1. **Понятие нейронной сети. Виды нейронных сетей. Связь с биологией: дендрит, аксон, синапс.**

**Нейронная сеть** — это последовательность нейронов, соединенных между собой синапсами. Структура нейронной сети пришла в мир программирования прямиком из биологии. Благодаря такой структуре, машина обретает способность анализировать и даже запоминать различную информацию.

**Нейрон**- вычислительная единица которая получает инф. Производит действия и передает дальше.

Виды:

**Однослойная**-с входа на выход

**Многослойная** –имеются скрытые слои

Вид нейронных сетей:

1. **Полносвязные** — каждый нейрон передает выходной сигнал и самому себе, и остальным нейронам;

2. **Многослойные** — совокупность **нейронных сетей** образует слой с единым входным сигналом;

3. **Слабосвязные** — нейроны располагаются в узлах прямоугольной или гесксогональной решетки.

**Дендрит**-нейрон на входном слое

**Аксон**-нейрон на выходном слое

**Синапс**- связь между нейронами

1. **Искусственный нейрон МакКаллока-Питтса.**

Первая вычислительная модель нейрона была предложена Уорреном Му Каллохом (нейробиологом) и Уолтером Питтсом (логиком) в 1943 году. Важно понимать, что в нейроне Мак-Каллока и Питтса нет привычных нам изменяемых синаптических весов. Принцип работы нейрона Мак-Каллока и Питтса формулируется следующим образом: нейрон x срабатывает на шаге t, если соблюдаются два условия:

1) нет ни одного нейрона, связанного с нейроном x ингибирующей связью, сработавшего на шаге t – 1.

2) число нейронов, связанных с нейроном x возбуждающей связью и сработавших на шаге t − 1, больше 0.

1. **Персептрон.**

1957 г.-Розенблант – персептрон.

Персептрон- мат.модель для распознавания образов.

Состоит из 3 слоев:

Сенсорные нейроны,входная инф.,acсоциативные нейроны ,реагирующие нейроны.

1. Понятие функции активации. Виды функции активации.

Функция активации-обрабатывает сумму весов на на нейронах и выдает выходной сигнал (от 0 до 1)

Примеры функций активации:

Функция Хевисайда(результат 0 или 1)::

Сигмоидальная функция(результат от 0 до 1):

Гиперболический тангенс(результат от -1 до 1)

1. **Тренировочный сет. Итерация. Эпоха.**

**Тренировочный сет**-последовательность данных ,которыми оперирует нейронная сеть.

**Итерация** – счетчик который увеличивается каждый раз, когда нейронная сеть проходит один тренировочный сет.

**Эпоха**-показатель натренированности нейронной сети ,уведичивается каждый раз когда нейр.сеть проходит цикл полного набора тренировочных сетов.

Чтобы проводить тренировку сети правильно нужно выполнят сеты, последовательно **увеличивая показатель эпохи.**

В процессе тренировки будут выявляться **ошибки**.

Это **процентный** **показатель** расхождения между полученным и желаемым результатом.

Этот показатель должен **уменьшаться в процессе увеличения показателя эпохи**, в противном случае где-то ошибка разработчика.

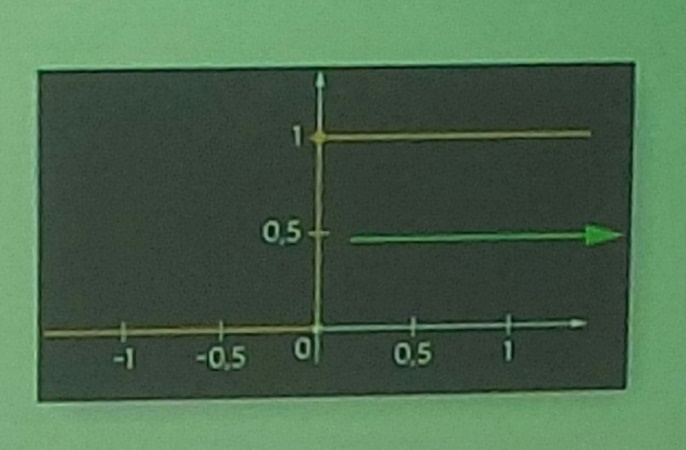
1. **Нейрон смещения.**

В нейронных сетях есть еще один вид нейронов – **нейрон смещения**

Он отличатся от основного вида нейронов тем, что его вход и выход в любом случае равняется единице

