Глубинное обучение
Лекция 4: Обучение нейросетей – оптимизация и регуляризация

Лектор: Антон Осокин

ФКН ВШЭ, 2020

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Магия нейросетей

План лекции

- Обучение нейросетей
  - Стохастический градиентный спуск
  - Learning rate, batch size
  - Инерция в градиентном спуске
  - Как отслеживать обучение?
- Регуляризация
  - L2 регуляризация (weight decay), dropout
  - Аугментация данных
  - Регуляризация через плохую оптимизацию
  - Ансамбли
- Оптимизация
  - Batch normalization и другие нормировки
  - Индивидуальные скорости: RMSprop, Adam

Задача приближения функции (обучение с учителем)

- Вход: объекты $x_1, \ldots, x_N \in \mathbb{R}^d$, ответы $y_1, \ldots, y_N \in \mathbb{Y}$
- Семейство функций – нейросети f(x, θ)
  - Параметры θ линейного слоя (Wx + b): W , b (weights, biases)
- Задача – настроить параметры по выборке
- Функция потерь $\ell(f(x, \theta), y)$
- Задача обучения: $\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \ell(f(x, \theta), y)$
- Регуляризованный эмпирический риск

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \theta), y_i) + \mathcal{R}(\theta)$$

Обучение нейросети

- Регуляризованный эмпирический риск

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \theta), y_i) + \mathcal{R}(\theta)$$

- Задача оптимизации – сложная (не выпуклая)
- Обычно функция дифференцируема
- Стохастическая оптимизация первого порядка
  - Stochastic gradient descent (SGD)

$$\theta_{t+1} \leftarrow \theta_t - \gamma \left(\nabla_\theta \left[\ell(f(x_i, \theta_t), y_i) + \mathcal{R}(\theta_t) \right] \right)$$

  - Вычисление градиента по параметрам – back-propagation (метод обратного распространения ошибки)

[Rumelhart&McClelland, 1986]

Основная функция потерь – Log loss (cross entropy, neg. log-likelihood)

- Функция потерь по умолчанию для классификации

$$y_i \in \mathbb{Y} = \{1, \dots, K\}$$

$$\ell(f(x,\theta), y) = -\log\left(\frac{\exp f_y(x,\theta)}{\sum_{s=1}^{K}\exp f_s(x,\theta)}\right)$$

$$\ell(f(x,\theta), y) = -f_y(x,\theta) + \log\left(\sum_{s=1}^{K}\exp f_s(x,\theta)\right)$$

- Оффтоп: наивная реализация очень неустойчива
  - Экспонента большого числа, разность близких чисел
  - Правильная реализация:

$$\ell(f(x,\theta), y) = -f_y(x,\theta) + F + \log\left(\sum_{s=1}^{K}\exp(f_s(x,\theta) - F)\right)$$

$$F := \max_{s=1,\dots,K} f_s(x,\theta)$$

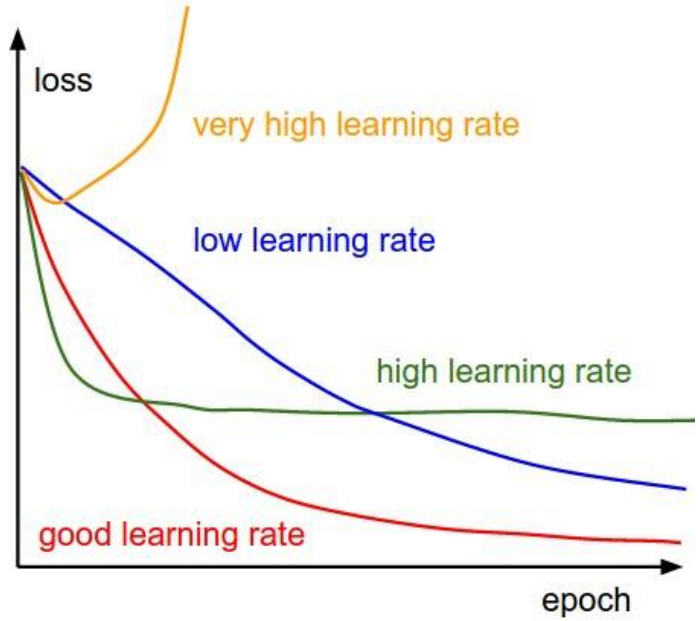
  - PyTorch:
    - `nn.CrossEntropyLoss(scores)`
    - `nn.NLLLoss(nn.LogSoftmax(scores))`

