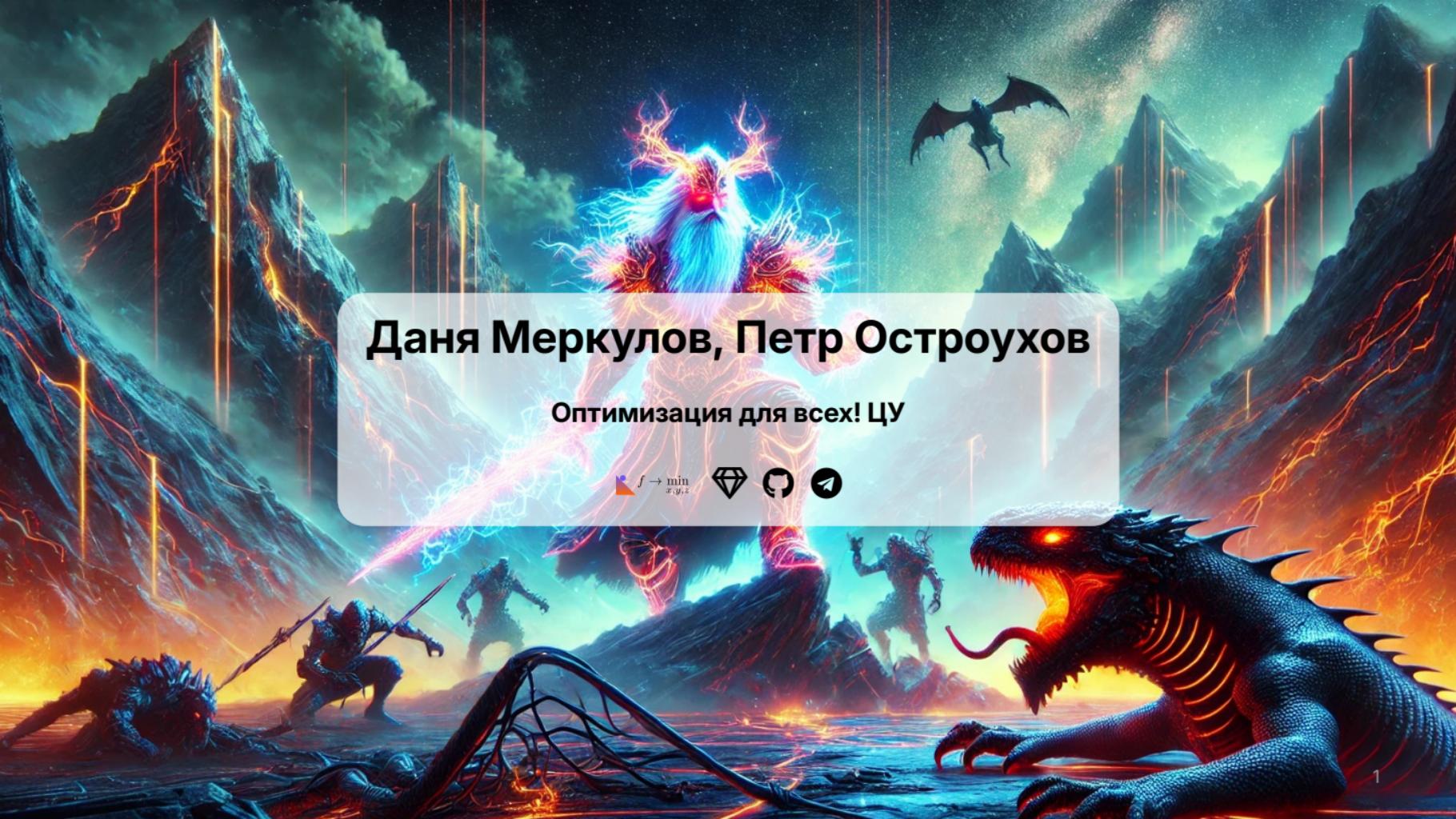


# Стохастический градиентный спуск. Адаптивные методы. Оптимизация нейронных сетей

МЕТОДЫ ВЫПУКЛОЙ ОПТИМИЗАЦИИ

НЕДЕЛЯ 14

Даня Меркулов  
Пётр Остроухов



# Даня Меркулов, Петр Остроухов

Оптимизация для всех! ЦУ



# Задача с конечной суммой

# Задача с конечной суммой

Рассмотрим задачу минимизации среднего значения функции на конечной выборке:

$$\min_{x \in \mathbb{R}^p} f(x) = \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

# Задача с конечной суммой

Рассмотрим задачу минимизации среднего значения функции на конечной выборке:

$$\min_{x \in \mathbb{R}^p} f(x) = \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

Шаг градиентного спуска для этой задачи:

$$x_{k+1} = x_k - \frac{\alpha_k}{n} \sum_{i=1}^n \nabla f_i(x) \quad (\text{GD})$$

# Задача с конечной суммой

Рассмотрим задачу минимизации среднего значения функции на конечной выборке:

$$\min_{x \in \mathbb{R}^p} f(x) = \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

Шаг градиентного спуска для этой задачи:

$$x_{k+1} = x_k - \frac{\alpha_k}{n} \sum_{i=1}^n \nabla f_i(x) \quad (\text{GD})$$

- Сходимость с постоянным  $\alpha$  или линейным поиском.

# Задача с конечной суммой

Рассмотрим задачу минимизации среднего значения функции на конечной выборке:

$$\min_{x \in \mathbb{R}^p} f(x) = \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

Шаг градиентного спуска для этой задачи:

$$x_{k+1} = x_k - \frac{\alpha_k}{n} \sum_{i=1}^n \nabla f_i(x) \tag{GD}$$

- Сходимость с постоянным  $\alpha$  или линейным поиском.
- Стоимость итерации линейна по  $n$ . Для ImageNet  $n \approx 1.4 \cdot 10^7$ , для WikiText  $n \approx 10^8$ . Для FineWeb  $n \approx 15 \cdot 10^{12}$  токенов.

# Задача с конечной суммой

Рассмотрим задачу минимизации среднего значения функции на конечной выборке:

$$\min_{x \in \mathbb{R}^p} f(x) = \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

Шаг градиентного спуска для этой задачи:

$$x_{k+1} = x_k - \frac{\alpha_k}{n} \sum_{i=1}^n \nabla f_i(x) \tag{GD}$$

- Сходимость с постоянным  $\alpha$  или линейным поиском.
- Стоимость итерации линейна по  $n$ . Для ImageNet  $n \approx 1.4 \cdot 10^7$ , для WikiText  $n \approx 10^8$ . Для FineWeb  $n \approx 15 \cdot 10^{12}$  токенов.

# Задача с конечной суммой

Рассмотрим задачу минимизации среднего значения функции на конечной выборке:

$$\min_{x \in \mathbb{R}^p} f(x) = \min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

Шаг градиентного спуска для этой задачи:

$$x_{k+1} = x_k - \frac{\alpha_k}{n} \sum_{i=1}^n \nabla f_i(x) \quad (\text{GD})$$

- Сходимость с постоянным  $\alpha$  или линейным поиском.
- Стоимость итерации линейна по  $n$ . Для ImageNet  $n \approx 1.4 \cdot 10^7$ , для WikiText  $n \approx 10^8$ . Для FineWeb  $n \approx 15 \cdot 10^{12}$  токенов.

Перейдем от вычисления полного градиента к его несмещенной оценке. На каждой итерации будем выбирать индекс  $i_k$  случайно и равномерно:

$$x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k) \quad (\text{SGD})$$

При  $p(i_k = i) = \frac{1}{n}$  стохастический градиент является несмещенной оценкой полного градиента:

$$\mathbb{E}[\nabla f_{i_k}(x)] = \sum_{i=1}^n p(i_k = i) \nabla f_i(x) = \sum_{i=1}^n \frac{1}{n} \nabla f_i(x) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) = \nabla f(x)$$

Это означает, что математическое ожидание стохастического градиента совпадает с истинным градиентом  $f(x)$ .

# Результаты для градиентного спуска

Стохастическая итерация в  $n$  раз дешевле, но сколько шагов потребуется для достижения заданной точности?

# Результаты для градиентного спуска

Стохастическая итерация в  $n$  раз дешевле, но сколько шагов потребуется для достижения заданной точности?

Если  $\nabla f$  липшицев, справедливы оценки:

Условие	GD	SGD
PL	$\mathcal{O}(\log(1/\varepsilon))$	$\mathcal{O}(1/\varepsilon)$
Выпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$
Невыпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$

# Результаты для градиентного спуска

Стохастическая итерация в  $n$  раз дешевле, но сколько шагов потребуется для достижения заданной точности?

Если  $\nabla f$  липшицев, справедливы оценки:

Условие	GD	SGD
PL	$\mathcal{O}(\log(1/\varepsilon))$	$\mathcal{O}(1/\varepsilon)$
Выпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$
Невыпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$

- SGD имеет низкую стоимость итерации, но низкую скорость сходимости.

# Результаты для градиентного спуска

Стохастическая итерация в  $n$  раз дешевле, но сколько шагов потребуется для достижения заданной точности?

Если  $\nabla f$  липшицев, справедливы оценки:

Условие	GD	SGD
PL	$\mathcal{O}(\log(1/\varepsilon))$	$\mathcal{O}(1/\varepsilon)$
Выпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$
Невыпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$

- SGD имеет низкую стоимость итерации, но низкую скорость сходимости.
  - Сублинейная скорость даже в сильно выпуклом случае.

# Результаты для градиентного спуска

Стохастическая итерация в  $n$  раз дешевле, но сколько шагов потребуется для достижения заданной точности?

Если  $\nabla f$  липшицев, справедливы оценки:

Условие	GD	SGD
PL	$\mathcal{O}(\log(1/\varepsilon))$	$\mathcal{O}(1/\varepsilon)$
Выпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$
Невыпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$

- SGD имеет низкую стоимость итерации, но низкую скорость сходимости.
  - Сублинейная скорость даже в сильно выпуклом случае.
  - Оценки скорости не могут быть улучшены при стандартных предположениях.

# Результаты для градиентного спуска

Стохастическая итерация в  $n$  раз дешевле, но сколько шагов потребуется для достижения заданной точности?

Если  $\nabla f$  липшицев, справедливы оценки:

Условие	GD	SGD
PL	$\mathcal{O}(\log(1/\varepsilon))$	$\mathcal{O}(1/\varepsilon)$
Выпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$
Невыпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$

- SGD имеет низкую стоимость итерации, но низкую скорость сходимости.
  - Сублинейная скорость даже в сильно выпуклом случае.
  - Оценки скорости не могут быть улучшены при стандартных предположениях.
  - Оракул возвращает несмешенную аппроксимацию градиента с ограниченной дисперсией.

# Результаты для градиентного спуска

Стохастическая итерация в  $n$  раз дешевле, но сколько шагов потребуется для достижения заданной точности?

Если  $\nabla f$  липшицев, справедливы оценки:

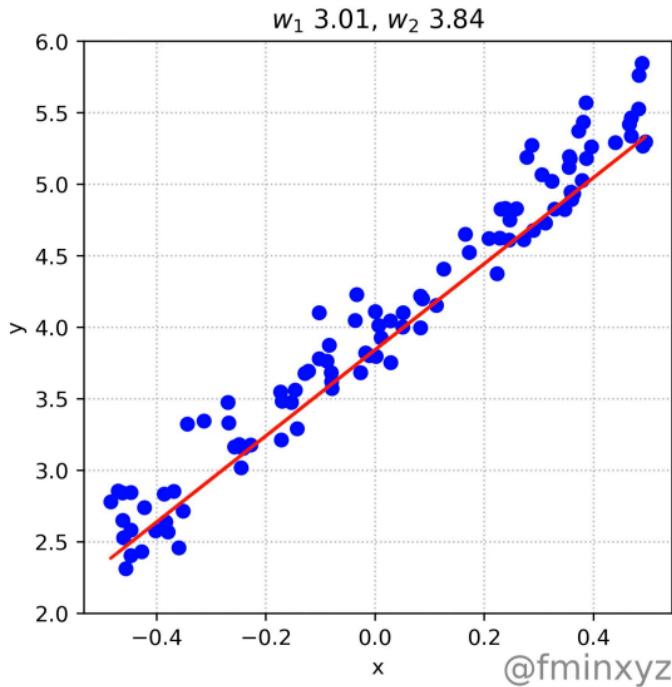
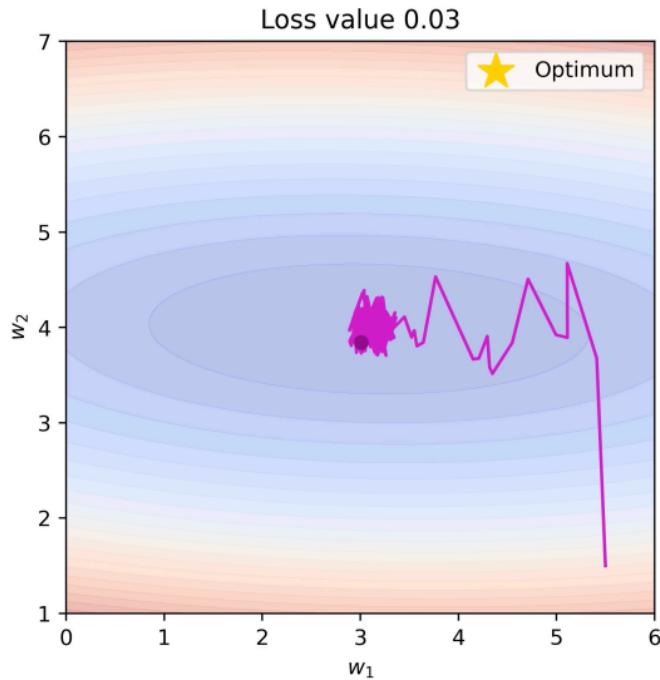
Условие	GD	SGD
PL	$\mathcal{O}(\log(1/\varepsilon))$	$\mathcal{O}(1/\varepsilon)$
Выпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$
Невыпуклый	$\mathcal{O}(1/\varepsilon)$	$\mathcal{O}(1/\varepsilon^2)$

- SGD имеет низкую стоимость итерации, но низкую скорость сходимости.
  - Сублинейная скорость даже в сильно выпуклом случае.
  - Оценки скорости не могут быть улучшены при стандартных предположениях.
  - Оракул возвращает несмешенную аппроксимацию градиента с ограниченной дисперсией.
- Методы с ускорением и квазиньютоновские методы не улучшают асимптотическую скорость в стохастическом случае, влияя лишь на константы (узкое место — дисперсия, а не число обусловленности).

# **Стохастический градиентный спуск (SGD)**

# Типичное поведение

Stochastic Gradient Descent. Batch = 2



@fminxyz

# Гладкий PL-случай с постоянным шагом

- i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

# Гладкий выпуклый случай

## Вспомогательные обозначения

Для (возможно) неконстантной последовательности шагов  $(\alpha_t)_{t \geq 0}$  определим *взвешенное среднее*

$$\bar{x}_k \stackrel{\text{def}}{=} \frac{1}{\sum_{t=0}^{k-1} \alpha_t} \sum_{t=0}^{k-1} \alpha_t x_t, \quad k \geq 1.$$

Везде ниже  $f^* \equiv \min_x f(x)$  и  $x^* \in \arg \min_x f(x)$ .

# Гладкий выпуклый случай с постоянным шагом

- i** Пусть  $f$  — выпуклая функция (не обязательно гладкая), а дисперсия стохастического градиента ограничена  $\mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \leq \sigma^2 \forall k$ . Если SGD использует постоянный шаг  $\alpha_t \equiv \alpha > 0$ , то для любого  $k \geq 1$

$$\mathbb{E}[f(\bar{x}_k) - f^*] \leq \frac{\|x_0 - x^*\|^2}{2\alpha k} + \frac{\alpha \sigma^2}{2}$$

где  $\bar{x}_k = \frac{1}{k} \sum_{t=0}^{k-1} x_t$ .

При выборе постоянного  $\alpha = \frac{\|x_0 - x^*\|}{\sigma\sqrt{k}}$  (зависящего от  $k$ ) имеем

$$\mathbb{E}[f(\bar{x}_k) - f^*] \leq \frac{\|x_0 - x^*\|\sigma}{\sqrt{k}} = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right).$$

# Гладкий выпуклый случай с убывающим шагом

$$\alpha_k = \frac{\alpha_0}{\sqrt{k+1}}, \quad 0 < \alpha_0 \leq \frac{1}{4L}$$

**i** При тех же предположениях, но с убывающим шагом  $\alpha_k = \frac{\alpha_0}{\sqrt{k+1}}$

$$\boxed{\mathbb{E}[f(\bar{x}_k) - f^*] \leq \frac{5\|x_0 - x^*\|^2}{4\alpha_0\sqrt{k}} + 5\alpha_0\sigma^2 \frac{\log(k+1)}{\sqrt{k}}} = \mathcal{O}\left(\frac{\log k}{\sqrt{k}}\right).$$



# Мини-батч SGD

# Мини-батч SGD

Детерминированный метод использует все  $n$  градиентов:

$$\nabla f(x_k) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k).$$

# Мини-батч SGD

Детерминированный метод использует все  $n$  градиентов:

$$\nabla f(x_k) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k).$$

Стохастический метод аппроксимирует это, используя только один элемент:

$$\nabla f_{ik}(x_k) \approx \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k).$$

# Мини-батч SGD

Детерминированный метод использует все  $n$  градиентов:

$$\nabla f(x_k) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k).$$

Стохастический метод аппроксимирует это, используя только один элемент:

$$\nabla f_{ik}(x_k) \approx \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k).$$

Распространённый вариант — использовать выборку элементов  $B_k$  («**мини-батч**»):

$$\frac{1}{|B_k|} \sum_{i \in B_k} \nabla f_i(x_k) \approx \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k),$$

особенно полезно для векторизации и распараллеливания.