При этом входных синапсов такие нейроны не имеют

Расположение таких нейронов происходит по одному на слой и не более, также они **не могут соединяться с синапсами друг с другом**

Размещать такие нейроны на выходном слое не целесообразно

*Для чего они нужны?*

Бывают ситуации, в которых нейросеть просто не сможет найти верное решение из-за того, что нужная точка будет **находиться вне пределов досягаемости**

Именно для этого и нужны такие нейроны, чтобы иметь возможность **сместить область определения**

1. **Описание обучения нейронной сети методом обратного распространения ошибки на основе градиентного спуска. Пример.**

**МЕТОД ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБОК**

В процессе поступления информации **нейронная сеть** последовательно передаёт её от одного

нейрона к другому посредством **синапсов**, до того момента, пока информация не окажется на

выходном слое и не будет выдана как результат.

Такой способ называется **передачей вперёд.**

После того как результат получен вычисляется ошибка и на её основании выполняется **обратная передача**.

Суть которой - последовательно изменить вес синапсов начиная с выходного и **продвигаясь к входному слою.**

При этом значение веса меняется в сторону **лучшего результата.**

Для использования такого метода обучения подойдут только те **функции активации**, которые можно **дифференцировать**.

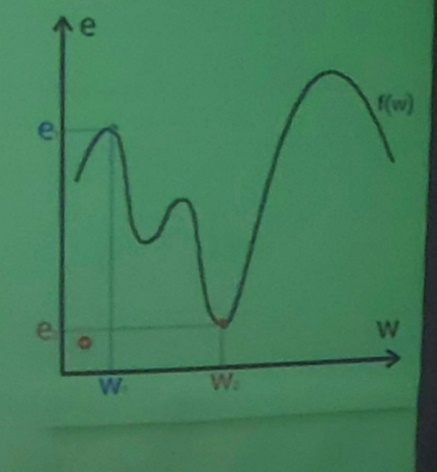
Так как обратное распространение вычисляется с помощью **высчитывания разницы результатов и**

**умножения его на производную функции от входного значения.**

1. **Метод градиентного спуска**

Это способ нахождения локального **минимума или максимума функции** с помощью движения вдоль градиента

График функции **f(w)** является зависимостью ошибки(е)от выбранного веса(w).

Точка (w2,е2) означает, что выбрав вес w2 мы получим самую маленькую ошибку – е2 и как следствие, самый лучший результат из всех возможных.

У каждого веса в нейросети будет **свой график** и градиент и у каждого надо найти глобал мин.

**Градиент** – это вектор, который определяет крутизну склона и указывает его направление относительно какой-либо из точек на поверхности или графике.

Чтобы **найти градиент** нужно взять производную от графика по данной точке.

Если ошибка равна **100%,** то мы находимся на самой вершине и если ошибка **0%,** то в низине.

Оказавшись в определённой точке со значением веса, которыйраспределяется в случайном порядке, вычисляется **градиент** иопределяется **направление движения** **спуска**, и так в каждойследующей точке, пока не достигается **локальный минимум**.

Чтобы справиться с этим необходимо установить **нужное значение момента**, которое позволит преодолеть частьграфика и **достигнуть нужной точки**.

1. **Метод упругого распространения.**

Алгоритм использует в качестве основы обучение **по эпохам** и применяет только **знаки производных** частного случая **для корректировки весовых коэффициентов.**

Если на этапе расчётов **производная меняет свой знак**, значит, изменение было слишком большим и локальный минимум **был упущен** и нужно произвести откат, то есть вес вернуть **в обратную позицию**, а величину изменения **уменьшить**.

Если знак производной не изменился, то величина изменения веса,наоборот, увеличивается для большей сходимости.

Если основные параметры коррекции веса зафиксировать, то настройкиглобальных параметров можно избежать.

Для этих параметров есть рекомендуемые значения, однако, никакихограничений на их выбор не накладывается.

Чтобы вес не принимал слишком большие или малые значения используются установленные ограничения величины коррекции

Если производная функции в конкретной точке меняет знак с плюса наминус, означает, что ошибка возрастает и вес требует коррекции ипроисходит его **уменьшение**, в противном случае - **увеличение**.

