Машинное обучение

Нейроны, нейросетевые подходы, модели

Практическое применение

- обработка данных (в широком смысле слова);
- экономика и бизнес (например, прогнозирование рынка валют, цен);
- медицина (диагностика, создание лекарств, обработка изображений);
- авионика (автопилоты, детекторы радаров, БПЛА, дроны);
- связь (сжатие видео, кодирование, оптимизация, маршрутизация);
- рекомендательные системы (ассистенты, боты, новости, реклама);
- автоматизация (производство, контроль, мониторинг, устойчивость);
- робототехника (распознавание сцены, объектов, маршрут);
- социология, политология (анализ, прогноз, кластеризация);
- охранные системы (идентификация, обнаружение вторжений);
- компьютерные игры (боты, AI) и многое другое…

История

40-е гг. XX века работы Уоррена Мак-Каллока и Уолтера Питтса, заложившие основы исследований нейронных сетей (изучение биологических процессов мозга, применение НС для создания ИИ);

Конец 40-х г. Дональд Хебб выдвинул теорию обучения ИНС на основе нейронной пластичности (способности к восстановлению клеток и связей);

1962 г. Фрэнк Розенблатт предложил перцептрон для классификации символов;

1965 г. первый нейрокомпьютер Розенблатта “Марк-1” на основе перцептрона;

60-е гг. Александр Петров и Михаил Бонгард занимаются изучением сферы практического применения перцептрона;

1969 г. Марвин Ли Мински и Сеймур Паперт публикуют доказательство того, что возможности перцептрона имеют существенные ограничения;

1982 г. Джон Джозеф Хопфилд достиг двусторонней передачи информации между нейронами и изобрел ассоциативную нейронную сеть;

80-е гг. получен мощный “технологический импульс” в развитии теории нейронных сетей благодаря появлению ПЭВМ.

История

2009 г. модель мозга кошки, правда медленнее в 643 раза =(^,^)=.

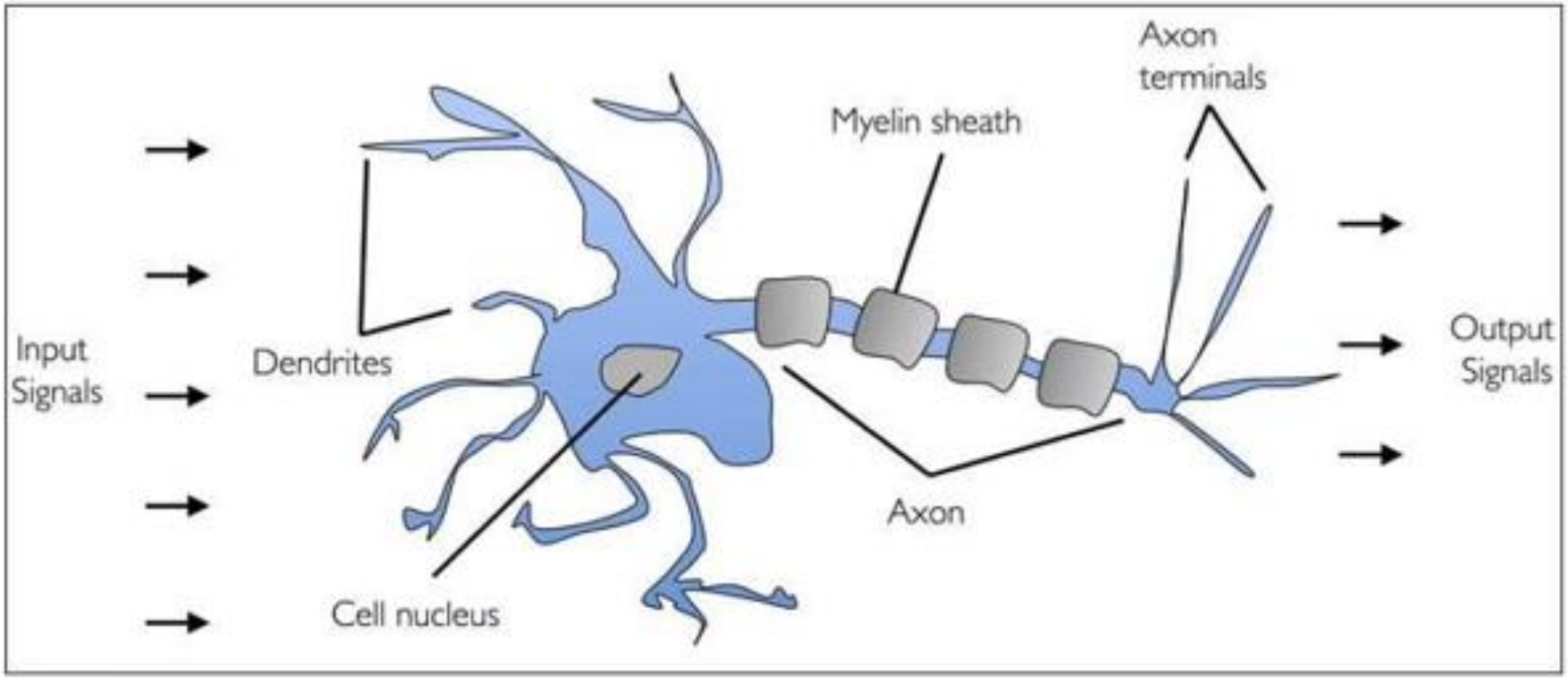
2011 г. в IBM создают первый нейропроцессор 256 нейронов и 262144 синапсов.

2012 г. симуляция нейрокомпьютера для моделирования деятельности мозга человека: 530 миллиардов нейронов и 137 триллионов синапсов (в 1542 раза медленнее реального времени: 1 572 864 процессорных ядер и 1.5 петабайта памяти.

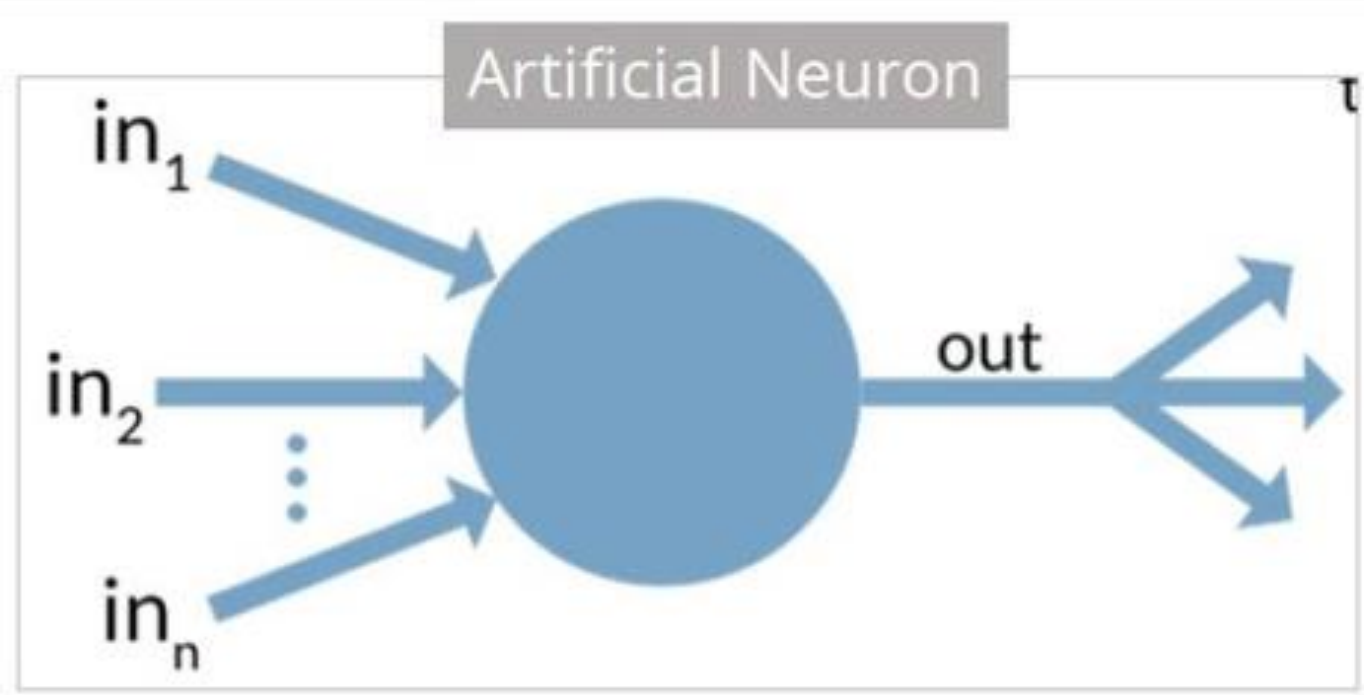
2014 г. IBM представила первый нейросинаптический процессор для архитектуры TrueNorth: 1 миллион программируемых нейронов в 4 096 нейросинаптических вычислительных ядер, 256 миллионов синапсов, 5.4 миллиарда транзисторов (70 МВт).