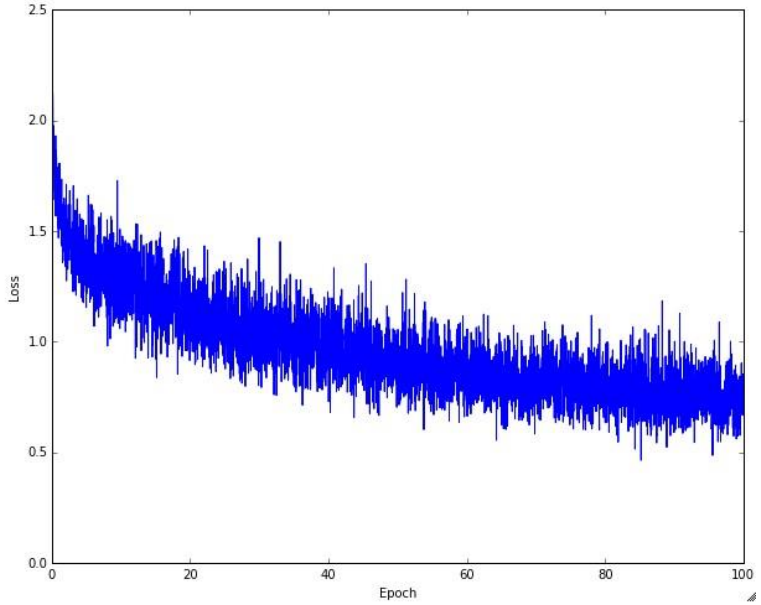
Основные параметры обучения

Stochastic gradient descent (SGD)

$$\theta_{t+1} \leftarrow \theta_t - \gamma\left(\nabla_\theta\left[\ell(f(x_i, \theta_t), y_i) + \mathcal{R}(\theta_t)\right]\right)$$

- γ – learning rate (скорость обучения, LR)
  - Большой или маленький шаг?





Images: cs231n

Основные параметры обучения

Stochastic gradient descent (SGD)

$$\theta_{t+1} \leftarrow \theta_t - \gamma\left(\nabla_\theta\left[\ell(f(x_i, \theta_t), y_i) + \mathcal{R}(\theta_t)\right]\right)$$

- γ – learning rate (скорость обучения, LR)
  - Большой или маленький шаг?
- Теория говорит, что фикс. LR сходится к шару

- На практике:
  - Выбирают макс. LR, такой что метод не расходится
  - Далее LR уменьшают по расписанию или ReduceOnPlateau
  - Цикличные LR [Loshchilov&Hutter, 2017; Smith&Topin 2017]

Размер батча (batch size)

Градиент обычно усредняют по нескольким примерам

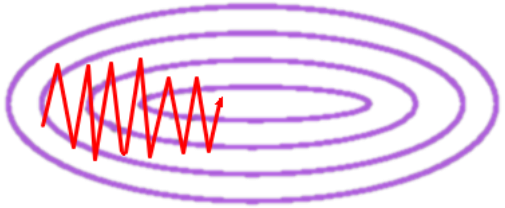
$$\theta_{t+1} \leftarrow \theta_t - \gamma \left(\frac{1}{B} \sum_{b=1}^{B} \nabla_\theta \left[\ell(f(x_{i_b}, \theta_t), y_{i_b}) + \mathcal{R}(\theta_t) \right] \right)$$

- Почему?
  - Градиенты становятся менее шумными
  - Теория оптимизации говорит, что этого делать не надо [Shalev-Shwartz et al., 2010]
  - Аппаратное ускорение от работы в батче (GPU, кластеры, etc)
  - Некоторые слои (batchnorm) непосредственно используют батч
- Рекомендация (для GPU) – макс. батч, который дает линейное ускорение и влезает в память
  - Для ImageNet – это 32, 64, 128
  - При очень больших батчах проблемы со сходимостью
  - Некоторые модели не обучаются с маленькими батчами
  - LR может быть связан с batch size: чем больше батч, тем больше и LR

Инерция в оптимизации (momentum)

- SGD ведёт себя нестабильно, особенно в «каньонах»

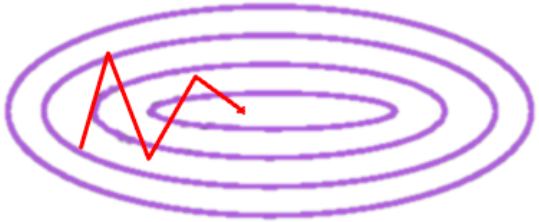
$$\theta_{t+1} \leftarrow \theta_t - \gamma \nabla_\theta \ell_t(\theta_t)$$

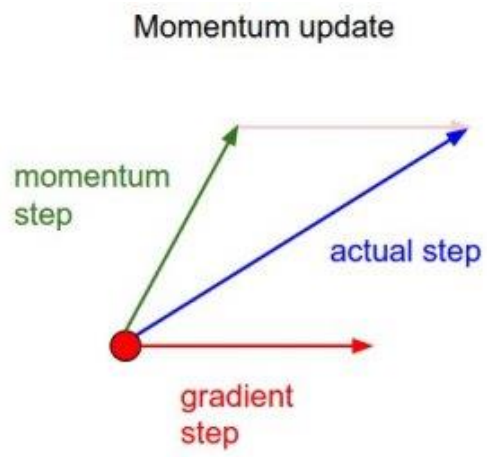


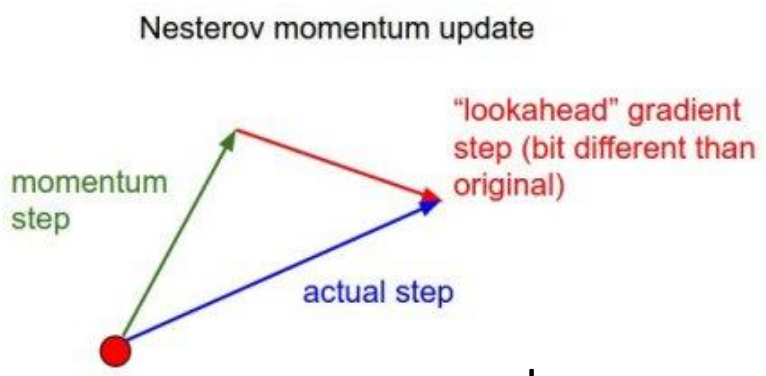
- Выход – использовать инерцию (momentum, heavy-ball)

$$m_{t+1} \leftarrow \rho m_t + \nabla_\theta \ell_t(\theta_t)$$

$$\theta_{t+1} \leftarrow \theta_t - \gamma m_t$$







Images credit: cs231n

Как отслеживать обучение? Графики!

[Tensorboard, visdom, etc]

1. Значение функции потерь в зависимости от итерации
   - Проверка идёт ли оптимизация
2. Ошибка классификации

Обучающая выборка
не покажет переобучение!

3. Потери(лосс) / ошибка на валидационной выборке
   - ВАЖНО: Использовать валидацию, а не тест
   - Можно выбрать итерацию с наилучшей моделью
   - Тогда по тестовой выборке можно оценить метод
   - Валидация – не часто и не редко (10% времени)
   - Можно использовать метрики из задачи (BLEU)

Регуляризация

Граница размыта:

- регуляризация
- оптимизация
- архитектура

L2 регуляризация (weight decay для SGD)

- L2-норма на параметры

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \theta), y_i) + \frac{\lambda}{2} \|\theta\|_2^2$$

- Препятствует росту параметров

- Обычно встроена в оптимизатор (в SGD – это weight decay)

$$\theta_{t+1} \leftarrow \theta_t(1 - \lambda) - \gamma \nabla_\theta \ell_t(\theta_t)$$

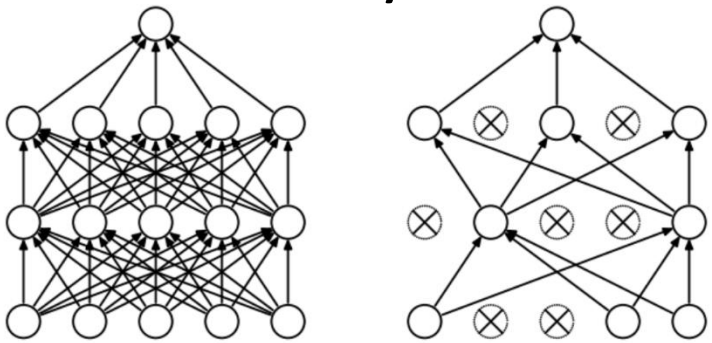
- Для других оптимизаторов weight decay != L2

- Обычные значения 10^{-3}, 10^{-4} (по умолчанию – 0)
- L1 – примерно такой же эффект

Dropout

[Srivastava et al., 2014]

- Случайное обнуление активаций



(a) Standard Neural Net (b) After applying dropout.

- Препятствует «коадаптации» узлов, связано с ансамблями сетей
- Полносвязные слои/эмбеддинги, реже свёртки
- ВАЖНО! в режиме теста обычно выключается
  - Активации обнуляются с вероятностью p
  - Активации домножаются на 1/(1-p) на обучении (p – вероятность 0)
- Неявная регуляризация!
  - Мешает оптимизации, но помогает обучению

Аугментация данных

[Yaeger et al., 1996]

- Из существующих данных искусственно сделать ещё
- Для изображений – a must (важная часть AlexNet)
  - Left/right flip
  - Random crop/rescaling
  - Случайные модификации цвета
  - Синтез синтетических данных
  - mixup [Zhang et al., 2017, arXiv:1710.09412]
  - RandAugment [Cubuk et al., 2019, arXiv:1909.13719]
- Для звука – случайный шум фона, тональность [Hannun et al., 2014]
- Для текстов – сложнее (дискретные данные)
  - Word dropout [Iyyer et al., 2015]
  - Замена на синонимы по словарю [Zhang&LeCun, 2016]

Batch normalization

[Ioffe&Szegedy, 2015]

- Internal Covariate Shift = изменение распределения активаций, вызванное изменением параметров

- Whitening – поворот на i.i.d. с 0-mean и 1-std
  - Оценка матрицы ковариаций Σ, $W = \text{chol}(\Sigma^{-1})$, $x := W x$
  - Упрощение: вычитание среднего и деление на std
  - Улучшает обучение (всегда в предобработке данных)

- Batchnorm – упрощенный whitening на слоях
  - Mean, std оцениваются по батчу
  - Mean, std – дифференцируемы

Batch normalization

[Ioffe&Szegedy, 2015]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

На тесте работает по другому!

Используются предпосчитанные оценки средних и дисперсий.

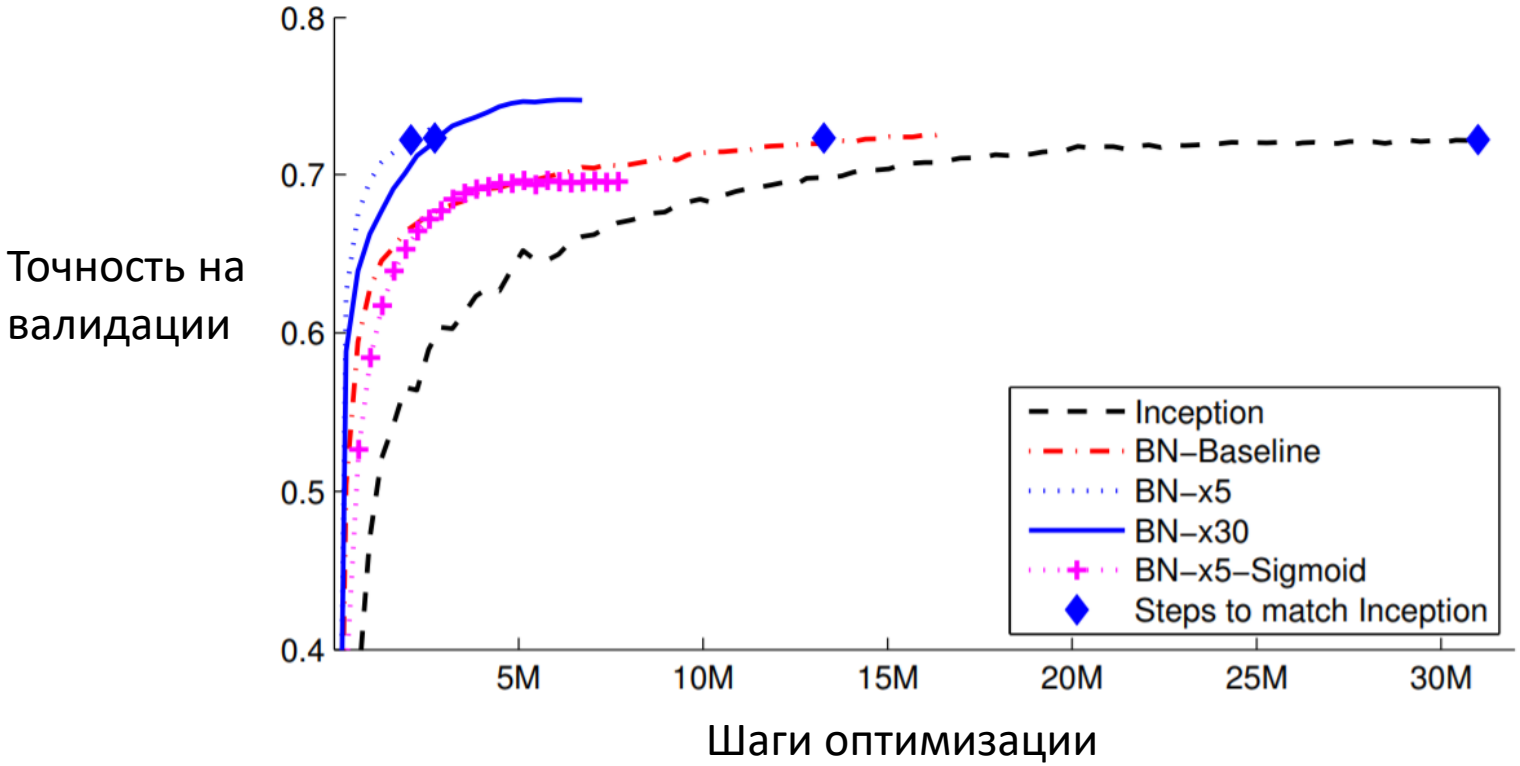
Оценки средних и дисперсий вычисляются скользящим средним во время обучения.

- Объекты перестают быть i.i.d.!

Следствия batch normalization

[Ioffe&Szegedy, 2015]

- Можно увеличить длины шага (LR)
- Можно убрать dropout
- Уменьшить L2-регуляризацию
- Позволяет обучать очень глубокие модели (ResNet)



Плохая оптимизация – регуляризация!

