Глубокое обучение для обработки изображений

Лекция 7

Вспомним лекцию про обучении НС

При обучении нейронных сетей мы хотим найти оптимальные параметры модели. Для этого используются **градиентные методы**; с помощью них мы итеративно обновляем веса нашей модели. Для обновления весов нам необходимо знать градиенты функции потерь, которые можно найти с помощью алгоритма **обратного распространения ошибки (backpropagation).**

Процесс обучения будет состоять таким образом из следующих шагов:

- На вход подаются данные, которые передаются по сети в прямом направлении, в результате чего получаются выходные данные (**Forward pass**)
- Сигнал ошибки передается в обратном направлении, находятся значения градиентов (**Backward pass**)
- С помощью оптимизатора (SGD, Adam и др.) обновляем значения весов

Вспомним лекцию про обучении НС

При обучении нейронных сетей мы хотим найти оптимальные параметры модели. Для этого используются **градиентные методы**; с помощью них мы итеративно обновляем веса нашей модели. Для обновления весов нам необходимо знать градиенты функции потерь, которые можно найти с помощью алгоритма **обратного распространения ошибки (backpropagation).**

Процесс обучения будет состоять таким образом из следующих шагов:

- На вход подаются данные, которые передаются по сети в прямом направлении, в результате чего получаются выходные данные (**Forward pass**)
- Сигнал ошибки передается в обратном направлении, находятся значения градиентов (**Backward pass**)
- С помощью оптимизатора (SGD, Adam и др.) обновляем значения весов

Градиентный спуск (Gradient Descent)

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α

Повторять для каждой эпохи:

Найти градиент: $\boldsymbol{g} \leftarrow \frac{1}{N} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}_i, \boldsymbol{\theta}_i), \boldsymbol{y}_i)$

Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{g}$

Самая базовая разновидность градиентных методов обучения нейронных сетей, в котором весь тренировочный датасет используется для осуществления одного обновления весов сети.

Хранение градиентов в памяти может быть затратным. Также можно улучшить/ускорить сходимость, разбивая входные данные на батчи и обновляя веса не в конце эпохи, а после прохода через сеть каждого батча.

Стохастический градиентный спуск (SGD)

Пусть N – количество объектов в датасете. **Батч (batch)** представляет собой часть данных с $m < N$ объектами: данные $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ и соответствующий вектор лейблов **y.**

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α

Повторять для каждой эпохи:

Повторять Для каждого батча:

Найти оценку градиента: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}_i, \boldsymbol{\theta}_i), \boldsymbol{y}_i)$

Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{g}$

Стохастический градиентный спуск (SGD)

Пусть N – количество объектов в датасете. **Батч (batch)** представляет собой часть данных с $m < N$ объектами: данные $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ и соответствующий вектор лейблов **y.**

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α

Повторять для каждой эпохи:

Повторять Для каждого батча:

Найти оценку градиента: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}_i, \boldsymbol{\theta}_i), \boldsymbol{y}_i)$

Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{g}$

Методы, в которых $m = 1$, называют **stochastic методами**. Если размер батча лежит между 1 и N, то **minibatch stochastic,** однако в настоящее время слово minibatch опускают

Метод инерции (Импульс, Momentum)

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α, параметр импульса μ, начальное значение $\boldsymbol{\vartheta}$

Повторять для каждой эпохи:

Повторять Для каждого батча:

Найти оценку градиента: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x_i}, \boldsymbol{\theta}), \boldsymbol{y}_i)$

Обновить импульс: $\boldsymbol{\vartheta} \leftarrow \mu\boldsymbol{\vartheta} - \alpha\boldsymbol{g}$

Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\vartheta}$

Для улучшения сходимости SGD можно использовать информацию о предыдущих шагах алгоритма. В методе инерции вводится переменная $\boldsymbol{\vartheta}$, которая играет роль вектора “скорости”, указывающей направление и скорость движения точки в пространстве параметров.

