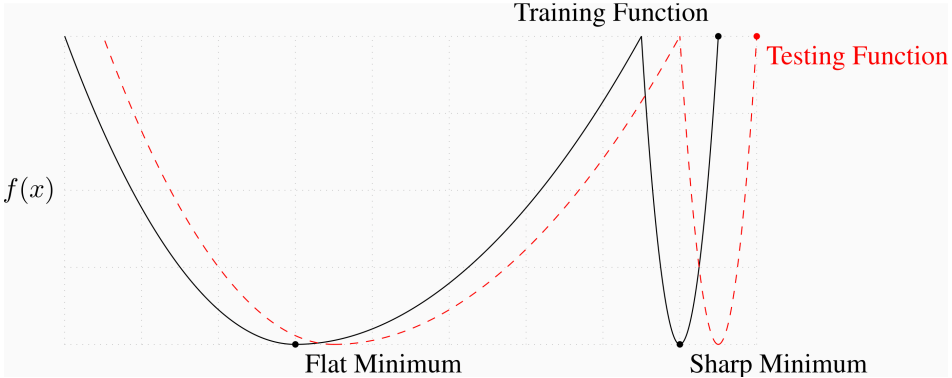
Оптимизация нейронных сетей: Sharpness-Aware Minimization, Mode Connectivity, Grokking, Double Descent.

Семинар

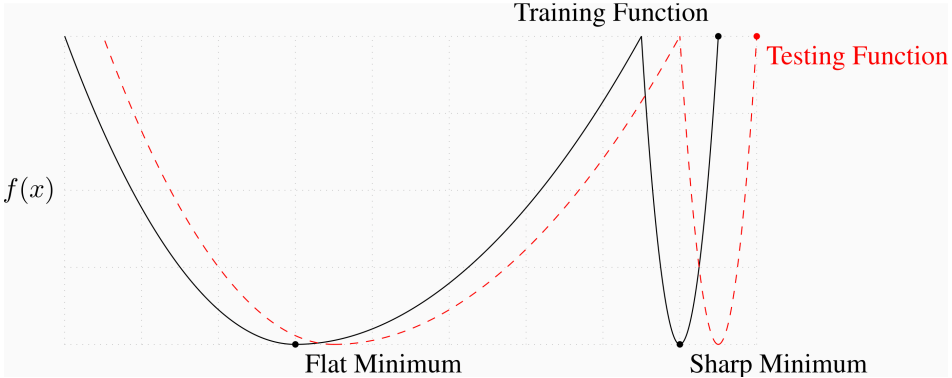
Оптимизация для всех! ЦУ

SAM

Плоский минимум vs Острый минимум



Плоский минимум vs Острый минимум



Question

Что не так с острым минимумом?

Sharpness-Aware Minimization [1]

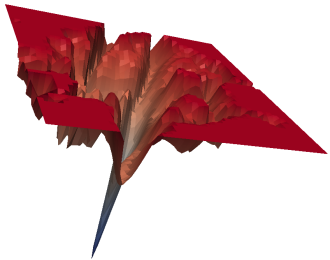


Рис. 1: Острый минимум, к которому сошлась ResNet, обученная с помощью SGD.

[1] Foret, Pierre, et al. "Sharpness-aware minimization for efficiently improving generalization." (2020)

Sharpness-Aware Minimization [1]

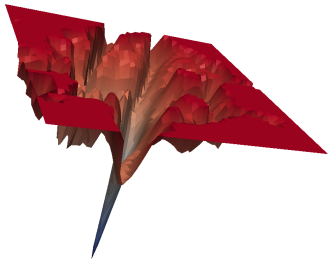


Рис. 1: Острый минимум, к которому сошлась ResNet, обученная с помощью SGD.

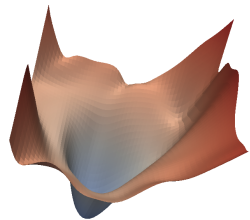


Рис. 2: Широкий минимум, к которому сошлась та же ResNet, обученная с помощью SAM.