Этот чип делает 46 миллиардов синаптических операций в сек. на 1 Вт, но человеческий мозг имеет более 85 миллиардов нейронов и более 100 триллионов синапсов. Теоретически, имея 85 тысяч таких процессоров, можно воссоздать деятельность человеческого мозга…

Понятие нейрона и перцептрона



“Математический” нейрон



“Математический” нейрон

Искусственный нейрон-это математическая функция, основанная на модели биологических нейронов, где каждый нейрон принимает входные данные, взвешивает их отдельно, суммирует и передает эту сумму через нелинейную функцию для получения выходных данных.

Биологический нейрон аналогичен искусственным нейронам в следующих терминах:

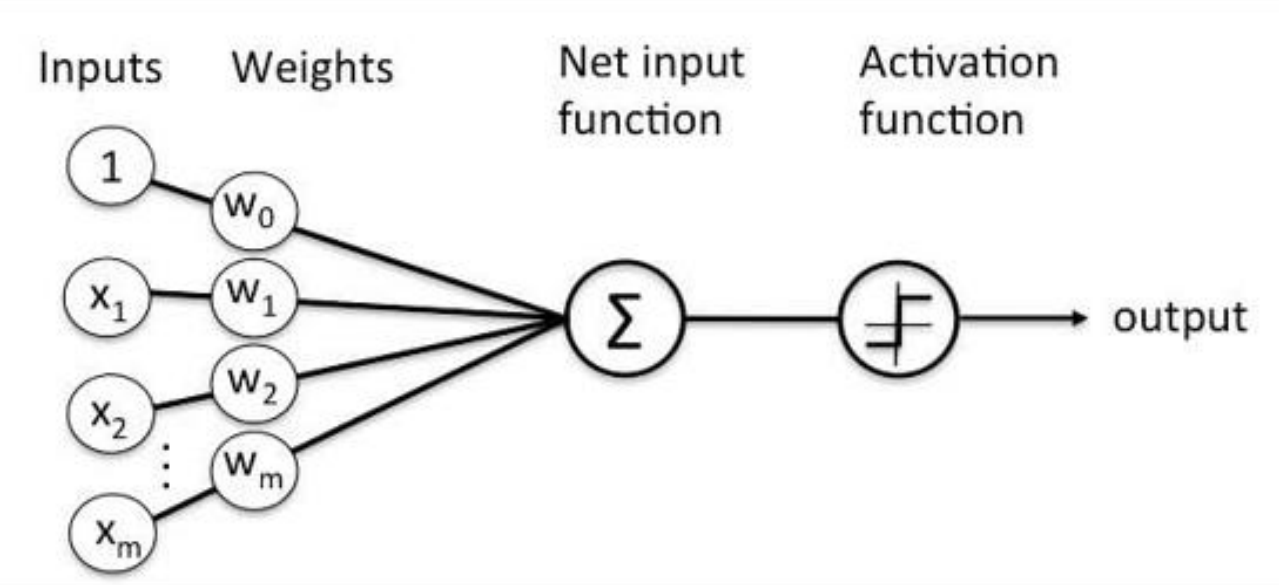
- Клеточное ядро или сома называется Узел.
- Дендриты называются Входами.
- Синапсы называются Весами или Связями.
- Аксоны называются Выходами.

Характеристики нейрона

- простейший нейрон - просто математическая функция;
- это элементарная единица и строительный блок любой искусственной нейронной сети;
- один или несколько входов всегда отдельно “взвешиваются”;
- входные данные суммируются и передаются через нелинейную функцию для получения выходных данных;
- каждый нейрон имеет внутреннее состояние (сигнал активации);
- каждое соединение содержит информацию о входном сигнале;
- каждый нейрон соединен с другими нейронами по каналам связи.

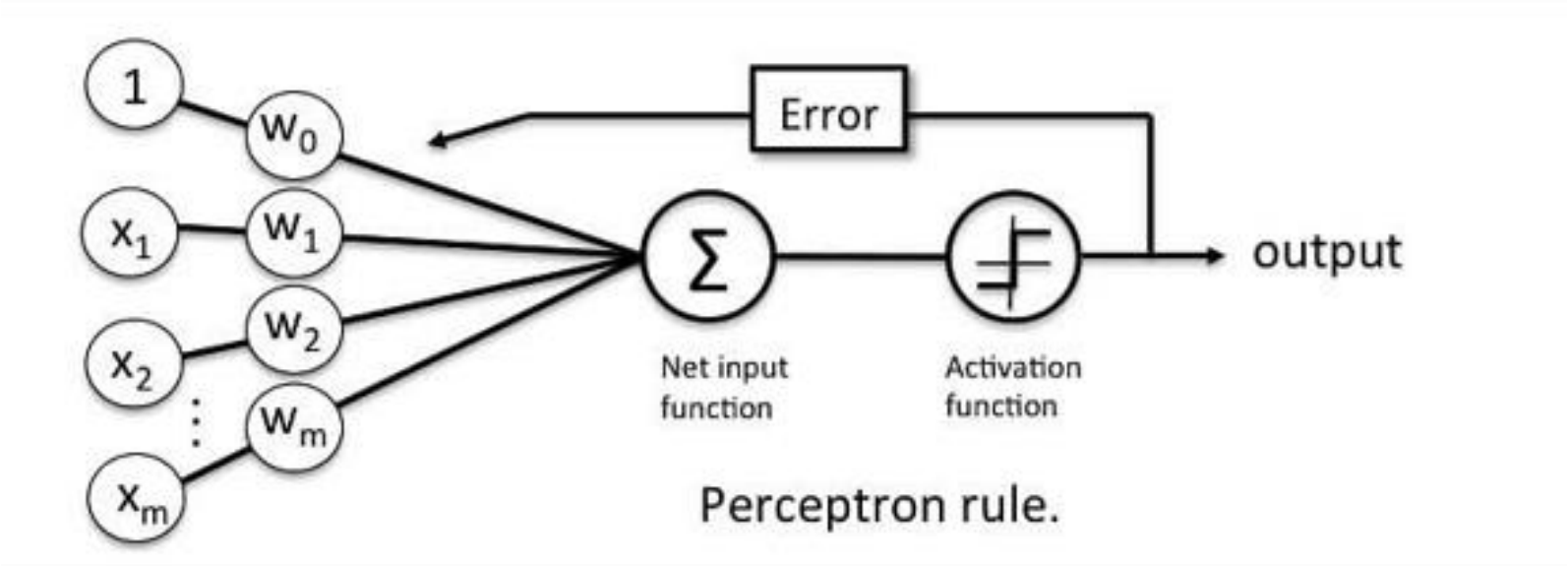
“Формальный” перцептрон

Перцептрон представлен Фрэнком Розенблаттом в 1957 г. Это “алгоритм” контролируемого обучения двоичных классификаторов.