# Мини-батч SGD

Детерминированный метод использует все  $n$  градиентов:

$$\nabla f(x_k) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k).$$

Стохастический метод аппроксимирует это, используя только один элемент:

$$\nabla f_{ik}(x_k) \approx \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k).$$

Распространённый вариант — использовать выборку элементов  $B_k$  («**мини-батч**»):

$$\frac{1}{|B_k|} \sum_{i \in B_k} \nabla f_i(x_k) \approx \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k),$$

особенно полезно для векторизации и распараллеливания.

Например, имея 16 ядер, можно взять  $|B_k| = 16$  и вычислить 16 градиентов параллельно.

# Мини-батч как градиентный спуск с ошибкой

Метод SGD с батчем  $B_k$  («мини-батч») использует итерации:

$$x_{k+1} = x_k - \alpha_k \left( \frac{1}{|B_k|} \sum_{i \in B_k} \nabla f_i(x_k) \right).$$

# Мини-батч как градиентный спуск с ошибкой

Метод SGD с батчем  $B_k$  («мини-батч») использует итерации:

$$x_{k+1} = x_k - \alpha_k \left( \frac{1}{|B_k|} \sum_{i \in B_k} \nabla f_i(x_k) \right).$$

Рассмотрим это как «градиентный метод с ошибкой»:

$$x_{k+1} = x_k - \alpha_k (\nabla f(x_k) + e_k),$$

где  $e_k$  — разница между аппроксимированным и истинным градиентом.

# Мини-батч как градиентный спуск с ошибкой

Метод SGD с батчем  $B_k$  («мини-батч») использует итерации:

$$x_{k+1} = x_k - \alpha_k \left( \frac{1}{|B_k|} \sum_{i \in B_k} \nabla f_i(x_k) \right).$$

Рассмотрим это как «градиентный метод с ошибкой»:

$$x_{k+1} = x_k - \alpha_k (\nabla f(x_k) + e_k),$$

где  $e_k$  — разница между аппроксимированным и истинным градиентом.

Если выбрать  $\alpha_k = \frac{1}{L}$ , то, согласно лемме о спуске:

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|^2 + \frac{1}{2L} \|e_k\|^2,$$

для любой ошибки  $e_k$ .

# Влияние ошибки на скорость сходимости

Оценка прогресса при  $\alpha_k = \frac{1}{L}$  и ошибке градиента  $e_k$ :

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|^2 + \frac{1}{2L} \|e_k\|^2.$$

## Идея SGD и батчей



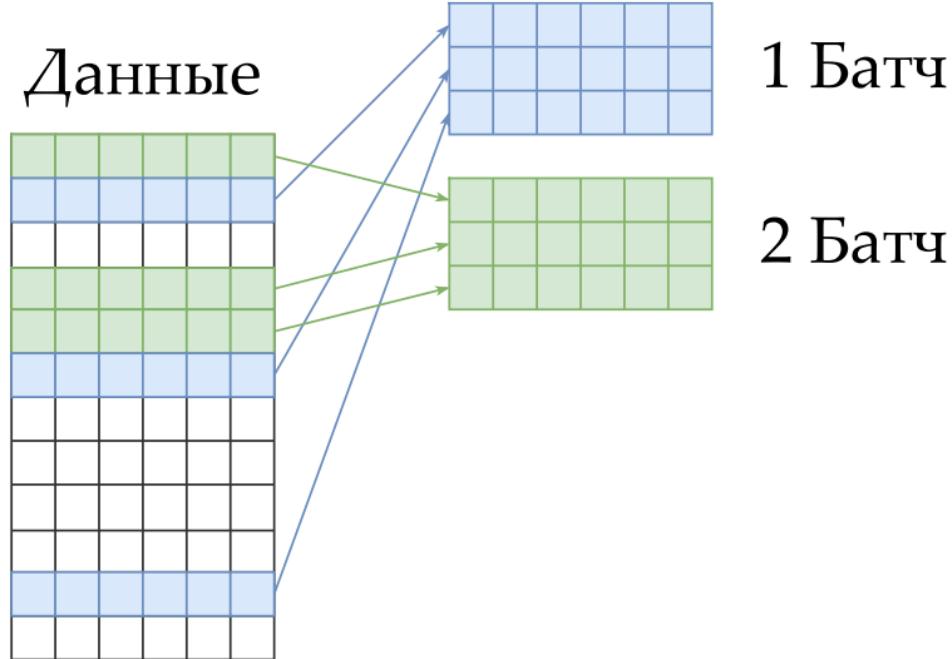
## Данные



# Идея SGD и батчей



# Идея SGD и батчей



# Идея SGD и батчей



# Идея SGD и батчей



# Основная проблема SGD

$$f(x) = \frac{\mu}{2} \|x\|_2^2 + \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-y_i \langle a_i, x \rangle)) \rightarrow \min_{x \in \mathbb{R}^n}$$

Strongly convex binary logistic regression. m=200, n=10, mu=1.



# Основные результаты сходимости SGD

- Пусть  $f$  —  $L$ -гладкая  $\mu$ -сильно выпуклая функция, а дисперсия стохастического градиента ограничена ( $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ ). Тогда траектория SGD с постоянным шагом  $\alpha < \frac{1}{2\mu}$  будет гарантировать:

$$\mathbb{E}[f(x_{k+1}) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

# Основные результаты сходимости SGD

- i** Пусть  $f$  —  $L$ -гладкая  $\mu$ -сильно выпуклая функция, а дисперсия стохастического градиента ограничена ( $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ ). Тогда траектория SGD с постоянным шагом  $\alpha < \frac{1}{2\mu}$  будет гарантировать:

$$\mathbb{E}[f(x_{k+1}) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

- i** Пусть  $f$  —  $L$ -гладкая  $\mu$ -сильно выпуклая функция, а дисперсия стохастического градиента ограничена ( $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ ). Тогда SGD с убывающим шагом  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$  будет сходиться сублинейно:

$$\mathbb{E}[f(x_{k+1}) - f^*] \leq \frac{L\sigma^2}{2\mu^2(k+1)}$$

# Summary

- SGD с постоянным шагом не сходится к оптимуму даже в PL (или сильно выпуклом) случае.

# Summary

- SGD с постоянным шагом не сходится к оптимуму даже в PL (или сильно выпуклом) случае.
- SGD сходится сублинейно со скоростью  $\mathcal{O}\left(\frac{1}{k}\right)$  для PL-случая.

# Summary

- SGD с постоянным шагом не сходится к оптимуму даже в PL (или сильно выпуклом) случае.
- SGD сходится сублинейно со скоростью  $\mathcal{O}\left(\frac{1}{k}\right)$  для PL-случая.
- Ускорение Нестерова/Поляка не улучшает скорость сходимости.

# Summary

- SGD с постоянным шагом не сходится к оптимуму даже в PL (или сильно выпуклом) случае.
- SGD сходится сублинейно со скоростью  $\mathcal{O}\left(\frac{1}{k}\right)$  для PL-случая.
- Ускорение Нестерова/Поляка не улучшает скорость сходимости.
- Двухфазный метод Ньютона достигает  $\mathcal{O}\left(\frac{1}{k}\right)$  без сильной выпуклости.

# Адаптивность или масштабирование



# Adagrad (Duchi, Hazan, and Singer 2010/Streeter and MacMahan 2010)

Популярный адаптивный метод. Обозначим  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$ . Правило обновления для  $j = 1, \dots, p$ :

$$v_j^{(k)} = v_j^{k-1} + (g_j^{(k)})^2$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}$$

# Adagrad (Duchi, Hazan, and Singer 2010/Streeter and MacMahan 2010)

Популярный адаптивный метод. Обозначим  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$ . Правило обновления для  $j = 1, \dots, p$ :

$$v_j^{(k)} = v_j^{k-1} + (g_j^{(k)})^2$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}$$

**Заметки:**

- AdaGrad не требует настройки шага обучения:  $\alpha > 0$  — фиксированная константа, и шаг обучения автоматически уменьшается в ходе итераций.

# Adagrad (Duchi, Hazan, and Singer 2010/Streeter and MacMahan 2010)

Популярный адаптивный метод. Обозначим  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$ . Правило обновления для  $j = 1, \dots, p$ :

$$v_j^{(k)} = v_j^{k-1} + (g_j^{(k)})^2$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}$$

**Заметки:**

- AdaGrad не требует настройки шага обучения:  $\alpha > 0$  — фиксированная константа, и шаг обучения автоматически уменьшается в ходе итераций.
- Шаг обучения для редких информативных признаков убывает медленно.

# Adagrad (Duchi, Hazan, and Singer 2010/Streeter and MacMahan 2010)

Популярный адаптивный метод. Обозначим  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$ . Правило обновления для  $j = 1, \dots, p$ :

$$v_j^{(k)} = v_j^{k-1} + (g_j^{(k)})^2$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}$$

## Заметки:

- AdaGrad не требует настройки шага обучения:  $\alpha > 0$  — фиксированная константа, и шаг обучения автоматически уменьшается в ходе итераций.
- Шаг обучения для редких информативных признаков убывает медленно.
- Может существенно превосходить SGD на разреженных задачах.

# Adagrad (Duchi, Hazan, and Singer 2010/Streeter and MacMahan 2010)

Популярный адаптивный метод. Обозначим  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$ . Правило обновления для  $j = 1, \dots, p$ :

$$v_j^{(k)} = v_j^{k-1} + (g_j^{(k)})^2$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}$$

## Заметки:

- AdaGrad не требует настройки шага обучения:  $\alpha > 0$  — фиксированная константа, и шаг обучения автоматически уменьшается в ходе итераций.
- Шаг обучения для редких информативных признаков убывает медленно.
- Может существенно превосходить SGD на разреженных задачах.
- Основной недостаток — монотонное накопление квадратов градиентов в знаменателе. AdaDelta, Adam, AMSGrad и др. улучшают это, популярны в обучении глубоких нейронных сетей.

# Adagrad (Duchi, Hazan, and Singer 2010/Streeter and MacMahan 2010)

Популярный адаптивный метод. Обозначим  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$ . Правило обновления для  $j = 1, \dots, p$ :

$$v_j^{(k)} = v_j^{k-1} + (g_j^{(k)})^2$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}$$

## Заметки:

- AdaGrad не требует настройки шага обучения:  $\alpha > 0$  — фиксированная константа, и шаг обучения автоматически уменьшается в ходе итераций.
- Шаг обучения для редких информативных признаков убывает медленно.
- Может существенно превосходить SGD на разреженных задачах.
- Основной недостаток — монотонное накопление квадратов градиентов в знаменателе. AdaDelta, Adam, AMSGrad и др. улучшают это, популярны в обучении глубоких нейронных сетей.
- Константа  $\epsilon$  обычно устанавливается в  $10^{-6}$  для предотвращения деления на ноль.

# RMSProp (Tieleman and Hinton, 2012)

Модификация AdaGrad, устраняющая проблему агрессивного монотонного убывания шага. Использует экспоненциальное скользящее среднее квадратов градиентов для настройки шага по каждой координате переменной. Пусть  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$  и правило обновления для  $j = 1, \dots, p$ :

$$v_j^{(k)} = \gamma v_j^{(k-1)} + (1 - \gamma)(g_j^{(k)})^2$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}$$

# RMSProp (Tieleman and Hinton, 2012)

Модификация AdaGrad, устраняющая проблему агрессивного монотонного убывания шага. Использует экспоненциальное скользящее среднее квадратов градиентов для настройки шага по каждой координате переменной. Пусть  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$  и правило обновления для  $j = 1, \dots, p$ :

$$\begin{aligned}v_j^{(k)} &= \gamma v_j^{(k-1)} + (1 - \gamma)(g_j^{(k)})^2 \\x_j^{(k)} &= x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}\end{aligned}$$

## Заметки:

- RMSProp нормирует шаг обучения на корень из скользящего среднего квадратов градиентов.

# RMSProp (Tieleman and Hinton, 2012)

Модификация AdaGrad, устраняющая проблему агрессивного монотонного убывания шага. Использует экспоненциальное скользящее среднее квадратов градиентов для настройки шага по каждой координате переменной. Пусть  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$  и правило обновления для  $j = 1, \dots, p$ :

$$\begin{aligned}v_j^{(k)} &= \gamma v_j^{(k-1)} + (1 - \gamma)(g_j^{(k)})^2 \\x_j^{(k)} &= x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}\end{aligned}$$

## Заметки:

- RMSProp нормирует шаг обучения на корень из скользящего среднего квадратов градиентов.
- Обеспечивает более тонкую настройку шагов обучения, чем AdaGrad, что делает его подходящим для неизотропных задач.

# RMSProp (Tieleman and Hinton, 2012)

Модификация AdaGrad, устраняющая проблему агрессивного монотонного убывания шага. Использует экспоненциальное скользящее среднее квадратов градиентов для настройки шага по каждой координате переменной. Пусть  $g^{(k)} = \nabla f_{i_k}(x^{(k-1)})$  и правило обновления для  $j = 1, \dots, p$ :

$$\begin{aligned}v_j^{(k)} &= \gamma v_j^{(k-1)} + (1 - \gamma)(g_j^{(k)})^2 \\x_j^{(k)} &= x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{v_j^{(k)} + \epsilon}}\end{aligned}$$

## Заметки:

- RMSProp нормирует шаг обучения на корень из скользящего среднего квадратов градиентов.
- Обеспечивает более тонкую настройку шагов обучения, чем AdaGrad, что делает его подходящим для неизотропных задач.
- Широко используется при обучении нейронных сетей, особенно рекуррентных.

# Adam (Kingma and Ba, 2014)<sup>12</sup>



Объединяет элементы из AdaGrad и RMSProp. Использует экспоненциальное скользящее среднее как градиентов, так и их квадратов.

EMA:

$$m_j^{(k)} = \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)}$$

$$v_j^{(k)} = \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2$$

Коррекция смещения:

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}$$

$$\hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

Обновление:

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

# Adam (Kingma and Ba, 2014) <sup>12</sup>



Объединяет элементы из AdaGrad и RMSProp. Использует экспоненциальное скользящее среднее как градиентов, так и их квадратов.

EMA:

$$\begin{aligned}m_j^{(k)} &= \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)} \\v_j^{(k)} &= \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2\end{aligned}$$

Коррекция смещения:  $\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}$

$$\hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

Обновление:  $x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$

**Заметки:**

- Компенсирует смещение к нулю на начальных итерациях, наблюдаемое в других методах (например, RMSProp), что делает оценки более точными.

# Adam (Kingma and Ba, 2014) <sup>12</sup>



Объединяет элементы из AdaGrad и RMSProp. Использует экспоненциальное скользящее среднее как градиентов, так и их квадратов.

EMA:

$$\begin{aligned}m_j^{(k)} &= \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)} \\v_j^{(k)} &= \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2\end{aligned}$$

Коррекция смещения:

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}$$

$$\hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

Обновление:

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

**Заметки:**

- Компенсирует смещение к нулю на начальных итерациях, наблюдаемое в других методах (например, RMSProp), что делает оценки более точными.
- Одна из самых цитируемых научных работ в мире.

# Adam (Kingma and Ba, 2014) <sup>12</sup>



Объединяет элементы из AdaGrad и RMSProp. Использует экспоненциальное скользящее среднее как градиентов, так и их квадратов.

EMA:

$$\begin{aligned}m_j^{(k)} &= \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)} \\v_j^{(k)} &= \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2\end{aligned}$$

Коррекция смещения:

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}$$

$$\hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

Обновление:

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

**Заметки:**

- Компенсирует смещение к нулю на начальных итерациях, наблюдаемое в других методах (например, RMSProp), что делает оценки более точными.
- Одна из самых цитируемых научных работ в мире.
- В 2018-2019 годах вышли статьи, указывающие на ошибку в оригинальной статье

# Adam (Kingma and Ba, 2014) 12



Объединяет элементы из AdaGrad и RMSProp. Использует экспоненциальное скользящее среднее как градиентов, так и их квадратов.

EMA:

$$\begin{aligned}m_j^{(k)} &= \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)} \\v_j^{(k)} &= \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2\end{aligned}$$

Коррекция смещения:

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}$$

$$\hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

Обновление:

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

**Заметки:**

- Компенсирует смещение к нулю на начальных итерациях, наблюдаемое в других методах (например, RMSProp), что делает оценки более точными.
- Одна из самых цитируемых научных работ в мире.
- В 2018-2019 годах вышли статьи, указывающие на ошибку в оригинальной статье
- Не сходится для некоторых простых задач (даже выпуклых)

# Adam (Kingma and Ba, 2014) <sup>12</sup>



Объединяет элементы из AdaGrad и RMSProp. Использует экспоненциальное скользящее среднее как градиентов, так и их квадратов.

EMA:

$$\begin{aligned}m_j^{(k)} &= \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)} \\v_j^{(k)} &= \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2\end{aligned}$$

Коррекция смещения:

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}$$

$$\hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

Обновление:

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

**Заметки:**

- Компенсирует смещение к нулю на начальных итерациях, наблюдаемое в других методах (например, RMSProp), что делает оценки более точными.
- Одна из самых цитируемых научных работ в мире.
- В 2018-2019 годах вышли статьи, указывающие на ошибку в оригинальной статье
- Не сходится для некоторых простых задач (даже выпуклых)
- Почему-то очень хорошо работает для некоторых сложных задач

# Adam (Kingma and Ba, 2014)<sup>1</sup><sup>2</sup>



Объединяет элементы из AdaGrad и RMSProp. Использует экспоненциальное скользящее среднее как градиентов, так и их квадратов.

EMA:

$$\begin{aligned}m_j^{(k)} &= \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)} \\v_j^{(k)} &= \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2\end{aligned}$$

Коррекция смещения:

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}$$

$$\hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

Обновление:

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon}$$

**Заметки:**

- Компенсирует смещение к нулю на начальных итерациях, наблюдаемое в других методах (например, RMSProp), что делает оценки более точными.
- Одна из самых цитируемых научных работ в мире.
- В 2018-2019 годах вышли статьи, указывающие на ошибку в оригинальной статье
- Не сходится для некоторых простых задач (даже выпуклых)
- Почему-то очень хорошо работает для некоторых сложных задач
- Работает для языковых моделей значительно лучше, чем для задач компьютерного зрения. Почему?