* инициализируется величина коррекции,
* вычисляются частные производные,
* подсчитывается новое значение коррекции значений весов,
* корректируются веса, если условие остановки алгоритма невыполняются процесс переходит к вычислению производных и зановоповторяет цикл.

1. **Параметры и гиперпараметры нейронной сети. Правила Хэбба.**

**Гиперпараметры** - параметры алгоритмов, значения которых устанавливаются **перед** запуском процесса обучения.

В этом смысле они и отличаются от обычных параметров,вычисляемых в процессе обучения.

**Гиперпараметры** используются для управления процессом обучения.

К гиперпараметрам можно отнести:

* **крутизну активационной функции,**
* **число скрытых слоёв и нейронов в них,**
* **коэффициент скорости обучения,**
* **момент, число итераций обучения, уровень ошибки.**

Одним из самых простых и понятных способов переобучения стало так называемое **Правило Хэбба.**

Предположим, у нас есть обучающая выборка, состоящая из объектов **обоих классов**.

Мы предоставляем персептрону сначала один объект, смотрим навыходной сигнал.

Пусть **0** соответствует первому классу, а **1** - второму классу.

Если выходной сигнал правильный, то берем второй объект изавершаем обучение.

Но если сигнал неверен?

Необходимо выполнить **корректировку весов**.

И тут возможны два варианта:

Выходной сигнал **неверен и равен нулю** - нужно **увеличить веса** всех активных входов, **добавив** к каждому величину входного сигнала**(Первое правило Хэбба)**

Выходной сигнал **неверен и равен единице** - нужно **уменьшить веса**всех активных входов, **отняв** от каждого величину входного сигнала**(Второе правило Хэбба)**

1. **Алгоритм обратного распространения ошибки на основе градиентного спуска. Достоинства и недостатки алгоритма.**

**Алгоритм обратного распространения ошибки** – это один из методов обучения многослойных нейронных сетей **прямого распространения**

**Шаг 1** Сеть функционирует в нормальном режиме – вычисляются выходные данные.

**Шаг 2** Сравнение выходных данных с известными выходными данными для данного входного набора.

Вычисление вектора ошибки.

**Шаг 3** Использование вектора ошибки для изменения весовых коэффициентов выходного слоя

Для уменьшения вектора ошибки при повторной подачк того же набора входных данных

**Шаг 4** Изменение весовых коэфф скрытого слоя

**Шаг 5** Если в сети существует еще один слой, то с ним проводятся аналогичные действия

**ДОСТОИНСТВА АЛГОРИТМА**

* Достаточно высокая эффективность.
* Через каждый нейрон проходит информация только о связных с ним нейронах, поэтому этот алгоритм легко реализуется на вычислительных устройствах с параллельной архитектурой.
* Высокая степень общности. Алгоритм легко записать для произвольного числа слоёв, произвольной размерности входов и выходов, произвольной функции потерь и произвольных функций активации.
* Этот метод не накладывает никаких ограничений на используемый метод оптимизации. Его можно применять вместе с методом скорейшего спуска, сопряженных градиентов, Ньютона-Рафсона и др.

**НЕДОСТАТКИ АЛГОРИТМА**

* Несмотря на многочисленные успешные применения обратного распространения, оно не является универсальным решением.
* Больше всего неприятностей приносит неопределённо долгий процесс обучения.
* В сложных задачах для обучения сети могут потребоваться дни или даже недели, она может и вообще не обучиться.
* Причиной может быть одна из описанных ниже.

1. **Задача распрознавания рукописных чисел. Постановка задачи, активация нейронов, назначение скрытых слоев. Библиотеки глубокого обучения.**

**Постановка задачи распознавания цифр**

Нейронная сеть принимает на вход изображение любой цифры в виде массива **28х28 пикселей** и выдает на выходе саму «сущность» - **цифру от 0 до 9**

**Активация нейронов.**

Так как наша сетка состоит из **28x28=784 пикселей**, пусть есть784 нейрона, содержащиеразличные числа **от до 1**: чемближе пиксель к белому цвету,тем ближе соответствующее число к единице.

