SGD доработки

Юлия Пономарева
Data Scientist

Проверка связи

Отправьте «✚», если меня видно и слышно

Если у вас нет звука или изображения:

- перезагрузите страницу
- попробуйте зайти заново
- откройте трансляцию в другом браузере (используйте Google Chrome или Microsoft Edge)
- с осторожностью используйте VPN, при подключении через VPN видеопотоки могут тормозить

Цели занятия

1. Познакомиться с оптимизаторами Adagrad, RMSProp, Adam
2. Обсудить шедулеры для изменения скорости обучения
3. Изучить способы борьбы с переобучением в нейросетях
4. Посмотреть на аугментации

План занятия

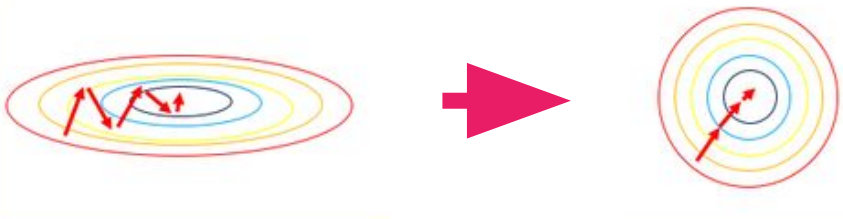
1. Оптимизаторы Adagrad, RMSProp, Adam
2. Шедулеры LinearLR, CyclicLR
3. Регуляризация в нейросетях Dropout, BatchNorm
4. Аугментации данных
5. Итоги занятия

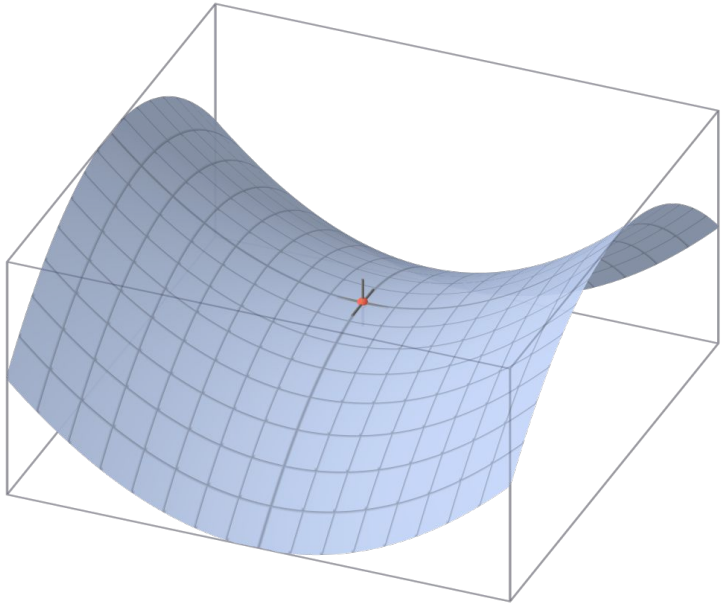
Оптимизаторы

Оптимизаторы

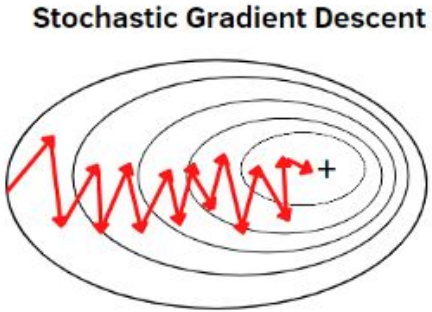
В НС при оптимизации функции потерь существует две основные проблемы:

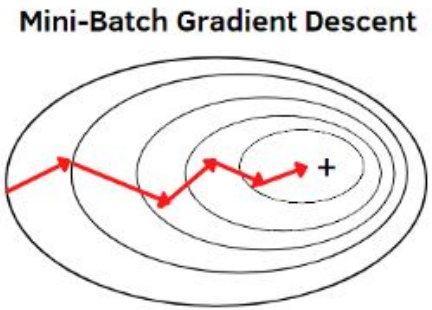
- седловые точки
- разные масштабы данных