[1] Foret, Pierre, et al. "Sharpness-aware minimization for efficiently improving generalization." (2020)

Sharpness-Aware Minimization [1]

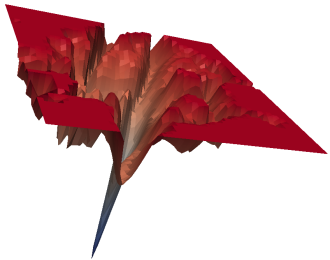


Рис. 1: Острый минимум, к которому сошлась ResNet, обученная с помощью SGD.

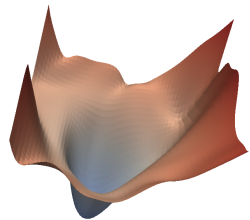


Рис. 2: Широкий минимум, к которому сошлась та же ResNet, обученная с помощью SAM.

Sharpness-Aware Minimization (SAM) — это процедура, целью которой является улучшение обобщающей способности модели путем одновременной минимизации значения функции потерь и **остроты (sharpness) функции потерь**.

[1] Foret, Pierre, et al. "Sharpness-aware minimization for efficiently improving generalization." (2020)

Постановка задачи обучения

Обучающая выборка, полученная $i.i.d.$ из распределения D:

$$S = \{(x_i, y_i)\}_{i=1}^{n},$$

где x_i — вектор признаков, а y_i — метка.

Постановка задачи обучения

Обучающая выборка, полученная $i.i.d.$ из распределения D:

$$S = \{(x_i, y_i)\}_{i=1}^{n},$$

где x_i — вектор признаков, а y_i — метка.

Функция потерь на обучающей выборке:

$$L_S = \frac{1}{n} \sum_{i=1}^{n} l(w, x_i, y_i),$$

где l — функция потерь для одного объекта, w — параметры.

Постановка задачи обучения

Обучающая выборка, полученная $i.i.d.$ из распределения D:

$$S = \{(x_i, y_i)\}_{i=1}^{n},$$

где x_i — вектор признаков, а y_i — метка.

Функция потерь на обучающей выборке:

$$L_S = \frac{1}{n} \sum_{i=1}^{n} l(w, x_i, y_i),$$

где l — функция потерь для одного объекта, w — параметры.

Функция потерь на генеральной совокупности (population loss):

$$L_D = \mathbb{E}_{(x,y)}[l(w, x, y)]$$

Что такое sharpness (острота)?

Theorem

Для любого $\rho > 0$, с высокой вероятностью по обучающей выборке S, сгенерированной из распределения D,

$$L_D(w) \leq \max_{\|\epsilon\|_2 \leq \rho} L_S(w + \epsilon) + h\left(\|w\|_2^2 / \rho^2\right),$$

где $h : \mathbb{R}_+ \to \mathbb{R}_+$ — строго возрастающая функция (при некоторых технических условиях на $L_D(w)$).

Что такое sharpness (острота)?

Theorem

Для любого $\rho > 0$, с высокой вероятностью по обучающей выборке S, сгенерированной из распределения D,

$$L_D(w) \leq \max_{\|\epsilon\|_2 \leq \rho} L_S(w+\epsilon) + h\left(\|w\|_2^2/\rho^2\right),$$

где $h : \mathbb{R}_+ \to \mathbb{R}_+$ — строго возрастающая функция (при некоторых технических условиях на $L_D(w)$).

Добавляя и вычитая $L_S(w)$:

$$\left[\max_{\|\epsilon\|_2 \leq \rho} L_S(w+\epsilon) - L_S(w)\right] + L_S(w) + h\left(\|w\|_2^2/\rho^2\right)$$

Слагаемое в квадратных скобках отражает **остроту (sharpness)** L_S в точке w, измеряя, как быстро может возрасти ошибка обучения при переходе от w к близкому значению параметров.

Sharpness-Aware Minimization

Функция h заменяется на более простую константу λ. Авторы предлагают выбирать значения параметров, решая следующую задачу Sharpness-Aware Minimization (SAM):

$$\min_w L_S^{SAM}(w) + \lambda \|w\|_2^2 \quad \text{где} \quad L_S^{SAM}(w) \triangleq \max_{\|\epsilon\|_p \leq \rho} L_S(w+\epsilon),$$

с гиперпараметром $\rho \geq 0$ и p из $[1, \infty]$ (небольшое обобщение, хотя $p = 2$ эмпирически является лучшим выбором).

Как минимизировать L_S^{SAM}?