<sup>1</sup>Adam: A Method for Stochastic Optimization

<sup>2</sup>On the Convergence of Adam and Beyond

# AdamW (Loshchilov & Hutter, 2017)

Решает проблему  $\ell_2$ -регуляризации в Adam. Стандартная  $\ell_2$ -регуляризация добавляет  $\lambda \|x\|^2$  к функции потерь, что дает добавку  $\lambda x$  к градиенту. В Adam эта добавка масштабируется адаптивным шагом обучения  $(\sqrt{\hat{v}_j} + \epsilon)$ , связывая затухание весов (weight decay) с величиной шага. AdamW отделяет затухание весов от адаптации шага.

Правило обновления:

$$m_j^{(k)} = \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)}$$

$$v_j^{(k)} = \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2$$

$$\hat{m}_j = \frac{m_j^{(k)}}{1 - \beta_1^k}, \quad \hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k}$$

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \left( \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon} + \lambda x_j^{(k-1)} \right)$$

# AdamW (Loshchilov & Hutter, 2017)

Решает проблему  $\ell_2$ -регуляризации в Adam. Стандартная  $\ell_2$ -регуляризация добавляет  $\lambda \|x\|^2$  к функции потерь, что дает добавку  $\lambda x$  к градиенту. В Adam эта добавка масштабируется адаптивным шагом обучения  $(\sqrt{\hat{v}_j} + \epsilon)$ , связывая затухание весов (weight decay) с величиной шага. AdamW отделяет затухание весов от адаптации шага.

Правило обновления:

$$\begin{aligned} m_j^{(k)} &= \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)} \\ v_j^{(k)} &= \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2 \\ \hat{m}_j &= \frac{m_j^{(k)}}{1 - \beta_1^k}, \quad \hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k} \\ x_j^{(k)} &= x_j^{(k-1)} - \alpha \left( \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon} + \lambda x_j^{(k-1)} \right) \end{aligned}$$

**Заметки:**

- Слагаемое затухания весов  $\lambda x_j^{(k-1)}$  добавляется после адаптивного шага по градиенту.

# AdamW (Loshchilov & Hutter, 2017)

Решает проблему  $\ell_2$ -регуляризации в Adam. Стандартная  $\ell_2$ -регуляризация добавляет  $\lambda \|x\|^2$  к функции потерь, что дает добавку  $\lambda x$  к градиенту. В Adam эта добавка масштабируется адаптивным шагом обучения  $(\sqrt{\hat{v}_j} + \epsilon)$ , связывая затухание весов (weight decay) с величиной шага. AdamW отделяет затухание весов от адаптации шага.

Правило обновления:

$$\begin{aligned} m_j^{(k)} &= \beta_1 m_j^{(k-1)} + (1 - \beta_1) g_j^{(k)} \\ v_j^{(k)} &= \beta_2 v_j^{(k-1)} + (1 - \beta_2) (g_j^{(k)})^2 \\ \hat{m}_j &= \frac{m_j^{(k)}}{1 - \beta_1^k}, \quad \hat{v}_j = \frac{v_j^{(k)}}{1 - \beta_2^k} \\ x_j^{(k)} &= x_j^{(k-1)} - \alpha \left( \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \epsilon} + \lambda x_j^{(k-1)} \right) \end{aligned}$$

**Заметки:**

- Слагаемое затухания весов  $\lambda x_j^{(k-1)}$  добавляется после адаптивного шага по градиенту.
- Широко используется в обучении трансформаторов и других крупных моделей. Вариант по умолчанию для Hugging Face Trainer.

# Много методов

Rosenbrock Function.  
Adaptive stochastic gradient algorithms.  
Learning rate 0.003



# **Оптимизация для глубокого обучения с практической точки зрения**

# Как сравнивать методы? Бенчмарк AlgoPerf<sup>3 4</sup>



- **Бенчмарк AlgoPerf:** Сравнивает алгоритмы обучения нейросетей в двух режимах:

# Как сравнивать методы? Бенчмарк AlgoPerf<sup>3 4</sup>



- **Бенчмарк AlgoPerf:** Сравнивает алгоритмы обучения нейросетей в двух режимах:
  - Внешняя настройка (*External Tuning*): моделирует подбор гиперпараметров при ограниченных ресурсах (5 запусков, квазислучайный поиск). Оценка — медианное минимальное время достижения цели по 5 наборам задач.

# Как сравнивать методы? Бенчмарк AlgoPerf<sup>3 4</sup>



- **Бенчмарк AlgoPerf:** Сравнивает алгоритмы обучения нейросетей в двух режимах:
  - Внешняя настройка (*External Tuning*): моделирует подбор гиперпараметров при ограниченных ресурсах (5 запусков, квазислучайный поиск). Оценка — медианное минимальное время достижения цели по 5 наборам задач.
  - Самонастройка (*Self-Tuning*): моделирует автоматический подбор на одной машине (фиксированный или внутренний подбор, бюджет ×3). Оценка — медианное время выполнения по 5 наборам задач.

# Как сравнивать методы? Бенчмарк AlgoPerf<sup>3 4</sup>



- **Бенчмарк AlgoPerf:** Сравнивает алгоритмы обучения нейросетей в двух режимах:
  - **Внешняя настройка (External Tuning):** моделирует подбор гиперпараметров при ограниченных ресурсах (5 запусков, квазислучайный поиск). Оценка — медианное минимальное время достижения цели по 5 наборам задач.
  - **Самонастройка (Self-Tuning):** моделирует автоматический подбор на одной машине (фиксированный или внутренний подбор, бюджет ×3). Оценка — медианное время выполнения по 5 наборам задач.
- **Оценка:** результаты агрегируются с помощью профилей производительности. Профили показывают долю задач, решённых за время, не превышающее множитель  $\tau$  относительно самой быстрой посылки. Итоговая оценка — нормированная площадь под кривой профиля (1.0 = самая быстрая на всех задачах).

# Как сравнивать методы? Бенчмарк AlgoPerf<sup>3</sup> <sup>4</sup>



- **Бенчмарк AlgoPerf:** Сравнивает алгоритмы обучения нейросетей в двух режимах:
  - **Внешняя настройка (External Tuning):** моделирует подбор гиперпараметров при ограниченных ресурсах (5 запусков, квазислучайный поиск). Оценка — медианное минимальное время достижения цели по 5 наборам задач.
  - **Самонастройка (Self-Tuning):** моделирует автоматический подбор на одной машине (фиксированный или внутренний подбор, бюджет ×3). Оценка — медианное время выполнения по 5 наборам задач.
- **Оценка:** результаты агрегируются с помощью профилей производительности. Профили показывают долю задач, решённых за время, не превышающее множитель  $\tau$  относительно самой быстрой посылки. Итоговая оценка — нормированная площадь под кривой профиля (1.0 = самая быстрая на всех задачах).
- **Затраты ресурсов:** оценка требует  $\sim 49,240$  часов суммарно на 8x NVIDIA V100 GPUs (в среднем  $\sim 3469$  ч/внешняя настройка,  $\sim 1847$  ч/самонастройка).

---

<sup>3</sup>Benchmarking Neural Network Training Algorithms

<sup>4</sup>Accelerating neural network training: An analysis of the AlgoPerf competition

# Бенчмарк AlgoPerf

**Сводка фиксированных базовых задач в бенчмарке AlgoPerf.** Функции потерь включают кросс-энтропию (CE), среднюю абсолютную ошибку (L1) и функцию потерь CTC (Connectionist Temporal Classification). Дополнительные метрики оценки: индекс структурного сходства (SSIM), коэффициент ошибок (ER), доля ошибок по словам (WER), средняя усреднённая точность (mAP) и метрика BLEU (*bilingual evaluation understudy*). Бюджет времени выполнения соответствует правилам внешней настройки; правила самонастройки допускают обучение, в 3 раза более длительное.

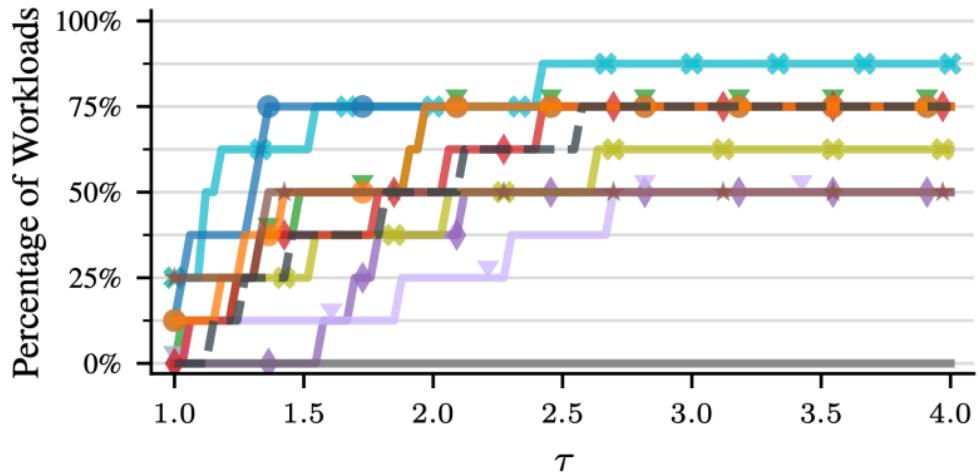
Задача	Датасет	Модель	Функция потерь	Метрика	Целевое значение (валидация)	Бюджет времени
Clickthrough rate prediction	CRITEO 1TB	DLRMSMALL	CE	CE	0.123735	7703
MRI reconstruction	FASTMRI	U-NET	L1	SSIM	0.7344	8859
Image classification	IMAGENET	ResNet-50	CE	ER	0.22569	63,008
		ViT	CE	ER	0.22691	77,520
Speech recognition	LIBRISPEECH	Conformer	CTC	WER	0.085884	61,068
		DeepSpeech	CTC	WER	0.119936	55,506
Molecular property prediction	OGBG	GNN	CE	mAP	0.28098	18,477
Translation	WMT	Transformer	CE	BLEU	30.8491	48,151

# Бенчмарк AlgoPerf



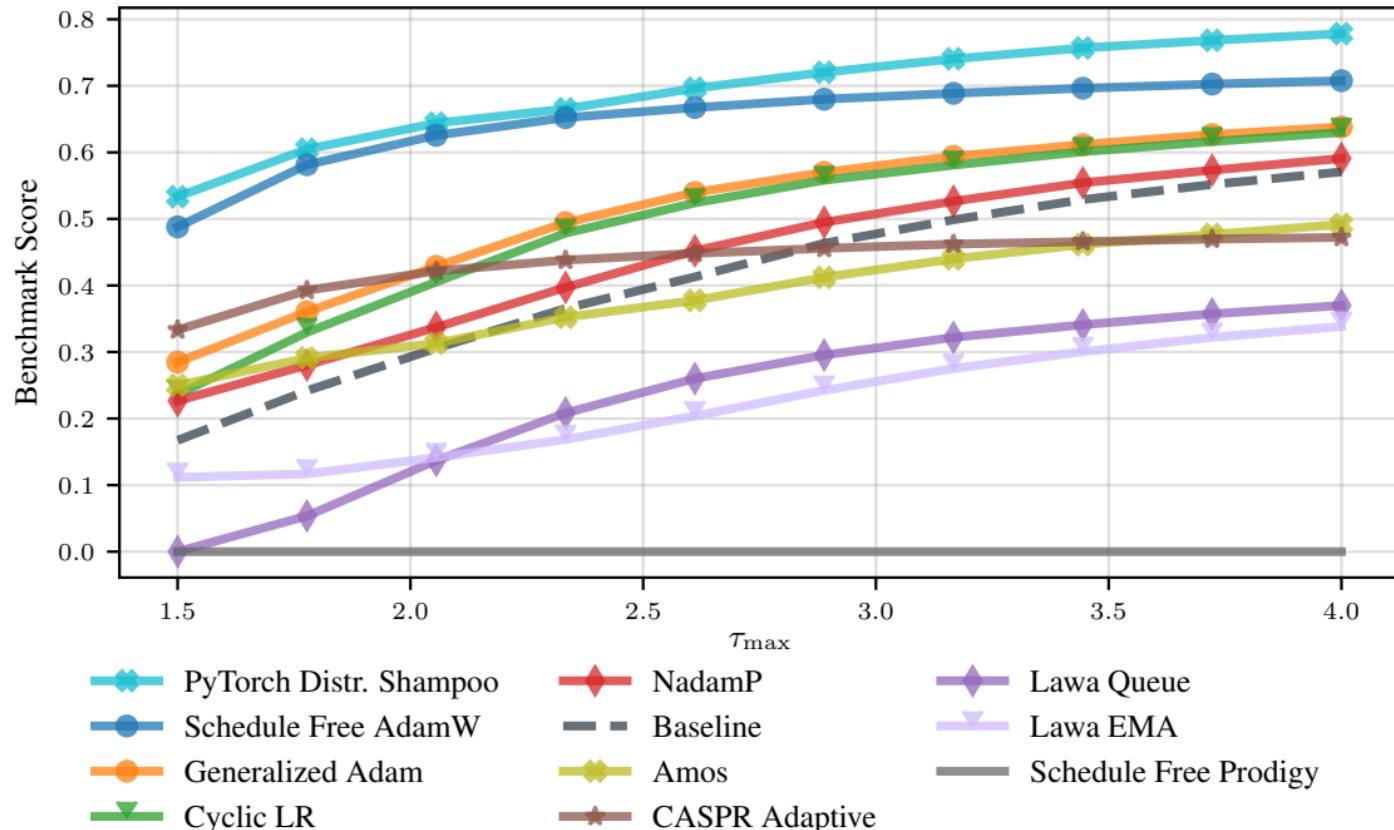
Submission	Line	Score
PYTORCH DISTRIBUTED SHAMPOO		0.7784
SCHEDULE FREE ADAMW		0.7077
GENERALIZED ADAM		0.6383
CYCLIC LR		0.6301
NADAMP		0.5909
<b>BASELINE</b>		<b>0.5707</b>
AMOS		0.4918
CASPR ADAPTIVE		0.4722
LAWA QUEUE		0.3699
LAWA EMA		0.3384
SCHEDULE FREE PRODIGY		0

