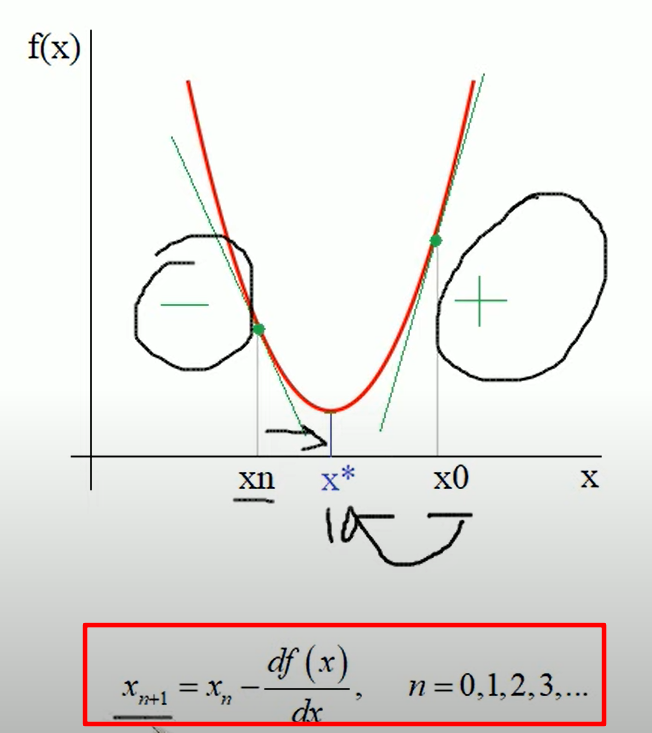
1. [Метод градиентного спуска](#_Метод_градиентного_спуска)
2. Модификации градиентного спуска, использующиеся в нейронных сетях. SGD, AdaGrad, Adam, метод Нестерова.
3. Функции активации ReLU, PReLU.
4. Функции активации Sigmoid, гиперболический тангенс.
5. Однослойная нейросеть и ее обучение.
6. Двуслойная нейросеть и ее обучение.
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.

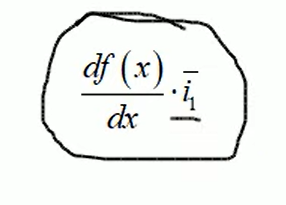
x\* – искомый минимум. Производная справа от минимума принимает положительное значение, слева – отрицательное. Чтобы сходиться к минимуму необходимо в каждой итерации вычитать из предыдущего значения x (абсцисс) значение производной в текущей точке xn ..

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



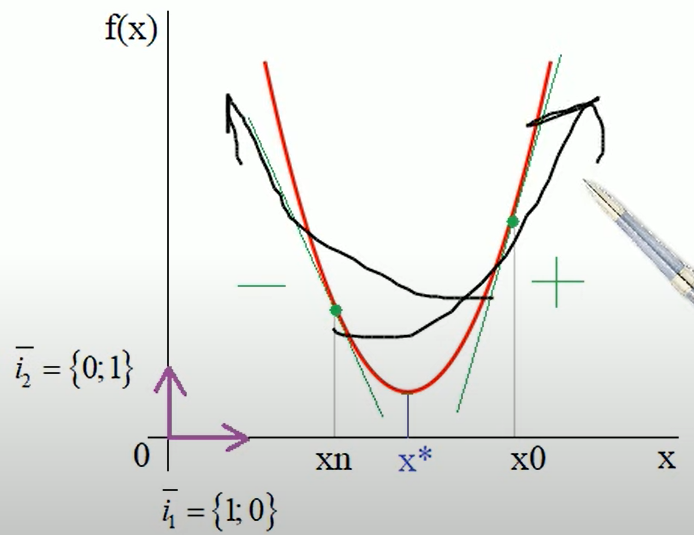
Градиент обозначает в каком направлении необходимо двигаться, чтобы найти точку экстремума x\*.

Градиент находится по след формуле:

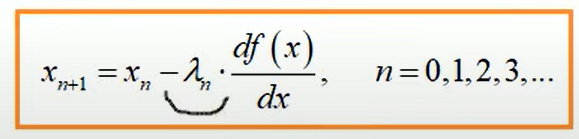
, где i – единичный вектор .

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:

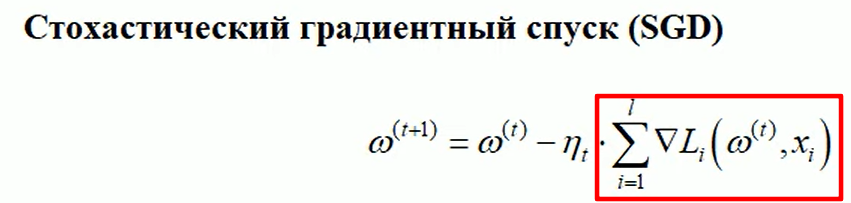
,

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.