Импульс Нестерова (Nesterov Momentum)

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α, параметр импульса μ, начальное значение $\boldsymbol{\vartheta}$

Повторять для каждой эпохи:

Повторять Для каждого батча:

Найти промежуточное значение: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \mu\boldsymbol{\vartheta}$

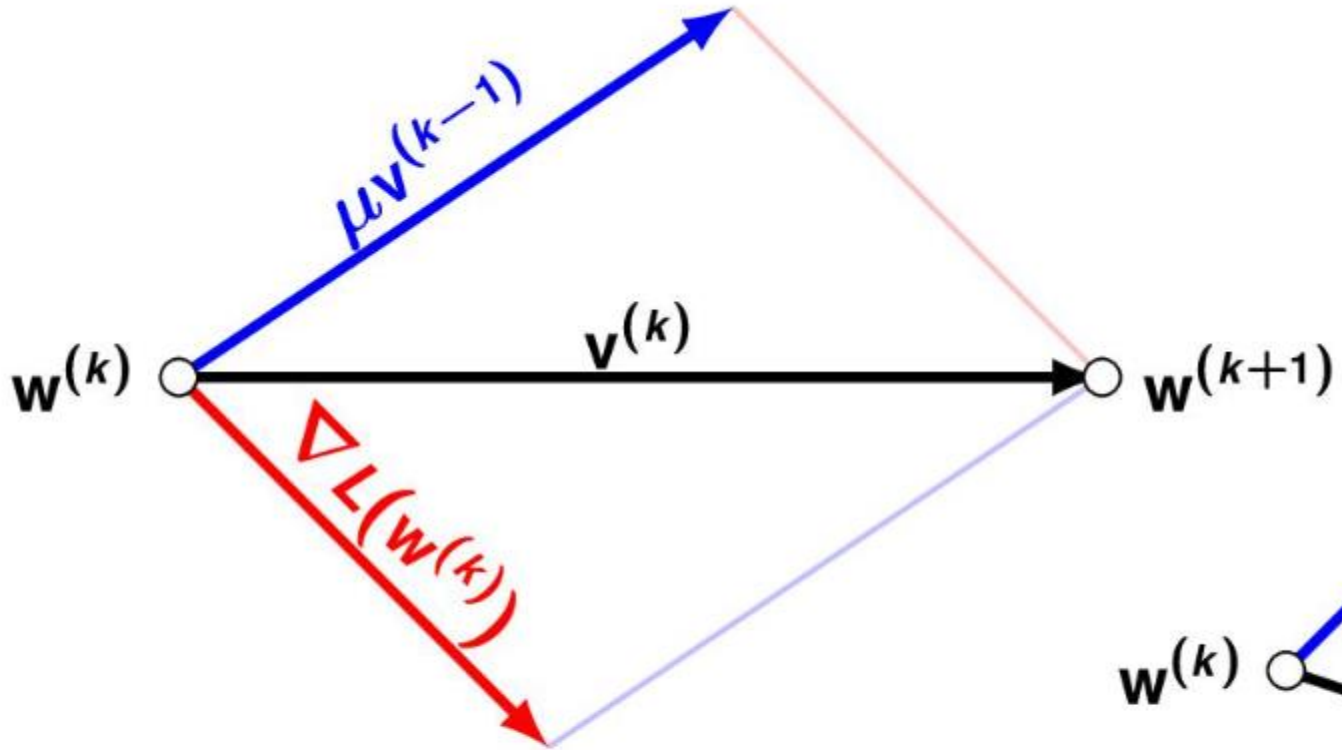
Найти оценку градиента: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}'} \sum_i L(f(\boldsymbol{x}_i, \boldsymbol{\theta}'), \boldsymbol{y}_i)$

Обновить импульс: $\boldsymbol{\vartheta} \leftarrow \mu\boldsymbol{\vartheta} - \alpha\boldsymbol{g}$

