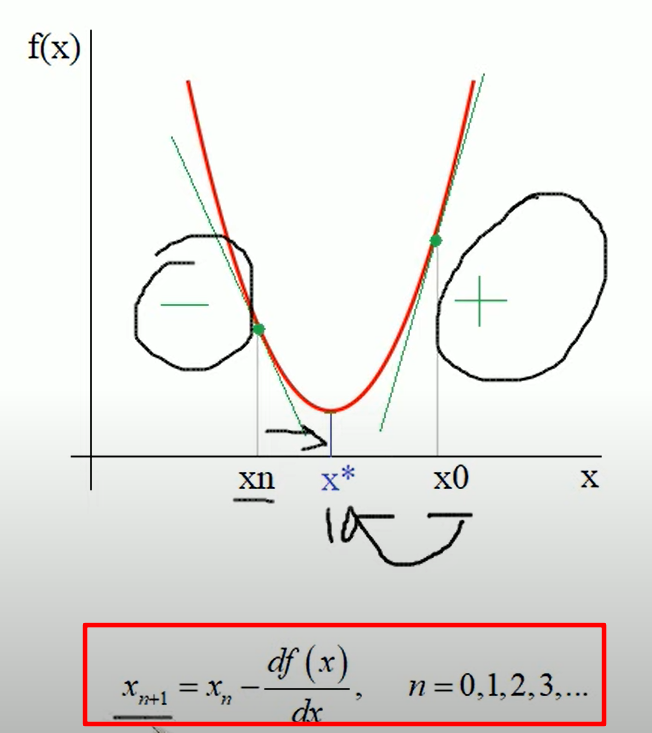
1. [Метод градиентного спуска](#_Метод_градиентного_спуска)
2. [Модификации градиентного спуска, использующиеся в нейронных сетях. SGD, AdaGrad, Adam, метод Нестерова.](#_Модификации_градиентного_спуска,)
3. [Функции активации ReLU, PReLU](#_Функции_активации_ReLU,).
4. [Функции активации Sigmoid, гиперболический тангенс.](#_Функции_активации_Sigmoid,)
5. [Однослойная нейросеть и ее обучение](#_Однослойная_нейросеть_и).
6. [Двуслойная нейросеть и ее обучение.](#_Двуслойная_нейросеть_и)
7. Пакет pytorch, основные возможности.
8. Вычислительный граф.
9. Переобучение. Использование TensorBoard. Dropout.
10. Обучение с подкреплением. Reinforce. Softmax и эпсилон жадная стратегия.
11. Сверточные нейросети (CNN).
12. Рекуррентные нейросети (RNN).
13. Генетический алгоритм.

# Метод градиентного спуска

Градиентный спуск — метод нахождения минимального значения функции потерь. Минимизация любой функции означает поиск самой глубокой впадины в этой функции.

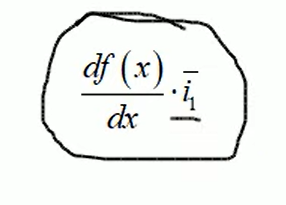
x\* – искомый минимум. Производная справа от минимума принимает положительное значение, слева – отрицательное. Чтобы сходиться к минимуму необходимо в каждой итерации вычитать из предыдущего значения x (абсцисс) значение производной в текущей точке xn ..

На каждой итерации алгоритма мы должны получать новое значение на оси абсцисс, используя прежнее значение минус значение производной в точке xn.



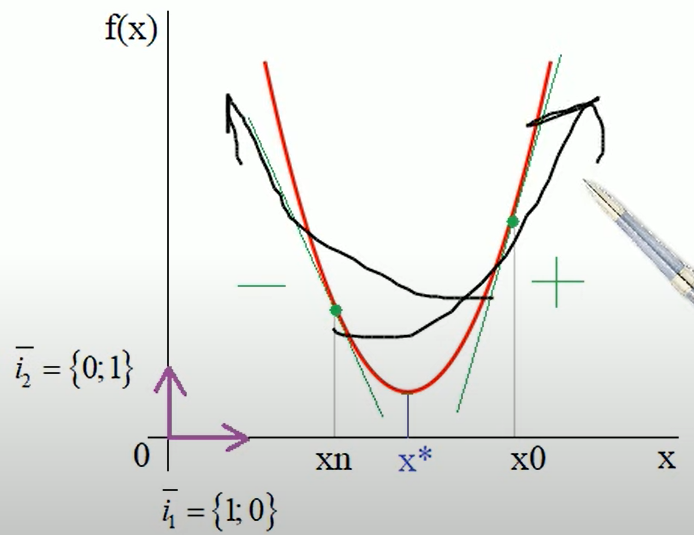
Градиент обозначает в каком направлении необходимо двигаться, чтобы найти точку экстремума x\*.