Mini Batch GD





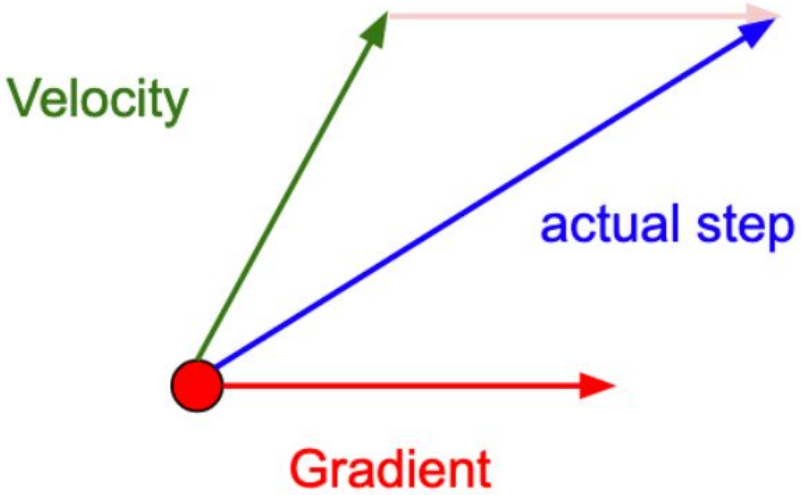
Momentum

Правило обновления весов w с градиентом g, когда импульс == 0:

$$w^k = w^{k-1} - \eta * \nabla Q(w^{k-1}, X)$$

Правило обновления весов w с градиентом g, когда импульс > 0:

$$velocity = momentum * velocity - \eta * \nabla Q(w^{k-1}, X)$$
$$w^k = w^{k-1} + velocity$$



Momentum

Засчет добавления импульса получается сглаживание оптимизации.

t_1	t_2	..	t_n
g_1	g_2	..	g_n

$$\gamma = 0.5$$

$$v_1 = g_1$$
$$v_2 = \gamma * v_1 + g_2 = 0.5 * g_1 + g_2$$
$$v_3 = \gamma * v_2 + g_3 = \gamma(\gamma * v_1 + g_2) + g_3 = \gamma^2 * g_1 + \gamma * g_2 + g_3 = 0.25 * g_1 + 0.5 * g_2 + g_3$$

Adagrad

МФТИ

В Adagrad используются разные скорости обучения в зависимости от итерации:

$$w^k = w^{k-1} - \eta_k \nabla Q(w^{k-1}, X)$$

$$\eta_k = \frac{\eta}{\sqrt{\alpha_k + \epsilon}}$$

где ϵ - маленькое число, чтобы не было деления на ноль.

$$\alpha_k = \sum_{i=0}^{k} \nabla Q(w^{k-1}, X)^2$$

Обновление весов:

$$w^k = w^{k-1} - \eta_k \nabla Q(w^{k-1}, X)$$

Получается, что когда α становится большим числом, то η_k становится меньше, то есть с увеличением итерации - уменьшается скорость обучения, а значит и уменьшается скорость изменения весов.

Но есть одна очень большая проблема - чем больше итераций, тем α больше, скорость обучения меньше, это приведет к тому, что изменение весов может стать совсем незаметным. Но это решаемо с помощью RMSProp.

RMSProp

Этот оптимизатор исправляет проблему с неизменяемыми весами в Adagrad засчет введения ограничения на градиенты весов.

$$\eta_k = \frac{\eta}{\sqrt{W_{avg_k} + \epsilon}}$$

где ϵ - маленькое число, чтобы не было деления на ноль.

$$W_{avg_0} = 0$$

$$W_{avg_k} = \rho * W_{avg_{k-1}} + (1 - \rho)\nabla Q(w^{k-1}, X)^2$$

Обновление весов:

$$w^k = w^{k-1} - \eta_k \nabla Q(w^{k-1}, X)$$

Adam

Здесь соединились два плюса предыдущих оптимизаторов:

1. Импульс (дает сглаживание оптимизации)
2. Постепенное уменьшение скорости обучения

$$V_k = \beta_1 * V_{k-1} + g_k$$

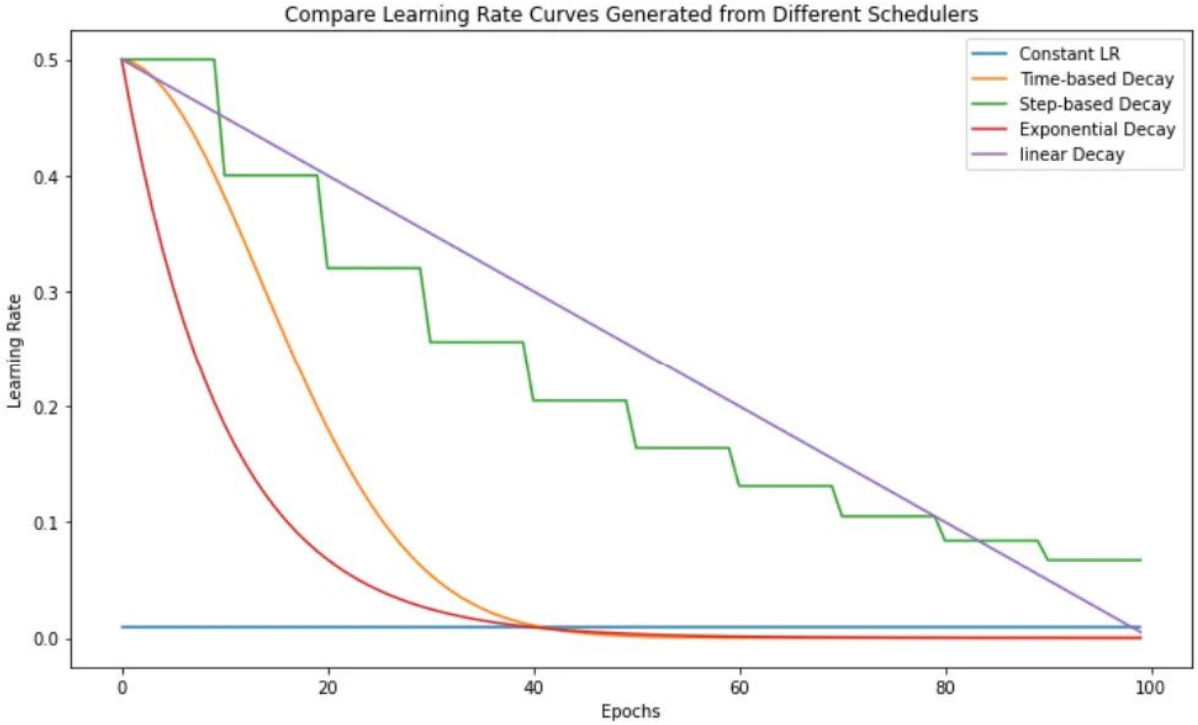
$$W_{avg_k} = \beta_2 * W_{avg_{k-1}} + (1 - \beta_2)\nabla Q(w^{k-1}, X)^2$$

Обновление весов:

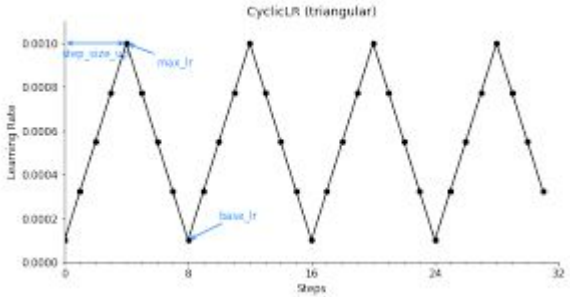
$$w^k = w^{k-1} - \frac{\eta * V_k}{\sqrt{W_{avg_k} + \epsilon}}$$