(a) External tuning leaderboard

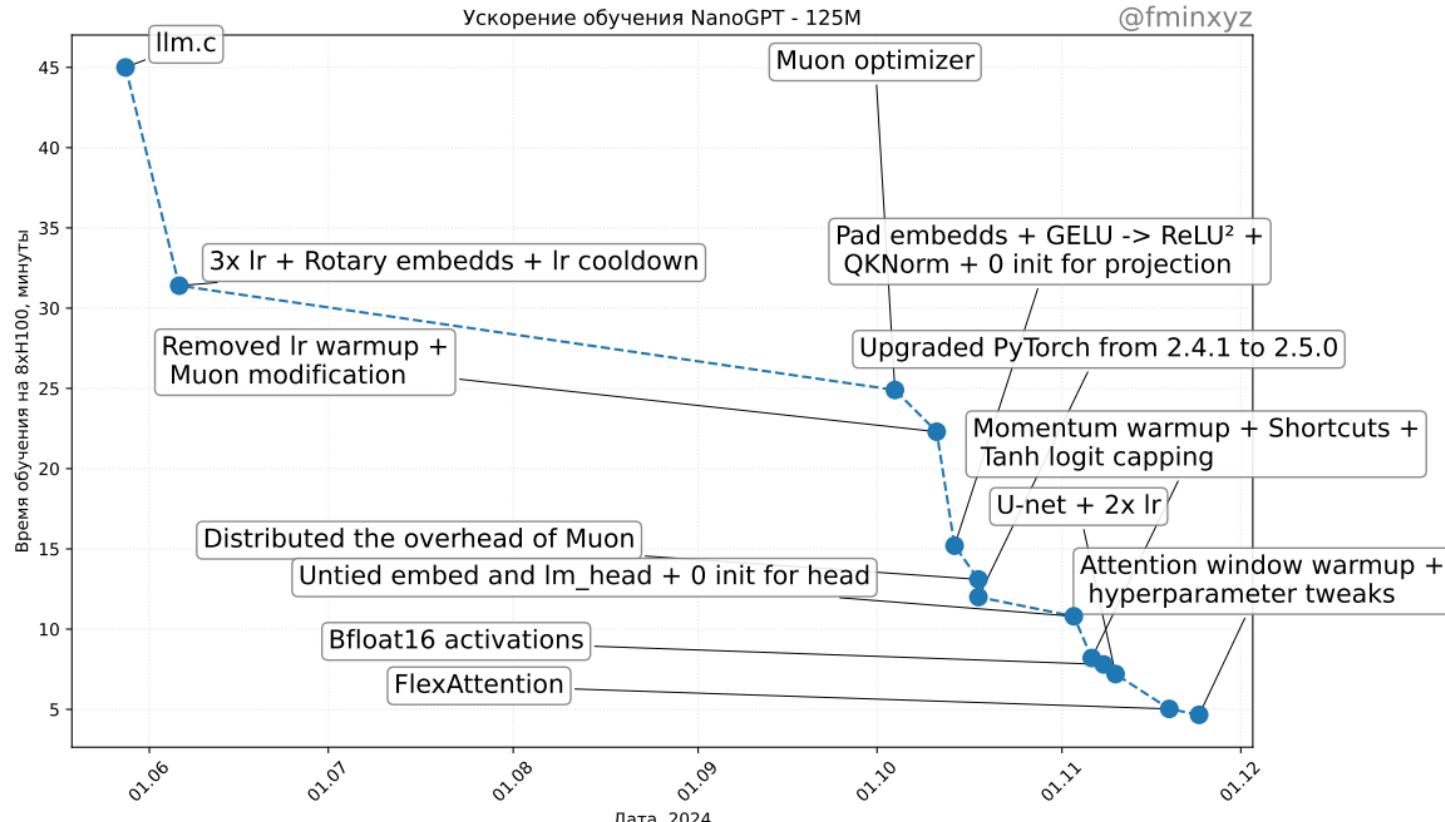


(b) External tuning performance profiles

# Бенчмарк AlgoPerf



# NanoGPT speedrun



# Работают ли трюки, если увеличить размер модели?

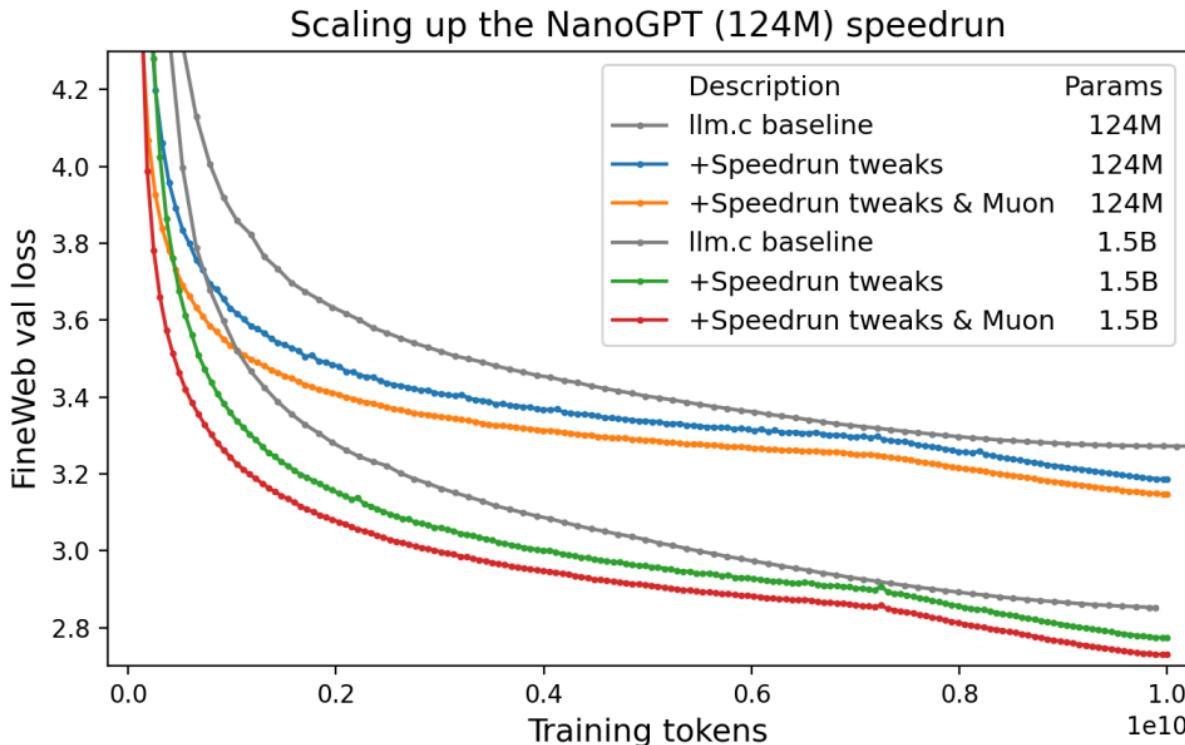


Рисунок 2. Источник

# Работают ли трюки, если увеличить размер модели?

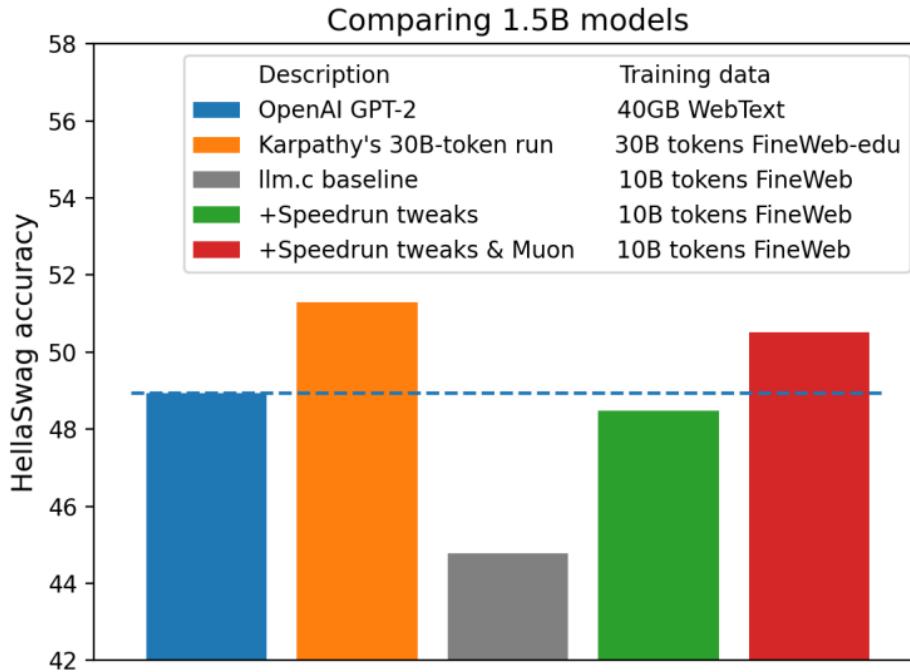


Рисунок 3. Источник



# Неожиданные истории

# Adam работает хуже для CV, чем для LLM? <sup>5</sup>

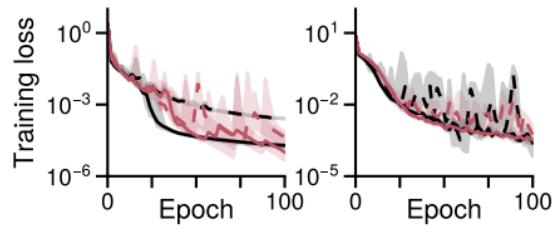


Рисунок 4. CNNs on MNIST and CIFAR10

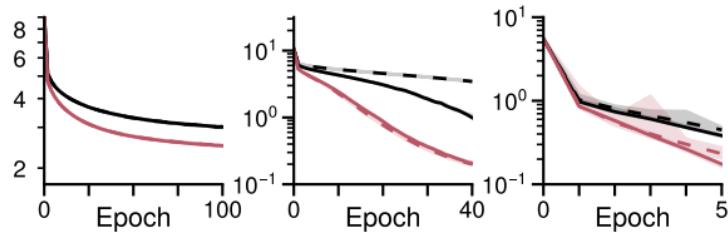


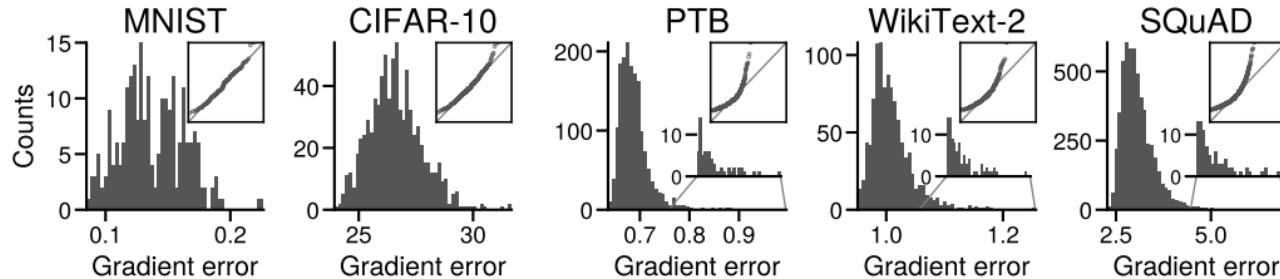
Рисунок 5. Transformers on PTB, WikiText2, and SQuAD

Чёрные линии — SGD, красные — Adam.

<sup>5</sup>Linear attention is (maybe) all you need (to understand transformer optimization)

# Почему Adam работает хуже для CV, чем для LLM? <sup>6</sup>

Потому что шум градиентов в языковых моделях имеет тяжелые хвосты?



<sup>6</sup>Linear attention is (maybe) all you need (to understand transformer optimization)

# Почему Adam работает хуже для CV, чем для LLM?<sup>7</sup>



Нет! Распределение меток имеет тяжёлые хвосты!

В компьютерном зрении датасеты часто сбалансированы: 1000 котиков, 1000 песелей и т.д.

В языковых датасетах почти всегда не так: слово *the* встречается часто, слово *tie* — на порядки реже.

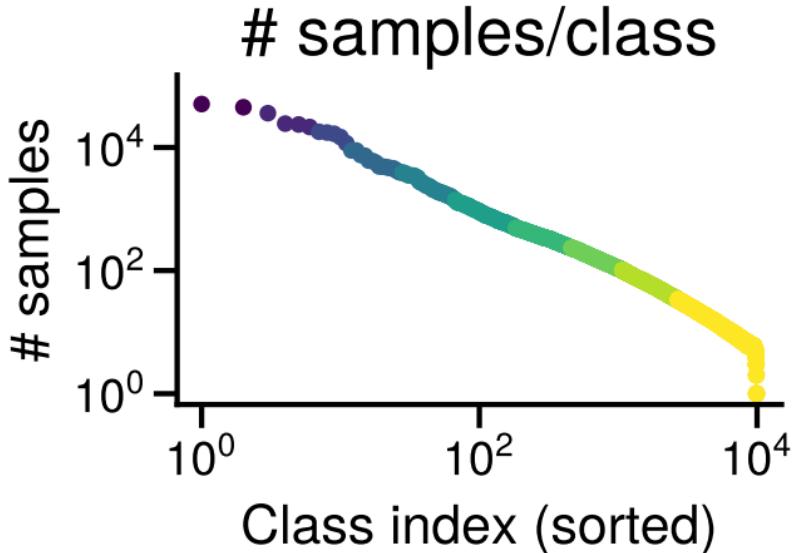


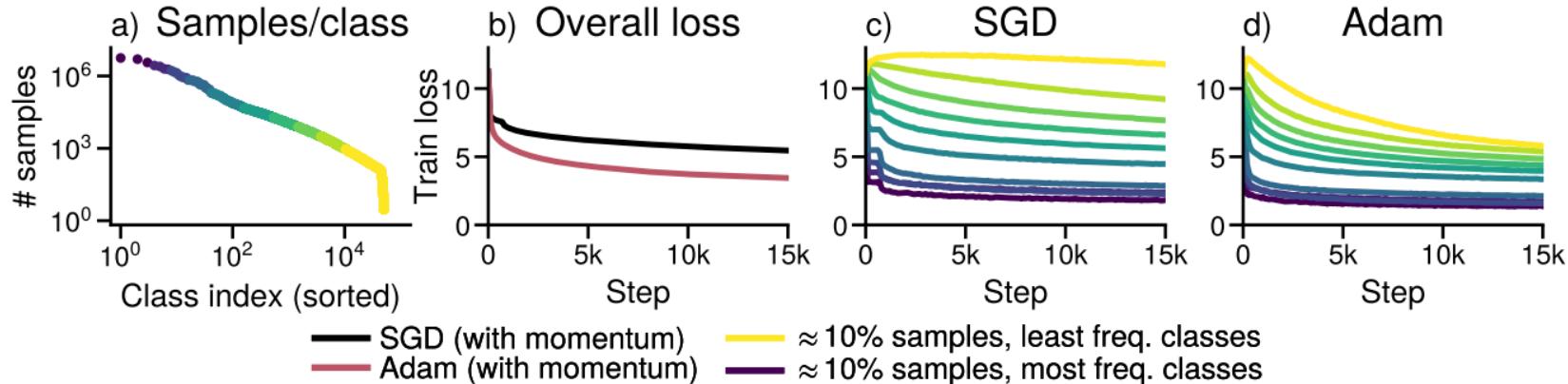
Рисунок 6. Распределение частоты токенов в PTB

<sup>7</sup>Heavy-Tailed Class Imbalance and Why Adam Outperforms Gradient Descent on Language Models

# Почему Adam работает хуже для CV, чем для LLM? <sup>8</sup>



SGD медленно прогрессирует на редких классах



SGD не добивается прогресса на низкочастотных классах, в то время как Adam добивается. Обучение GPT-2 S на WikiText-103. (a) Распределение классов, отсортированных по частоте встречаемости, разбитых на группы, соответствующие  $\approx 10\%$  данных. (b) Значение функции потерь при обучении. (c, d) Значение функции потерь при обучении для каждой группы при использовании SGD и Adam.

<sup>8</sup>Heavy-Tailed Class Imbalance and Why Adam Outperforms Gradient Descent on Language Models

# Влияние инициализации



Правильная инициализация нейронной сети важна. Функция потерь нейронной сети сильно невыпукла; оптимизировать её для достижения «хорошего» решения трудно, это требует тщательной настройки.

# Влияние инициализации



Правильная инициализация нейронной сети важна. Функция потерь нейронной сети сильно невыпукла; оптимизировать её для достижения «хорошего» решения трудно, это требует тщательной настройки.

- Не инициализируйте все веса одинаково — почему?



# Влияние инициализации



Правильная инициализация нейронной сети важна. Функция потерь нейронной сети сильно невыпукла; оптимизировать её для достижения «хорошего» решения трудно, это требует тщательной настройки.

- Не инициализируйте все веса одинаково — почему?
- Случайная инициализация: инициализируйте случайно, например, из гауссовского распределения  $N(0, \sigma^2)$ , где стандартное отклонение  $\sigma$  зависит от числа нейронов в слое. Это обеспечивает нарушение симметрии (*symmetry breaking*).

# Влияние инициализации



Правильная инициализация нейронной сети важна. Функция потерь нейронной сети сильно невыпукла; оптимизировать её для достижения «хорошего» решения трудно, это требует тщательной настройки.

- Не инициализируйте все веса одинаково — почему?
- Случайная инициализация: инициализируйте случайно, например, из гауссовского распределения  $N(0, \sigma^2)$ , где стандартное отклонение  $\sigma$  зависит от числа нейронов в слое. Это обеспечивает нарушение симметрии (*symmetry breaking*).
- Можно найти более полезные советы здесь

<sup>9</sup>On the importance of initialization and momentum in deep learning Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton

# Влияние инициализации весов нейронной сети на сходимость методов<sup>10</sup>

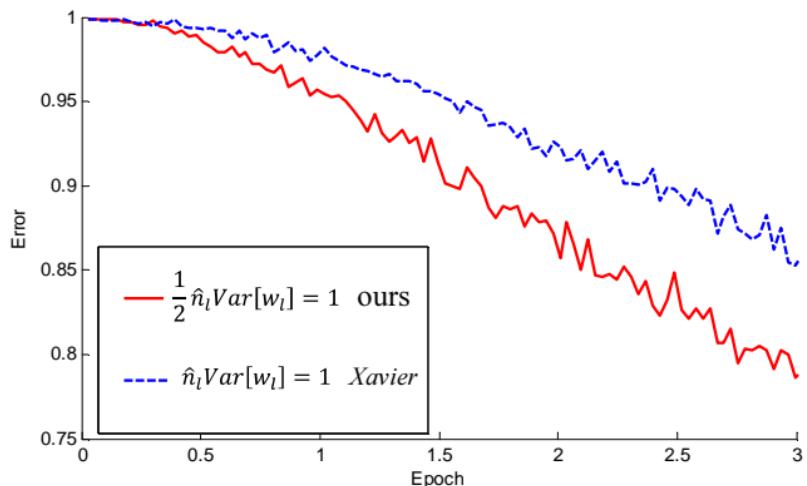


Рисунок 7. 22-layer ReLU net: good init converges faster

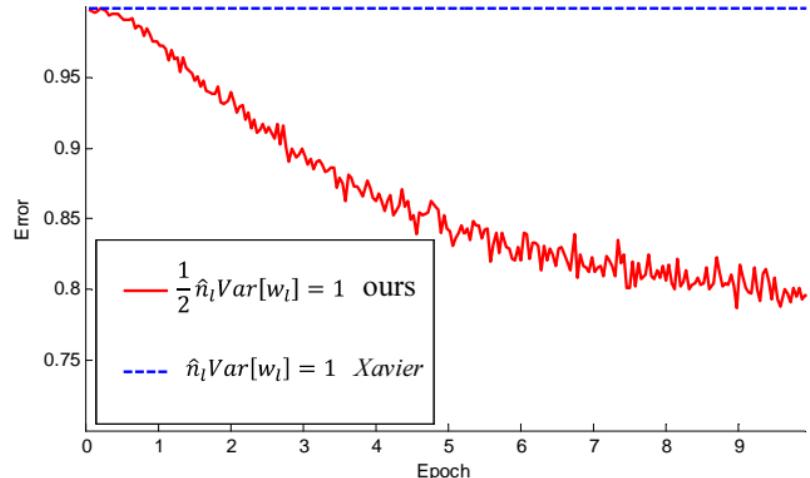


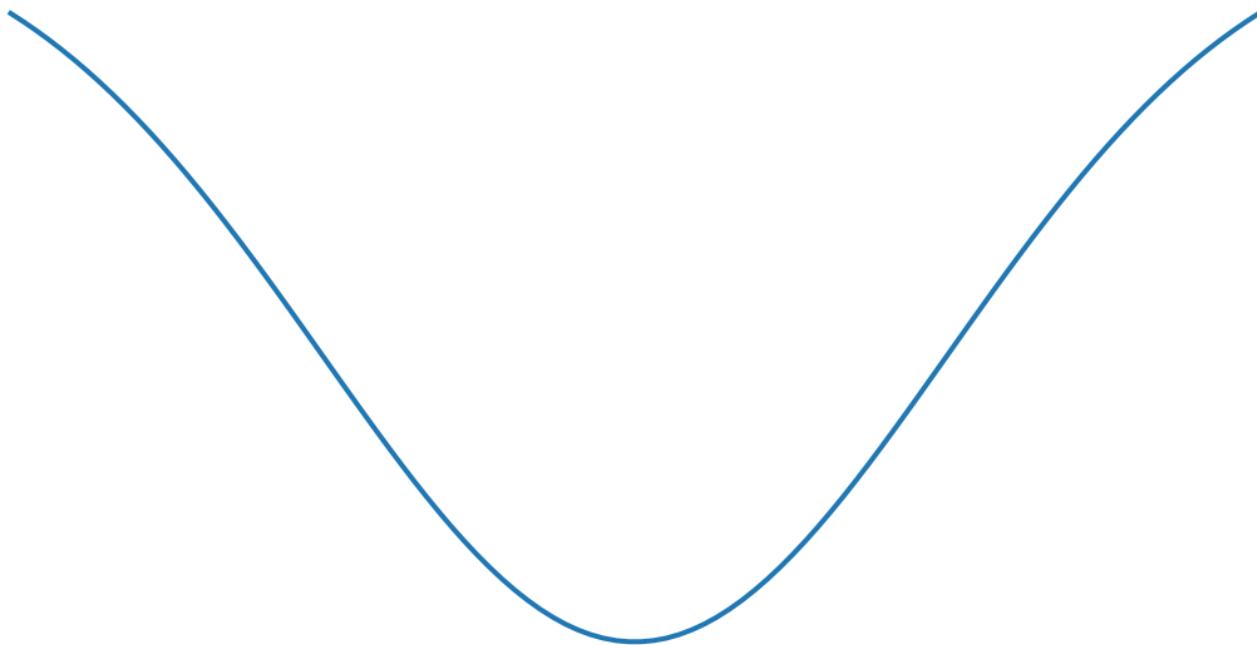
Рисунок 8. 30-layer ReLU net: good init is able to converge