Эти заполняющие сетку числа называются **активациями нейронов.**

**Назначение скрытых слоев**

* Выбор количества скрытых слоев исодержащихся в них нейронов произволен однако обычно они выбираются из определённых представлений о задаче,решаемой нейронной сетью.
* Принцип работы нейросети состоит в том, что активация в одном слое определяет активацию в следующем.
* Возбуждаясь, некоторая группа нейронов вызывает возбуждение другой группы.

**Библиотеки глубокого изучения**

**TensorFlow, theano, Microsoft CNTK, Caffe**

1. **Обучение нейронной сети для распознавания чисел. Архитектура сети, используемые функции активации. Фрагмент кода построения архитектуры и обучения сети.**

В результате обучения нейросеть должна правильным образом различать числа из ранее не представленных, тестовых данных. Соответственно в качестве проверки обучения нейросети можно использовать отношение числа актов корректного распознавания цифр к количеству элементов тестовой выборки.

Концептуально задача обучения нейросети сводится к нахождению минимума определенной функции – **функции стоимости**

Математически эта функция представляет сумму квадратов разностей между реальными значениями активации выходного слоя и их же идеальными значениями. Например, в случае тройки активация должна быть нулевой для всех нейронов, кроме соответствующего тройке, у которого она равна единице.  
Получается, что для одного изображения мы можем определить одно текущее значение функции стоимости.

Если нейросеть **обучена**, это значения будет **небольшим**, в идеале стремящимся к нулю, и **наоборот**: чем больше значение функции стоимости, **тем хуже обучена** нейросеть.

чтобы впоследствии определить, **насколько хорошо  произошло обучение** нейросети, необходимо определить **среднее значение функции стоимости** для всех изображений обучающей выборки.

Это довольно трудная задача. Если наша нейросеть имеет на входе 784 пикселя, на выходе 10 значений и требует для их расчета 13 тыс. параметров, то функция стоимости является функцией от этих 13 тыс. параметров, выдает одно единственное значение стоимости, которое мы хотим минимизировать, и при этом в качестве параметров выступает вся обучающая выборка.

Как изменить все эти веса и сдвиги, чтобы нейросеть обучалась?

Использовать **Функцию Градиентный спуск**. Описана в ответе на вопрос 8

**def** test\_set(hidden\_out\_w, input\_hidden\_w):  
 data\_file = open(**'mnist\_train.csv'**, **'r'**)  
 trening\_list = data\_file.readlines()  
 data\_file.close()  
 **for** record **in** trening\_list:  
 all\_values = record.aplit(**','**)  
 inputs = (numpy.asfarray(all\_values[1:])/255.0\*0.99)+0.01  
 targets = numpy.zeros(10)+0.01  
 targets[int(all\_values[0])] = 0.99

1. **Проблема переобучения нейронной сети. Улучшение качества обучения нейронной сети. Фрагмент кода, где происходит улучшение качества обучения нейронной сети**

Для обучения используются три набора данных:

**Обучающая выборка** - набор данных, который используетсядля обучения сети.

**Проверочная выборка** - набор данных, которыйиспользуется в процессе обучения для подборагиперпараметров сети.

**Тестовая выборка** - набор данных, который используется для оценки качества работы сети после завершения обучения.

Переобучение

#обучаем сеть

history = model.fit(x train, train,

batch size=200,

epochs=100,

validation split=0.2,

verbose=1)

#Оцениваем качество обучения сети на тестовых данных

scores = model.evaluate(x test, y test, verbose=1)

print("Доля верных ответов на тестовых данных, в процентах: ",

round(scores[1] \* 100, 4))

🡪Доля верных ответов на тестовых данных, в процентах: 86.39

**Проблема переобучения**

* Нейронная сеть распознает особенности обучающей выборки, а не данных
* Необходима проверка работы сети на данных, не использованных дляобучения

**Проверочная выборка**

* Используется в процессе обучения для подбора гиперпараметров
* Рост ошибки на проверочной выборке говорит о переобучении

**Тестовая выборка**

* Проверка качества работы сети после завершения обучения

**Улучшение качества обучения**

* Чтобы улучшить качество обучения сети нужно изменить следующие гиперпараметры:
* **количество эпох обучения,**
* **размер мини-выборки,**
* **количество нейронов во входном слое,**
* **количество скрытых слоев.**
* Для этого нужно провести серию экспериментов, в каждом из которых следует менять один из гиперпараметров, и анализировать, как изменилось качество работы сети.