Типы перцептронов

Однослойные (различают только линейно разделяемые паттерны), многослойные или нейронные сети с двумя или более слоями.



Суть расчета

- перцептрон принимает несколько входных сигналов и, если их взвешенная сумма превышает определенный порог, то перцептрон либо выдает сигнал, либо не выдает;
- перцептрон может быть описан функцией (в векторной форме):

$$f(x) = \begin{cases} 1 \text{ если } w * x + b > 0 \\ 0 \text{ в иных случаях} \end{cases}$$

- здесь:
  - w - вектор весов
  - b - смещение
  - x - вектор входных данных
  - m - количество входов

$$\Sigma_{i=1}^{m} w_i x_i$$

Суть расчета

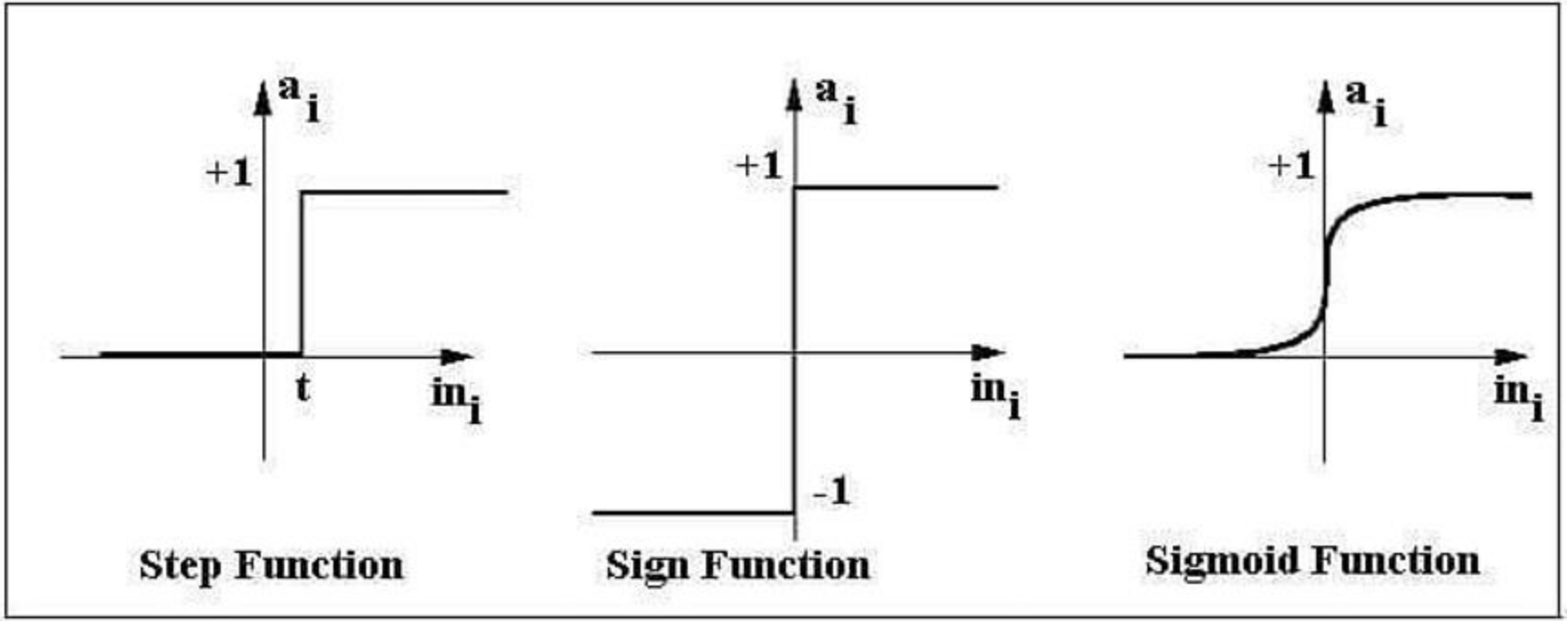
- входные данными могут быть представлены как 0 или 1 (или 1 и -1 в зависимости от функции активации);
- персептрон принимает входные данные, “модерирует” их с определенными значениями веса, затем применяет функцию преобразования для вывода конечного результата.

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Функция активации

- функция активации применяет правило шага (преобразует числовой вывод в +1 или -1), чтобы проверить, больше ли выход функции взвешивания нуля или нет.
- функция шага (step) срабатывает выше определенного значения выходного сигнала нейрона; в противном случае она выводит ноль.
- функция знака (sign) выводит +1 или -1 в зависимости от того, больше ли выход нейрона нуля или нет.
- функция сигмоид (sigmoid) - это S-образная кривая, которая выводит значение от 0 до 1.

Функция активации



Функция активации

Для перцептронов характерны следующие типы выходов.

Персептрон с булевым выходом:

Входы: x1...xn

Выход: o(x1.... xn)

$$o(x_1, \dots, x_n) = \begin{cases} 1 \text{ если } w_o + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ в иных случаях} \end{cases}$$

Веса: wi=> вклад входного сигнала xi в выход персептрона;

w0=> смещение или пороговое значение

Если $\sum w.x > 0$, выход равен +1, в противном случае -1. Нейрон срабатывает только тогда, когда взвешенный входной сигнал достигает определенного порогового значения.

$$o(\vec{x}) = sgn(\vec{w}\vec{x})$$

$$sgn(y) = \begin{cases} 1 \text{ если } y > 0 \\ -1 \text{ в иных случаях} \end{cases}$$

Значение +1 указывает на то, что нейрон активирован. Значение -1 указывает на то, что нейрон не сработал.

"sgn" означает функцию знака с выходом +1 или -1.

Обучение перцептрона

В Правиле обучения персептрона прогнозируемый результат сравнивается с известным результатом. Если он не совпадает, ошибка распространяется в обратном направлении, чтобы произошла корректировка веса.

Давайте теперь обсудим функцию принятия решений на базе перцептрона.

Функция принятия решения $\varphi(z)$ персептрона определяется как линейная комбинация векторов x и w.

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

Значение z в функции принятия решения задается как:

$$z = w_1 x_1 + \ldots + w_m x_m$$

Функция принятия решения равна +1, если z больше порогового значения θ, и равна -1 в противном случае.

Обучение перцептрона

$$\phi(x) = \begin{cases} 1 \text{ если } z \geq \Theta \\ -1 \text{ в иных случаях} \end{cases}$$

Это, по сути, алгоритм работы перцептрона.

Для простоты порог θ можно перенести влево и представить в виде w0x0, где w0= -θ и x0= 1.