<sup>10</sup>Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

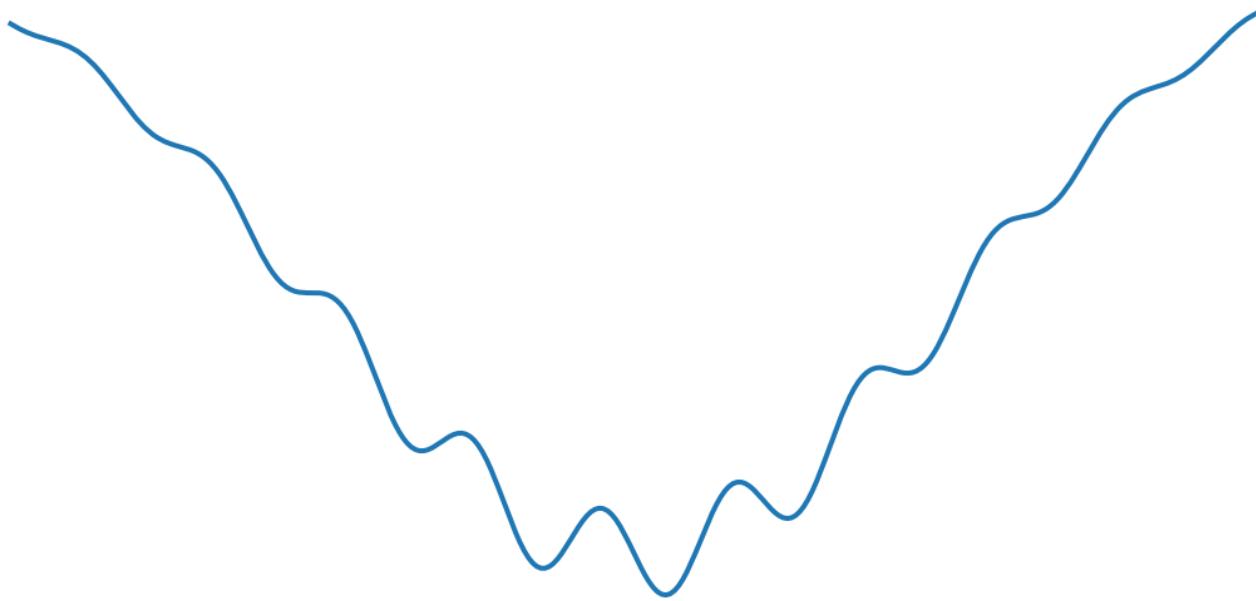


# Весёлые истории

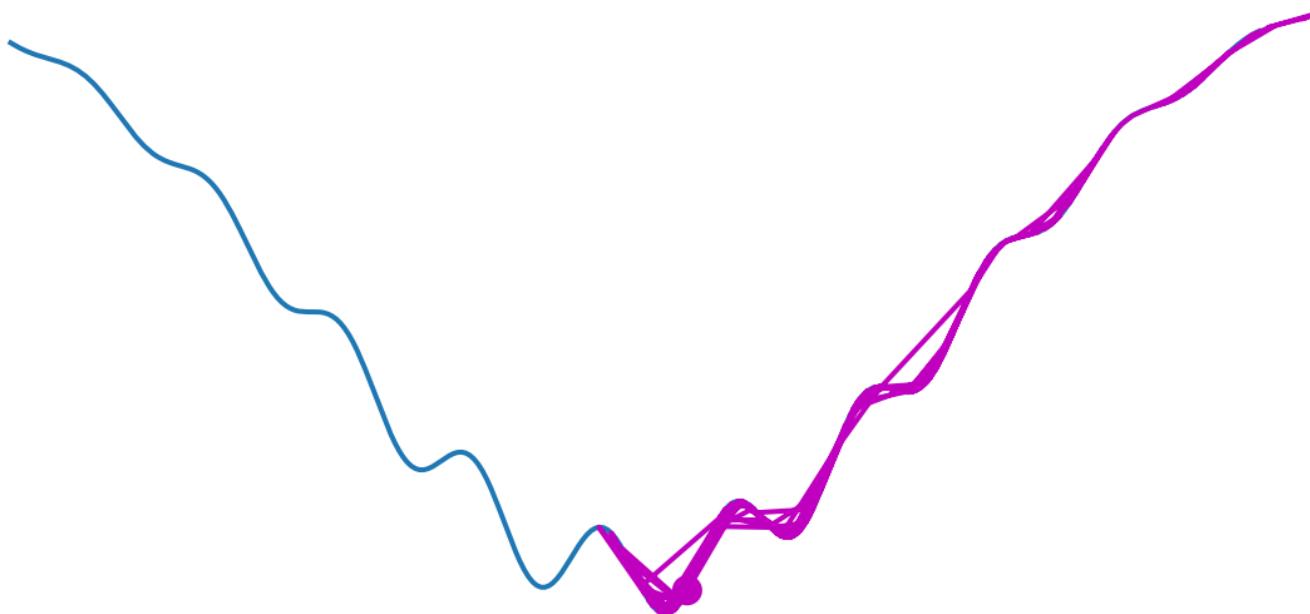
Градиентный спуск сходится к локальному минимуму



Градиентный спуск  
сходится к локальному минимуму



Стохастический градиентный спуск  
выпрыгивает из локальных минимумов



# Визуализация с помощью проекции на прямую

- Обозначим через  $w_0$  начальные веса нейронной сети. Веса, полученные после обучения, обозначим  $\hat{w}$ .

# Визуализация с помощью проекции на прямую

- Обозначим через  $w_0$  начальные веса нейронной сети. Веса, полученные после обучения, обозначим  $\hat{w}$ .

# Визуализация с помощью проекции на прямую

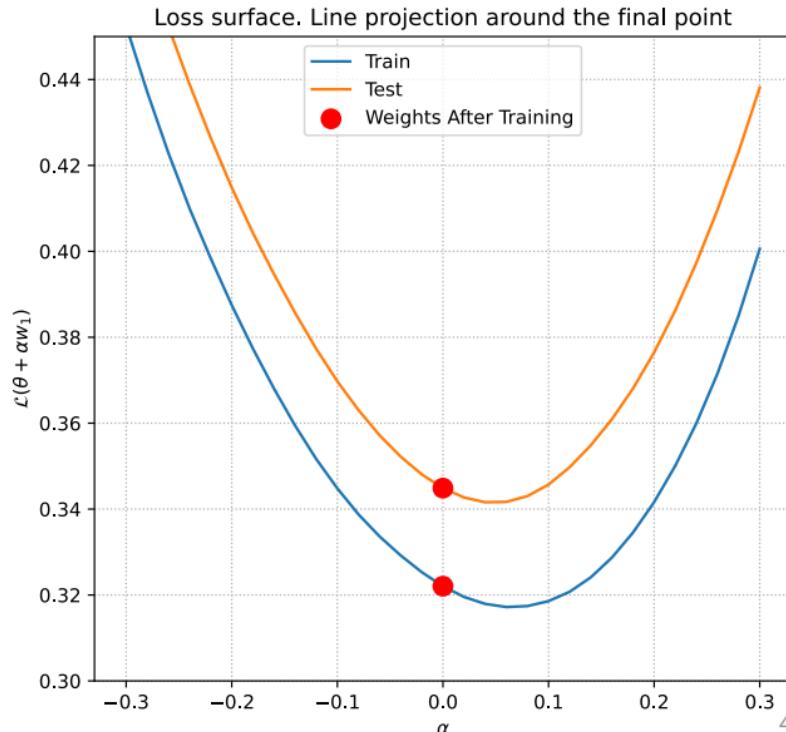
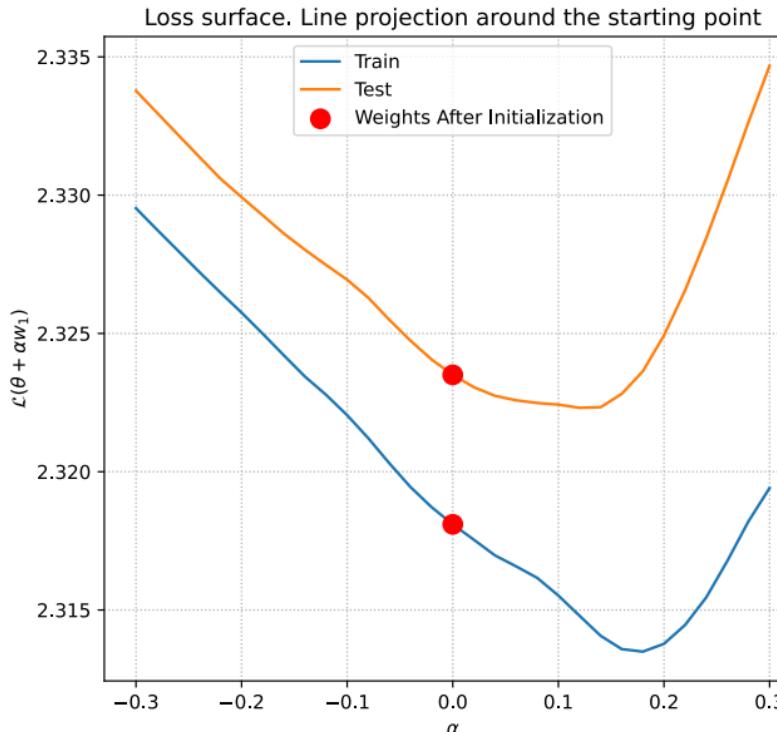
- Обозначим через  $w_0$  начальные веса нейронной сети. Веса, полученные после обучения, обозначим  $\hat{w}$ .
- Сгенерируем случайное направление  $w_1 \in \mathbb{R}^p$  той же размерности, затем вычислим значение функции потерь вдоль этого направления:

$$L(\alpha) = L(w_0 + \alpha w_1), \quad \text{где } \alpha \in [-b, b].$$

# Проекция функции потерь нейронной сети на прямую



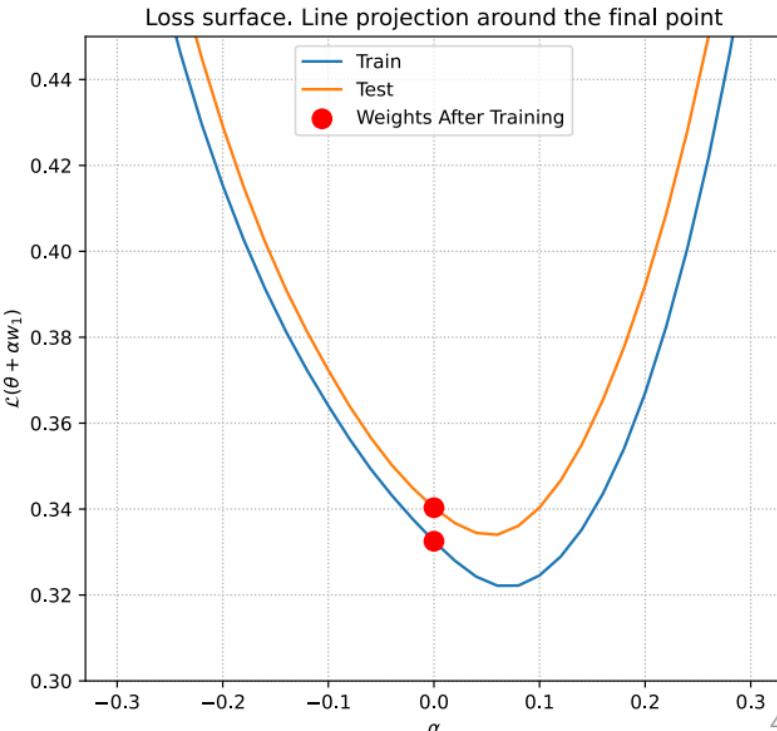
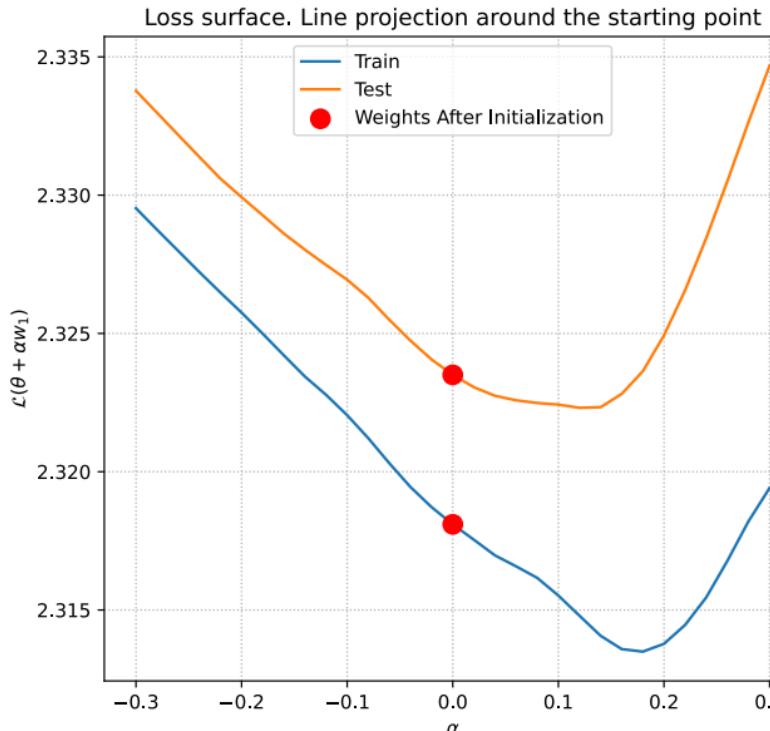
No Dropout



# Проекция функции потерь нейронной сети на прямую



Dropout 0.2



# Проекция функции потерь нейронной сети на плоскость

- Мы можем расширить эту идею и построить проекцию поверхности потерь на плоскость, которая задается 2 случайными векторами.

# Проекция функции потерь нейронной сети на плоскость

- Мы можем расширить эту идею и построить проекцию поверхности потерь на плоскость, которая задается 2 случайными векторами.

# Проекция функции потерь нейронной сети на плоскость

- Мы можем расширить эту идею и построить проекцию поверхности потерь на плоскость, которая задается 2 случайными векторами.
- Два случайных гауссовых вектора в пространстве большой размерности с высокой вероятностью ортогональны.

$$L(\alpha, \beta) = L(w_0 + \alpha w_1 + \beta w_2), \quad \text{где } \alpha, \beta \in [-b, b]^2.$$

# Проекция функции потерь нейронной сети на плоскость

- Мы можем расширить эту идею и построить проекцию поверхности потерь на плоскость, которая задается 2 случайными векторами.
- Два случайных гауссовых вектора в пространстве большой размерности с высокой вероятностью ортогональны.

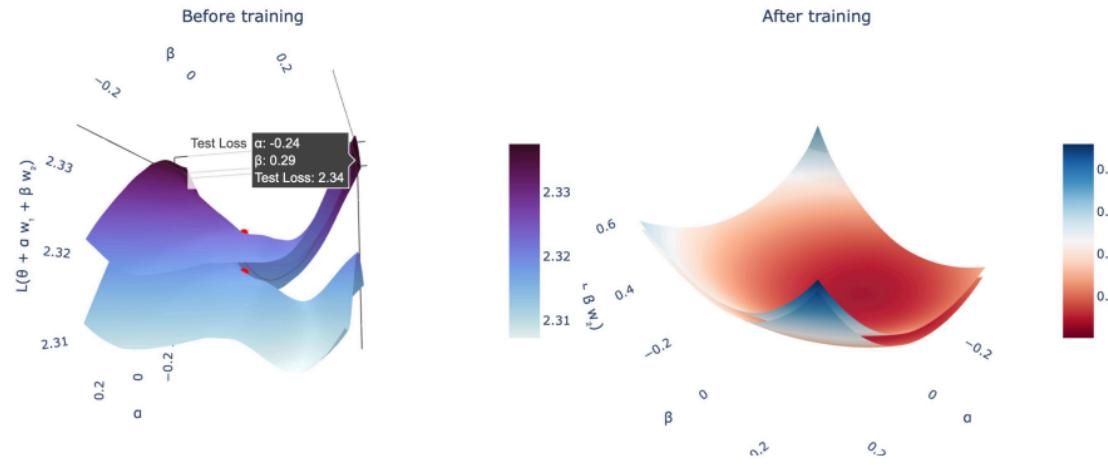
$$L(\alpha, \beta) = L(w_0 + \alpha w_1 + \beta w_2), \quad \text{где } \alpha, \beta \in [-b, b]^2.$$

# Проекция функции потерь нейронной сети на плоскость

- Мы можем расширить эту идею и построить проекцию поверхности потерь на плоскость, которая задается 2 случайными векторами.
- Два случайных гауссовых вектора в пространстве большой размерности с высокой вероятностью ортогональны.

$$L(\alpha, \beta) = L(w_0 + \alpha w_1 + \beta w_2), \quad \text{где } \alpha, \beta \in [-b, b]^2.$$

No Dropout. Plane projection of loss surface.