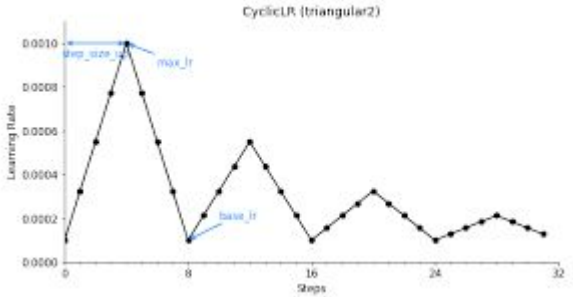
Скорость обучения

LR Schedulers



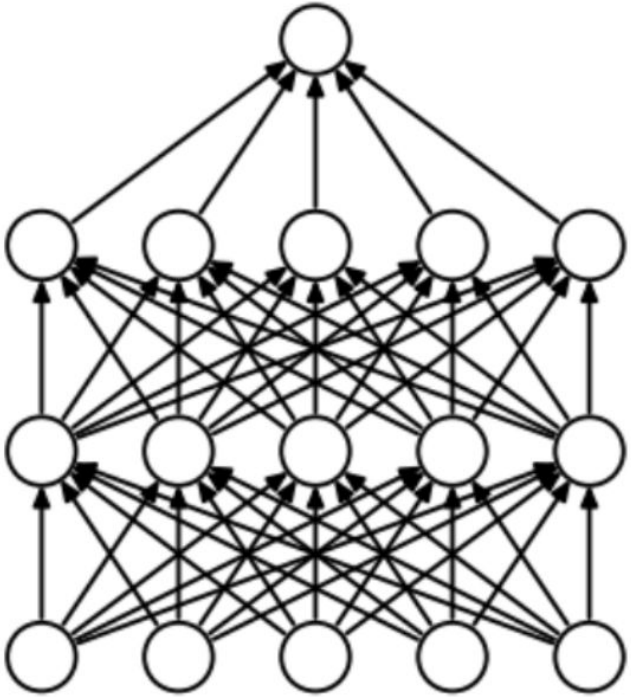
CycleLR


CyclicLR (triangular)
Learning Rate
Steps
max_lr
base_lr

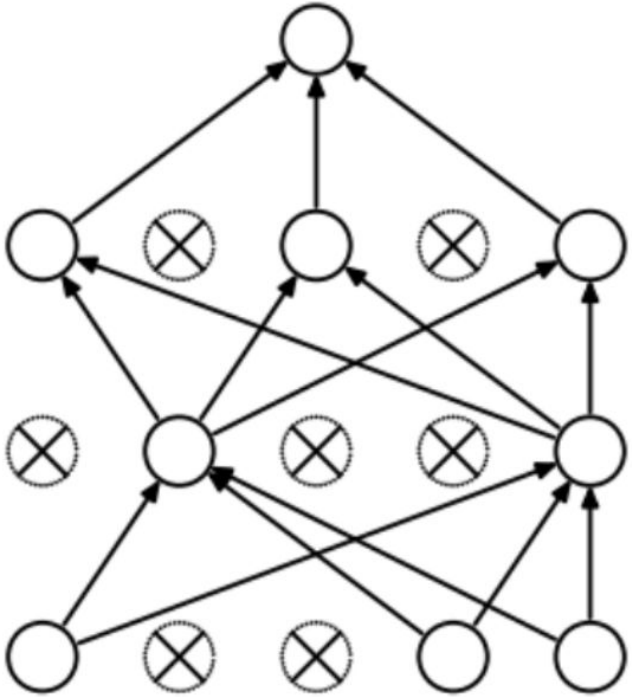

CyclicLR (triangular2)
Learning Rate
Steps
max_lr
base_lr

Регуляризация в NN

Слой Dropout



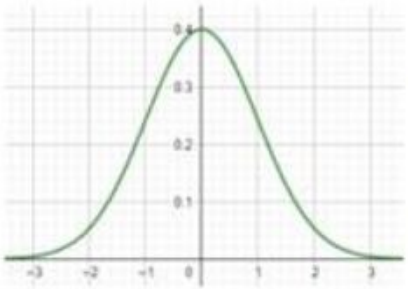
(a) Standard Neural Net



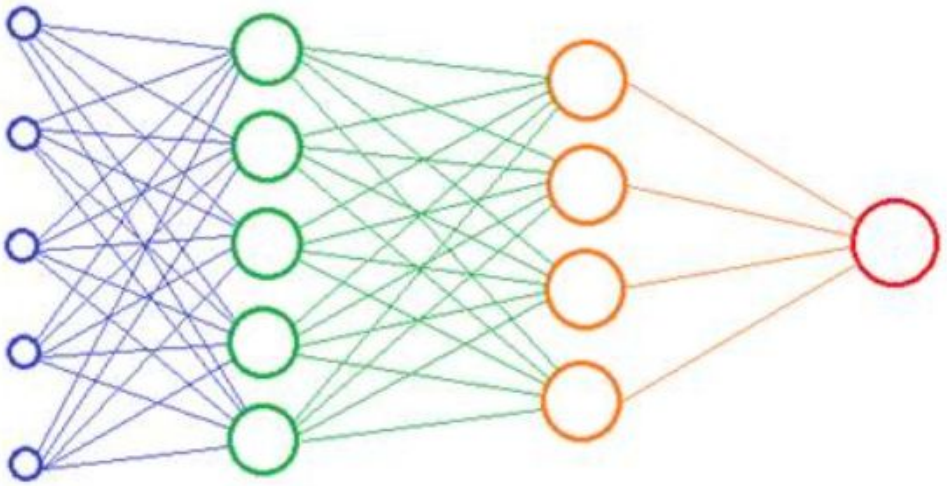
(b) After applying dropout.