Градиент находится по след формуле:

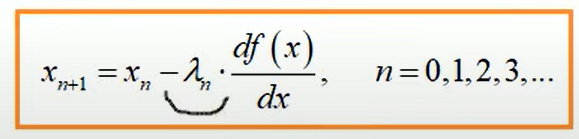
, где i – единичный вектор .

Т.е. если градиент положительный – значит что он направлен в доль оси абсцисс (вправо), поэтому мы добавляем минус, чтобы двигаться к точке оптимума (минимума).

Значение производной может быть очень большим и мы можем уйти далеко влево или вправо. Для этого производную дополнительно умножают на небольшое число (в данном примере лямбда, но вообще альфа у нас была)



Таким образом получаем следующую формулу:

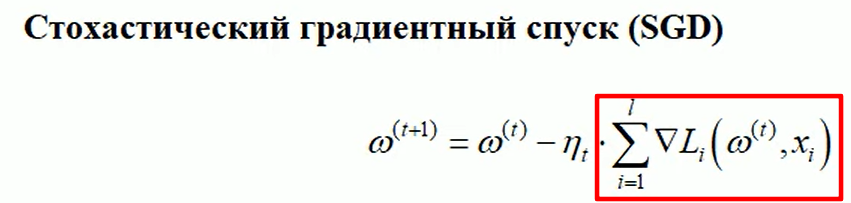
,

Где xn – значение ф-ии в предыдщую итерацию, (лямбдаn или шаг сходимости алгоритма) – шаг с которым мы перемещаемся по оси ординат в поиске минимума, (df(x)/dx) – производная в данной точке.

# Модификации градиентного спуска, использующиеся в нейронных сетях. SGD, AdaGrad, Adam, метод Нестерова.

## Метод стохастического градиентного спуска (SGD)

Сумма градиентов от функции потерь по всем объектам обучающей выборки.

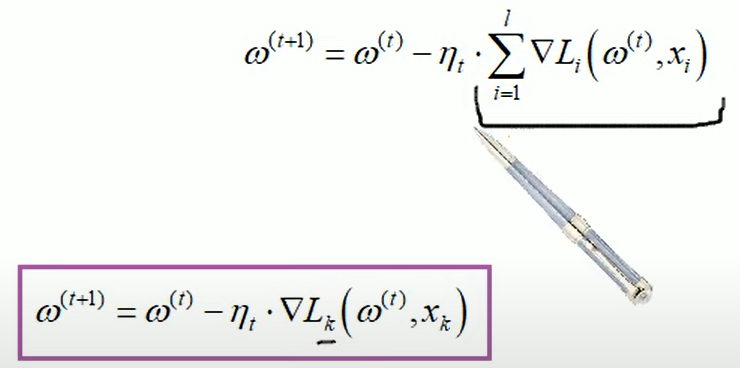


Обучающая выборка может состоять из огромного числа элементов (l), поэтому при практической реализации градиент, заменяют на псевдоградиент.

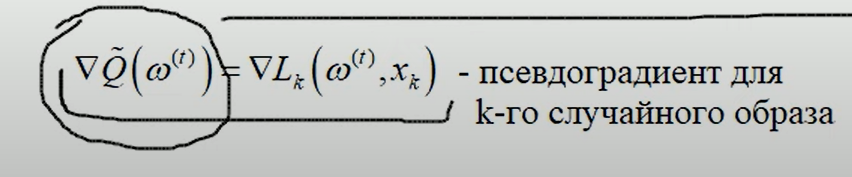
Псевдоградиент должен обладать одним главным свойством: в среднем псевдоградиент должен образовывать острый угол в n-мерном признаковом пространстве с истинным градиентом.