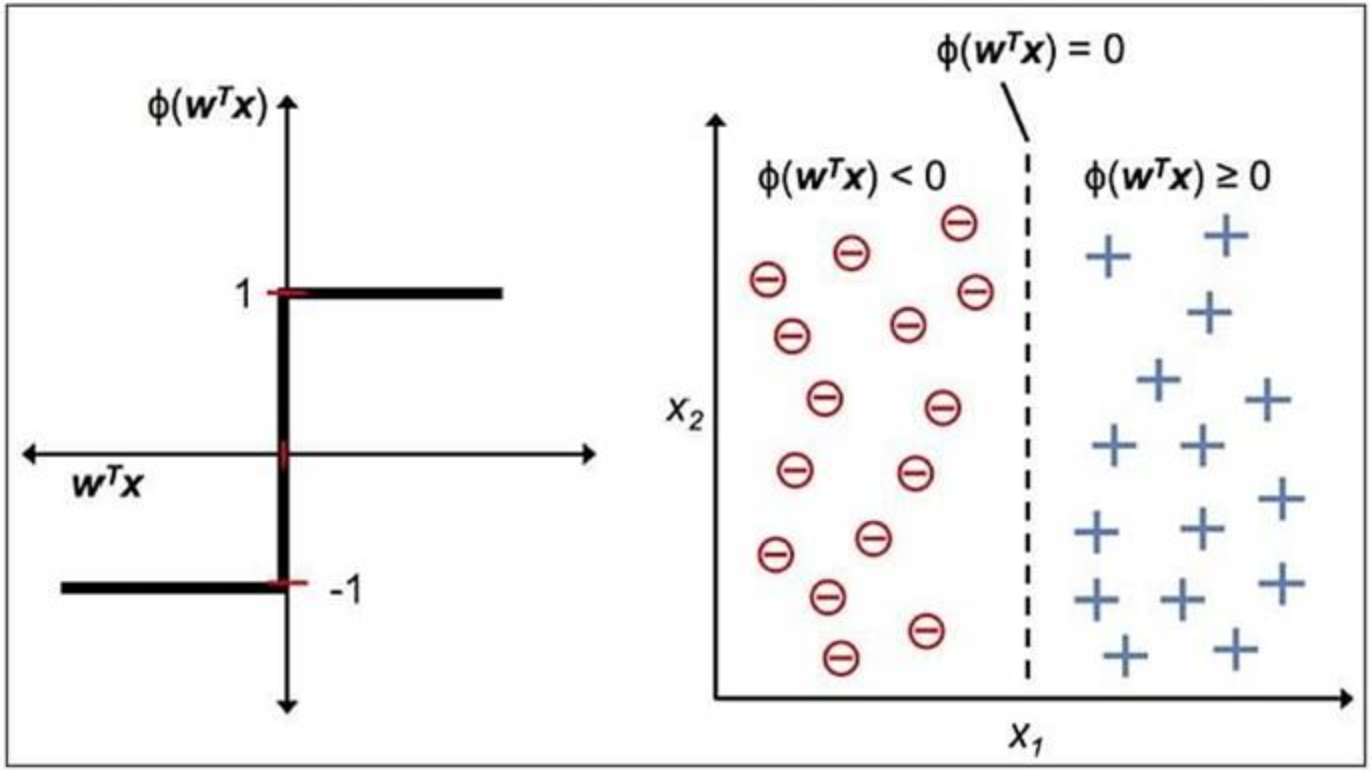
$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

Значение w0 называется единицей смещения.

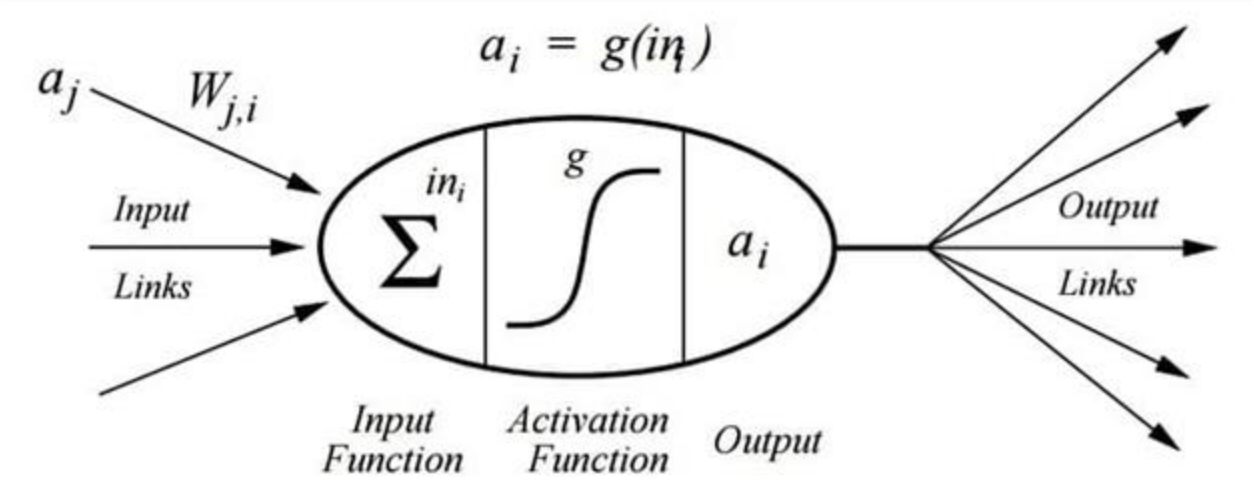
Затем функция принятия решений становится:

$$\phi(z) = \begin{cases} 1 \text{ если } z \geq 0 \\ -1 \text{ в иных случаях} \end{cases}$$

Выход перцептрона



Перцептрон, как "ядро" вычислений



$$a_i = g(\sum_j W_{j,i} a_j)$$

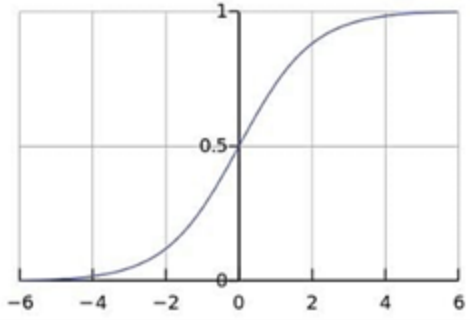
Сигмоидальная функция

Сигмоидная функция - это математическая функция с сигмоидной кривой (S-образной кривой). Это частный случай логистической функции и определяется функцией, приведенной ниже:

$$\phi_{logistic}(z) = \frac{1}{1+e^{-z}}$$

Здесь значение z равно:

$$z = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \sum_{i=0}^{m} w_i x_i = w^T x$$



Простой расчет

simplenn.py > ...

```python
import numpy as np

def net_input(x, w):
    return np.dot(x, w)

def logistic_function(z):
    return 1.0 / (1.0 + np.exp(-z))

def logistic_activation(x, w):
    z = net_input(x, w)
    return logistic_function(z)

X = np.array([1.0, 1.4, 2.5])

W = np.array([0.4, 0.3, 0.5])

print(f'P(y=1|x) = {logistic_activation(X, W)}')

# -> P(y=1|x) = 0.8879529614430097
```

Понятие нейронной сети

- нейронные сети состоят из “слоев” - объединений нейронов одного “уровня”;
- они похожи на “переключатели”, которые включаются или выключаются при подаче входного сигнала через сеть;
- выходные сигналы каждого слоя одновременно являются входными данными для последующего слоя, начиная с первого;
- сети могут быть многослойными и содержать множество “скрытых” слоев;
- сопоставление регулируемых весов модели с входными функциями - это то, как мы придаем значение этим функциям в отношении того, как нейронная сеть работает с данными.

Типы нейронных сетей

Общая классификация искусственных нейронных сетей (ANN):

- однослойные
- многослойные со скрытыми полными слоями;
- многослойные со скрытыми “неполными” слоями.

Виды искусственных нейронных сетей

Перцептрон Розенблатта;
Многослойный перцептрон Розенблатта;
Сеть Джордана;
Сеть Хэмминга;
Сеть Хопфилда;
Нейронный газ;
Неокогнитрон;
Осцилляторная нейронная сеть;
Сеть радиально-базисных функций (RBF-сеть);
Сеть Д.Смирнова;
Вероятностная нейронная сеть Решетова;
Сети адаптивного резонанса;
Нечёткий многослойный перцептрон;
Сплайн-модель Хакимова;
Многослойный перцептрон Румельхарта;
Сеть Элмана;
Сеть Ворда;
Сеть Кохонена;
Когнитрон;
Хаотическая нейронная сеть;
Сеть встречного распространения;
Сеть обобщённой регрессии;
Вероятностная сеть;
Сиамская нейронная сеть;
Свёрточная нейронная сеть;
Импульсная нейронная сеть.

Тезисы

- Нейронные сети обычно включает в себя большое количество процессоров, работающих параллельно и расположенных по уровням. Первый уровень получает исходную входную информацию-аналогично зрительным нервам в зрительной обработке человека или нейронам на коже.
- Каждый последующий уровень получает выходные данные от предыдущего уровня, а не исходные данные - точно так же нейроны человека, расположенные дальше от зрительного нерва, получают сигналы от тех, кто находится ближе к нему. Последний уровень производит вывод системы для внешних программ или устройств.
- Каждый узел обработки имеет свою собственную небольшую область знаний, включая то, что он видел, и любые правила, с которыми он был первоначально запрограммирован или разработан. Уровни тесно взаимосвязаны, что означает, что каждый узел на уровне n будет подключен ко многим узлам на уровне n-1-его входам-и на уровне n+1, который предоставляет входные данные для этих узлов. В выходном слое может быть один или несколько узлов, с которых можно прочитать полученный ответ.