Для минимизации L_S^{SAM} используется эффективная аппроксимация его градиента. Первым шагом рассматривается разложение Тейлора первого порядка для $L_S(w+\epsilon)$:

$$\epsilon^*(w) \triangleq \underset{\|\epsilon\|_p \leq \rho}{\arg\max} \{L_S(w+\epsilon)\} \approx \underset{\|\epsilon\|_p \leq \rho}{\arg\max} \left\{L_S(w) + \epsilon^T \nabla_w L_S(w)\right\} = \underset{\|\epsilon\|_p \leq \rho}{\arg\max} \left\{\epsilon^T \nabla_w L_S(w)\right\}.$$

Как минимизировать L_S^{SAM}?

Для минимизации L_S^{SAM} используется эффективная аппроксимация его градиента. Первым шагом рассматривается разложение Тейлора первого порядка для $L_S(w+\epsilon)$:

$$\epsilon^*(w) \triangleq \underset{\|\epsilon\|_p \leq \rho}{\arg\max} \left\{ L_S(w+\epsilon) \right\} \approx \underset{\|\epsilon\|_p \leq \rho}{\arg\max} \left\{ L_S(w) + \epsilon^T \nabla_w L_S(w) \right\} = \underset{\|\epsilon\|_p \leq \rho}{\arg\max} \left\{ \epsilon^T \nabla_w L_S(w) \right\}.$$

Последнее выражение — это просто argmax скалярного произведения векторов ϵ и $\nabla_w L_S(w)$, и хорошо известно, какой аргумент его максимизирует:

$$\hat{\epsilon}(w) = \rho \, \mathrm{sign}\left(\nabla_w L_S(w)\right) \left|\nabla_w L_S(w)\right|^{q-1} / \left(\|\nabla_w L_S(w)\|_q^q \right)^{1/p},$$

где $1/p + 1/q = 1$.

Как минимизировать L_S^{SAM}?

Для минимизации L_S^{SAM} используется эффективная аппроксимация его градиента. Первым шагом рассматривается разложение Тейлора первого порядка для $L_S(w+\epsilon)$:

$$\epsilon^*(w) \triangleq \underset{\|\epsilon\|_p \le \rho}{\arg\max} \{L_S(w+\epsilon)\} \approx \underset{\|\epsilon\|_p \le \rho}{\arg\max} \left\{L_S(w) + \epsilon^T \nabla_w L_S(w)\right\} = \underset{\|\epsilon\|_p \le \rho}{\arg\max} \left\{\epsilon^T \nabla_w L_S(w)\right\}.$$

Последнее выражение — это просто argmax скалярного произведения векторов ϵ и $\nabla_w L_S(w)$, и хорошо известно, какой аргумент его максимизирует:

$$\hat{\epsilon}(w) = \rho \, \text{sign}\left(\nabla_w L_S(w)\right) \left|\nabla_w L_S(w)\right|^{q-1} / \left(\|\nabla_w L_S(w)\|_q^q\right)^{1/p},$$

где $1/p + 1/q = 1$.

Таким образом

$$\nabla_w L_S^{SAM}(w) \approx \nabla_w L_S(w+\hat{\epsilon}(w)) = \left.\frac{d(w+\hat{\epsilon}(w))}{dw} \nabla_w L_S(w)\right|_{w+\hat{\epsilon}(w)}$$

$$= \left.\nabla_w L_S(w)\right|_{w+\hat{\epsilon}(w)} + \left.\frac{d\hat{\epsilon}(w)}{dw} \nabla_w L_S(w)\right|_{w+\hat{\epsilon}(w)}$$

Sharpness-Aware Minimization

Современные фреймворки могут легко вычислить предыдущее приближение. Однако для ускорения вычислений члены второго порядка можно отбросить, получая:

$$\nabla_w L_S^{SAM}(w) \approx \nabla_w L_S(w)\big|_{w+\hat{\epsilon}(w)}$$

Sharpness-Aware Minimization

Современные фреймворки могут легко вычислить предыдущее приближение. Однако для ускорения вычислений члены второго порядка можно отбросить, получая:

$$\nabla_w L_S^{SAM}(w) \approx \nabla_w L_S(w)\big|_{w+\hat{\epsilon}(w)}$$

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Algorithm 1: SAM algorithm

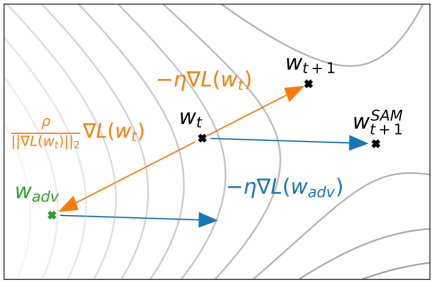


Figure 2: Schematic of the SAM parameter update.