Самым простым способом упростить вычисление градиента является:

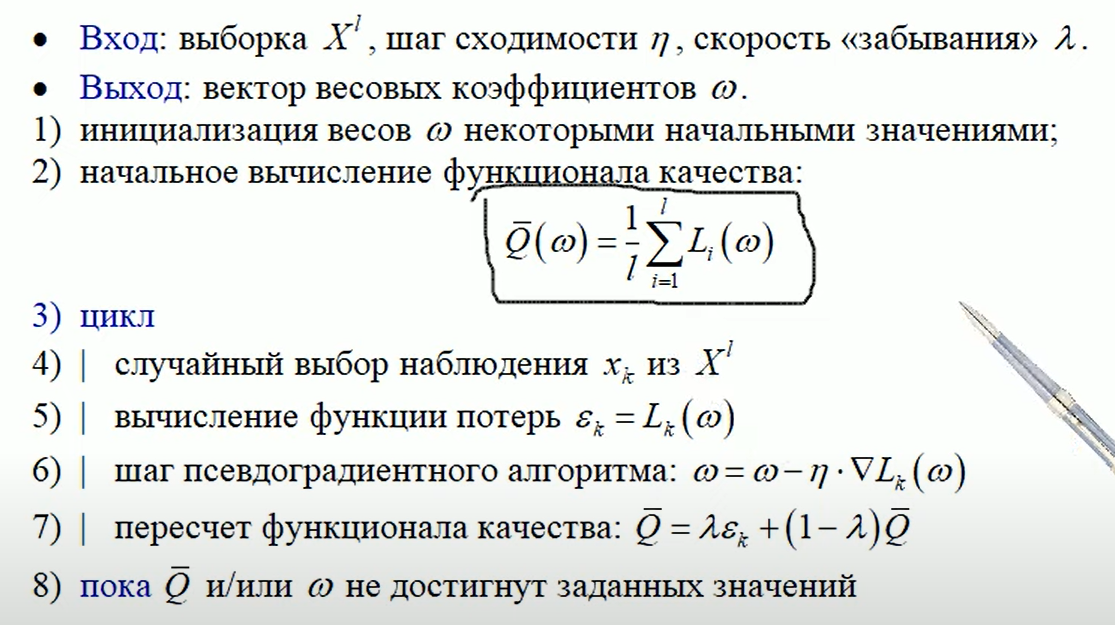
Брать из всей суммы только одно наблюдение и по нему вычислять текущее приблежение



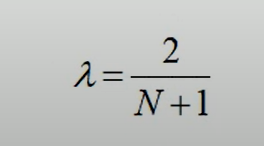
На каждой итерации берем случайный объект из обучающей выборки, вычисляем антиградиент только по выбранному объекту и корректируем набор весов. В таком случае градиент определяется как градиент функции потерь для k-го элемента.



Таким образом алгоритм будет иметь следующий вид:



Изначально функционал качества вычисляется по всем значениям функции потерь обучающей выборки.

В цикле при пересчете функционала качества для оптимизации алгоритма используется экспоненциальная скользящая средняя (для чего и нужен параметр «забывания», это значение обычно вычисляется по след. правилу , где N – интервал сглаживания. Т.е. будут учитвываться последние N элементов выборки)

Основным недостатком SGD алгоритма является существенная неравномерность сходимости к точке минимума.

Для решения этого недостатка локальные градиенты вычисляются по пакетам наблюдений (батчам).

## AdaGrad

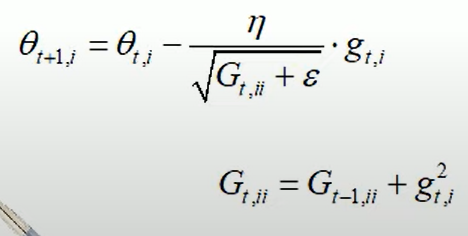
AdaGrad – adaptive gradient.

Базовая идея алгоритма заключается в том, что некоторые параметры могут быстрее достигать своего оптимума (минимума), чем другие. Т.е. те параметры, что близки к оптимуму следует менять с меньшим шагом.

Шаг для параметров зависит от величины их флуктуации колебаний: чем они больше, тем меньше шаг. Такой алгоритм хорошо подходит для сильно разреженных данных.



