Оптимизаторы – алгоритмы, используемые для незначительного изменения параметров с целью ускорения работы модели

**Adagrad** – оптимизатор, меняющий скорость обучения $\eta$ для каждого параметра на каждом шаге. Работает на производной функции ошибки.

$g_t = \nabla_\theta J(\theta_t)$ - производная функции

$G_t = G_t + g_t^2$

$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t+\varepsilon}} \cdot g_t$ - обновление параметров

$\eta$ - скорость обучения, которая изменяется для заданного параметра $\theta_i$ в данный момент времени на основе предыдущих градиентов, рассчитанных для данного параметра.

**Adadelta** – расширение Adagrad. Ограничивает окно накопленных прошлых градиентов до некоторого фиксированного размера, вместо того чтобы хранить их все. Используется экспоненциальная скользящая средняя, а не сумма градиентов.

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

$RMS[g]_t = \sqrt{E[g^2]_t + \varepsilon}$

$\theta_{t+1} = \theta_t - \frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} \cdot g_t$

**Adam** – работает с импульсами первого и второго порядка. Идея заключается в уменьшении скорости во избежание проскакивания минимума. В дополнение к хранению экспоненциальных скользящих средних (как в Adadelta) сохраняются экспоненциальные скользящие средние прошлых градиентов.

$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$, $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \cdot \hat{m}_t$

| Название | Формула | Недостатки |
|---|---|---|
| Gradient Descent | $\theta = \theta - \alpha\nabla J(\theta)$ | Можно застрять в локальных минимумах<br>Веса изменяются после вычисления градиента для всего набора данных<br>Требуется большой объем памяти для вычисления градиента для всего набора данных |
| Stochastic Gradient Descent | $\theta = \theta - \alpha\nabla J(\theta, x_i, y_i)$, $x_i, y_i$ – обучающие примеры | Высокая дисперсия параметров модели<br>Чтобы получить ту же конвергенцию, что и градиентный спуск, |

| | | |
|---|---|---|
| | | необходимо медленно снижать значение скорости обучения |
| Mini-Batch Gradient Descent | $\theta = \theta - \alpha\nabla J(\theta, B_i)$, $B_i$ – батчи обучающих примеров | Можно застрять в локальных минимумах<br>При неоптимальном выборе скорости обучения потребуется много времени для схождения градиентного снижения<br>(верно и для предыдущих) |
| SGD + Momentum | $v = \gamma v + \eta\nabla J(\theta)$<br>$\theta = \theta - \alpha v$ | Добавляется параметр, который нужно выбирать вручную и достаточно точно<br>Скорость обучения постоянна<br>(верно и для предыдущих) |
| Adagrad | $g_t = \nabla_\theta J(\theta_t)$,<br>$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot g_t$ | Вычислительно дорого (считается производная второго порядка)<br>Скорость обучения всегда снижается (т.е. обучение медленное) |
| Adadelta | $RMS[g]_t = \sqrt{E[g^2]_t + \varepsilon}$,<br>$\theta_{t+1} = \theta_t - \frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} \cdot g_t$ | Вычислительно дорого |
| Adam | $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$, $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$,<br>$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \cdot \hat{m}_t$ | Вычислительно дорого |
| RMSProp | $\theta_{t+1} = \theta_t - \frac{\eta}{RMS[g]_t} \cdot g_t$ | Вычислительно дорого |
| Nesterov Accelerated Gradient | $v = \gamma v + \eta\nabla J(\theta - \gamma v)$<br>$\theta = \theta - v$ | Низкая скорость обучения |