Морозов Леонид 519/2

I) Описание основных оптимизаторов

а) Gradient Descent (GD) - градиентный спуск:

Веса изменяются следующим образом (на каждой эпохе):

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_\theta J(\theta_t)$$

θ_{t+1}, θ_t — вектор весов до и после изменения; α — скорость обучения (learning rate); $J(\theta_t)$ — функция потерь

б) Stochastic Gradient Descent (SGD) - стохастический градиентный спуск:

Веса изменяются следующим образом (на каждой эпохе, для каждого входного объекта (sample)):

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_\theta J(\theta_t)$$

в) Mini-Batch Gradient Descent

Веса изменяются следующим образом (на каждой эпохе, для каждой выборке из N входных объектов):

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_\theta \cdot J(\theta_t)$$

Лучше на каждом шаге выбирать случайные N входных объектов (помогает избежать ситуации, когда алгоритм застревает в какой-то окрестности некоторого значения вектора весов, т.е. когда значение функции потерь почти не меняется)

г) SGD + Momentum (стохастический градиентный спуск с использованием импульса)

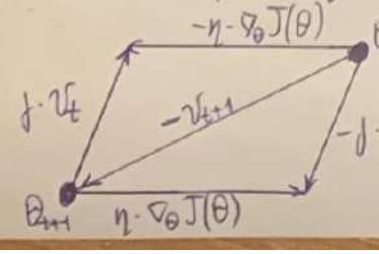
Веса изменяются следующим образом (на каждой эпохе, для каждого входного объекта):

$$v_{t+1} = \gamma \cdot v_t + \eta \cdot \nabla_\theta J(\theta_t)$$

$$\theta_{t+1} = \theta_t - v_{t+1}$$

По сути такой оптимизатор при очередном изменении весов учитывается их изменение на предыдущем шаге алгоритма:

$$\theta_{t+1} = \theta_t - (\gamma \cdot v_t + \eta \cdot \nabla_\theta J(\theta)) = \theta_t - \gamma \cdot v_t - \eta \cdot \nabla_\theta J(\theta)$$



Таким образом реализуется следующая идея: если изменение происходило определённое время в определённом направлении, то, скорее всего, следует производить изменения в том же направлении и в будущем.

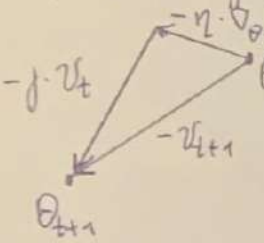
д) NAG (Nesterov's accelerated gradient - метод Нестерова)

Веса изменяются так:

$$v_{t+1} = \gamma \cdot v_t + \eta \cdot \nabla_\theta J(\theta_t - v_t)$$

$$\theta_{t+1} = \theta_t - v_{t+1}$$

По сути этот алгоритм является ускоренным SGD+Momentum: добавляется идея „заглядывания вперёд", когда градиент вычисляется не в текущей точке, а изменённой текущей точке так же как и на предыдущем шаге (v_t). Это помогает снизить влияние выбросов на процесс обучения



Таким образом, этот алгоритм позволяет быстрее „двигаться" туда, где градиент больше, и медленнее, если наоборот.

е) Adagrad (adaptive gradient)

$$g_t = \nabla_\theta J(\theta_t), \quad G_t = G_{t-1} + g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} g_t \qquad \varepsilon << 1, \text{ нужно для того, чтобы избежать деления на } 0$$

Основная идея данного алгоритма заключается в накоплении квадратов изменений весов (в G_t). Таким образом, координаты вектора весов будут меняться быстрее с большим значением градиента, и медленнее с маленьким значением градиента. Также уменьшаются изменения координат вектора весов, которые часто изменялись.

ж) RMSProp (root mean square propagation)

$g_t = \nabla_\theta J(\theta_t)$ $G_t = \gamma G_{t-1} + (1-\gamma) g_t^2 = E[g^2]_t$ — экспоненциально затухающее бегущее среднее

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} g_t \qquad RMS[g]_t = \sqrt{E[g^2]_t + \varepsilon}$$

Основная идея такая же, что и у Adagrad, однако в этом алгоритме решена проблема в потенциально увеличивающемся G_t, что со временем приведёт к слишком маленьким изменениям весов, за счёт бегущего среднего.

3) Adadelta

$$g_t = \nabla_\theta J(\theta_t) \quad E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2, \quad RMS[g]_t = \sqrt{E[g^2]_t + \varepsilon}$$

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

Отличается от RMSProp только добавлением в числитель $RMS[\Delta\theta]_{t-1}$. Это сделано для того, чтобы размерности θ и $\Delta\theta$ совпадали (в Adagrad и RMSProp не совпадали)

4) Adam (adaptive moment estimation)

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad g_t = \nabla_\theta J(\theta_t)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

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

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

Этот алгоритм является комбинацией RMSProp, AdaGrad, SGD+Momentum.

II) Сравнение оптимизаторов

Оптимизатор	Недостатки оптимизатора
GD	Из-за больших значений $\nabla_\theta J(\theta_t)$ может вообще не сойтись без специально подобранного learning rate.
SGD	Очень долго считать градиенты для каждого объекта, также нужно эмпирически подбирать learning rate
Mini-batch GD	Быстрее, чем SGD, т.к. меньше градиентов нужно вычислять. Недостатком Mini-Batch GD, как и предыд. оптимизаторов является „застревание" в локальных минимумах или седловых точках. Также нужно подбирать learning rate.
SGD+Momentum NAG	Долго обучается для некрутых (с малым значением производных) наклонов поверхности, порождаемой функцией потерь.
Adagrad	Со временем G_t увеличится настолько, что веса почти не будут меняться ⇒ остановится процесс обучения
RMSProp, Adadelta Adam	Ликвидируется проблема Adagrad. В RMSProp отличаются размерности θ и $\Delta\theta$, что плохо. Общим недостатком для RMSProp, Adadelta, Adam является плохая сходимость на специфичных поверхностях. Этот недостаток - общий для всех, только у других оптимизаторов таких поверхностей больше.