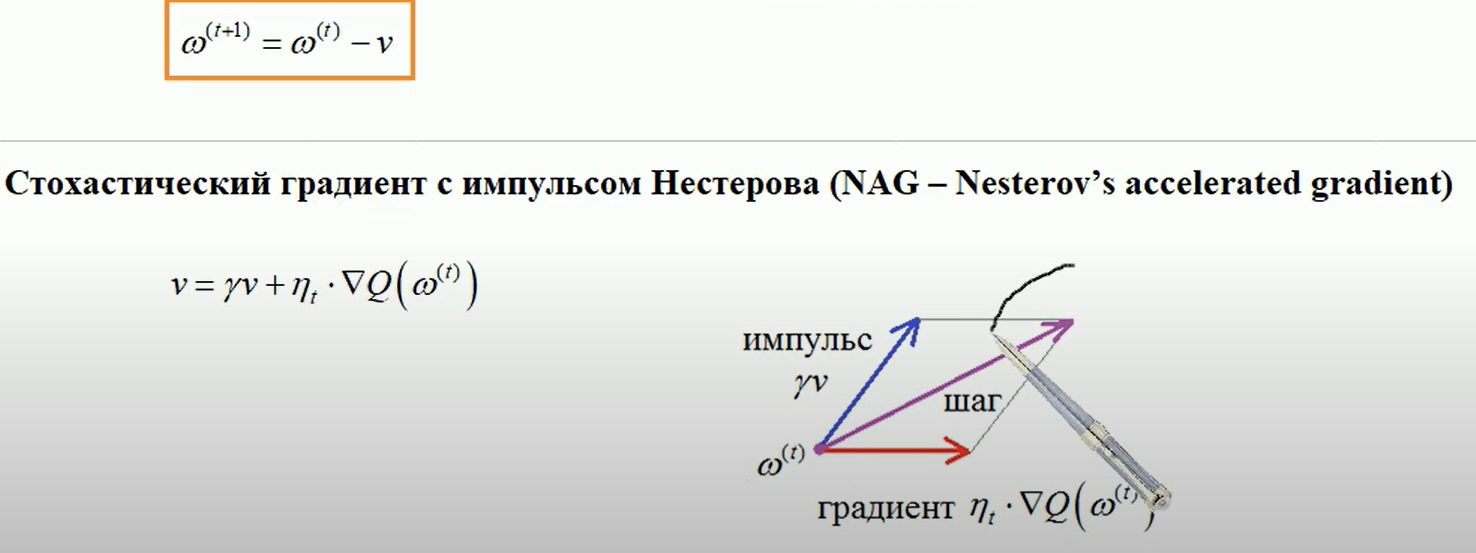
Градиент вычисляется отдельно для каждого i-го параметра, а затем применяем его для оптимизации.



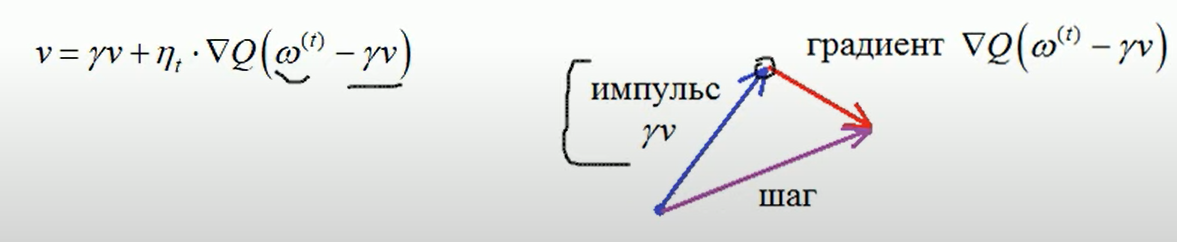
В числителе шаг обучения (ню) уменьшается на величину суммы квадратов градиентов вычисленных на предыдущих шагах, эпсилон – это некоторая положительная величина для избежания деления на 0.

## метод Нестерова

Для расчета градиента используется метот экспоненциальной скользящей средней.