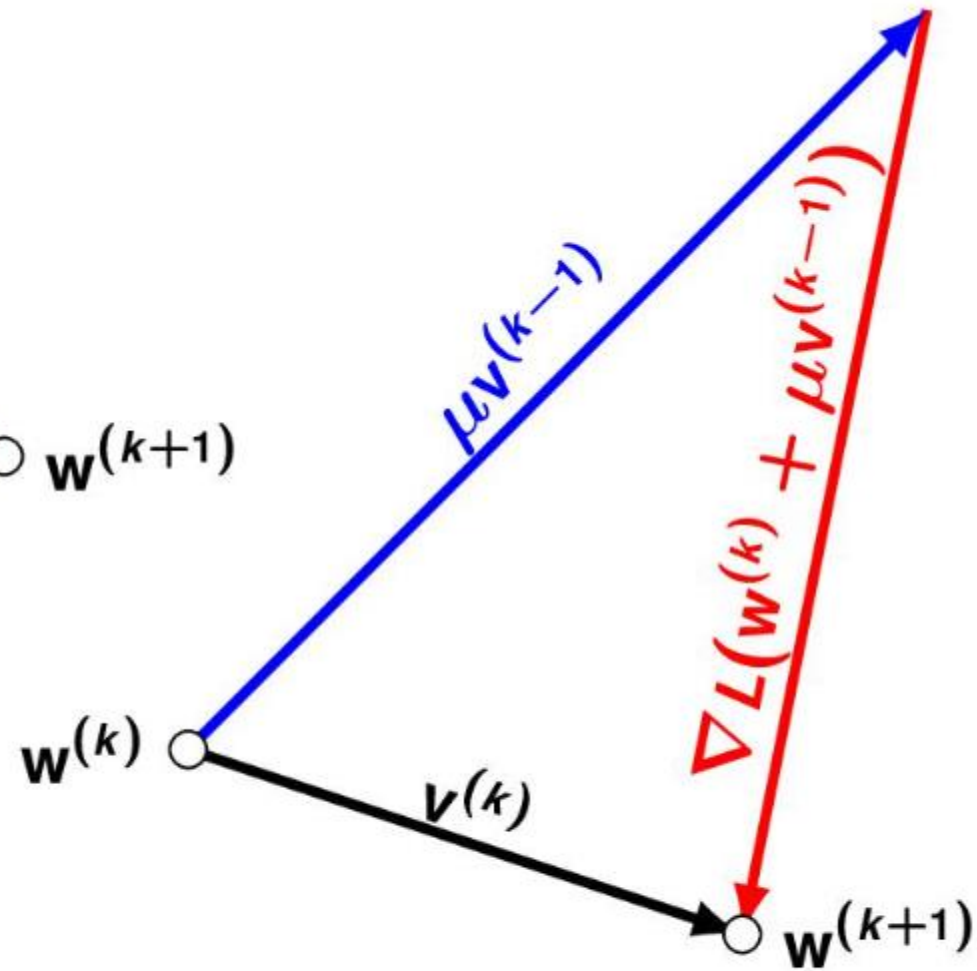
Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\vartheta}$

Единственное отличие от метода инерции в том, что градиент вычисляется после добавления к вектору весов вектора импульса/скорости.

Можно сказать, что алгоритм пытается добавить корректирующее слагаемое к стандартному методу momentum, вычисляя градиент в точке, в которой мы бы оказались, следуя вектору импульса.



Momentum



Nesterov Momentum

AdaGrad (Adaptive Gradient)

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α, аккумулированный градиент $\boldsymbol{G} = 0$.

Повторять для каждой эпохи:

Повторять Для каждого батча:

Найти оценку градиента: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}_i, \boldsymbol{\theta}), \boldsymbol{y}_i)$

Обновить $\boldsymbol{G}$: $\boldsymbol{G} \leftarrow \boldsymbol{G} + \boldsymbol{g} \odot \boldsymbol{g}$

Найти поправку к весам: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\alpha}{\sqrt{\boldsymbol{G}+\delta}} \odot \boldsymbol{g}$

Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

AdaGrad является алгоритмом, предназначенным для адаптивного изменения learning rate в процессе обучения.