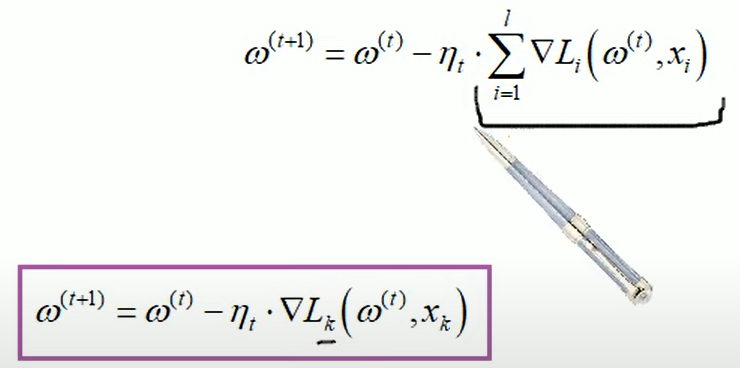


Обучающая выборка может состоять из огромного числа элементов (l), поэтому при практической реализации градиент, заменяют на псевдоградиент.

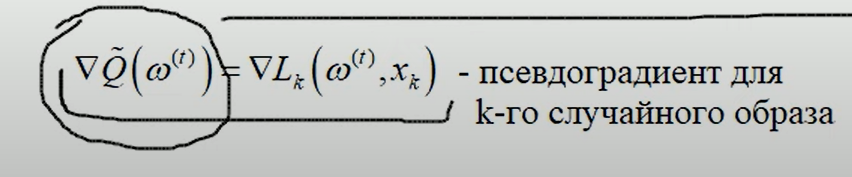
Псевдоградиент должен обладать одним главным свойством: в среднем псевдоградиент должен образовывать острый угол в n-мерном признаковом пространстве с истинным градиентом.

Самым простым способом упростить вычисление градиента является:

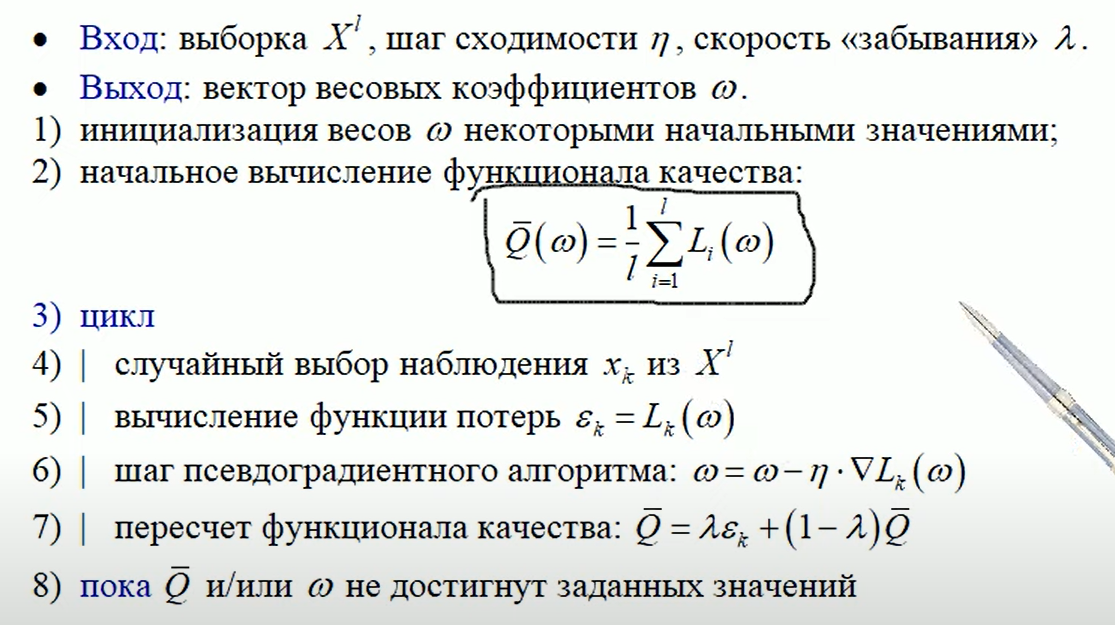
Брать из всей суммы только одно наблюдение и по нему вычислять текущее приблежение



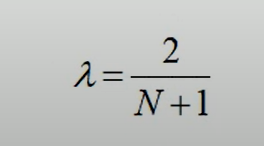
На каждой итерации берем случайный объект из обучающей выборки, вычисляем антиградиент только по выбранному объекту и корректируем набор весов. В таком случае градиент определяется как градиент функции потерь для k-го элемента.



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

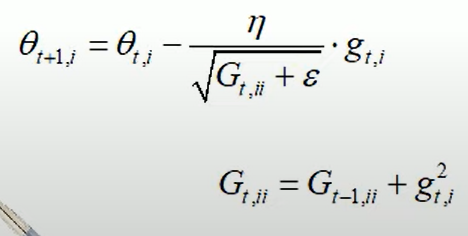
AdaGrad – adaptive gradient.

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.