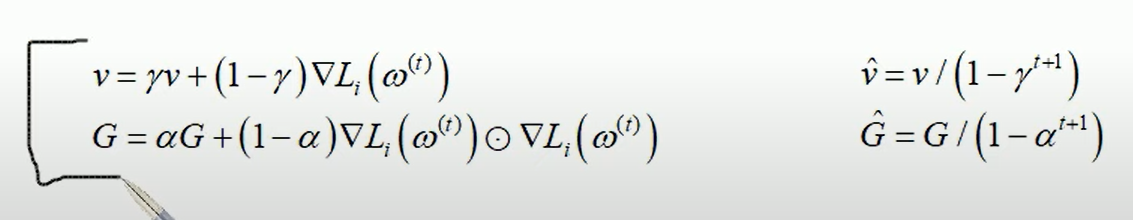


В данном алгоритме градиент функционала качества считается не в точке w(t) , а в точке, на которую мы смещаемся благодаря импульсу yv.

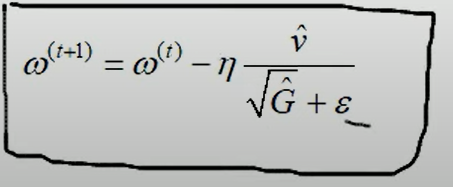


## Adam

Adam — adaptive moment estimation, ещё один оптимизационный алгоритм. Он сочетает в себе и идею накопления движения и идею более слабого обновления весов для типичных признаков. Получен объединением метода импульсов и RMSProp



Вычисляем экспоненциальные скользящие средние, затем выполняем их нормировку, после чего используем эти параметры для корректировки весов:



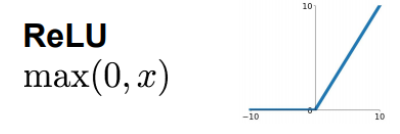
# Функции активации ReLU, PReLU

Функция активации (англ. activation function) a(x) определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов и порогового значения.

## Relu

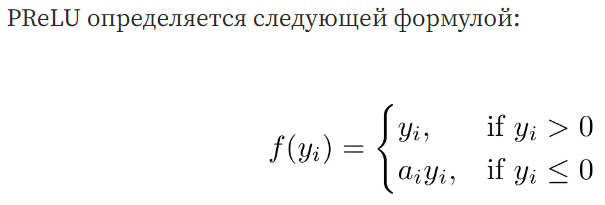
ReLU или Rectified Linear Unit вычисляет функцию f(x) = max(0,x), то есть просто выдаёт значения «ноль» и «не ноль». Это решает проблему обнуления градиента для положительных чисел. Кроме того, ReLU очень просто вычисляется: примерно в шесть раз быстрее сигмоиды и тангенса. Однако, в ней снова отсутствует нулевое центрирование.

Другой очевидный недостаток — градиент по-прежнему «умирает» при отрицательных входных данных. Это может привести к тому, что половина нейронов будет неактивна и не сможет обновляться.



## PReLu

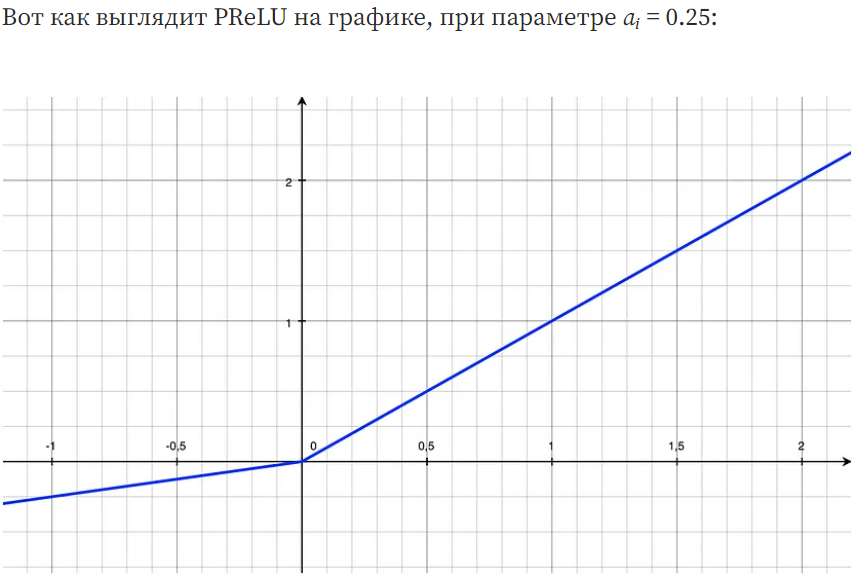
PReLU или Parametric ReLU — это логическое развитие обыкновенной активационной функции ReLU. Основное отличие между ними состоит в том, что у PReLU есть обучаемые параметры.