Конкретные типы нейронных сетей

Нейронные сети с обратной связью: один из простейших вариантов нейронных сетей. Они передают информацию в одном направлении, через различные входные узлы, пока она не попадет в выходной узел.

Сеть может иметь или не иметь скрытых слоев узлов (что делает ее функционирование более понятным). Она подготовлена для обработки данных даже в условиях большого количества шума. Подобные типы вычислительной модели нейронной сети используется в таких технологиях, как распознавание лиц и компьютерное зрение.

Сейчас популярно создавать отдельные типы нейронных сетей для отдельных задач.

Конкретные типы нейронных сетей

Рекуррентные нейронные сети RNN: более сложные. Они сохраняют выходные данные обрабатывающих узлов и возвращают результат обратно в модель. Именно так, как говорят, модель учится предсказывать результат слоя.

Каждый узел в модели RNN действует как ячейка памяти, продолжая вычисления и выполнение операций. Эта нейронная сеть начинается с того же переднего распространения, что и сеть прямой передачи, но затем запоминает всю обработанную информацию, чтобы повторно использовать ее в будущем.

Если прогноз сети неверен, то система самообучается и продолжает работать над правильным прогнозом во время обратного распространения. Этот тип ANN часто используется при преобразовании текста в речь.

Конкретные типы нейронных сетей

Сверточные нейронные сети CNN: одна из самых популярных моделей, используемых сегодня. Эта вычислительная модель нейронной сети использует вариацию многослойных персептронов и содержит один или несколько сверточных слоев, которые могут быть либо полностью связаны, либо объединены. Эти сверточные слои создают карты объектов, которые записывают область изображения, которая в итоге разбивается на прямоугольники и отправляется для обработки.

Модель CNN особенно популярна в области распознавания изображений: она использовалась во многих самых передовых приложениях искусственного интеллекта, включая распознавание лиц, оцифровку текста и обработку естественного языка. Другие виды использования включают обнаружение перефразирования, обработку сигналов и классификацию изображений.

Конкретные типы нейронных сетей

Деконволюционные нейронные сети: используйте обратный процесс модели CNN. Они направлены на поиск потерянных функций или сигналов, которые изначально могли считаться несущественными для задачи системы CNN. Эта сетевая модель может быть использована при синтезе и анализе изображений.

Графовые нейронные сети: используются для обработки данных, представленных в виде графа. Используются для классификации, предиктивной аналитики, принятия решений.

Модульные нейронные сети: содержат несколько нейронных сетей, работающих отдельно друг от друга. Сети не взаимодействуют и не вмешиваются в деятельность друг друга во время вычислительного процесса. Следовательно, сложные или большие вычислительные процессы могут выполняться более эффективно с этим типом нейронных сетей.

Сеть Хопфилда

Сеть построена таким образом, что ее ответ на запомненные эталонные образы составляет эти же сами образы. Так что если образ немного изменить и подать обратно на вход, то он будет восстановлен и в виде отклика сети представлен в виде оригинального образа.

Это свойство сети позволяет говорить о сетях Хопфилда как о сетях с коррекцией ошибок и помех.

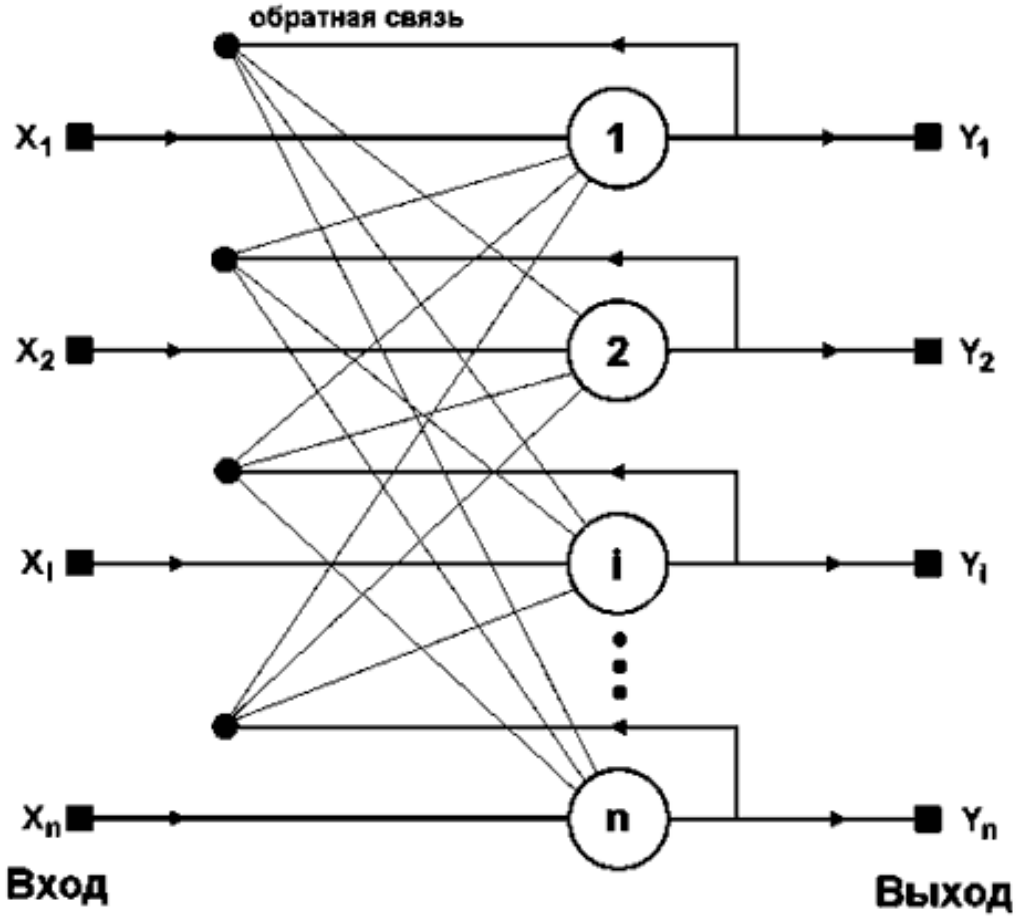
Сеть Хопфилда является примером однослойной нейронной сети и ее составляют из ограниченного числа N формальных нейронов.

Каждый нейрон из этой сети может принимать на входе одно из двух устойчивых состояний: 1 или – 1.

Таким образом поведение сети аналогично нейрону с пороговой функцией активации.

Благодаря этому сети Хопфилда нередко называют «спинами» или «биполярными».

Схема



Что внутри?

Сеть Хопфилда является полносвязной сетью, то есть каждый нейрон связан со всеми остальными нейронами.

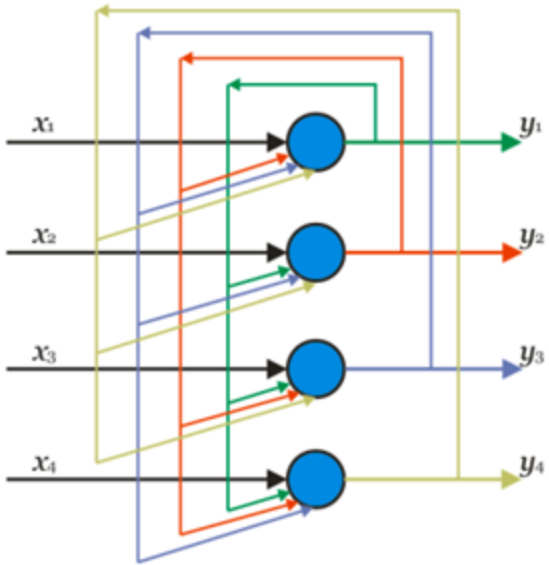
Взаимодействие между нейронами мы можем описать в виде следующего процесса.

$$E = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} x_i x_j$$

Здесь wij это элемент матрицы взаимодействия W. Данная матрица состоит из весов для связей между нейронами.

