Конспект лекции

А.Н. Шевляков

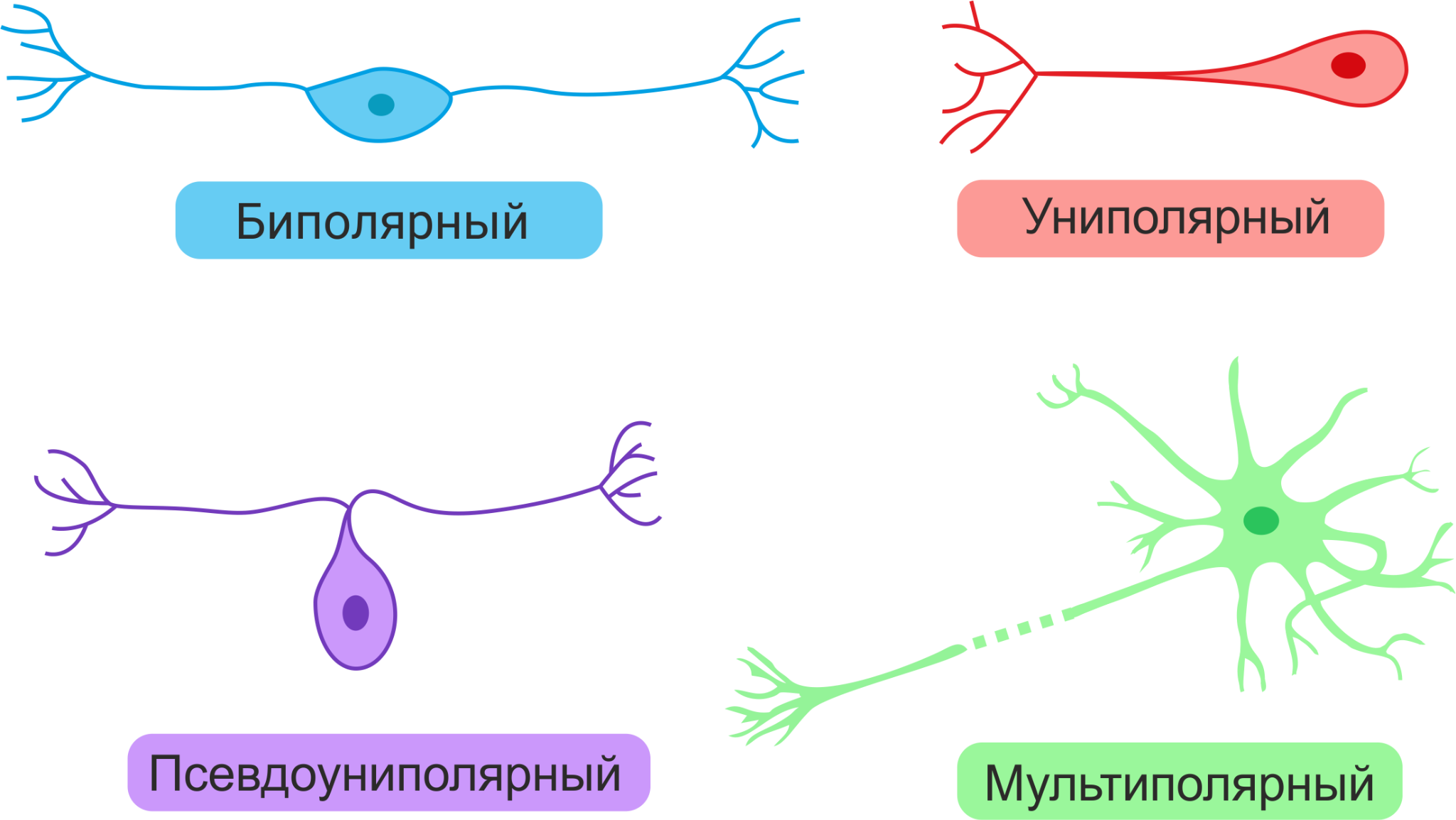
Введение в нейронные сети

Тренировка нейросети для задачи регрессии

# Искусственные нейронные сети

## Биологические нейроны

У него есть отростки (аксоны), которые ведут в другие нейроны. Из этих нейронов может приходить нервный импульс. Нейрон может генерировать импульс и передавать его соседям.