BatchNorm

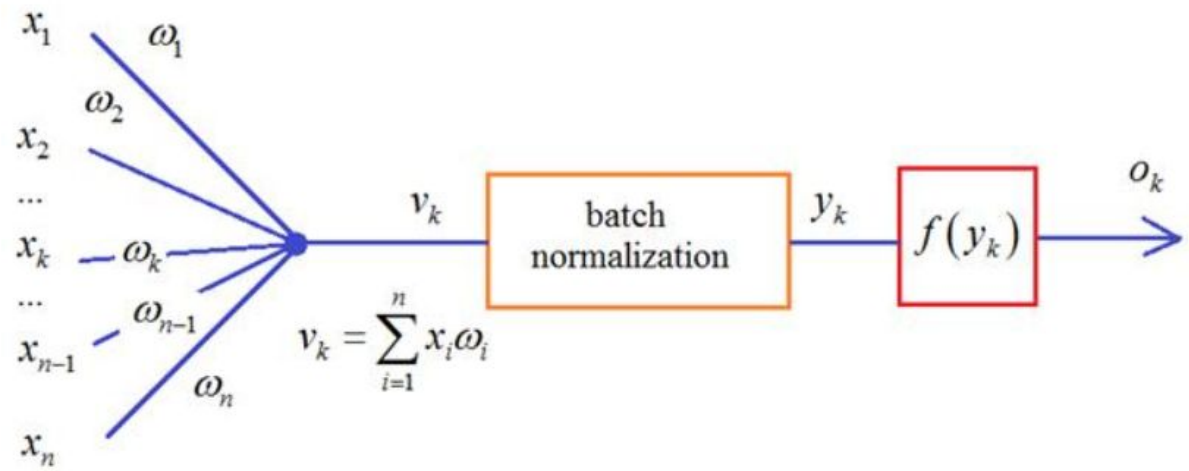
Начальное распределение входов



Распределение на выходах нейронов



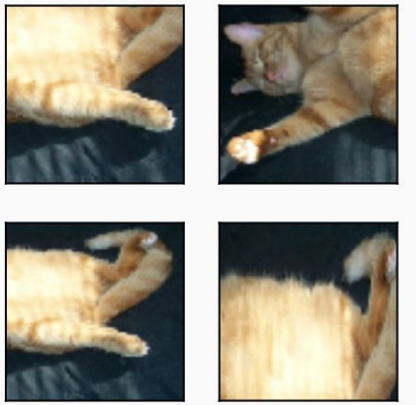
BatchNorm

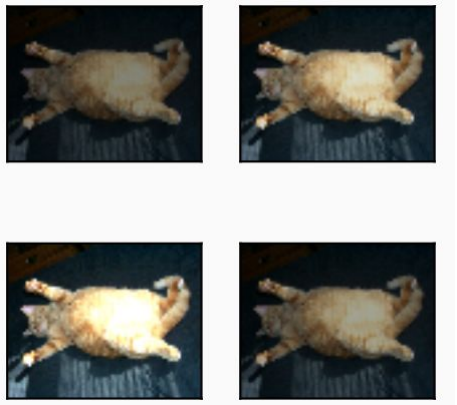


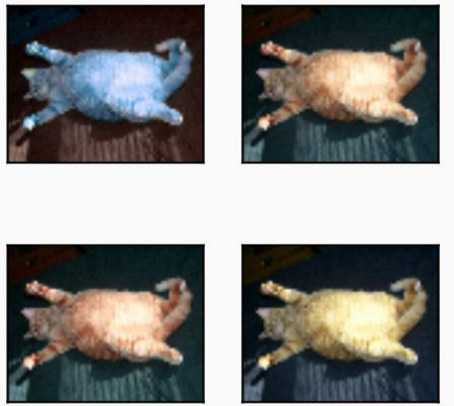
$$z_k^i = \frac{v_k^i - m_v}{\sqrt{\sigma_v^2 + \epsilon}}$$

$y_i^k = \gamma z_i^k + \beta$, где γ и β - это настраиваемые параметры, которые нужны для дополнительного масштабирования и смещения.

Аугментации



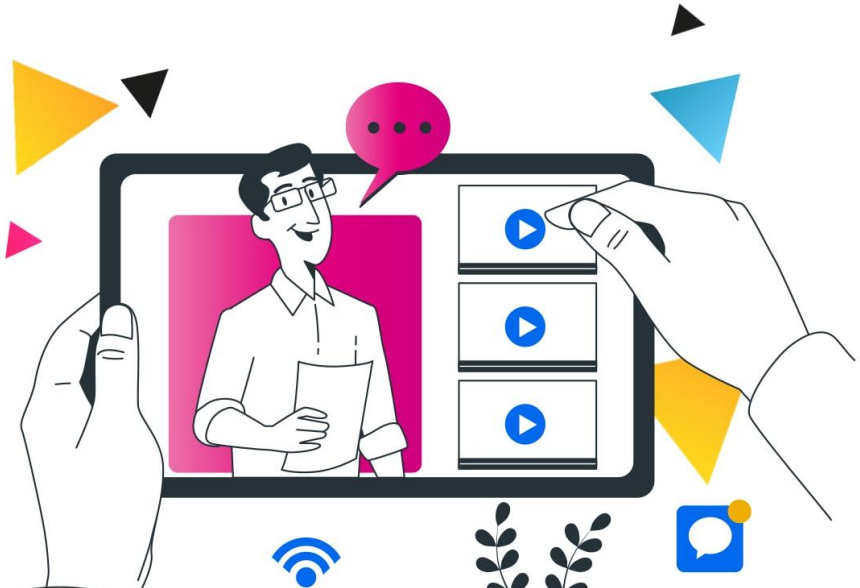




МФТИ

Практика

Ваши вопросы?



Итоги занятия

Итоги занятия

1. Познакомились с оптимизаторами Adagrad, RMSProp, Adam
2. Обсудили шедулеры для изменения скорости обучения
3. Изучили способы борьбы с переобучением в нейросетях
4. Посмотрели на аугментации

Дополнительные материалы

1. Оптимизаторы и шедулеры https://pytorch.org/docs/stable/optim.html
2. Аугментации в модуле transforms https://pytorch.org/vision/stable/transforms.html
3. Порядок слоев BatchNorm с Dropout https://www.reddit.com/r/MachineLearning/comments/67gonq/d_batch_normalization_before_or_after_relu/

МФТИ

Пожалуйста, оставьте свой отзыв о семинаре

До встречи!