# Может ли быть полезно изучение таких проекций? <sup>11</sup>

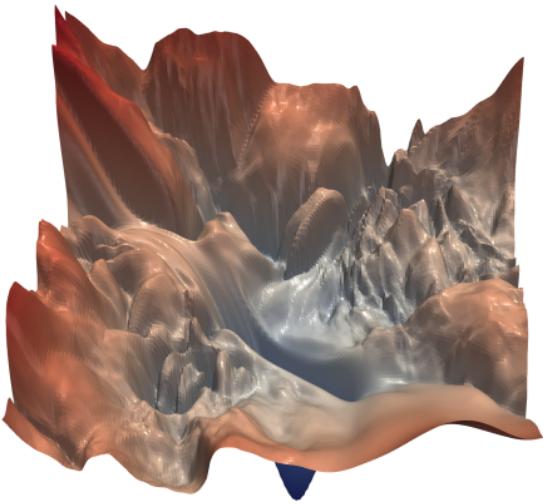


Рисунок 12. The loss surface of ResNet-56  
without skip connections

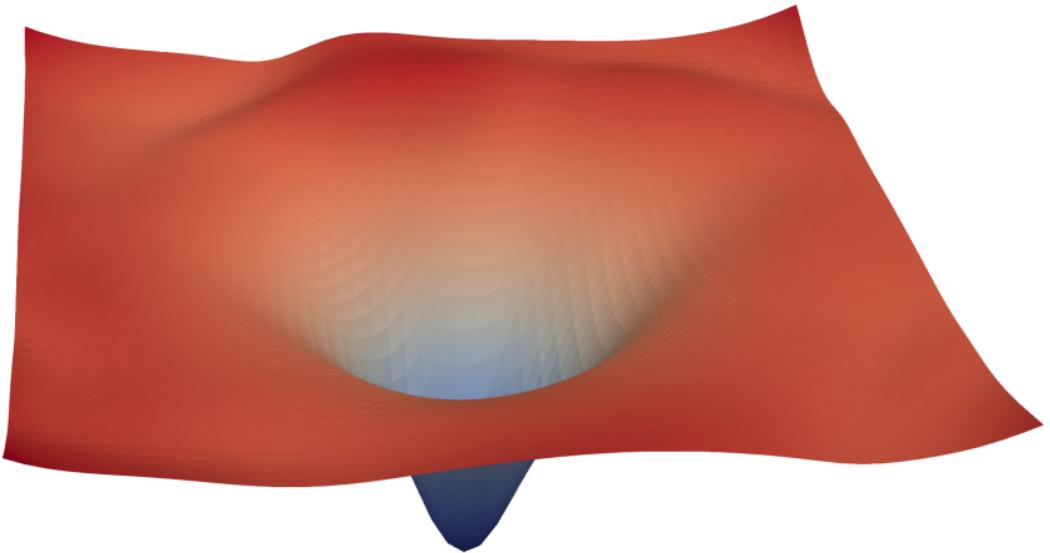


Рисунок 13. The loss surface of ResNet-56 with skip connections

<sup>11</sup>Visualizing the Loss Landscape of Neural Nets, Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein

# Может ли быть полезно изучение таких проекций, если серьезно? <sup>12</sup>

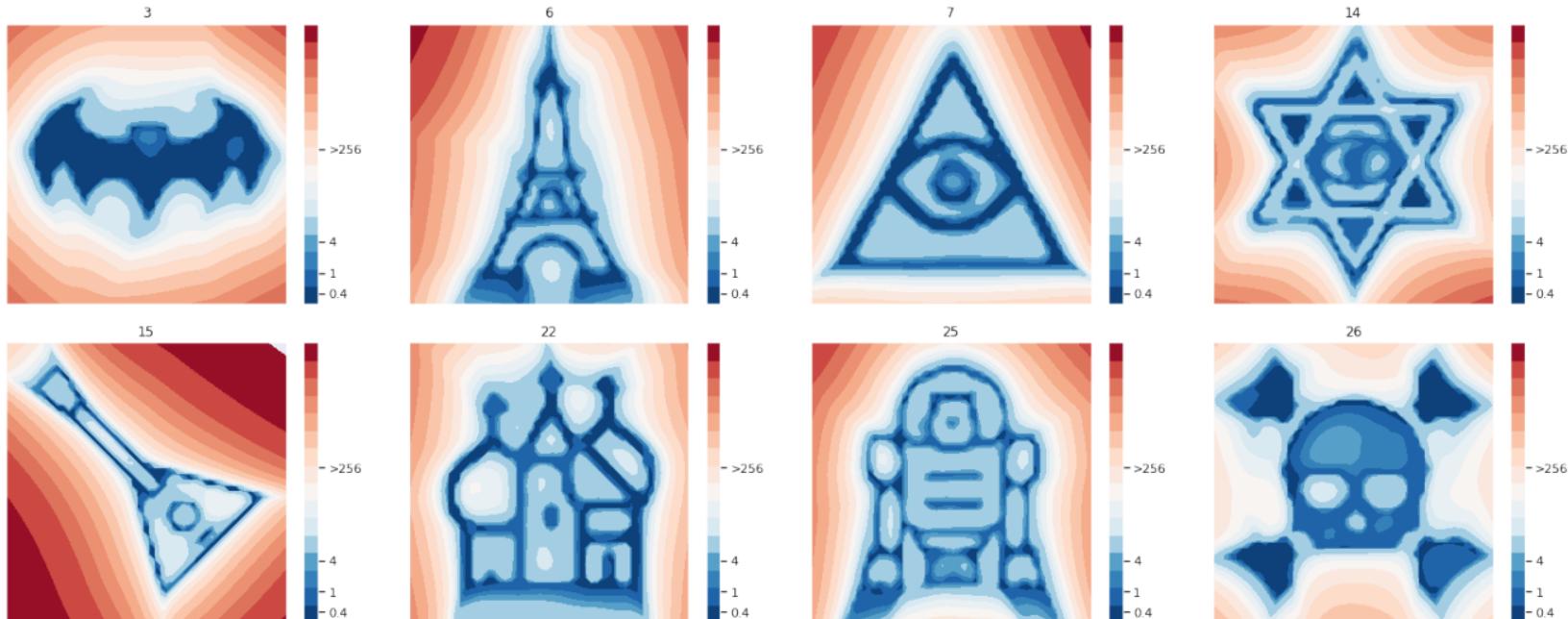
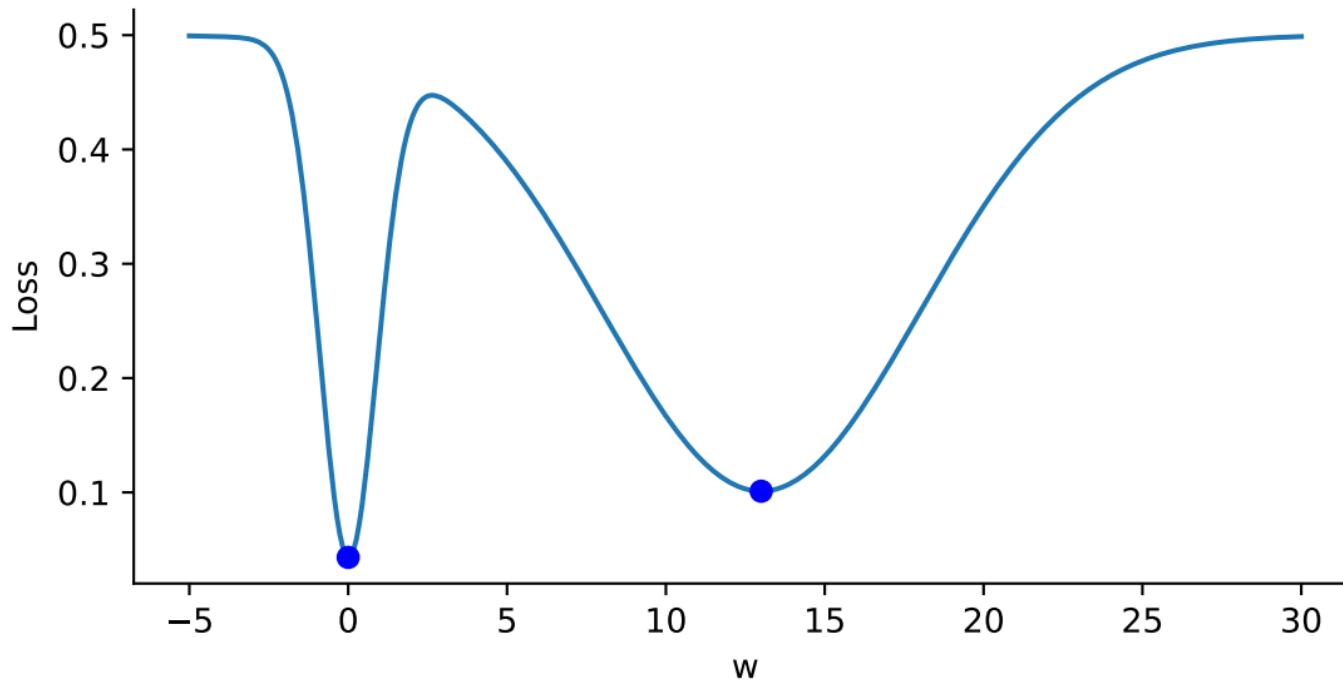


Рисунок 14. Examples of a loss landscape of a typical CNN model on FashionMNIST and CIFAR10 datasets found with MPO. Loss values are color-coded according to a logarithmic scale

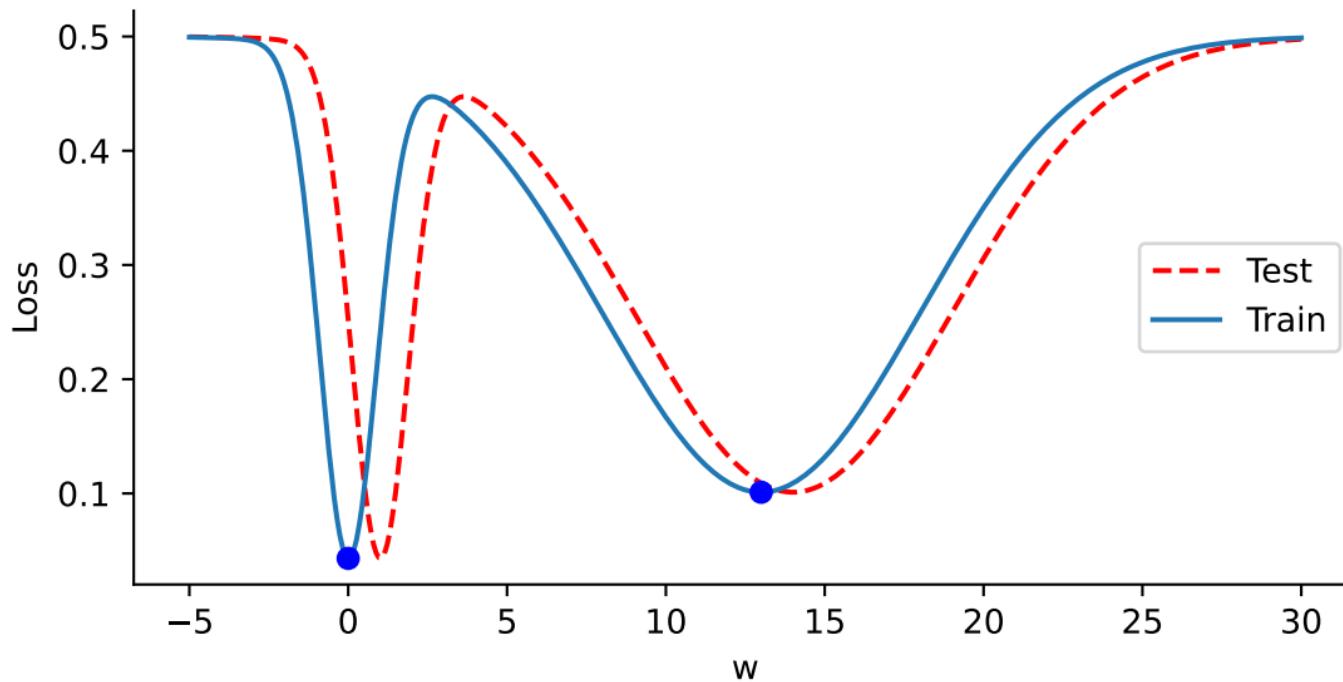
# Ширина локальных минимумов

Узкие и широкие локальные минимумы



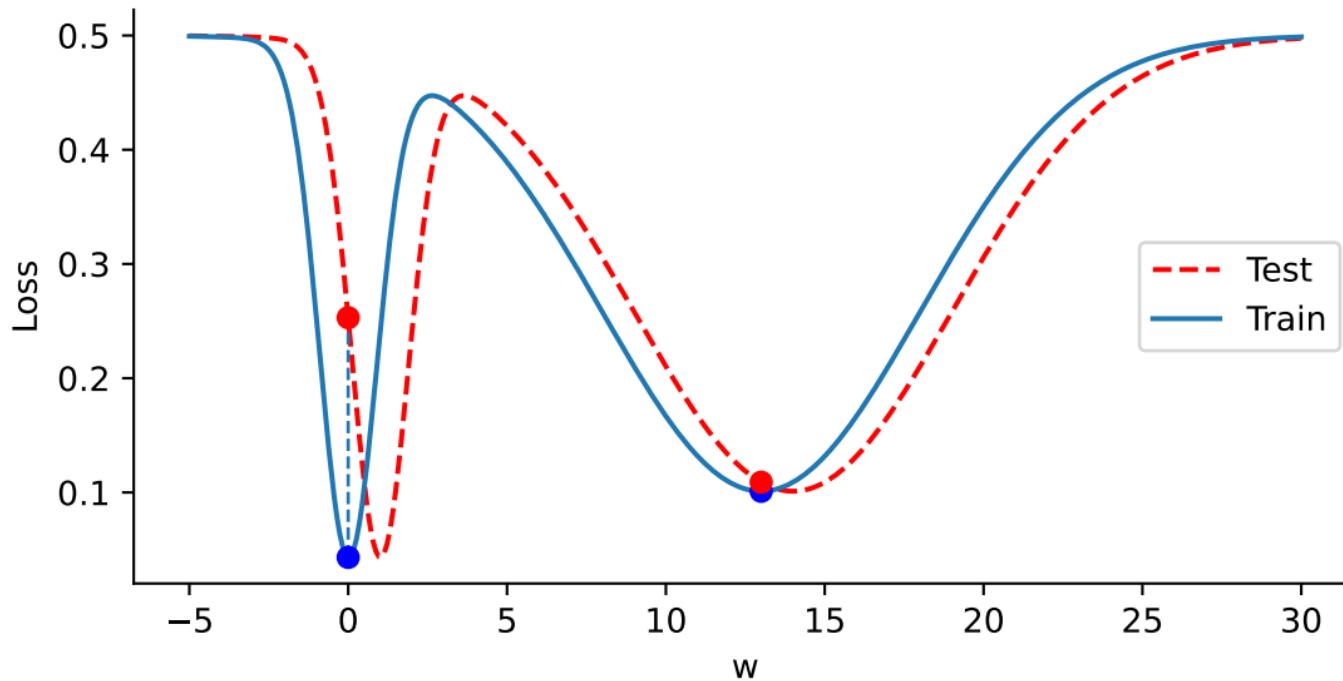
# Ширина локальных минимумов

Узкие и широкие локальные минимумы



# Ширина локальных минимумов

Узкие и широкие локальные минимумы





# Немного про LR schedulers

- В классическом подходе шаг обучения (learning rate, LR):



# Немного про LR schedulers

- В классическом подходе шаг обучения (learning rate, LR):
  - слишком маленький  $\Rightarrow$  медленная сходимость;

# Немного про LR schedulers

- В классическом подходе шаг обучения (learning rate, LR):
  - слишком маленький  $\Rightarrow$  медленная сходимость;
  - слишком большой  $\Rightarrow$  «перескакивание» областей малых значений функции потерь.

# Немного про LR schedulers

- В классическом подходе шаг обучения (learning rate, LR):
  - слишком маленький  $\Rightarrow$  медленная сходимость;
  - слишком большой  $\Rightarrow$  «перескакивание» областей малых значений функции потерь.
- Популярные стратегии выбора LR:

# Немного про LR schedulers

- В классическом подходе шаг обучения (learning rate, LR):
  - слишком маленький ⇒ медленная сходимость;
  - слишком большой ⇒ «перескакивание» областей малых значений функции потерь.
- Популярные стратегии выбора LR:
  - step decay;

# Немного про LR schedulers

- В классическом подходе шаг обучения (learning rate, LR):
  - слишком маленький  $\Rightarrow$  медленная сходимость;
  - слишком большой  $\Rightarrow$  «перескакивание» областей малых значений функции потерь.
- Популярные стратегии выбора LR:
  - step decay;
  - cosine decay;

# Немного про LR schedulers

- В классическом подходе шаг обучения (learning rate, LR):
  - слишком маленький  $\Rightarrow$  медленная сходимость;
  - слишком большой  $\Rightarrow$  «перескакивание» областей малых значений функции потерь.
- Популярные стратегии выбора LR:
  - step decay;
  - cosine decay;
  - циклические и осциллирующие схемы.

# Немного про LR schedulers

- В классическом подходе шаг обучения (learning rate, LR):
  - слишком маленький  $\Rightarrow$  медленная сходимость;
  - слишком большой  $\Rightarrow$  «перескакивание» областей малых значений функции потерь.
- Популярные стратегии выбора LR:
  - step decay;
  - cosine decay;
  - циклические и осциллирующие схемы.
- Интуитивная идея: большие LR в начале (исследование пространства), меньшие — в конце (уточнение минимума).