Процесс обучения нейронной сети Хопфилда состоит в том, что нужно сформировать матрицу весов W, которая должна запомнить определенное количество (m) эталонных образов.

Что внутри?



Эталонные образы – это N-мерные бинарные векторы вида $S_m = (s_{m1}, s_{m1}, \dots, s_{mN})$. Они и будут во время использования сети выражать отклик системы на комбинации входных сигналов, то есть будут формировать выходные данных y_i

Следует отметить, что в случае сети Хопфилда матрица связей между нейронами является симметричной, то есть $w_{ij} = w_{ji}$. Другое свойство сети выражается в том, что диагональные элементы матрицы равны нулю, то есть $w_{ii} = 0$.

Последнее свойство исключает обратные связи (петли) от нейрона к самому себе.

Сеть работает асинхронно. А матрица взаимодействия хранится на самих нейронах в виде весов при связях нейронов с другими нейронами.

Структура сети, а именно число нейронов в слое, определяется входными параметрами. Например, если у нейронной сети Хопфилда присутствует 10 входных параметров, то будет один слой с 10 нейронами.

Каждый нейрон связан с другими и не связан с самим собой, так что будет связь с остальными 9 нейронами. То есть в сети образуется 10 * 9 связей.

Для каждой связи будет свой весовой коэффициент w_{ij}, то есть таких весов будет 90. Эти 90 связей образуют матрицу взаимодействия и на нее будет направлен процесс обучения.

Обучение?

Обучение сети Хопфилда заключается в том, что находятся веса матрицы взаимодействий так, чтобы запомнить m векторов (эталонных образов, составляющих «память» системы).

Вычисление коэффициентов основано на следующем правиле: для всех запомненных образов Xi матрица связи должна удовлетворять уравнению Xi=WXi.

Если посмотреть внимательно на такое состояние, то можно увидеть, что именно при таком сочетании параметров, состояние сети Xi будет устойчивым – сеть в нем и остается.

Входные вектора для «запоминания» должны иметь вид массива бинарных значений.

Для расчета коэффициентов – весов необходимо воспользоваться следующей функцией

- $W = \frac{1}{N} \sum_i X_i X_i^T$

Здесь Xi – это запоминаемый вектор-столбец.

Формула будет более понятной, если мы ее запишем следующим образом

- $w_{ij} = \frac{1}{N} \sum_{d=1..m} X_{id} X_{jd}$

Здесь N – это размерность вектора, m – это число запоминаемых выходных векторов, d – это конкретный номер запоминаемого вектора, а X{id} – это компонента «i» запоминаемого выходного вектора «j».

Процесс обучения выполняется одну «эпоху» (то есть итерацию) и заключается в расчете этих выходных коэффициентов.

Почему их применяют

“Возможность параллельной обработки” означает, что сеть может выполнять более одного задания одновременно, если это возможно физически на используемых вычислительных средствах.

“Информация хранится во всей сети”, а не только в выделенной базе данных.

“Способность учиться” и “моделировать” нелинейные, сложные взаимосвязи помогает моделировать реальные отношения между входными и выходными данными.

“Отказоустойчивость” означает, что повреждение одной или нескольких ячеек ANN не остановит генерацию выходных данных, но может снизить точность принятия решения.

“Постепенное повреждение” означает, что сеть будет медленно деградировать с течением времени вместо того, чтобы система мгновенно разрушается от воздействия.

Почему их применяют

Способность производить расчет результата с неполными знаниями с потерей производительности зависит от того, насколько важна недостающая информация.

На входные переменные не накладывается никаких ограничений, например, на то, как они должны распределяться.

Машинное обучение означает, что ANN может извлекать уроки из событий и принимать решения на основе наблюдений.

Способность изучать скрытые взаимосвязи в данных, не требуя каких-либо фиксированных взаимосвязей, означает, что ANN может лучше моделировать сильно изменчивые данные и не постоянную дисперсию.

Способность обобщать и выводить невидимые взаимосвязи из невидимых данных означает, что ANNs может предсказывать вывод невидимых данных.

Их применяют, несмотря на...

Отсутствие правил для определения правильной структуры сети означает, что подходящую архитектуру искусственной нейронной сети можно найти только методом проб и ошибок и на собственном опыте.

Требования к процессорам с возможностями параллельной обработки делают нейронные сети аппаратно зависимыми.

Сеть работает с числовой информацией, поэтому все проблемы должны быть переведены в числовые значения, прежде чем они могут быть представлены ANN.

Отсутствие объяснений, лежащих в основе решений, является одним из самых больших недостатков ANNs. Неспособность объяснить, почему или как стоит за решением, иногда порождает отсутствие доверия к сети.

Выводы?

Алгоритм обучения сети Хопфилда существенно отличается от таких классических алгоритмов обучения перцептронов, как метод коррекции ошибки или метод обратного распространения ошибки. Отличие заключается в том, что вместо последовательного приближения к нужному состоянию с вычислением ошибок, все коэффициенты матрицы рассчитываются по одной формуле, за один цикл, после чего сеть сразу готова к работе.

Так что сеть Хопфилда является более «компактной» в плане обучения по отношению к другим нейронным сетям.

Сеть Хэмминга

Сеть Хэмминга это вид нейронной сети, который используется в первую очередь для классификации бинарных векторов. Основным критерием работы этой сети является применение расстояния Хэмминга.

Нужно отметить, что эта сеть является «наследником» или «развитием» сети Хопфилда.

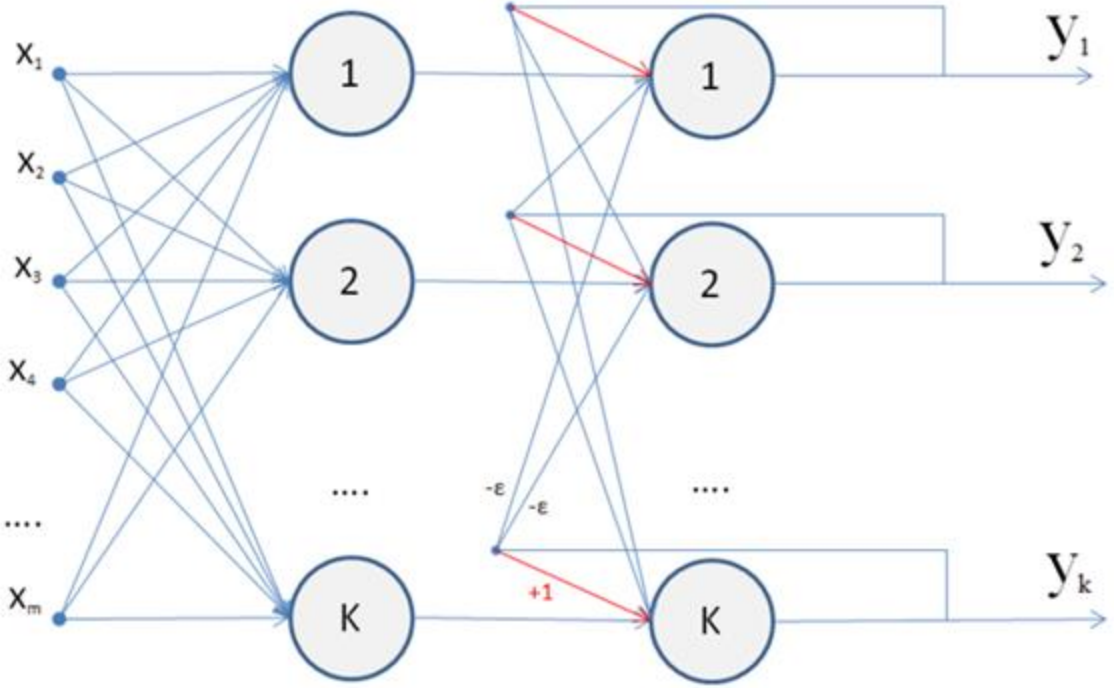
Сеть используется для того, чтобы соотнести бинарный вектор x=(x1, x2, xm) где xi={-1, 1} с одним из эталонных образцов или же определяет, что вектор не соответствует ни одному из имеющихся эталонов.

