Градиентный спуск (gradient descent)

Производная, частные производные, градиент. Методы оценки градиента.

Производная – предел отношения приращения функции к приращению её аргумента при стремлении приращения аргумента к нулю, если такой предел существует.

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

Частная производная – предел отношения приращения функции по выбранному аргументу к приращению этого аргумента при стремлении приращения к нулю, если такой предел существует.

Градиент – вектор, своим направлением указывающий направление наибольшего возрастания функции, по величине равный скорости роста этой функции в этом направлении.

Связь градиента с производными: градиент функции есть вектор, компоненты которого являются частными производными этой функции.

Градиентный спуск, проблема выбора шага.

Градиентный спуск — метод нахождения локального минимума с помощью движения вдоль **градиента**.

Алгоритм:

1. В градиентном спуске выбирается начальная точка θ
2. Вычислить градиент $\nabla f(\theta)$
3. ЕСЛИ $\nabla f(\theta) < \varepsilon$
   ТО ВЕРНУТЬ $\boldsymbol{\theta}$
4. θ рассчитывается:
   $\theta - \alpha * \nabla f(\theta)$
5. Вернуться к шагу 2

Размер шага алгоритма определяет, насколько мы собираемся двигать точку на функции потерь, и этот параметр называется «скоростью обучения (learning rate)».

Если этот размера шага слишком велик, то будем отдаляться от минимума. Если размер шага слишком мал, мы будем использовать слишком много итераций, чтобы достичь минимума.

Стохастический градиентный спуск (SGD).

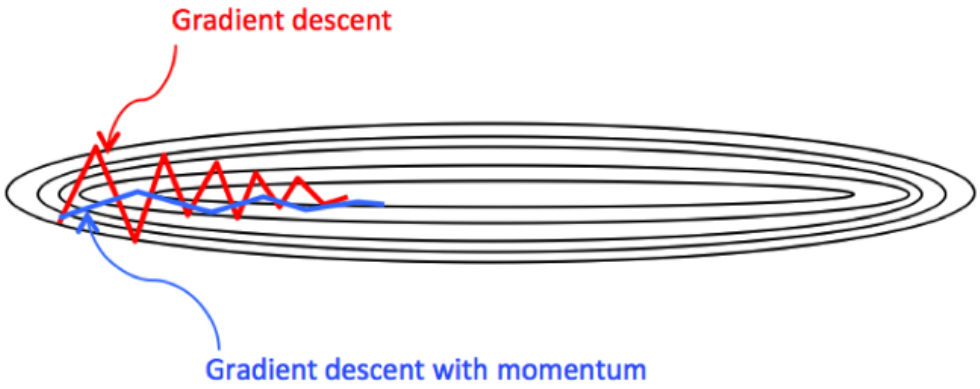
Стохастический градиентный спуск – оптимизационный алгоритм градиентного спуска, в котором градиент считается как градиент от случайно выбранного подмножества данных.

$$\theta = \theta - \alpha \nabla f\left(\theta; x^{(i)}; y^{(i)}\right)$$

Алгоритм позволяет повышать скорость одной итерации, однако это приведёт к более низкой скорости сходимости.

Использование момента

Алгоритм ускоряет стохастический градиентный спуск в соответствующем направлении и гасит лишние колебания. В алгоритме с моментом шаг может ускоряться в направлениях с малой кривизной и не становиться нестабильным в направлениях с большой кривизной.



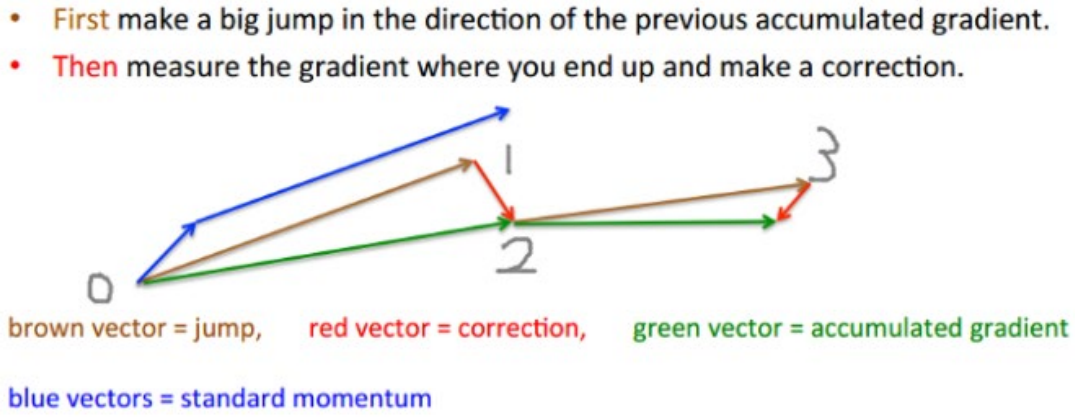
$$v_t = \gamma v_{t-1} + \alpha \nabla f(\theta)$$

$$\theta = \theta - v_t$$

Рекомендуемый параметр $\gamma = 0.9$

Метод Нестерова

Алгоритм Нестерова позволяет предвидеть подъёмы или спуски функции, тем самым значительно ускоряя алгоритм с моментом.



$$v_t = \gamma v_{t-1} + \alpha \nabla f(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$

Метод отжига

Алгоритм основывается на имитации физического процесса, который происходит при кристаллизации вещества, в том числе при отжиге металлов. Предполагается, что процесс протекает при постепенно понижающейся температуре.

Применяя его к градиентному спуску, мы каждый раз уменьшаем learning rate по мере приближения к минимуму функции.

Adagrad (адаптивный градиент)

Adagrad – алгоритм оптимизации на основе градиента, который адаптирует learning rate, основываясь на сумму квадратов прошлых градиентов.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \quad g_{t,i}$$

$G_{t,ii}$ – диагональная матрица, где каждый элемент диагонали является суммой квадратов градиентов.

ϵ – гарантирует, что знаменатель не будет равен нулю.

Однако learning rate в данном алгоритм быстро уменьшается, из-за чего алгоритм будет слишком долго сходиться.

Adadelta и RMSprop

В отличие от Adagrad, здесь сумма градиентов определяется как убывающее среднее всех квадратов градиентов, таким образом старые градиенты перестают со временем сильно влиять на текущую оптимизацию.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) {g_t}^2$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[g^2]_{t,i} + \epsilon}} \quad g_{t,i}$$

γ = 0.9
η= 0.001

Adam (Adaptive Moment Estimation) Адаптивная оценка момента

Adam рассчитывает адаптивную скорость обучения каждого параметра.

$$m_t = \beta_1 m_t - 1 + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_t - 1 + (1 - \beta_2) g_t^2$$

Так как величины в самом начале алгоритма близки к 0, используются их оценки:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \quad \hat{m}_t$$

Авторы предлагают значения по умолчанию 0,9 для β1, 0,999 для β2, и 10–8 для є.