**Важно:** соседи нейрона не равноправны.Важность соседа выражается в толщине отростка (импульс от важного соседа будет более сильным).

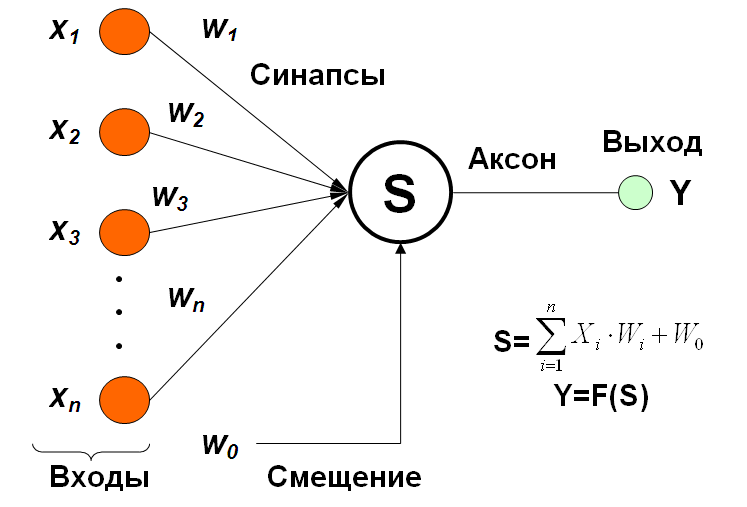
Каждый нейрон может находится в двух состояниях:возбужденном и невозбужденном. Если в нейрон пришел слишком большой сигнал, то нейрон переходит в возбужденное состояние.

Нейроны могут образовывать цикл**.** В этом случае импульс, испущенный из нейрона, может (после всех преобразований в нейронах цикла) снова вернуться в него.

## Искусственный нейрон

Как представить нейрон в виде математического объекта?

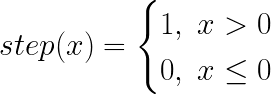
* У него должны быть входные и выходные значения.
* У него должна быть указана важность (сила) каждой связи с соседним нейроном.
* Он должен по достаточно простой формуле обрабатывать входные значения и передавать результат вычислений дальше.
* Выходное значение нейрона должно моделировать возбуждение нейрона.



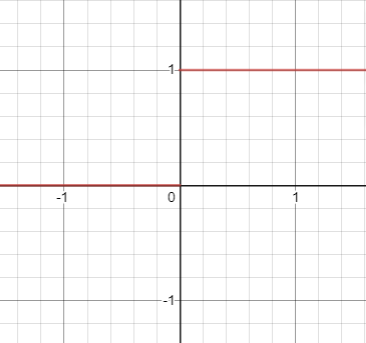
У искусственного нейрона (ИН) каждый вход имеет свой вес *wi*. Входные значения *xi* приходят либо из внешней среды, либо из других нейронов.

Важность каждого входа выражается весом связи *wi*. ИН считает взвешенную сумму входов, прибавляет **смещение** *w0*. Смысл смещения: **порог возбуждения**. Далее к результату применяется **функции активации** (ФА). Если результирующее значение достаточно велико, то считается, что нейрон возбудился.

Если строго следовать биологии, мы должны взять ступенчатую функцию



Она моделирует переход нейрона в возбужденное состояние.

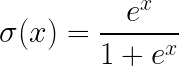


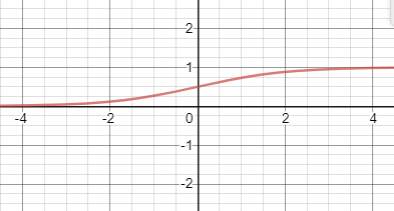
Недостатки функции-ступеньки step:

* Она разрывна.
* Её производная равна 0 во всех точках своей области определения.
* Следовательно, бесполезно использовать ГС при минимизации выражений, содержащих функцию-ступеньку.

Но позвольте, зачем нужно минимизировать ИН? Это нужно для поиска их оптимальных параметров (весов и смещений), необходимых для построения модели искусственного интеллекта.

Сигмоида является гладкой аппроксимацией ступеньки. В исторически первых НС применялась именно она.





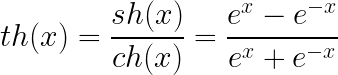
В предыдущих лекциях мы говорили о том, что сигмоида не безупречна.

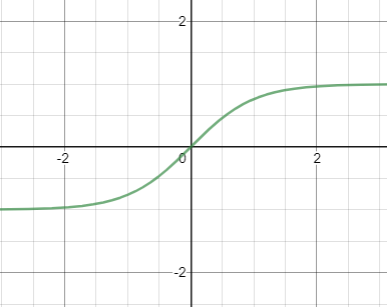
Когда у вас будет нейронная сеть, в которой несколько нейронов соединены последовательно, то возникнет суперпозиция сигмоид

\sigma(\sigma(\sigma(x)))

А у такой функции градиент практически нулевой.

**Гиперболический тангенс** тоже аппроксимирует ступеньку. Такая ФА реально применяется в больших НС (LeNet).

****



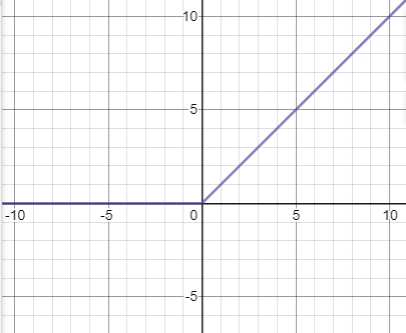
**ReLU – функция.** ReLU=Rectified linear unit

Достоинства: простота.

Недоумение: она совсем не похожа на ступеньку.

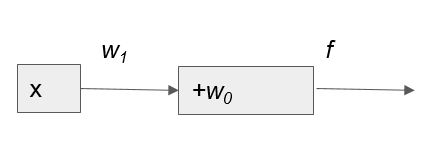
Практика показала, что часто оправдано моделировать ИН не двумя состояниями (возбудился или нет), а числовой «шкалойвозбуждения».

ReLU для этого идеально подходит.



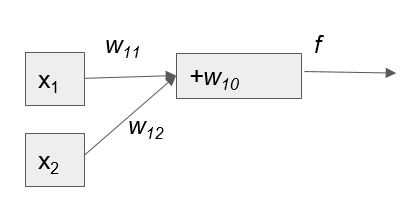


**Пример работы искусственного нейрона**



Если вход нейрона *x=1*, его вес *w1=2*, смещение *w0=-3*, ФА *f(x)=Relu(x)*, то на выходе будет значение *Relu(2\*1-3)=0*. То есть нейрон не возбуждается.

А вот если бы смещение было бы *w0=1,* то выход нейрона будет *Relu(2\*1+1)=3.*

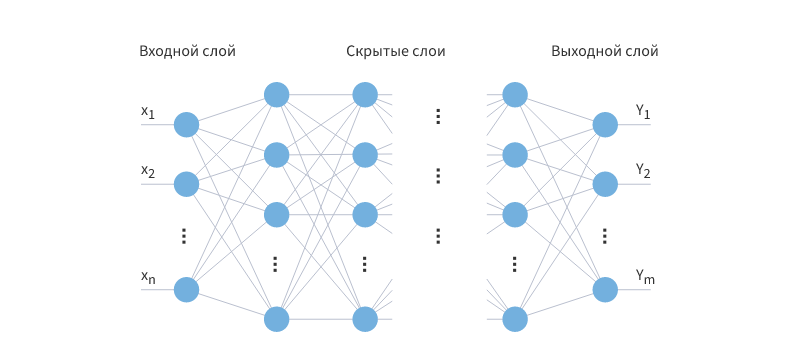


У ИН может быть несколько входов (каждый со своим весом). Входы умножаются на веса и суммируются.

Например, для *x1=1, x2=-1, w11=-2, w12=3, w10=10, f=Relu* получаем *Relu(1\*(-2)+(-1)\*3+10)=Relu(-5+10)=5.*

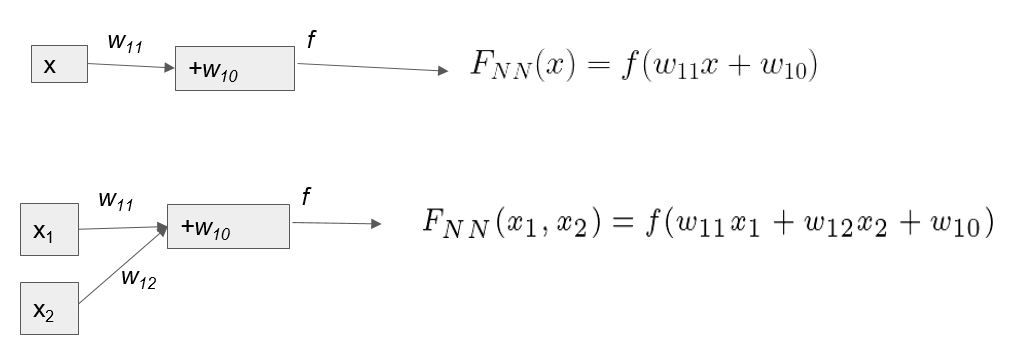
## Нейронные сети. Общие сведения

Что можно сделать с искусственными нейронами?Их можно соединить в нейронную сеть (НС) так, что выход одного нейрона является входом другого нейрона.

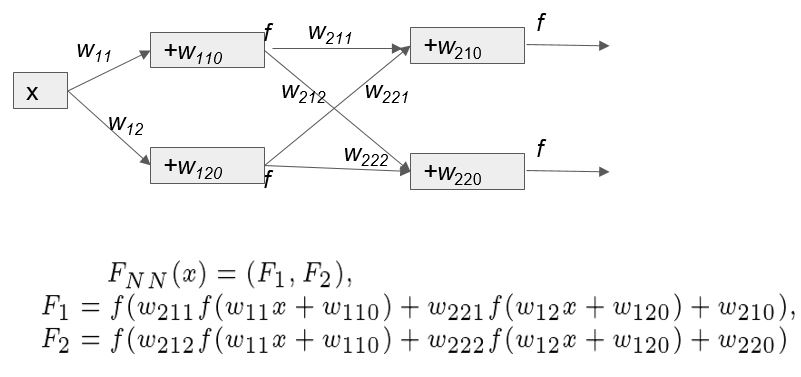


В НС есть два типа весов: **веса-связи** (веса связи между двумя нейронами) и **веса-смещения**.

С НС можно связать функцию *FNN(x),* которая преобразует вход сети в выход.



Если у НС несколько выходов (а такое бывает), то ее функция *FNN(x)* является вектором.



Если каждый нейрон некоторого слоя НС соединен со всеми нейронами следующего слоя, то такая пара слоев называется **полносвязной**. В паре полносвязных слоев может быть разное число нейронов.

Если два слоя полносвязны, то число весов в связях между ними равно n1n2, где n1,n2 - количество нейронов в этих слоях.

Нужно не забыть, что еще есть веса-смещения. Их число совпадает с общим числом нейронов в НС (кроме нейронов входного слоя).

Входной слой – особый. Там лежат входные данные. Размер входного слоя совпадает с размерностью данных. Если в НС подаются числа, то размерность входного слоя равна 1. Если в НС подаются пары чисел, то размерность входного слоя равна 2 и т.д.

Выходной слой – тоже особый. **То, что выходит с этого слоя, считается ответом все НС**. Размер выходного слоя может быть разным – в зависимости от типа ответа. Если ответ сети – единственное число, то на выходном слое будет 1 нейрон. Но это не обязательно...

Несколько выходов у НС оправданы, когда НС выдает массив чисел.Например, НС может выдать список вероятностей, что поданная на вход фотография содержит изображение собаки, кошки, автомобиля соответственно.

Во время тренировки НС ищутся оптимальные значения весов сети. Чем больше число весов НС, тем больше:

* время тренировки НС;
* требуется данных для НС;
* вероятность того, что НС обнаружит ложную закономерность (переобучение, overfitting).

## Различия между биологическими и искусственными нейронными сетями

**Различие № 1.** Сигнал в искусственной НС распространяется в одном направлении, то есть искусственные нейроны не могут образовывать циклы. Например, искусственная НС не подвержена галлюцинациям. Но это не самое важное. Искусственная нейронная сеть не может хорошо достраивать изображения.

**Различие № 2.** Практика показала, что для искусственных НС два состояния нейрона (возбужденный – невозбужденный) недостаточно. Поэтому de facto была принята непрерывная «шкала возбуждения», и теперь наиболее популярна такая функция активации (Relu).

**Различие № 3.** Для настройки весов искусственной НС используется алгоритм обратного распространения. Никаких аналогов такого алгоритма в биологических НС нет!

**Различие № 4.** Чтобы найти истинную закономерность в данных, человеческому мозгу требуется на порядок меньше данных.

# Задачи, решаемые нейросетями. Задача регрессии

## Задачи предсказания

**Постановка задачи.** Есть множество объектов *М* с известными значениями признака *Y*. Найти (предсказать, оценить) значение признака *Y* для нового объекта *А*. Признак Y называется **целевым**.



Каждая из этих задач требует особой архитектуры нейронной сети.

## Задача регрессии

Примеры:

* Предсказать возраст человека по его фото (анализу крови).
* Подсчитать количество людей (машин) на фото.
* Предсказать курс доллара (температуру воздуха) на завтра.
* Предсказать рост ребенка по росту его родителей.

Когда исследовали зависимость роста сына от роста его отца, то было замечено «рост сына приближался (**регрессировал**) к среднему росту мужчин». Поскольку это была исторически первая задача предсказания, то эффект регрессии дал название целому классу задач.

Нужно построить НС и так подогнать ее веса, чтобы функция *FNN(x)* давала более-менее точное значение для любого объекта *x* (*x* может быть не только числом, а числовым вектором или объектом более сложной природы: фото, текст и т.д.).

Входной слой НС соответствует размерности объектов *x.* Для задачи регрессии выходной слой состоит из одного нейрона.

## План решения задачи регрессии с помощью нейросети

Аналогичный план имеет место и для задачи классификации.

1. Взять **тренировочную выборку** (ТВ), то есть набор объектов с известными значениями целевого признака Y. Нейронная сеть в дальнейшем должна восстановить зависимость между нецелевыми признаками и целевым признаком.
2. Задать основные параметры нейронной сети:количество слоёв, количество нейронов на каждом слое, тип связи между слоями и т.д.
3. Выписать функцию *FNN(x).* Это выражение будет содержать вхождения весов *wi*.
4. Пусть *yi* – точное значение целевого признака *Y* для *i*-го объекта из тренировочной выборки, *FNN(Xi ) –* значение функции НС для *i*-го объекта из тренировочной выборки.
5. Составим квадрат разности (*FNN(Xi )-yi )2* . А затем и сумму по всем объектам тренировочной выборки:

***L(w)=(FNN(X1)-y1)2+*(*FNN(X2)-y2)2+...+*(*FNN(Xm)-ym)2*** – это суммарная ошибка НС на тренировочной выборке. *L(w)* называется функцией потерь для задачи регрессии.

1. Функция потерь *L(w)* содержит вхождения букв *wi* (весов НС). Относительно этих переменных мы находим точку минимума функции *L(w).*
2. Точка минимума определяет оптимальные веса НС.
3. Присваиваем весам НС найденные оптимальные значения.
4. Пусть теперь объект А не принадлежит тренировочной выборке. Тогда значение целевого признака Y для него вычисляется (предсказывается) по формуле *FNN(А)* с использованием найденных ранее оптимальных значений весов.

Тренировочная выборка (ТВ) строго определяет размер входного слоя. Если у объектов ТВ ***n*** нецелевых признаков, то размер входного слоя равен ***n*.**

Количество внутренних слоев – на ваше усмотрение. Слишком много (переобучение) и слишком мало слоев плохо. Выходной слой из одного нейрона.