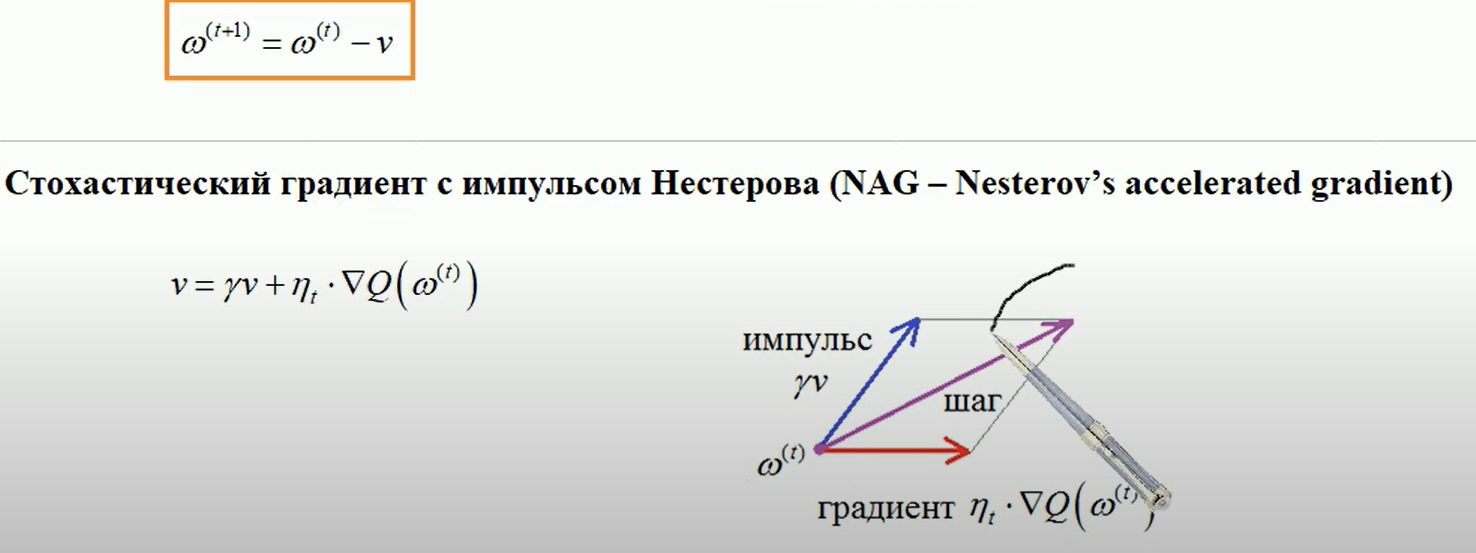
Градиент вычисляется отдельно для каждого i-го параметра, а затем применяем его для оптимизации.



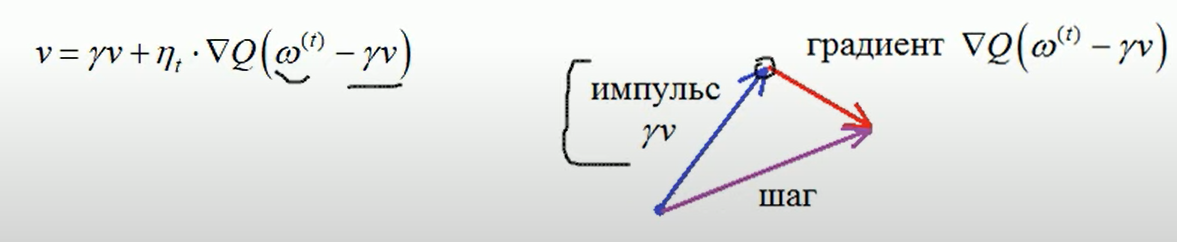
В числителе шаг обучения (ню) уменьшается на величину суммы квадратов градиентов вычисленных на предыдущих шагах, эпсилон – это некоторая положительная величина для избежания деления на 0.

## метод Нестерова

Для расчета градиента используется метот экспоненциальной скользящей средней.

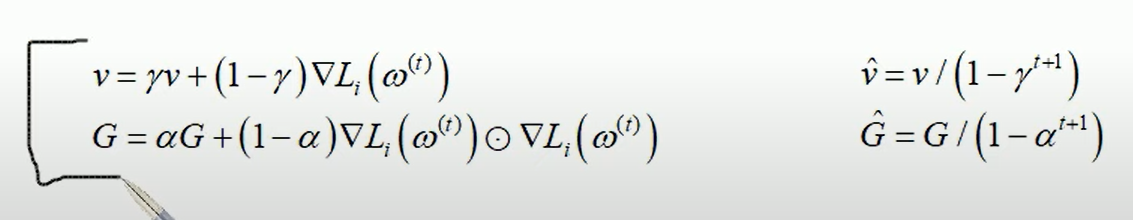


В данном алгоритме градиент функционала качества считается не в точке w(t) , а в точке, на которую мы смещаемся благодаря импульсу yv.



## 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:

