

Глубинное обучение

Лекция 4: Обучение нейросетей – оптимизация и регуляризация

Лектор: Антон Осокин

ФКН ВШЭ, 2018



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

Чёрная магия нейросетей

ФКН ВШЭ, 2018



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

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
- Оптимизация
 - Batch normalization и другие нормировки
 - Индивидуальные скорости: RMSprop, Adam

Задача приближения функции (обучение с учителем)

- Вход: объекты $x_1, \dots, x_N \in \mathbb{R}^d$, ответы $y_1, \dots, y_N \in \mathbb{Y}$
- Семейство функций – нейросети $f(x, \theta)$
 - 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)

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

- 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

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

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

- Оффтоп: наивная реализация очень неустойчива

- Экспонента большого числа, разность близких чисел

- Правильная реализация:

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

- PyTorch:

- `nn.CrossEntropyLoss(scores)`

- `nn.NLLLoss(nn.LogSoftmax(scores))`

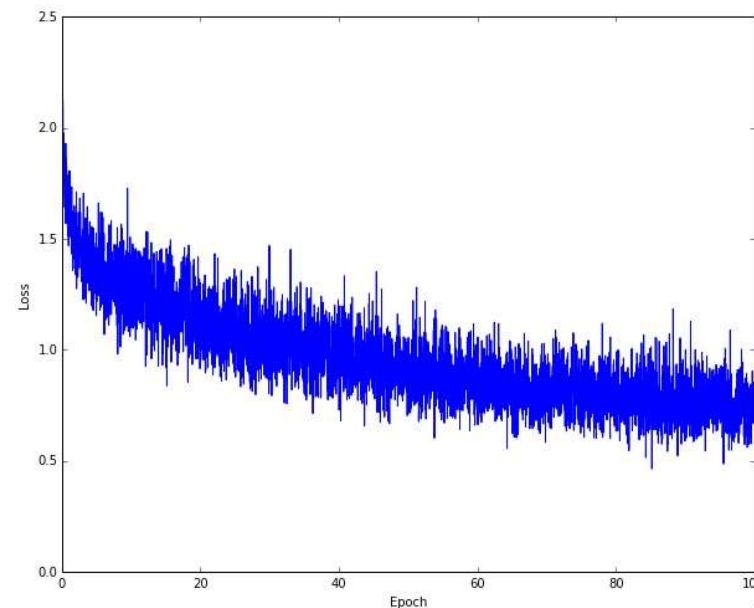
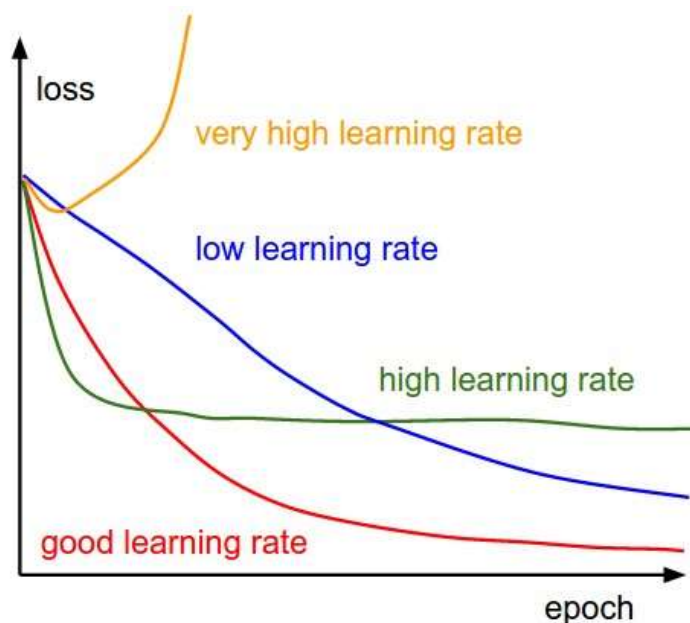
$$F := \max_{s=1, \dots, K} f_s(x, \theta)$$

Основные параметры обучения

Stochastic gradient descent (SGD)

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

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



Images:
[cs231n](#)

Основные параметры обучения

Stochastic gradient descent (SGD)

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

- γ – learning rate (скорость обучения, LR)
 - Большой или маленький шаг?
- Теория говорит, что фикс. LR сходится к шару
- На практике:
 - Выбирают макс. LR, что метод не расходится
 - Далее LR уменьшают по расписанию или ReduceOnPlateau

Размер батча (batch size)

Градиент обычно усредняют по нескольким примерам

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

- Почему?

- Градиенты становятся менее шумными
- Теория оптимизации говорит, что этого делать не надо

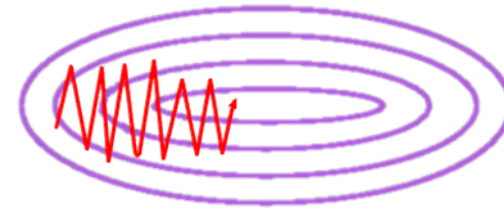
[Shalev-Shwartz et al., 2010]

- 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

Инерция в оптимизации

- SGD ведёт себя нестабильно, особенно в «каньонах»

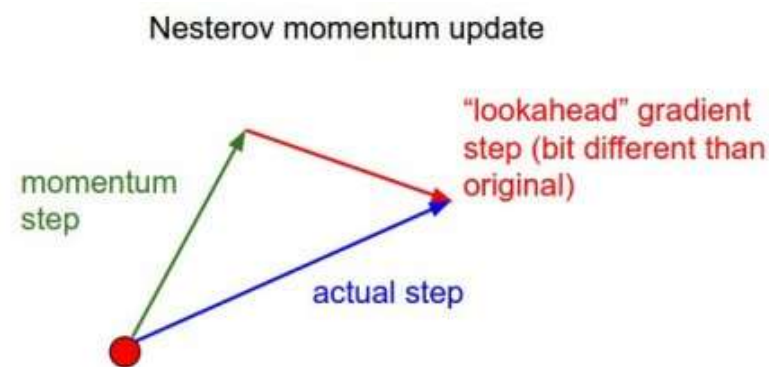
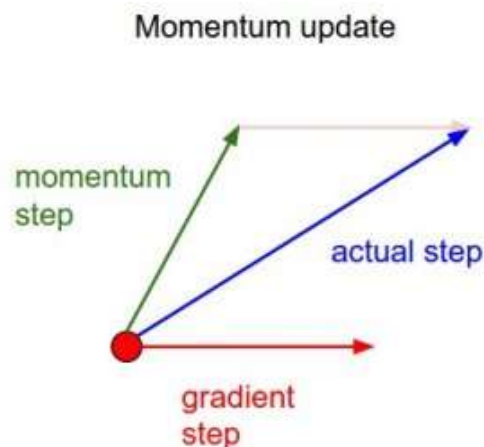
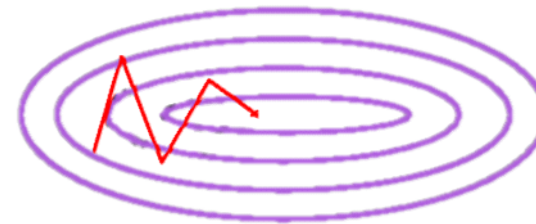
$$\theta_{t+1} \leftarrow \theta_t - \gamma \nabla_{\theta} \ell_t(\theta_t)$$



- 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](https://cs231n.github.io/)

Как отслеживать обучение? Графики!

[Tensorboard(X), 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)

- 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 (weight decay)

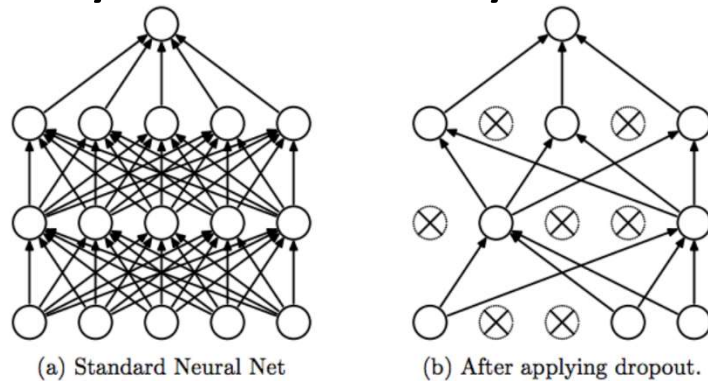
$$\theta_{t+1} \leftarrow \theta_t(1 - \lambda) - \gamma \nabla_{\theta} \ell_t(\theta_t)$$

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

Dropout

[Srivastava et al., 2014]

- Случайное обнуление активаций



- Препятствует «коадаптации» узлов, связано с ансамблями сетей
- Полносвязные слои/эмбединги, реже свёртки
- ВАЖНО! в режиме теста обычно выключается
 - Веса домножаются на $1/(1-p)$ для масштаба
- 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. Сдвиги – редко
 - Случайные модификации цвета
 - Синтез синтетических данных
- Для звука – случайный шум фона, тональность
[Hannun et al., 2014]
- Для текстов – реже (дискретные данные)
 - Замена на синонимы по словарю [Zhang& LeCun, 2016]

Плохая оптимизация – регуляризация!

- Ранний останов (early stopping)
 - Смотрим на ошибку на валидации
 - Останавливаемся/уменьшаем шаг, когда убывает
- Использовать SGD! [Bottou&Bousquet, 2008]
 - Approximation–Estimation–Optimization Tradeoff
$$\min_{\mathcal{F}, \rho, n} \mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} \quad \text{subject to} \quad \begin{cases} n \leq n_{\text{max}} \\ T(\mathcal{F}, \rho, n) \leq T_{\text{max}} \end{cases}$$
 - Approximation – regret лучшего NN в семействе
 - Estimation – лучший выбор NN по данным
 - Optimization – лучший NN за вычислительный бюджет
 - Large-scale: ограничение по вычислениям
 - SGD лучше по обобщающей способности в Large-scale
 - Sharp vs. flat minima
- CNN – лучше SGD, сложные модели – Adam

Оптимизация

Граница размыта:

- регуляризация
- оптимизация
- архитектура

Batch normalization

[Ioffe&Szegedy, 2015]

- Internal Covariate Shift = изменение распределения активаций, вызванное изменением параметров
- Whitening – перенормировка на 0-mean и 1-std
 - Вычитание среднего и деление на std
 - Улучшает обучение (всегда в предобработке данных)
- Batchnorm – whitening на промежуточных слоях
 - Mean, std оцениваются по батчу
 - Mean, std – дифференцируемы

Batch normalization

[Ioffe&Szegedy, 2015]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

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

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

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

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

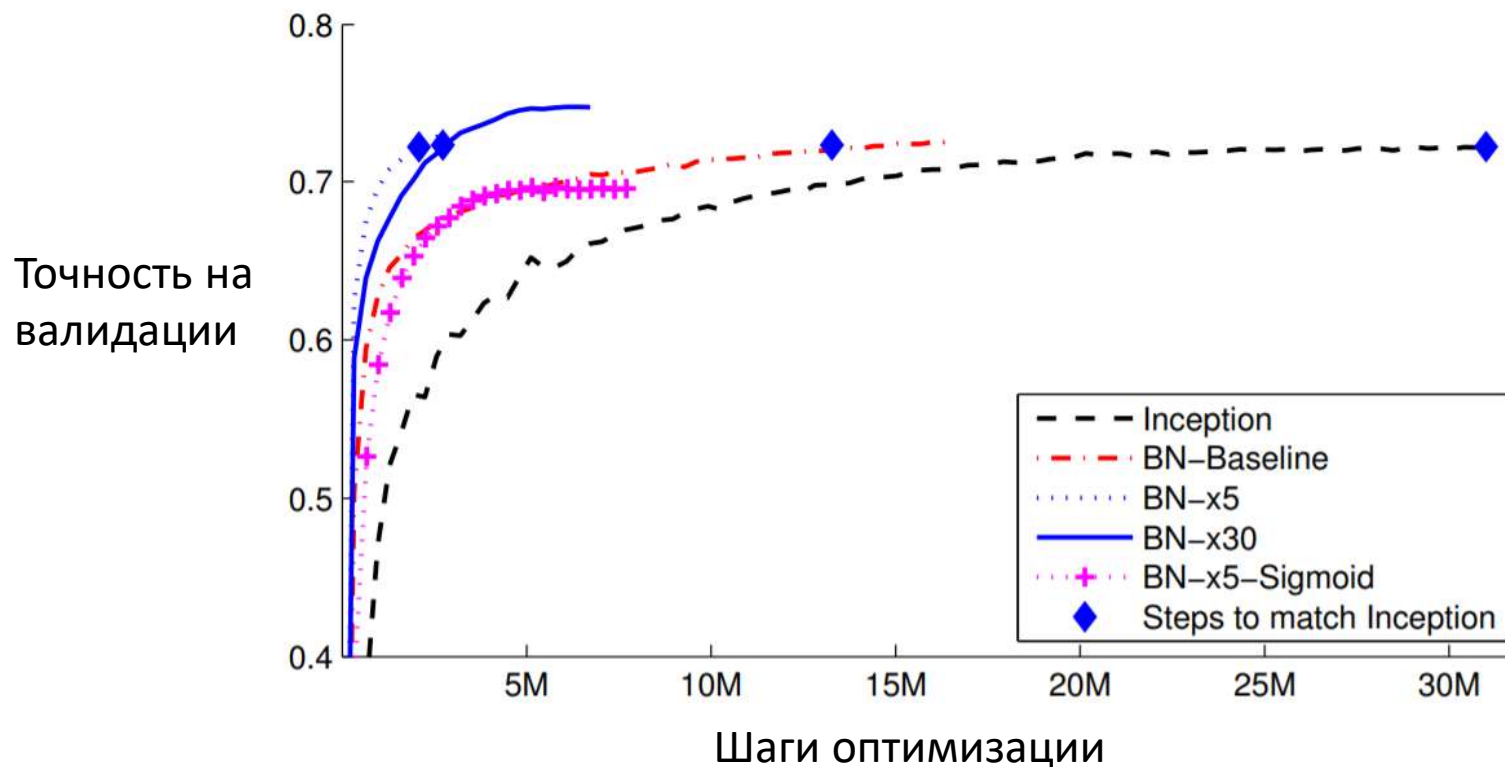
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

- 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)



Подбор LR для каждого параметра

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

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

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

– Параметры: $\beta = 0.99$, $\epsilon = 10^{-8}$, $\gamma = 10^{-2}$

- Adam [Kingma&Ba, 2015]

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

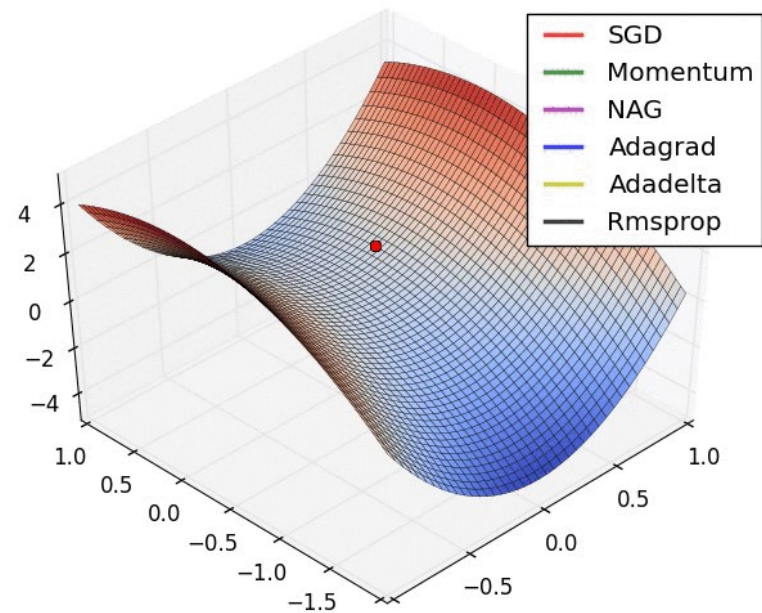
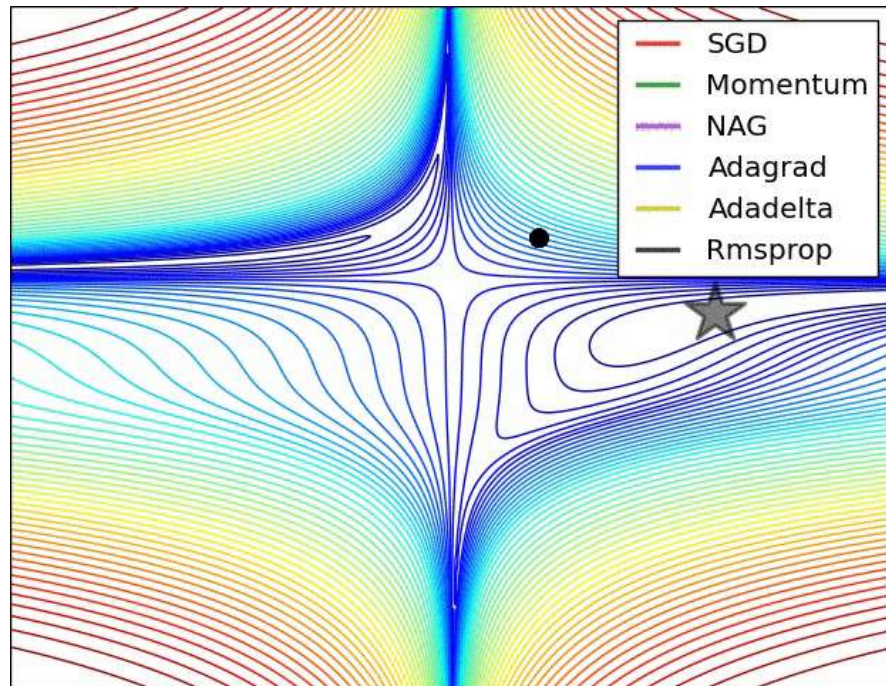
$$v_{t,i}^2 \leftarrow \beta_2 v_{t-1,i}^2 + (1 - \beta_2)(\nabla_{\theta_i} \ell_t(\theta_t))^2, \quad \hat{v}_{t,i}^2 \leftarrow \frac{v_{t,i}^2}{1 - \beta_2^t},$$

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

– Параметры: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\gamma = 10^{-3}$

[Karpathy constant](#) $\gamma = 3e-4$

Разные методы оптимизации

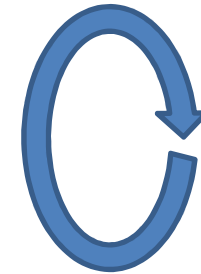


Images credit: [Alec Radford](#)

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 можно и нельзя!