В этой формуле yᵢ — это то, что приходит на вход активационной функции канала i, а aᵢ — это коэффициент, отвечающий за наклон отрицательной части. Индекс i у параметра aᵢ показывает, что нелинейная активация может быть разной для разных каналов. При этом, если aᵢ приравнять к нулю, активационная функция превратится в простую ReLU.

Формулу выше можно записать ещё и немного по-другому:





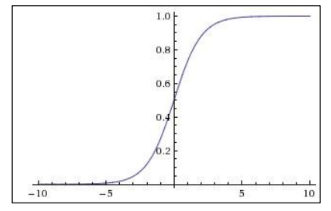
# Функции активации Sigmoid, гиперболический тангенс

Функция активации (англ. activation function) a(x) определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов и порогового значения.

## Sigmoid

Функция сигмоиды преобразовывает поступающие в неё значения в вещественный диапазон [0, 1]. То есть, если входные данные окажутся большими положительными значениями, то после преобразования они будут равны примерно единице, а отрицательные числа станут близки к нулю. Это довольно популярная функция, которую можно интерпретировать как частоту возбуждения нейрона





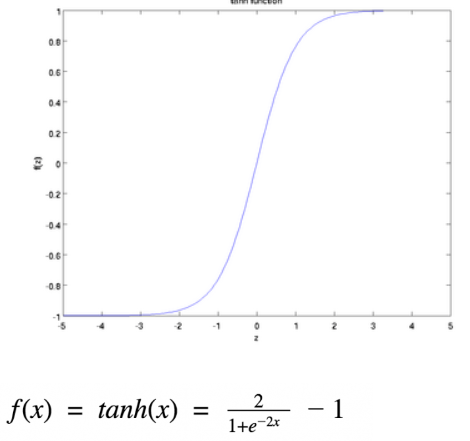
Сигмоида не работает для слишком высоких положительных или отрицательных данных.

Следует изначально подготавливать данные таким образом, чтобы их средним значением являлся ноль иначе это приведёт к тому, что все веса при обновлении также будут либо увеличены, либо уменьшены, и градиентный поток станет зигзагообразным.

Функцию exp() достаточно дорого считать

## гиперболический тангенс

Тангенс очень похож на сигмоиду, но обладает двумя существенными отличиями: он преобразует данные в диапазон [-1, 1] и имеет нулевое центрирование, что исключает вторую проблему сигмоиды. Значения градиента при обратном распространении по-прежнему могут обнуляться, тем не менее, использование тангенса обычно более предпочтительно.



Гиперболический тангенс очень похож на сигмоиду. И действительно, это скорректированная сигмоидная функция.

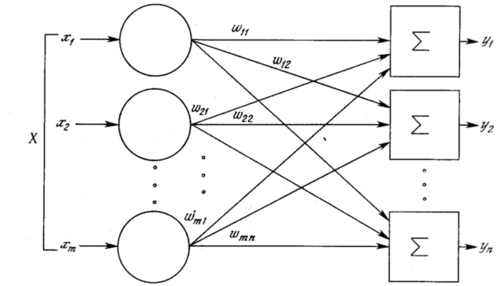


# Однослойная нейросеть и ее обучение

*Нейро́нная сеть* (также искусственная нейронная сеть, ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

*Однослойная нейронная* сеть (англ. 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.

Как видно из схемы однослойной нейронной сети, представленной снизу, сигналы x1,x2,…xn поступают на входной слой (который не считается за слой нейронной сети), а затем сигналы распределяются на выходной слой обычных нейронов. На каждом ребре от нейрона входного слоя к нейрону выходного слоя написано число — вес соответствующей связи.



*Нейрон* — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше.