- Ранний останов (early stopping)
  - Смотрим на ошибку на валидации
  - Останавливаемся/уменьшаем шаг, когда перестает убывать
- Использовать SGD!

Плохая оптимизация – регуляризация!

- Ранний останов (early stopping)
  - Смотрим на ошибку на валидации
  - Останавливаемся/уменьшаем шаг, когда перестает убывать
- Использовать SGD!

[Bottou&Bousquet, 2008]

  - Approximation–Estimation–Optimization Tradeoff

$$\min_{\mathcal{F},\rho,n} \mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} \quad \text{subject to} \begin{cases} n & \leq & n_{\max} \\ T(\mathcal{F},\rho,n) & \leq & T_{\max} \end{cases}$$

    - Approximation – ошибка лучшей NN в семействе
    - Estimation – лучший выбор NN по данным
    - Optimization – лучшая NN за вычислительный бюджет
  - Large-scale: ограничение по вычислениям
  - SGD лучше по обобщающей способности в Large-scale
  - Sharp vs. flat minima
- Практика: CNN – лучше SGD, сложные модели – Adam

Ансамбли нейросетей

- Deep ensembles [Lakshminarayanan et al., 2017]
  - Обучить несколько сетей
  - Усреднить вероятности
  - Требуется много ресурсов как на обучение, так и на тест

- Stochastic Weight Averaging (SWA) [Izmailov et al., 2018]
  - Усреднить веса сети с нескольких точек во одного обучения
  - Улучшает обобщающую способность

Оптимизация

Граница размыта:

- регуляризация
- оптимизация
- архитектура

Подбор LR для каждого параметра

- RMSprop [Tieleman&Hinton, slide 29 of lecture 6]

$$\theta_{t+1,i} \leftarrow \theta_{t,i} - \frac{\gamma}{\sqrt{v_{t,i}^2 + \varepsilon}} \nabla_{\theta_i} \ell_t(\theta_t),$$

$$v_{t,i}^2 \leftarrow \beta v_{t-1,i}^2 + (1-\beta)(\nabla_{\theta_i} \ell_t(\theta_t))^2$$

  - Параметры: β = 0.99, $\varepsilon = 10^{-8}$, $\gamma = 10^{-2}$

- Adam [Kingma&Ba, 2015]

$$\theta_{t+1,i} \leftarrow \theta_{t,i} - \frac{\gamma}{\sqrt{\hat{v}_{t,i}^2 + \varepsilon}} \hat{m}_{t,i},$$

$$v_{t,i}^2 \leftarrow \beta_2 v_{t-1,i}^2 + (1-\beta_2)(\nabla_{\theta_i} \ell_t(\theta_t))^2, \qquad \hat{v}_{t,i}^2 \leftarrow \frac{v_{t,i}^2}{1-\beta_2^t},$$

$$m_{t,i} \leftarrow \beta_1 m_{t-1,i} + (1-\beta_1)\nabla_{\theta_i} \ell_t(\theta_t) \qquad \hat{m}_{t,i} \leftarrow \frac{m_{t,i}}{1-\beta_1^t}$$

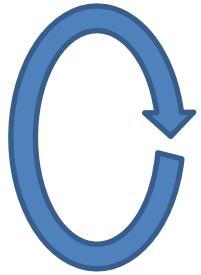
  - Параметры: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, $\gamma = 10^{-3}$ Karpathy constant γ = 3e-4
  - Увеличение ε если нестабильно

Еще есть разные версии:
AMSgrad, AdamW, RAdam, etc.

Выводы: оптимизация и регуляризация

Процесс:

- Архитектура и функция потерь
- Оптимизация потерь на обучении
- Регуляризация, чтобы улучшить обобщение



HINTs:

- Метод оптимизации часто является частью модели!
- Осторожно с гиперпараметрами!
- Диагностика:
  - Отношение нормы апдейта к норме весов
  - Гистограммы активаций
  - Визуализация всего!