Результаты SAM

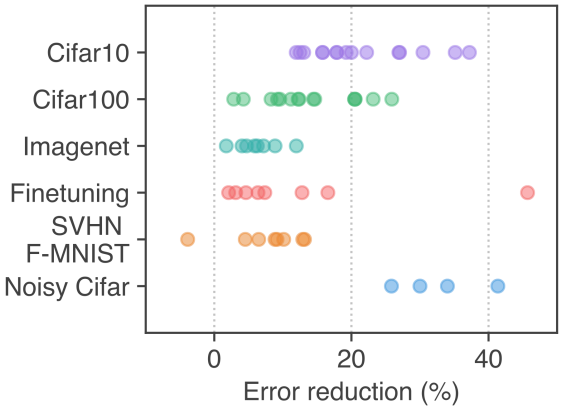


Рис. 4: Снижение частоты ошибок, полученное при переходе на SAM. Каждая точка соответствует отдельному набору данных / модели / аугментации данных.

Связность мод (Mode Connectivity)

Связность мод (Mode Connectivity) [2]

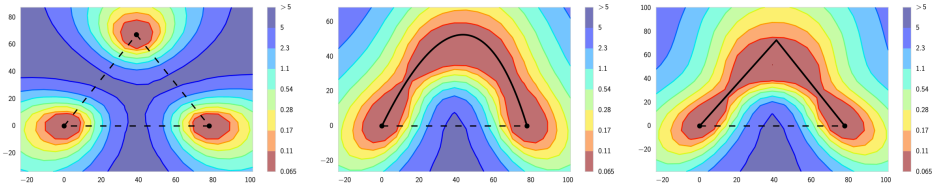


Рис. 5: Поверхность функции потерь (кросс-энтропия с l_2-регуляризацией) для ResNet-164 на CIFAR-100 как функция весов сети в двумерном подпространстве. На каждом графике горизонтальная ось фиксирована и проходит через оптимумы двух независимо обученных сетей. Вертикальная ось меняется между графиками при смене плоскостей (определенных в основном тексте). Слева: Три оптимума для независимо обученных сетей. В центре и справа: Квадратичная кривая Безье и ломаная с одним изгибом, соединяющие два нижних оптимума с левого графика вдоль пути с почти постоянной функцией потерь. Заметьте, что на каждом графике прямой линейный путь между модами привел бы к высоким потерям.

[2] Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. Advances in neural information processing systems, 31.

Процедура поиска кривой

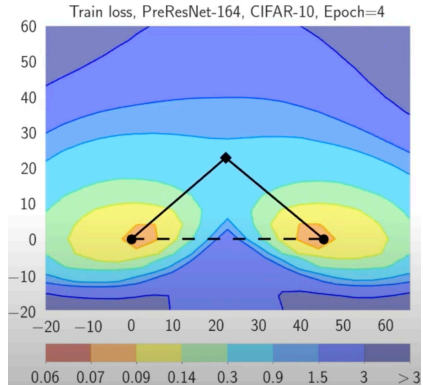
- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

$$\mathcal{L}(w)$$

$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t\sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$



Процедура поиска кривой

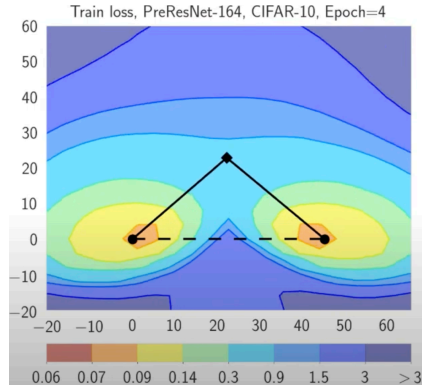
- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
  $\phi_\theta(\cdot) : [0, 1] \to \mathbb{R}^{|\text{net}|}$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

$$\mathcal{L}(w)$$



$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$

Процедура поиска кривой

- Веса предобученных сетей:

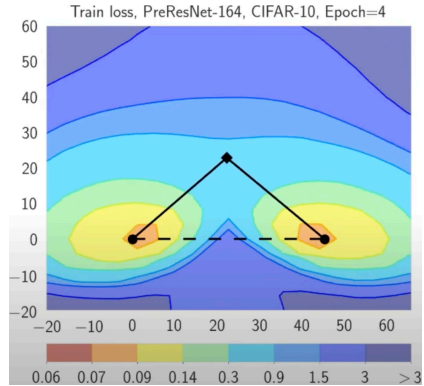
$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
$\phi_\theta(\cdot) : [0, 1] \to \mathbb{R}^{|\text{net}|}$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

- Функция потерь DNN:

$$\mathcal{L}(w)$$



$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$

Процедура поиска кривой

- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
$\phi_\theta(\cdot) : [0, 1] \to \mathbb{R}^{|\text{net}|}$

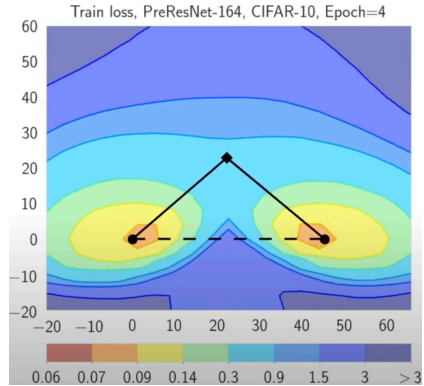
$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

- Функция потерь DNN:

$$\mathcal{L}(w)$$

- Минимизируем усредненную функцию потерь по θ:

$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$



Процедура поиска кривой

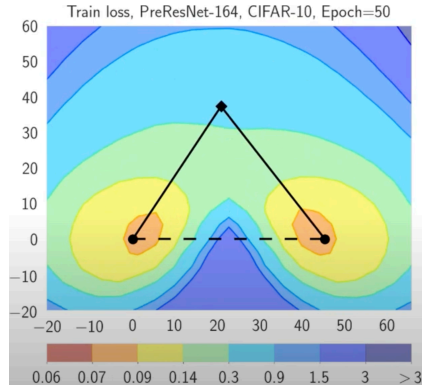
- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

$$\mathcal{L}(w)$$

$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t\sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$



Процедура поиска кривой

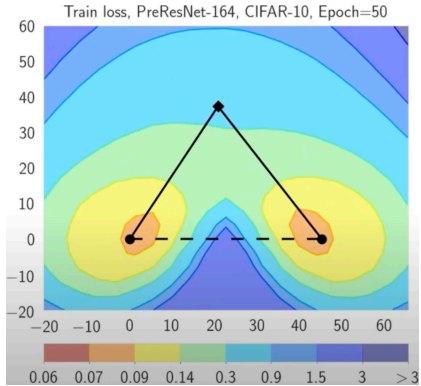
- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
$\phi_\theta(\cdot) : [0, 1] \to \mathbb{R}^{|\text{net}|}$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

$$\mathcal{L}(w)$$



$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$

Процедура поиска кривой

- Веса предобученных сетей:

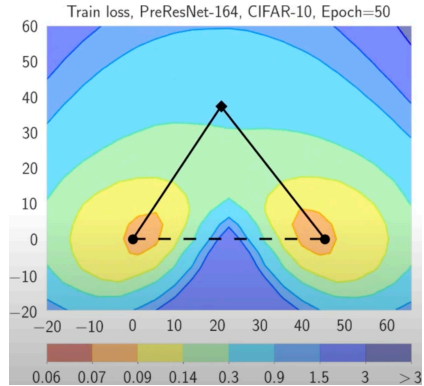
$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
$\phi_\theta(\cdot) : [0, 1] \to \mathbb{R}^{|\text{net}|}$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

- Функция потерь DNN:

$$\mathcal{L}(w)$$



$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$

Процедура поиска кривой

- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
$\phi_\theta(\cdot) : [0, 1] \to \mathbb{R}^{|\text{net}|}$