Идея состоит в том, чтобы уменьшать α для параметров с большим значением частной производной и увеличивать темп обучения вдоль координаты градиента с малым значением производной.

AdaGrad (Adaptive Gradient)

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α, аккумулированный градиент $\boldsymbol{G} = 0$.

Повторять для каждой эпохи:

Повторять Для каждого батча:

Найти оценку градиента: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}_i, \boldsymbol{\theta}), \boldsymbol{y}_i)$

Обновить $\boldsymbol{G}$: $\boldsymbol{G} \leftarrow \boldsymbol{G} + \boldsymbol{g} \odot \boldsymbol{g}$

Найти поправку к весам: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\alpha}{\sqrt{\boldsymbol{G}+\delta}} \odot \boldsymbol{g}$

Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

AdaGrad является алгоритмом, предназначенным для адаптивного изменения learning rate в процессе обучения.

Идея состоит в том, чтобы уменьшать α для параметров с большим значением частной производной и увеличивать темп обучения вдоль координаты градиента с малым значением производной.

Главный недостаток: если начальные градиенты велики, то темп обучения будет малым на протяжении всего дальнейшего обучения

RMSProp (Root Mean Squared Propagation)

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α, аккумулированный градиент $\boldsymbol{G} = 0$, параметр сглаживания ρ

Повторять для каждой эпохи:

Повторять Для каждого батча:

Найти оценку градиента: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}_i, \boldsymbol{\theta}), \boldsymbol{y}_i)$

Обновить $\boldsymbol{G}$: $\boldsymbol{G} \leftarrow \rho\boldsymbol{G} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$

Найти поправку к весам: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\alpha}{\sqrt{\boldsymbol{G}+\delta}} \odot \boldsymbol{g}$

Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

RMSProp улучшает AdaGrad, используя вместо простого накопления квадрата градиента *экспоненциальное взвешенное скользящее среднее.* Темп обучения уменьшается не так быстро, поскольку RMSProp не использует историю градиентов с очень давних шагов благодаря сглаживанию.

Adam (Adaptive Moment Estimation)

Дано: вектор параметров $\boldsymbol{\theta}$, темп обучения α, переменные $\boldsymbol{G} = 0$, $\boldsymbol{\vartheta} = 0$, шаг по “времени” $t = 0$

Повторять для каждой эпохи:

Повторять Для каждого батча:

Найти оценку градиента: $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}_i, \boldsymbol{\theta}), \boldsymbol{y}_i)$

$t \leftarrow t + 1$

Обновить оценку первого момента: $\boldsymbol{\vartheta} \leftarrow \beta_1 \boldsymbol{\vartheta} + (1 - \beta_1)\boldsymbol{g}$

Обновить оценку второго момента: $\boldsymbol{G} \leftarrow \beta_2 \boldsymbol{G} + (1 - \beta_2)\boldsymbol{g} \odot \boldsymbol{g}$

Скорректировать смещение в первом моменте: $\boldsymbol{\vartheta}' \leftarrow \frac{\boldsymbol{\vartheta}}{1-\beta_1^t}$

Скорректировать смещение во втором моменте: $\boldsymbol{G}' \leftarrow \frac{\boldsymbol{G}}{1-\beta_2^t}$

Найти поправку к весам: $\Delta\boldsymbol{\theta} \leftarrow -\alpha \frac{\boldsymbol{\vartheta}'}{\sqrt{\boldsymbol{G}'+\delta}}$

Обновить веса: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

Adam объединяет идеи использования импульса и адаптивного шага. Как правило из используемых в алгоритме гиперпараметров подбирают только α. Значения остальных обычно фиксируют.

Adam является сильным оптимизационным алгоритмом и зачастую используется как оптимизатор по умолчанию