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!)<sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!)<sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?
- Авторы предлагают **экспоненциально растущую** стратегию LR:

$$\eta_t = \eta_0(1 + \alpha)^t, \quad \alpha > 0$$

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!)<sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?
- Авторы предлагают **экспоненциально растущую** стратегию LR:

$$\eta_t = \eta_0(1 + \alpha)^t, \quad \alpha > 0$$

- Несмотря на быстрое «взрывание» шага, обучение **всё ещё возможно**.

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!)<sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?
- Авторы предлагают **экспоненциально растущую** стратегию LR:

$$\eta_t = \eta_0(1 + \alpha)^t, \quad \alpha > 0$$

- Несмотря на быстрое «взрывание» шага, обучение **всё ещё возможно**.
- Экспериментальный факт:

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!)<sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?
- Авторы предлагают **экспоненциально растущую** стратегию LR:

$$\eta_t = \eta_0(1 + \alpha)^t, \quad \alpha > 0$$

- Несмотря на быстрое «взрывание» шага, обучение **всё ещё возможно**.
- Экспериментальный факт:
  - стандартные архитектуры для CIFAR-10 (например, PreResNet-32) успешно обучаются с ExpLR;

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!)<sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?
- Авторы предлагают **экспоненциально растущую** стратегию LR:

$$\eta_t = \eta_0(1 + \alpha)^t, \quad \alpha > 0$$

- Несмотря на быстрое «взрывание» шага, обучение **всё ещё возможно**.
- Экспериментальный факт:
  - стандартные архитектуры для CIFAR-10 (например, PreResNet-32) успешно обучаются с ExpLR;
  - при корректном выборе  $\alpha$  траектория оптимизации оказывается близка к классической стратегии: фиксированный LR + weight decay.

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!)<sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?
- Авторы предлагают **экспоненциально растущую** стратегию LR:

$$\eta_t = \eta_0(1 + \alpha)^t, \quad \alpha > 0$$

- Несмотря на быстрое «взрывание» шага, обучение **всё ещё возможно**.
- Экспериментальный факт:
  - стандартные архитектуры для CIFAR-10 (например, PreResNet-32) успешно обучаются с ExpLR;
  - при корректном выборе  $\alpha$  траектория оптимизации оказывается близка к классической стратегии: фиксированный LR + weight decay.
- Наблюдение: нормализация + weight decay создают эффект, напоминающий «эффективное увеличение» LR в процессе обучения.

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!) <sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?
- Авторы предлагают **экспоненциально растущую** стратегию LR:

$$\eta_t = \eta_0(1 + \alpha)^t, \quad \alpha > 0$$

- Несмотря на быстрое «взрывание» шага, обучение **всё ещё возможно**.
- Экспериментальный факт:
  - стандартные архитектуры для CIFAR-10 (например, PreResNet-32) успешно обучаются с ExpLR;
  - при корректном выборе  $\alpha$  траектория оптимизации оказывается близка к классической стратегии: фиксированный LR + weight decay.
- Наблюдение: нормализация + weight decay создают эффект, напоминающий «эффективное увеличение» LR в процессе обучения.

<sup>13</sup> Exponential Learning Rate Schedules for Deep Learning (2020)

# Немного про LR schedulers: Экспоненциально растущий LR (ExpLR) (??!??!)<sup>13</sup>

- Вопрос авторов: действительно ли уменьшение LR является необходимым условием успешного обучения глубоких сетей?
- Авторы предлагают **экспоненциально растущую** стратегию LR:

$$\eta_t = \eta_0(1 + \alpha)^t, \quad \alpha > 0$$

- Несмотря на быстрое «взрывание» шага, обучение **всё ещё возможно**.
- Экспериментальный факт:
  - стандартные архитектуры для CIFAR-10 (например, PreResNet-32) успешно обучаются с ExpLR;
  - при корректном выборе  $\alpha$  траектория оптимизации оказывается близка к классической стратегии: фиксированный LR + weight decay.
- Наблюдение: нормализация + weight decay создают эффект, напоминающий «эффективное увеличение» LR в процессе обучения.

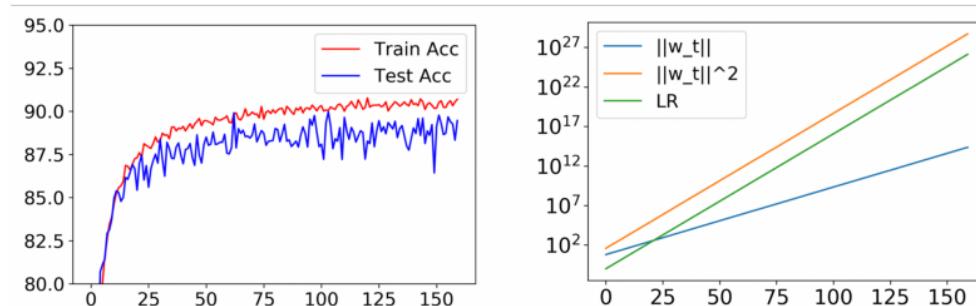
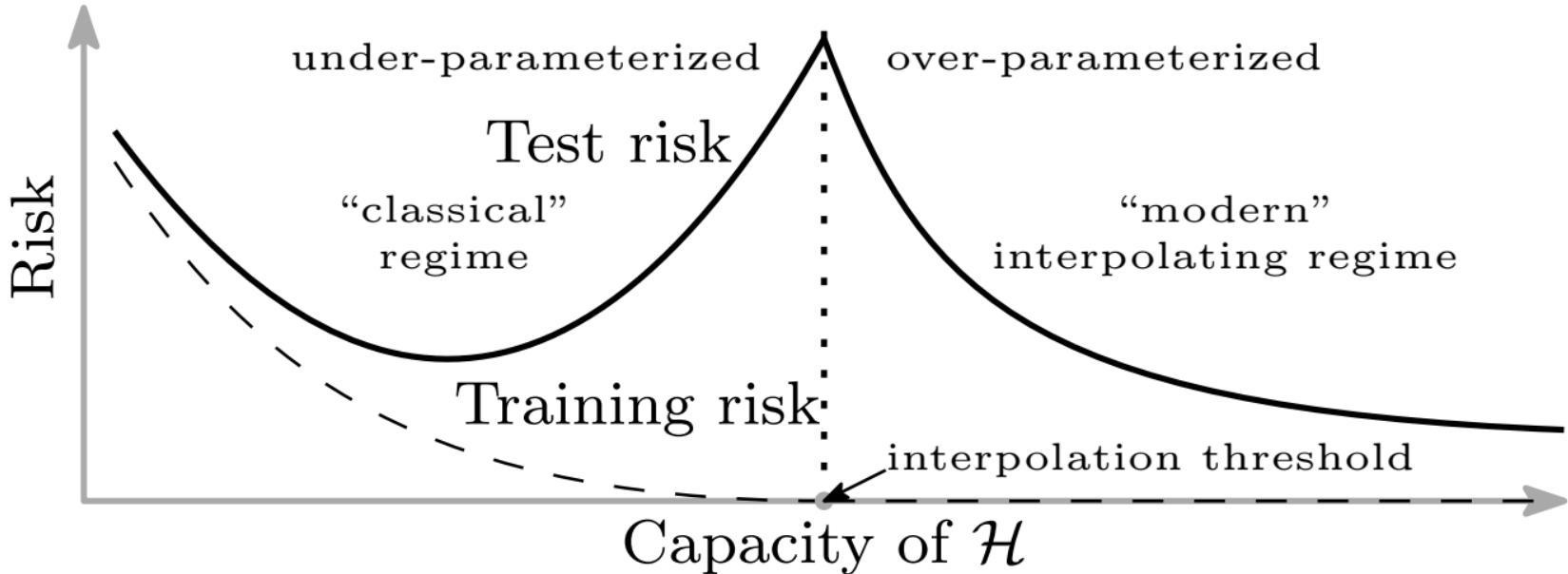


Рисунок 15. Обучение PreResNet32 на CIFAR10. Траектория с фиксированным LR и WD совпадает с ExpLR ( $\tilde{\eta}_t = 0.1 \times 1.481^t$ ) без WD. Справа: норма весов растет экспоненциально,  $|w_t|_2^2 / \tilde{\eta}_t = \text{const}$ .

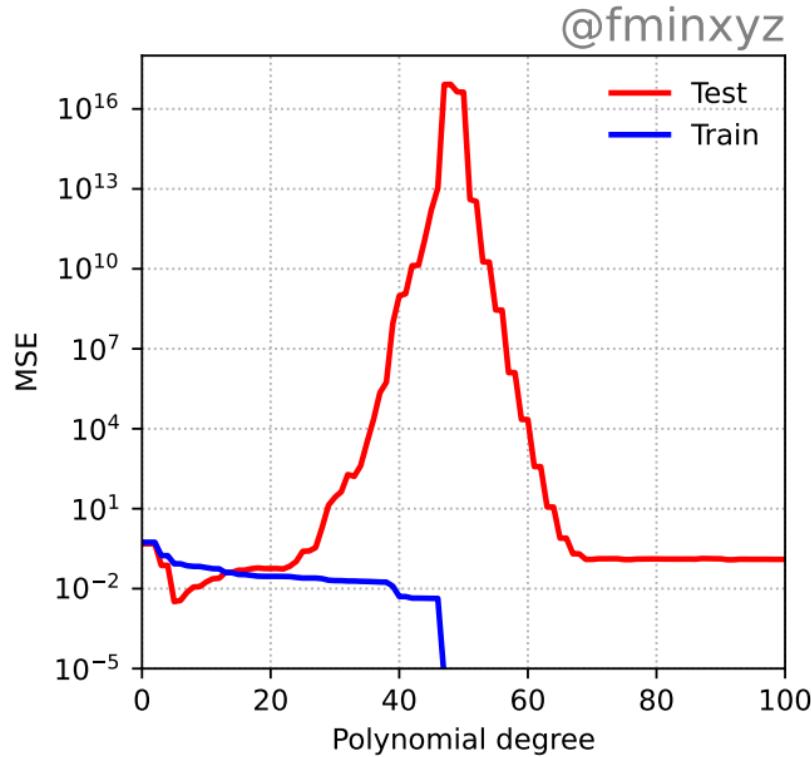
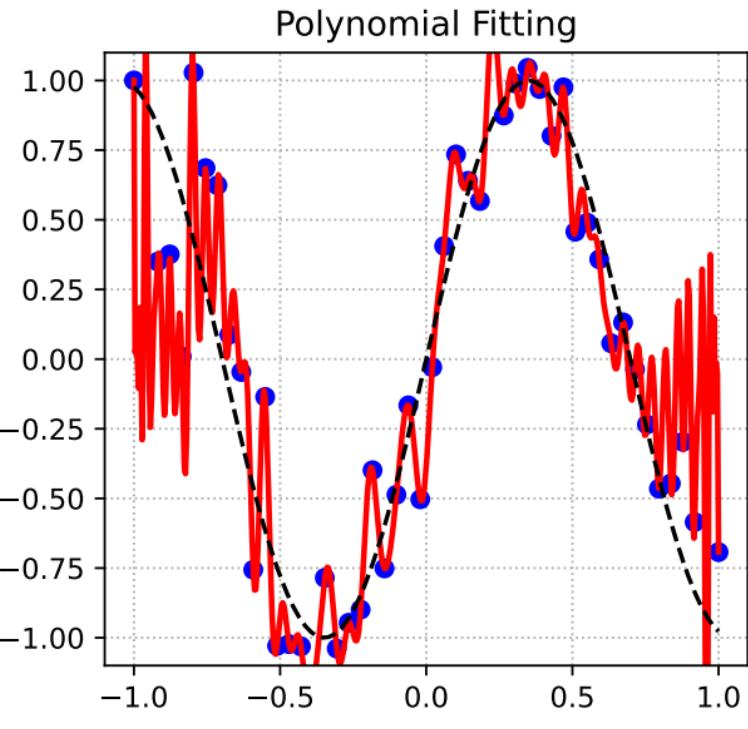
<sup>13</sup> Exponential Learning Rate Schedules for Deep Learning (2020)

# Double Descent<sup>14</sup>



<sup>14</sup>Reconciling modern machine learning practice and the bias-variance trade-off, Mikhail Belkin, Daniel Hsu, Siyuan Ma, Soumik Mandal

# Double Descent



# Бонус: доказательства

# Следствие из липшицевости градиента

Из липшицевости градиента следует:

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2} \|x_{k+1} - x_k\|^2$$

# Следствие из липшицевости градиента

Из липшицевости градиента следует:

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2} \|x_{k+1} - x_k\|^2$$

Используя (SGD):

$$f(x_{k+1}) \leq f(x_k) - \alpha_k \langle \nabla f(x_k), \nabla f_{i_k}(x_k) \rangle + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(x_k)\|^2$$

# Следствие из липшицевости градиента

Из липшицевости градиента следует:

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2} \|x_{k+1} - x_k\|^2$$

Используя (SGD):

$$f(x_{k+1}) \leq f(x_k) - \alpha_k \langle \nabla f(x_k), \nabla f_{i_k}(x_k) \rangle + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(x_k)\|^2$$

Теперь возьмем матожидание по  $i_k$ :

$$\mathbb{E}[f(x_{k+1})] \leq \mathbb{E}[f(x_k) - \alpha_k \langle \nabla f(x_k), \nabla f_{i_k}(x_k) \rangle + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(x_k)\|^2]$$

# Следствие из липшицевости градиента

Из липшицевости градиента следует:

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2} \|x_{k+1} - x_k\|^2$$

Используя (SGD):

$$f(x_{k+1}) \leq f(x_k) - \alpha_k \langle \nabla f(x_k), \nabla f_{i_k}(x_k) \rangle + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(x_k)\|^2$$

Теперь возьмем матожидание по  $i_k$ :

$$\mathbb{E}[f(x_{k+1})] \leq \mathbb{E}[f(x_k) - \alpha_k \langle \nabla f(x_k), \nabla f_{i_k}(x_k) \rangle + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(x_k)\|^2]$$

Используя линейность матожидания:

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \alpha_k \langle \nabla f(x_k), \mathbb{E}[\nabla f_{i_k}(x_k)] \rangle + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2]$$

# Следствие из липшицевости градиента

Из липшицевости градиента следует:

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2} \|x_{k+1} - x_k\|^2$$

Используя (SGD):

$$f(x_{k+1}) \leq f(x_k) - \alpha_k \langle \nabla f(x_k), \nabla f_{i_k}(x_k) \rangle + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(x_k)\|^2$$

Теперь возьмем матожидание по  $i_k$ :

$$\mathbb{E}[f(x_{k+1})] \leq \mathbb{E}[f(x_k) - \alpha_k \langle \nabla f(x_k), \nabla f_{i_k}(x_k) \rangle + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(x_k)\|^2]$$

Используя линейность матожидания:

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \alpha_k \langle \nabla f(x_k), \mathbb{E}[\nabla f_{i_k}(x_k)] \rangle + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2]$$

Так как выбор индекса равномерен, стохастический градиент несмещён:  $\mathbb{E}[\nabla f_{i_k}(x_k)] = \nabla f(x_k)$ :

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \tag{1}$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

Воспользуемся неравенством (1):

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2]$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

Воспользуемся неравенством (1):

$$\mathbb{E}[f(x_{k+1})] \leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2]$$

$$\text{PL: } \|\nabla f(x_k)\|^2 \geq 2\mu(f(x_k) - f^*)$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

Воспользуемся неравенством (1):

$$\begin{aligned} \mathbb{E}[f(x_{k+1})] &\leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{PL: } \|\nabla f(x_k)\|^2 \geq 2\mu(f(x_k) - f^*) &\leq f(x_k) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \end{aligned}$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

Воспользуемся неравенством (1):

$$\begin{aligned} \mathbb{E}[f(x_{k+1})] &\leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{PL: } \|\nabla f(x_k)\|^2 &\geq 2\mu(f(x_k) - f^*) \quad \leq f(x_k) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \end{aligned}$$

Вычтем  $f^*$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

Воспользуемся неравенством (1):

$$\begin{aligned} \mathbb{E}[f(x_{k+1})] &\leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{PL: } \|\nabla f(x_k)\|^2 \geq 2\mu(f(x_k) - f^*) &\leq f(x_k) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{Вычтем } f^* \quad \mathbb{E}[f(x_{k+1})] - f^* &\leq (f(x_k) - f^*) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \end{aligned}$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

Воспользуемся неравенством (1):

$$\begin{aligned} \mathbb{E}[f(x_{k+1})] &\leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{PL: } \|\nabla f(x_k)\|^2 \geq 2\mu(f(x_k) - f^*) &\leq f(x_k) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{Вычтем } f^* \quad \mathbb{E}[f(x_{k+1})] - f^* &\leq (f(x_k) - f^*) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{Переставляем} \quad &\leq (1 - 2\alpha_k \mu)[f(x_k) - f^*] + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \end{aligned}$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

Воспользуемся неравенством (1):

$$\begin{aligned} \mathbb{E}[f(x_{k+1})] &\leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{PL: } \|\nabla f(x_k)\|^2 &\geq 2\mu(f(x_k) - f^*) \quad \leq f(x_k) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{Вычтем } f^* \quad \mathbb{E}[f(x_{k+1})] - f^* &\leq (f(x_k) - f^*) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{Переставляем} \quad &\leq (1 - 2\alpha_k \mu)[f(x_k) - f^*] + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \end{aligned}$$

Ограниченност дисперсии:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с постоянным шагом  $\alpha < \frac{1}{2\mu}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq (1 - 2\alpha\mu)^k [f(x_0) - f^*] + \frac{L\sigma^2\alpha}{4\mu}.$$