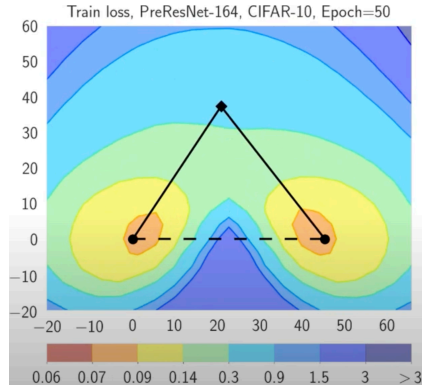
$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

- Функция потерь DNN:

$$\mathcal{L}(w)$$

- Минимизируем усредненную функцию потерь по θ:

$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$



Процедура поиска кривой

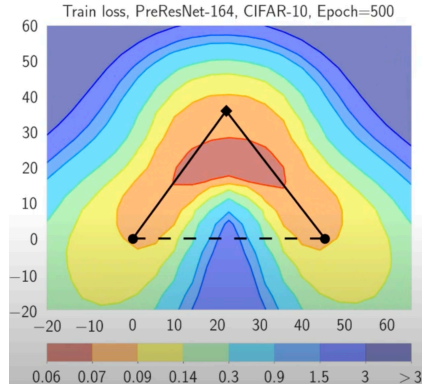
- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

$$\mathcal{L}(w)$$

$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t\sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$



Процедура поиска кривой

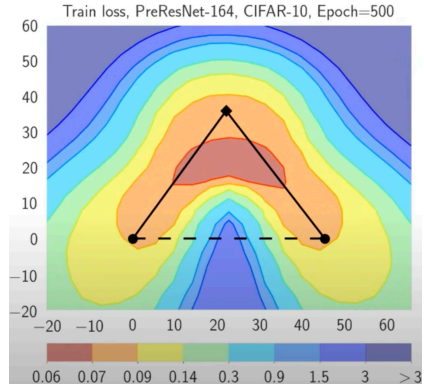
- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
  $\phi_\theta(\cdot) : [0, 1] \to \mathbb{R}^{|\text{net}|}$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

$$\mathcal{L}(w)$$



$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$

Процедура поиска кривой

- Веса предобученных сетей:

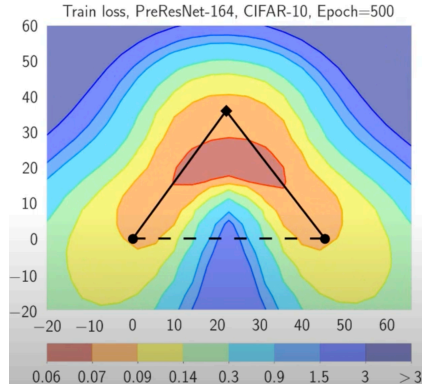
$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
  $\phi_\theta(\cdot) : [0,1] \to \mathbb{R}^{|\text{net}|}$

$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

- Функция потерь DNN:

$$\mathcal{L}(w)$$



$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t\sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$

Процедура поиска кривой

- Веса предобученных сетей:

$$\widehat{w}_1, \widehat{w}_2 \in \mathbb{R}^{|\text{net}|}$$

- Определим параметрическую кривую:
$\phi_\theta(\cdot) : [0, 1] \to \mathbb{R}^{|\text{net}|}$

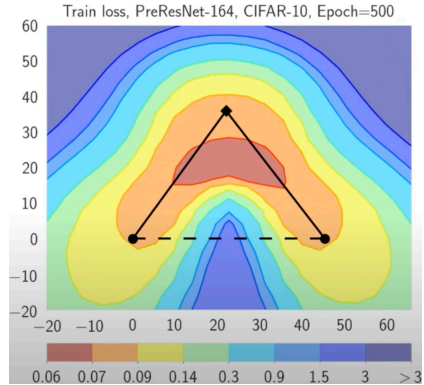
$$\phi_\theta(0) = \widehat{w}_1, \quad \phi_\theta(1) = \widehat{w}_2$$

- Функция потерь DNN:

$$\mathcal{L}(w)$$

- Минимизируем усредненную функцию потерь по θ:

$$\underset{\theta}{\text{minimize}} \;\; \ell(\theta) = \int_0^1 \mathcal{L}\left(\phi_\theta(t)\right) dt = \mathbb{E}_{t \sim U(0,1)} \mathcal{L}\left(\phi_\theta(t)\right)$$



Гроккинг (Grokking)