К тому же, в отличие от сети Хопфилда, выдаёт не сам образец, а его номер.

Напомним, что расстояние Хэмминга работает именно по такому принципу: (оно определяет число позиций, в которых соответствующие символы двух слов одинаковой длины различны).

Сеть Хэмминга — это многослойная (трёхслойная) нейронная сеть с обратной связью. Количество нейронов во втором и третьем слоях равно количеству классов классификации. Синапсы нейронов второго слоя соединены с каждым входом сети, нейроны третьего слоя связаны между собой отрицательными связями, кроме синапса, связанного с собственным аксоном каждого нейрона — он имеет положительную обратную связь.

Сеть Хэмминга



Сеть Хэмминга

Данная сеть также может быть обучена следующим образом.

Матрица весов первого слоя рассчитывается исходя из матрицы эталонов X следующим образом: $w_{ij} = \frac{1}{2} x_{ij}$. Здесь матрица KxM – это матрица эталонных образов. Каждая строка этой матрицы – соответствующий эталонный двоичный (бинарный) вектор.

$$f(s) = \begin{cases} 0, \text{для } s \leq 0, \\ s, \text{для } s \in (0, T], \\ \frac{M}{2}, \text{для } s > \frac{M}{2}. \end{cases}$$

Далее можно рассчитать второй слой нейронной сети как матрицу коэффициентов KxK следующего вида:

$$\begin{matrix} 1 & -\epsilon & \dots & -\epsilon \\ -\epsilon & 1 & \dots & -\epsilon \\ \dots & \dots & \dots & \dots \\ -\epsilon & -\epsilon & \dots & 1 \end{matrix}$$

Здесь каждый $\epsilon \in \left(0, \frac{1}{K}\right]$.

Важно понимать, что расчет всех значений проводится за один цикл, так что обучение сети занимает также одну эпоху.

Как работает?

Давайте посмотрим, как работает данная сеть.

На вход сети подается классифицируемый вектор {x}. Состояние нейронов первого слоя рассчитывается по формуле s1j=wji * xi.

Выходы нейронов первого слоя получаются после применения функции активации к их состоянию. Выходные результаты функций активации являются входными параметрами для второго слоя.

Далее, состояния нейронов второго слоя получаются из их предыдущего состояния, исходя из матрицы весовых коэффициентов второго слоя, и процедура повторяется итерационно до стабилизации вектора состояния второго слоя — пока норма разницы векторов двух последовательных итераций не станет меньше определенного значения Emax.

В случае, если в итоге один вектор положительный, а остальные отрицательные, то он указывает на подходящий образец. В случае же, если несколько векторов положительны, и при этом, не один из них не превышает Emax, то мы можем считать, что сеть Хэмминга не может классифицировать входной вектор, то есть отнести его достаточно точно ни к одному уже известному ей классу. Однако, положительные выходы указывают на наиболее похожие эталоны.

Самые современные сети

- ChatGPT (от англ. Generative Pre-trained Transformer «генеративный предварительно обученный трансформер») — чат-бот с искусственным интеллектом, разработанный компанией OpenAI и способный работать в диалоговом режиме, поддерживающий запросы на естественных языках. ChatGPT — большая языковая модель, для тренировки которой использовались методы обучения с учителем и обучения с подкреплением.
- ChatGPT был запущен 30 ноября 2022 года и привлёк внимание своими широкими возможностями: написание кода, создание текстов, возможности перевода, получения точных ответов и использование контекста диалога для ответов, хотя его фактическая точность и подверглась критике.

Самые современные сети

- Запросы для нейросетей, которые генерируют картинки.
- Сценарии (фильмы, игры, песни, ноты).
- Школьные эссе.
- Медицинские советы.
- План питания и тренировок.
- Переводы.
- Рутинные задачи (письма, расписания, списки дел).
- Пересказы.

Можете попробовать Yandex Алиса, навык "Давай придумаем".

Самые современные (модные) сети

- Midjourney — исследовательская компания и разрабатываемое ею одноименное программное обеспечение искусственного интеллекта, создающее изображения по текстовым описаниям. Наряду с конкурентами на рынке генерации изображений для персонализированных медиа — приложениями DALL-E от OpenAI и Stable Diffusion — использует технологии генеративно-состязательных сетей

Примеры

Изгой

Рыбак на крючке

Морковные деревья

Солнечное представление

Давайте рассмотрим задачу

Есть два класса, красный (снизу слева) и синий (справа вверху), и мы хотим разделить их, проведя между ними прямую линию. Или, более формально, мы хотим изучить набор параметров тета (theta), чтобы найти оптимальную гиперплоскость (прямую линию для наших данных), которая разделяет два класса.

Для линейной регрессии наша гипотеза (y_hat) была theta.X. Затем для двоичной классификации в логистической регрессии нам нужно было вывести вероятности от 0 до 1, поэтому мы изменили гипотезу как — сигмоидальную (theta.X). Мы применили сигмоидальную функцию к точечному произведению входных функций и параметров, потому что нам нужно было сжать наши выходные данные между 0 и 1.

Для алгоритма Персептрона мы применяем другую функцию над theta.X, которая является функцией единичного шага, которая определяется следующим образом.

$$h_\Theta(x) = g(\Theta^T x)$$

где

$$g(x) = \begin{cases} 1 \text{ если } z \geq 0 \\ 0 \text{ если } z < 0 \end{cases}$$

График функции

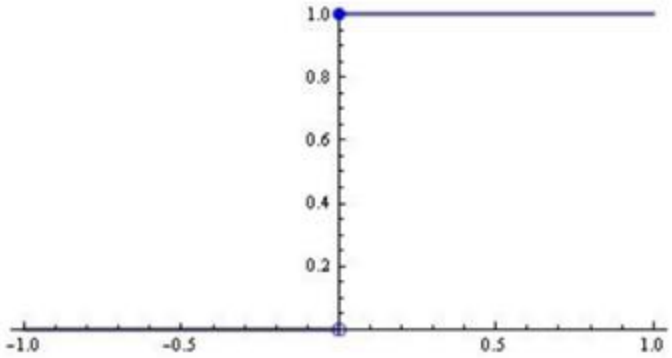
В отличие от логистической регрессии, которая выводит вероятность от 0 до 1, Персептрон выводит значения, которые в точности равны 0 или 1.

Эта функция говорит, что, если вывод (theta.X) больше или равно нулю, тогда модель классифицирует 1 (например, красный), а если результат меньше нуля, модель классифицирует как 0 (например, зеленый).

И именно так классифицирует алгоритм восприятия.

Давайте посмотрим на график функция единичного шага.

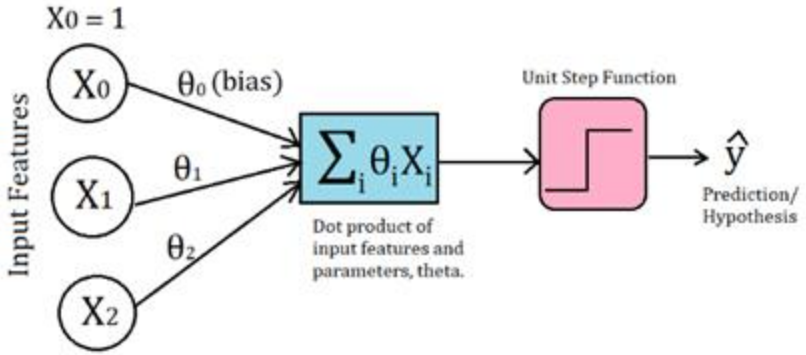
```
24  def step_func(z):
25      return 1.0 if (z > 0) else 0.0
```



Мы можем видеть, что для $z \geq 0$, $g(z) = 1$ и для $z < 0$, $g(z) = 0$.

