 5? False

1 это 5? False

2 это 5? False

3 это 5? False

4 это 5? False

6 это 5? False

7 это 5? False

8 это 5? False

9 это 5? False

Все предложенные цифры наша нейросеть смело отвергает, клеймя их False.

Итоговый результат программы - тестовая выборка. Как видно, сеть прекрасно распознала и цифру 5, и шесть ее искаженных вариаций.

Узнал 5? True

Узнал 5 — 1? True

Узнал 5 — 2? True

Узнал 5 — 3? True

Узнал 5 — 4? True

Узнал 5 — 5? True

Узнал 5 — 6? True

Тем самым, даже на упрощенном однослойном персептроне с одним скрытым слоем, который, представляет собой фактически один искусственный нейрон, мы смогли программно добиться эффекта обобщения. Сеть никогда не видела искаженные пятерки. Но она их узнала.

Напомню, что мы рассматривали упрощенный вариант сети, связи и входы которой могли быть только целыми числами. Название алгоритма обучения, который мы использовали: правила Хебба. Всего этих правил два, и они соответствуют двум пунктам уже описанного выше алгоритма обучения.

Правила Хебба (*Hebb's rule, Hebbian learning rule*) – два правила, составляющие алгоритм обучения персептронов для решения простейших задач классификации, когда входы могут быть равны только 0 или 1:

1 Правило. Если сигнал персептрона неверен и равен 0, то необходимо увеличить веса тех входов, на которые была подана единица.

2 Правило. Если сигнал персептрона неверен и равен 1, то необходимо уменьшить веса тех входов, на которые была подана единица.

11 Заключение

В данной выпускной квалификационной работе была создана однослойная нейронная сеть с одним скрытым слоем. Целью данной выпускной квалификационной работы является исследование возможного построения нейронной сети с использование стандартных утилит операционной системы в частности командной оболочки Bourne shell (BASH).

Актуальность этой работы заключается в том, что, необходимо было создать нейронную сеть в среде разработки, которая не требует для своего создания полновесных языков программирования, где все операции выполняется непосредственно операционной системой.

Результатом выпускной квалификационной работы является реализация программы, которая отвечает требованиям ТЗ, а значит построение локальных систем управления в среде разработки BASH возможно.

12 Библиографический список

1. Галушкин А. И. Теория нейронных сетей. Кн.: Учебное пособие для вузов – М.: ИПРЖР, 2000. -416 с

2. Штабов С. Д. Проектирование нечетких систем средствами MATLAB. – М.: Горячая линия – Телеком, 2007. -288с.

3. Агеев А. Д., Балухто А. Н., Бычков А. В., и др. Нейроматематика. Кн. 6: Учеб. Пособие для вузов – М.: ИНПРЖР, 2002. -448 с.

4. Минаев Ю. Н., Филимонова О. Ю., Бенамеур Лиес. Методы и алгоритмы решения задач идентификации и прогнозирования в условиях неопределенности в нейросетевом логическом базисе. – М.: Горячия линия – Телеком, 2003. – 205с.

5. Тейнсли Д. Unix і UNIX: программирование в shell. Руководство разработчика: Пер. с англ. — К.: Издательская группа BHV, 2001. — 464 с.

6. Киселев А. Advanced Bash-Scripting Guide. Пер. с англ.