## Улучшения градиентного спуска для нейросетей

ГС – универсальный метод поиска минимума, но функция потерь *L(w)* имеет весьма специфический вид, можно этим воспользоваться.

Итак, *L(w)* для задачи регрессии – это сумма квадратов

***L(w)=(FNN(X1)-y1)2+*(*FNN(X2)-y2)2+...+*(*FNN(Xm)-ym)2***

Каждое слагаемое этой суммы соответствует некоторому элементу ТВ. Это свойство позволяет улучшить алгоритм ГС.

**Стохастический ГС (СГС)**

1. Случайным образом выберем из суммы

***L(w)=(FNN(X1)-y1)2+*(*FNN(X2)-y2)2+...+*(*FNN(Xm )-ym )2*** одно слагаемое ***F=(FNN(Xi )-yi )2***

1. **Осуществим одну итерацию ГС для функции F.** Это приведёт к получению нового набора весов нашей НС. Возвращаемся на п.1

ГС на п.2 может осуществляться как по классической формуле, так и по «продвинутым» формулам (Momentum, Adagrad, Adam...)

**Плюсы**: СГС, как правило, находит более глубокие минимумы, чем классический ГС. Мы фактически заменяем минимизацию большой функции на минимизацию нескольких простых функций.

**1-я итерация СГС**

Итерация СГС начинается со случайного выбора слагаемого в *L(w)*. Допустим, что было выбрано первое слагаемое:

*L1(w)=*(*-w1+w0-1*)2 – делаем для него один шаг классического ГС:

*dL/dw1=-2(-w1+w0-1), dL/dw0=2(-w1+w0-1)*

*dL/dw1(0,0)=2 dL/dw0(0,0)=-2*

***a1=(0,0)-0.1\*(2,-2)=(-0.2, 0.2) –* вот новая точка спуска.**

После этого начинается вторая итерация СГС.

**2-я итерация СГС**

Допустим, что было выбрано второе слагаемое:

*L2(w)=*(*w0-0*)2 – делаем для него один шаг классического ГС:

*dL/dw1=0, dL/dw0=2w0*

*dL/dw1(-0.2,0.2)=0 dL/dw0(-0.2,0.2)=0.4*

***a2=(-0.2,0.2)-0.1\*(0,0.4)=(-0.2, 0.16) –* вот новая точка спуска.**

После этого начинается третья итерация СГС и так далее.

Обычно поступают так: в случайном порядке перебирают все слагаемые в функции потерь (их число совпадает с объемом тренировочной выборки) и для выбранного слагаемого вычисляют градиент и производят сдвиг по формуле ГС. То есть делают СГС.

Когда будут просмотрены все слагаемые в функции потерь, то говорят, что прошла одна эпоха обучения. Число итераций в эпохе равно размеру ТВ.

Потом начинается следующая эпоха и перебор слагаемых (в другом случайном порядке) повторяется.

Количество эпох является одним из входных параметров обучения НС.

Мы рассмотрели две крайности в способах минимизации функции потерь:

* **«обычный» ГС** (когда минимизируется вся функция потерь);
* **СГС** (когда на каждой итерации спуска минимизируется одно случайное слагаемое).

Но как всегда бывает: **истина посередине**, то есть оптимальна промежуточная стратегия – **СГС по мини-батчам**.

Фиксируется размер батча (группы) *k*. После этого на каждом шаге происходит выбор случайных k слагаемых из функции потерь и вычисляется градиент по сумме этих k слагаемых.

При *k=1* – это обычный СГС.

При k={объем тренировочной выборки} – это минимизация всей функции потерь. То есть это - классический ГС функции потерь.

Пусть k – размер мини-батча. Тогда объекты ТВ случайным образом разбиваются на m/k батчей (m — объем ТВ).

Далее делается m/k шагов ГС. Эпоха обучения на этом заканчивается. После этого происходит новое случайное разбиение ТВ на m/k батчей и т.д.