Воспользуемся неравенством (1):

$$\begin{aligned} \mathbb{E}[f(x_{k+1})] &\leq f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{PL: } \|\nabla f(x_k)\|^2 \geq 2\mu(f(x_k) - f^*) &\leq f(x_k) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{Вычтем } f^* \quad \mathbb{E}[f(x_{k+1})] - f^* &\leq (f(x_k) - f^*) - 2\alpha_k \mu(f(x_k) - f^*) + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{Переставляем} \quad &\leq (1 - 2\alpha_k \mu)[f(x_k) - f^*] + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \\ \text{Ограниченност дисперсии: } \mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2 &\leq (1 - 2\alpha_k \mu)[f(x_k) - f^*] + \frac{L\sigma^2\alpha_k^2}{2}. \end{aligned}$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с убывающим шагом  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq \frac{L\sigma^2}{2\mu^2 k}$$

1. Рассмотрим стратегию **убывающего шага** с  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$ . Получим:

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с убывающим шагом  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq \frac{L\sigma^2}{2\mu^2 k}$$

1. Рассмотрим стратегию **убывающего шага** с  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$ . Получим:

$$1 - 2\alpha_k \mu = \frac{(k+1)^2}{(k+1)^2} - \frac{2k+1}{(k+1)^2} = \frac{k^2}{(k+1)^2}$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с убывающим шагом  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq \frac{L\sigma^2}{2\mu^2 k}$$

1. Рассмотрим стратегию **убывающего шага** с  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$ . Получим:

$$1 - 2\alpha_k \mu = \frac{(k+1)^2}{(k+1)^2} - \frac{2k+1}{(k+1)^2} = \frac{k^2}{(k+1)^2} \quad \mathbb{E}[f(x_{k+1}) - f^*] \leq \frac{k^2}{(k+1)^2} [f(x_k) - f^*] + \frac{L\sigma^2(2k+1)^2}{8\mu^2(k+1)^4}$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с убывающим шагом  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq \frac{L\sigma^2}{2\mu^2 k}$$

1. Рассмотрим стратегию **убывающего шага** с  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$ . Получим:

$$\begin{aligned} 1 - 2\alpha_k \mu &= \frac{(k+1)^2}{(k+1)^2} - \frac{2k+1}{(k+1)^2} = \frac{k^2}{(k+1)^2} & \mathbb{E}[f(x_{k+1}) - f^*] &\leq \frac{k^2}{(k+1)^2} [f(x_k) - f^*] + \frac{L\sigma^2(2k+1)^2}{8\mu^2(k+1)^4} \\ (2k+1)^2 &< (2k+2)^2 = 4(k+1)^2 & \leq \frac{k^2}{(k+1)^2} [f(x_k) - f^*] + \frac{L\sigma^2}{2\mu^2(k+1)^2} \end{aligned}$$

# Сходимость. Гладкий PL-случай

**i** Пусть  $f$  —  $L$ -гладкая функция, удовлетворяющая условию Поляка-Лоясиевича (PL) с константой  $\mu > 0$ , а дисперсия стохастического градиента ограничена:  $\mathbb{E}[\|\nabla f_i(x_k)\|^2] \leq \sigma^2$ . Тогда стохастический градиентный спуск с убывающим шагом  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$  гарантирует

$$\mathbb{E}[f(x_k) - f^*] \leq \frac{L\sigma^2}{2\mu^2 k}$$

1. Рассмотрим стратегию **убывающего шага** с  $\alpha_k = \frac{2k+1}{2\mu(k+1)^2}$ . Получим:

$$\begin{aligned} 1 - 2\alpha_k \mu &= \frac{(k+1)^2}{(k+1)^2} - \frac{2k+1}{(k+1)^2} = \frac{k^2}{(k+1)^2} & \mathbb{E}[f(x_{k+1}) - f^*] &\leq \frac{k^2}{(k+1)^2} [f(x_k) - f^*] + \frac{L\sigma^2(2k+1)^2}{8\mu^2(k+1)^4} \\ (2k+1)^2 &< (2k+2)^2 = 4(k+1)^2 & \leq \frac{k^2}{(k+1)^2} [f(x_k) - f^*] + \frac{L\sigma^2}{2\mu^2(k+1)^2} \end{aligned}$$

2. Умножим обе части на  $(k+1)^2$  и обозначим  $\delta_f(k) \equiv k^2 \mathbb{E}[f(x_k) - f^*]$ . Получим:

$$\begin{aligned} (k+1)^2 \mathbb{E}[f(x_{k+1}) - f^*] &\leq k^2 \mathbb{E}[f(x_k) - f^*] + \frac{L\sigma^2}{2\mu^2} \\ \delta_f(k+1) &\leq \delta_f(k) + \frac{L\sigma^2}{2\mu^2}. \end{aligned}$$

# Сходимость. Гладкий PL-случай

3. Просуммируем предыдущее неравенство от  $i = 0$  до  $k$  и, учитывая, что  $\delta_f(0) = 0$ , получим:

что даёт искомую скорость сходимости.

## Сходимость. Гладкий PL-случай

3. Просуммируем предыдущее неравенство от  $i = 0$  до  $k$  и, учитывая, что  $\delta_f(0) = 0$ , получим:

$$\delta_f(i+1) \leq \delta_f(i) + \frac{L\sigma^2}{2\mu^2}$$

что даёт искомую скорость сходимости.

# Сходимость. Гладкий PL-случай

3. Просуммируем предыдущее неравенство от  $i = 0$  до  $k$  и, учитывая, что  $\delta_f(0) = 0$ , получим:

$$\delta_f(i+1) \leq \delta_f(i) + \frac{L\sigma^2}{2\mu^2}$$

$$\sum_{i=0}^k [\delta_f(i+1) - \delta_f(i)] \leq \sum_{i=0}^k \frac{L\sigma^2}{2\mu^2}$$

что даёт искомую скорость сходимости.

# Сходимость. Гладкий PL-случай

3. Просуммируем предыдущее неравенство от  $i = 0$  до  $k$  и, учитывая, что  $\delta_f(0) = 0$ , получим:

$$\delta_f(i+1) \leq \delta_f(i) + \frac{L\sigma^2}{2\mu^2}$$

$$\sum_{i=0}^k [\delta_f(i+1) - \delta_f(i)] \leq \sum_{i=0}^k \frac{L\sigma^2}{2\mu^2}$$

$$\delta_f(k+1) - \delta_f(0) \leq \frac{L\sigma^2(k+1)}{2\mu^2}$$

что даёт искомую скорость сходимости.

# Сходимость. Гладкий PL-случай

3. Просуммируем предыдущее неравенство от  $i = 0$  до  $k$  и, учитывая, что  $\delta_f(0) = 0$ , получим:

$$\delta_f(i+1) \leq \delta_f(i) + \frac{L\sigma^2}{2\mu^2}$$

$$\sum_{i=0}^k [\delta_f(i+1) - \delta_f(i)] \leq \sum_{i=0}^k \frac{L\sigma^2}{2\mu^2}$$

$$\delta_f(k+1) - \delta_f(0) \leq \frac{L\sigma^2(k+1)}{2\mu^2}$$

$$(k+1)^2 \mathbb{E}[f(x_{k+1}) - f^*] \leq \frac{L\sigma^2(k+1)}{2\mu^2}$$

что даёт искомую скорость сходимости.

# Сходимость. Гладкий PL-случай

3. Просуммируем предыдущее неравенство от  $i = 0$  до  $k$  и, учитывая, что  $\delta_f(0) = 0$ , получим:

$$\delta_f(i+1) \leq \delta_f(i) + \frac{L\sigma^2}{2\mu^2}$$

$$\sum_{i=0}^k [\delta_f(i+1) - \delta_f(i)] \leq \sum_{i=0}^k \frac{L\sigma^2}{2\mu^2}$$

$$\delta_f(k+1) - \delta_f(0) \leq \frac{L\sigma^2(k+1)}{2\mu^2}$$

$$(k+1)^2 \mathbb{E}[f(x_{k+1}) - f^*] \leq \frac{L\sigma^2(k+1)}{2\mu^2}$$

$$\mathbb{E}[f(x_k) - f^*] \leq \frac{L\sigma^2}{2\mu^2 k}$$

что даёт искомую скорость сходимости.

# Сходимость. Гладкий выпуклый случай с постоянным шагом

**i** Пусть  $f$  — выпуклая функция (не обязательно гладкая), а дисперсия стохастического градиента ограничена  $\mathbb{E}[\|\nabla f_{i_k}(x_k)\|^2] \leq \sigma^2 \quad \forall k$ . Если SGD использует постоянный шаг  $\alpha_t \equiv \alpha > 0$ , то для любого  $k \geq 1$

$$\mathbb{E}[f(\bar{x}_k) - f^*] \leq \frac{\|x_0 - x^*\|^2}{2\alpha k} + \frac{\alpha \sigma^2}{2}$$

где  $\bar{x}_k = \frac{1}{k} \sum_{t=0}^{k-1} x_t$ .

При выборе постоянного  $\alpha = \frac{\|x_0 - x^*\|}{\sigma\sqrt{k}}$  (зависящего от  $k$ ) имеем

$$\mathbb{E}[f(\bar{x}_k) - f^*] \leq \frac{\|x_0 - x^*\|\sigma}{\sqrt{k}} = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right).$$

# Сходимость. Гладкий выпуклый случай с постоянным шагом

1. Начнём с разложения квадрата расстояния до минимума:

$$\|x_{k+1} - x^*\|^2 = \|x_k - \alpha \nabla f_{i_k}(x_k) - x^*\|^2 = \|x_k - x^*\|^2 - 2\alpha \langle \nabla f_{i_k}(x_k), x_k - x^* \rangle + \alpha^2 \|\nabla f_{i_k}(x_k)\|^2.$$

# Сходимость. Гладкий выпуклый случай с постоянным шагом

- Начнём с разложения квадрата расстояния до минимума:

$$\|x_{k+1} - x^*\|^2 = \|x_k - \alpha \nabla f_{i_k}(x_k) - x^*\|^2 = \|x_k - x^*\|^2 - 2\alpha \langle \nabla f_{i_k}(x_k), x_k - x^* \rangle + \alpha^2 \|\nabla f_{i_k}(x_k)\|^2.$$

- Берём условное матожидание по  $i_k$  (обозначим  $\mathbb{E}_k[\cdot] = \mathbb{E}[\cdot | x_k]$ ), используем свойство  $\mathbb{E}_k[\nabla f_{i_k}(x_k)] = \nabla f(x_k)$ , ограниченность дисперсии  $\mathbb{E}_k[\|\nabla f_{i_k}(x_k)\|^2] \leq \sigma^2$  и выпуклость  $f$  (которая даёт  $\langle \nabla f(x_k), x_k - x^* \rangle \geq f(x_k) - f^*$ ):

$$\begin{aligned}\mathbb{E}_k[\|x_{k+1} - x^*\|^2] &= \|x_k - x^*\|^2 - 2\alpha \langle \nabla f(x_k), x_k - x^* \rangle + \alpha^2 \mathbb{E}_k[\|\nabla f_{i_k}(x_k)\|^2] \\ &\leq \|x_k - x^*\|^2 - 2\alpha(f(x_k) - f^*) + \alpha^2 \sigma^2.\end{aligned}$$



# Сходимость. Гладкий выпуклый случай с постоянным шагом

- Начнём с разложения квадрата расстояния до минимума:

$$\|x_{k+1} - x^*\|^2 = \|x_k - \alpha \nabla f_{i_k}(x_k) - x^*\|^2 = \|x_k - x^*\|^2 - 2\alpha \langle \nabla f_{i_k}(x_k), x_k - x^* \rangle + \alpha^2 \|\nabla f_{i_k}(x_k)\|^2.$$

- Берём условное матожидание по  $i_k$  (обозначим  $\mathbb{E}_k[\cdot] = \mathbb{E}[\cdot|x_k]$ ), используем свойство  $\mathbb{E}_k[\nabla f_{i_k}(x_k)] = \nabla f(x_k)$ , ограниченность дисперсии  $\mathbb{E}_k[\|\nabla f_{i_k}(x_k)\|^2] \leq \sigma^2$  и выпукłość  $f$  (которая даёт  $\langle \nabla f(x_k), x_k - x^* \rangle \geq f(x_k) - f^*$ ):

$$\begin{aligned}\mathbb{E}_k[\|x_{k+1} - x^*\|^2] &= \|x_k - x^*\|^2 - 2\alpha \langle \nabla f(x_k), x_k - x^* \rangle + \alpha^2 \mathbb{E}_k[\|\nabla f_{i_k}(x_k)\|^2] \\ &\leq \|x_k - x^*\|^2 - 2\alpha(f(x_k) - f^*) + \alpha^2 \sigma^2.\end{aligned}$$

- Переносим слагаемое с  $f(x_k)$  влево и берём полное матожидание:

$$2\alpha \mathbb{E}[f(x_k) - f^*] \leq \mathbb{E}[\|x_k - x^*\|^2] - \mathbb{E}[\|x_{k+1} - x^*\|^2] + \alpha^2 \sigma^2.$$

# Сходимость. Гладкий выпуклый случай с постоянным шагом

1. Начнём с разложения квадрата расстояния до минимума:

$$\|x_{k+1} - x^*\|^2 = \|x_k - \alpha \nabla f_{i_k}(x_k) - x^*\|^2 = \|x_k - x^*\|^2 - 2\alpha \langle \nabla f_{i_k}(x_k), x_k - x^* \rangle + \alpha^2 \|\nabla f_{i_k}(x_k)\|^2.$$

2. Берём условное матожидание по  $i_k$  (обозначим  $\mathbb{E}_k[\cdot] = \mathbb{E}[\cdot|x_k]$ ), используем свойство  $\mathbb{E}_k[\nabla f_{i_k}(x_k)] = \nabla f(x_k)$ , ограниченность дисперсии  $\mathbb{E}_k[\|\nabla f_{i_k}(x_k)\|^2] \leq \sigma^2$  и выпукłość  $f$  (которая даёт  $\langle \nabla f(x_k), x_k - x^* \rangle \geq f(x_k) - f^*$ ):

$$\begin{aligned}\mathbb{E}_k[\|x_{k+1} - x^*\|^2] &= \|x_k - x^*\|^2 - 2\alpha \langle \nabla f(x_k), x_k - x^* \rangle + \alpha^2 \mathbb{E}_k[\|\nabla f_{i_k}(x_k)\|^2] \\ &\leq \|x_k - x^*\|^2 - 2\alpha(f(x_k) - f^*) + \alpha^2 \sigma^2.\end{aligned}$$

3. Переносим слагаемое с  $f(x_k)$  влево и берём полное матожидание:

$$2\alpha \mathbb{E}[f(x_k) - f^*] \leq \mathbb{E}[\|x_k - x^*\|^2] - \mathbb{E}[\|x_{k+1} - x^*\|^2] + \alpha^2 \sigma^2.$$

4. Суммируем (теслекопирем) по  $t = 0, \dots, k-1$ :

$$\begin{aligned}\sum_{t=0}^{k-1} 2\alpha \mathbb{E}[f(x_t) - f^*] &\leq \sum_{t=0}^{k-1} (\mathbb{E}[\|x_t - x^*\|^2] - \mathbb{E}[\|x_{t+1} - x^*\|^2]) + \sum_{t=0}^{k-1} \alpha^2 \sigma^2 \\ &= \mathbb{E}[\|x_0 - x^*\|^2] - \mathbb{E}[\|x_k - x^*\|^2] + k \alpha^2 \sigma^2 \\ &\leq \|x_0 - x^*\|^2 + k \alpha^2 \sigma^2.\end{aligned}$$

# Сходимость. Гладкий выпуклый случай с постоянным шагом

5. Делим на  $2\alpha k$ :

$$\frac{1}{k} \sum_{t=0}^{k-1} \mathbb{E}[f(x_t) - f^*] \leq \frac{\|x_0 - x^*\|^2}{2\alpha k} + \frac{\alpha\sigma^2}{2}.$$

# Сходимость. Гладкий выпуклый случай с постоянным шагом

5. Делим на  $2\alpha k$ :

$$\frac{1}{k} \sum_{t=0}^{k-1} \mathbb{E}[f(x_t) - f^*] \leq \frac{\|x_0 - x^*\|^2}{2\alpha k} + \frac{\alpha\sigma^2}{2}.$$

6. Воспользуемся выпуклостью  $f$  и неравенством Йенсена для усреднённой точки  $\bar{x}_k = \frac{1}{k} \sum_{t=0}^{k-1} x_t$ :

$$\mathbb{E}[f(\bar{x}_k)] \leq \mathbb{E}\left[\frac{1}{k} \sum_{t=0}^{k-1} f(x_t)\right] = \frac{1}{k} \sum_{t=0}^{k-1} \mathbb{E}[f(x_t)].$$

Вычитая  $f^*$  из обеих частей, получаем:

$$\mathbb{E}[f(\bar{x}_k) - f^*] \leq \frac{1}{k} \sum_{t=0}^{k-1} \mathbb{E}[f(x_t) - f^*].$$



# Сходимость. Гладкий выпуклый случай с постоянным шагом

5. Делим на  $2\alpha k$ :

$$\frac{1}{k} \sum_{t=0}^{k-1} \mathbb{E}[f(x_t) - f^*] \leq \frac{\|x_0 - x^*\|^2}{2\alpha k} + \frac{\alpha\sigma^2}{2}.$$

6. Воспользуемся выпуклостью  $f$  и неравенством Йенсена для усреднённой точки  $\bar{x}_k = \frac{1}{k} \sum_{t=0}^{k-1} x_t$ :

$$\mathbb{E}[f(\bar{x}_k)] \leq \mathbb{E}\left[\frac{1}{k} \sum_{t=0}^{k-1} f(x_t)\right] = \frac{1}{k} \sum_{t=0}^{k-1} \mathbb{E}[f(x_t)].$$

Вычитая  $f^*$  из обеих частей, получаем:

$$\mathbb{E}[f(\bar{x}_k) - f^*] \leq \frac{1}{k} \sum_{t=0}^{k-1} \mathbb{E}[f(x_t) - f^*].$$

7. Объединяя (5) и (6), получаем искомую оценку:

$$\mathbb{E}[f(\bar{x}_k) - f^*] \leq \frac{\|x_0 - x^*\|^2}{2\alpha k} + \frac{\alpha\sigma^2}{2}.$$