А как же обучение?

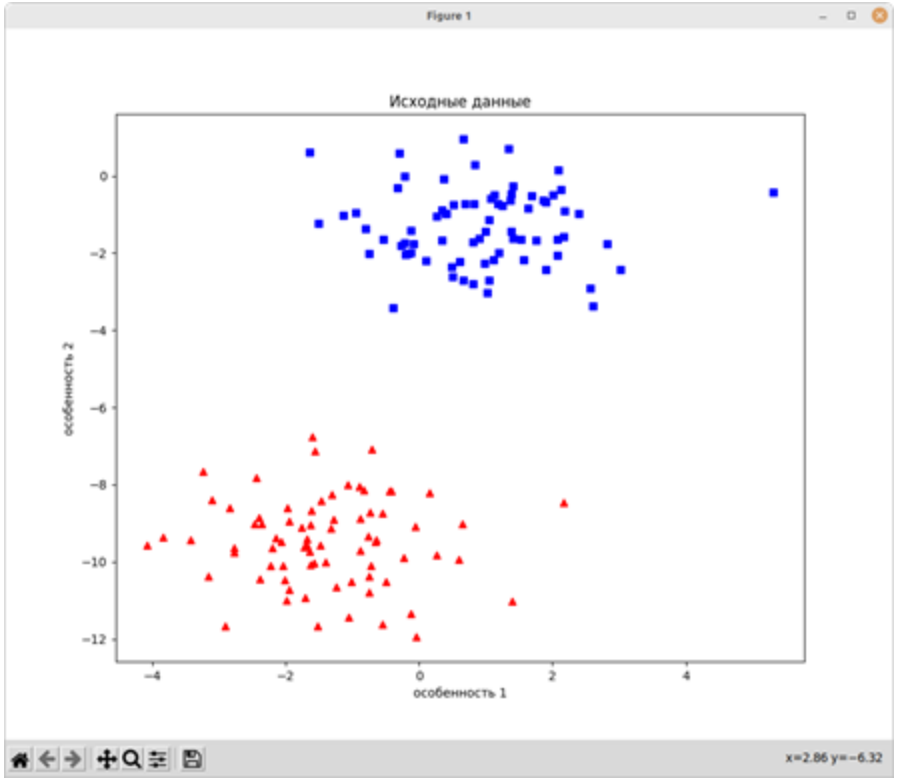
Мы можем понять принцип работы перцептрона, взглянув на приведенное ниже изображение.



Для каждого обучающего примера мы сначала берем точечное произведение входных функций и параметров, theta. Затем мы применяем функцию единичного шага для выполнения прогноза (y_hat).

И если прогноз неверен или, другими словами, модель неправильно классифицировала этот пример, мы обновляем параметры theta. Мы не обновляем, когда прогноз верен (или совпадает с истинным / целевым значением y).

Создаем перцептрон



perceptron.py > perceptron

```python
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as pl
# may require -> sudo apt-get install python3-tk

# Создадим набор данных - dataset
X, y = datasets.make_blobs(
    n_samples=150,
    n_features=2,
    centers=2,
    cluster_std=1.05,
    random_state=2
)

# Нарисуем первый график 1
figure = pl.figure(figsize=(10, 8))
pl.plot(X[:, 0][y == 0], X[:, 1][y == 0], 'r^')
pl.plot(X[:, 0][y == 1], X[:, 1][y == 1], 'bs')
pl.xlabel("особенность 1")
pl.ylabel("особенность 2")
pl.title('Исходные данные')
pl.show()
```

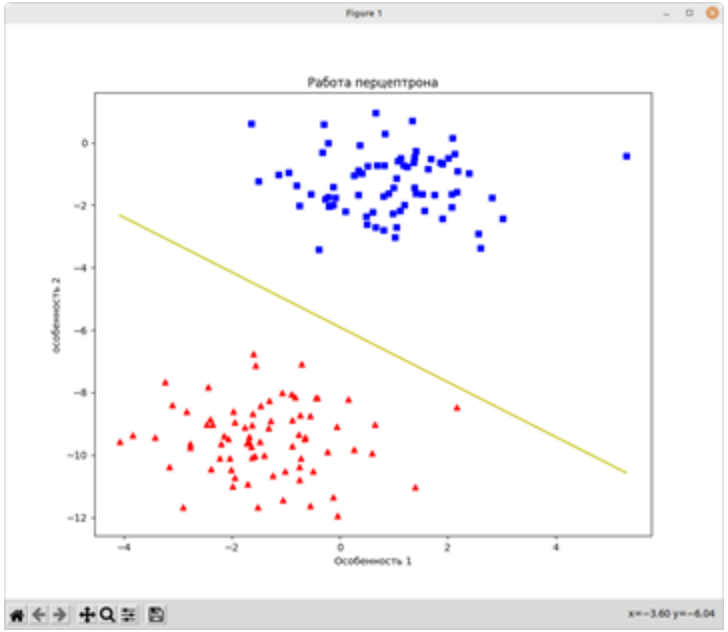
Что дальше?

```
27  def perceptron(X, y, lr, epochs):
28      # X --> Входы
29      # y --> Выходы (метки, центры)
30      # lr --> Скорость обучения
31      # epochs --> Количество итераций алгоритма
32      # m-> Количество примеров для обучения
33      # n-> Количество особенностей
34      m, n = X.shape
35
36      # Инициализация параметров (тета) в нули
37      # +1 в n+1 для срока смещения
38      theta = np.zeros((n+1, 1))
39
40      # Пустой список для хранения того, сколько примеров
41      # неправильно классифицируется на каждой итерации
42      n_miss_list = []
```

Что дальше?

```
44      # Обучение
45      for epoch in range(epochs):
46          # переменная для хранения образца,
47          # который неправильно классифицирован (#misclassified)
48          n_miss = 0
49
50          # Итерация по каждомо образцу
51          for idx, x_i in enumerate(X):
52              # Вставляем 1 для смещения (bias), X0 = 1.
53              x_i = np.insert(x_i, 0, 1).reshape(-1,1)
54
55              # Расчет гипотезы/предсказания
56              y_hat = step_func(np.dot(x_i.T, theta))
57
58              # Обновление если пример неправильно классифицирован
59              if (np.squeeze(y_hat) - y[idx]) != 0:
60                  theta += lr*((y[idx] - y_hat)*x_i)
61                  n_miss += 1
62
63          # Добавление к списку неправильно классифицированных
64          # на каждой итерации
65          n_miss_list.append(n_miss)
66
67      return theta, n_miss_list
```

Что дальше?




```
69  def plot_decision_boundary(X, theta):
70      # X --> Входы
71      # theta --> Параметры
72
73      # Линия разделения по уравнению y = mx + c
74      # Так что, вычисляем: mx + c = theta0.X0 + theta1.X1 + theta2.X2
75      # Решаем уравление для нахождения m и c
76      x1 = [min(X[:,0]), max(X[:,0])]
77      m = -theta[1]/theta[2]
78      c = -theta[0]/theta[2]
79      x2 = m*x1 + c
80
81      # Вывод графика № 2
82      figure = pl.figure(figsize=(10,8))
83      pl.plot(X[:, 0][y==0], X[:, 1][y==0], "r^")
84      pl.plot(X[:, 0][y==1], X[:, 1][y==1], "bs")
85      pl.xlabel("Особенность 1")
86      pl.ylabel("особенность 2")
87      pl.title('Работа перцептрона')
88      pl.plot(x1, x2, 'y-')
89      pl.show()
90
91  theta, miss_l = perceptron(X, y, 0.5, 100)
92  plot_decision_boundary(X, theta)
```