У каждого из нейронов есть 2 основных параметра: входные данные (input data) и выходные данные (output data). В случае входного нейрона: input=output. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего, она нормализуется, с помощью функции активации (пока что просто представим ее f(x)) и попадает в поле output.

*Функция активации* — это способ нормализации входных данных (мы уже говорили об этом ранее). То есть, если на входе у вас будет большое число, пропустив его через функцию активации, вы получите выход в нужном вам диапазоне.

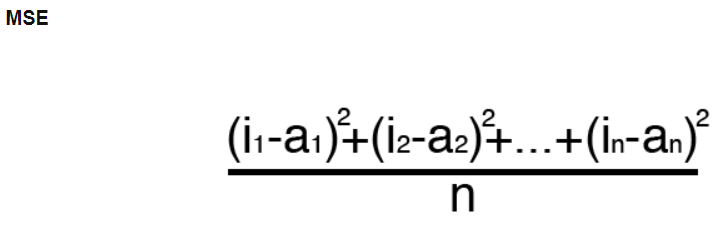
*Обучающая выборка* — конечный набор входных сигналов (иногда вместе с правильными выходными сигналами), по которым происходит обучение сети.

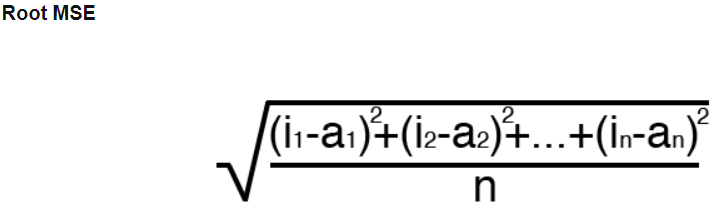
После обучения сети, то есть когда сеть выдает корректные результаты для всех входных сигналов из обучающей выборки, ее можно использовать на практике. Однако прежде чем сразу использовать нейронную сеть, обычно производят оценку качества ее работы на так называемой тестовой выборке.

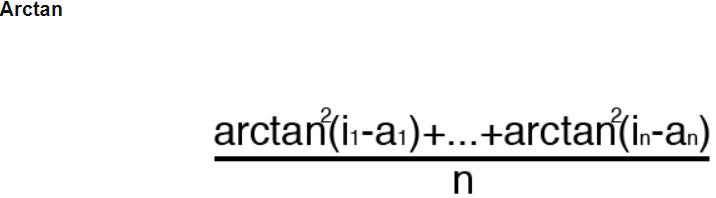
*Тестовая выборка* — конечный набор входных сигналов (иногда вместе с правильными выходными сигналами), по которым происходит оценка качества работы сети.

Само обучение нейронной сети можно разделить на два подхода: *обучение с учителем*. и *обучение без учителя*. В первом случае *веса меняются* так, чтобы ответы сети минимально отличались от уже готовых правильных ответов, а во втором случае сеть самостоятельно классифицирует входные сигналы.

*Ошибка* — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами. Ошибка формируется каждую эпоху и должна идти на спад. Если этого не происходит, значит, вы что-то делаете не так. Ошибку можно вычислить разными путями, но мы рассмотрим лишь три основных способа: *Mean Squared Error (далее MSE), Root MSE и Arctan*.







# Двуслойная нейросеть и ее обучение

*Нейро́нная сеть* (также искусственная нейронная сеть, ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

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

*Функция активации* — это способ нормализации входных данных (мы уже говорили об этом ранее). То есть, если на входе у вас будет большое число, пропустив его через функцию активации, вы получите выход в нужном вам диапазоне.

*Обучающая выборка* — 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 сразу использовать нейронную сеть, обычно производят оценку качества ее работы на так называемой тестовой выборке.

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

Само обучение нейронной сети можно разделить на два подхода: *обучение с учителем*. и *обучение без учителя*. В первом случае *веса меняются* так, чтобы ответы сети минимально отличались от уже готовых правильных ответов, а во втором случае сеть самостоятельно классифицирует входные сигналы.

*Ошибка* — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами. Ошибка формируется каждую эпоху и должна идти на спад. Если этого не происходит, значит, вы что-то делаете не так. Ошибку можно вычислить разными путями, но мы рассмотрим лишь три основных способа: *Mean Squared Error (далее MSE), Root MSE и Arctan*.