Гроккинг (Grokking)[3]

- После достижения нулевой ошибки на обучении веса продолжают изменяться в манере, напоминающей случайное блуждание

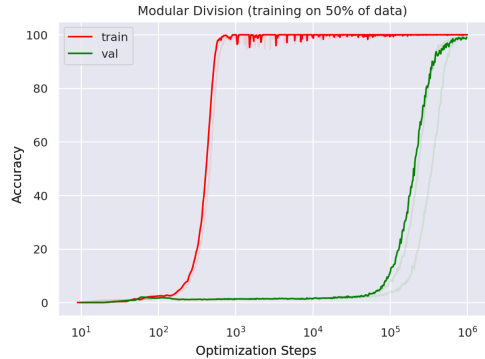


Рис. 6: Гроккинг: Яркий пример обобщения, наступающего намного позже переобучения на алгоритмическом наборе данных.

Гроккинг (Grokking)[3]

- После достижения нулевой ошибки на обучении веса продолжают изменяться в манере, напоминающей случайное блуждание
- Возможно, они медленно дрейфуют к более широкому минимуму

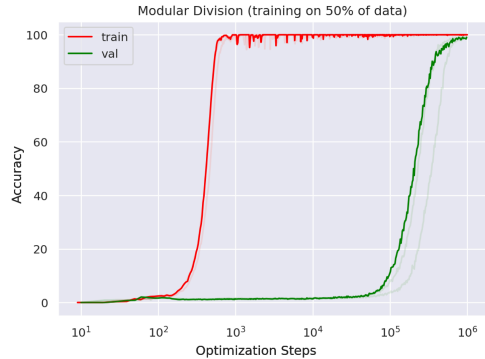


Рис. 6: Гроккинг: Яркий пример обобщения, наступающего намного позже переобучения на алгоритмическом наборе данных.

Гроккинг (Grokking)[3]

- После достижения нулевой ошибки на обучении веса продолжают изменяться в манере, напоминающей случайное блуждание
- Возможно, они медленно дрейфуют к более широкому минимуму
- Недавно открытый эффект гроккинга подтверждает эту гипотезу

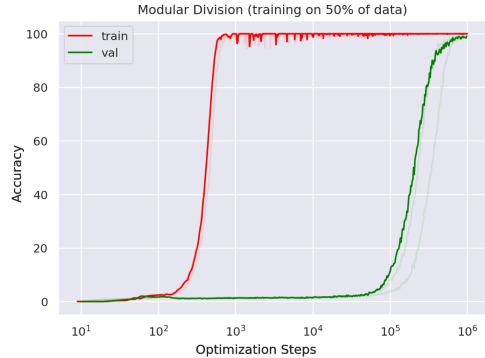


Рис. 6: Гроккинг: Яркий пример обобщения, наступающего намного позже переобучения на алгоритмическом наборе данных.

[3]Power, Alethea, et al. "Grokking: Generalization beyond overfitting on small algorithmic datasets." (2022).

Двойной спуск (Double Descent)

Двойной спуск (Double Descent)[4]

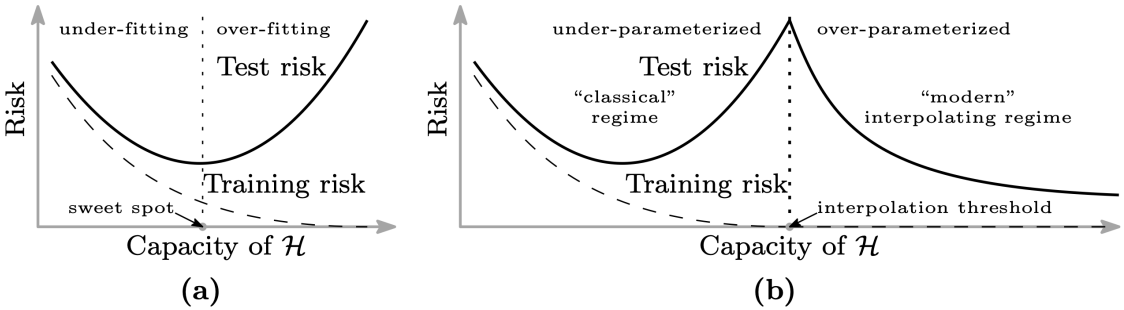
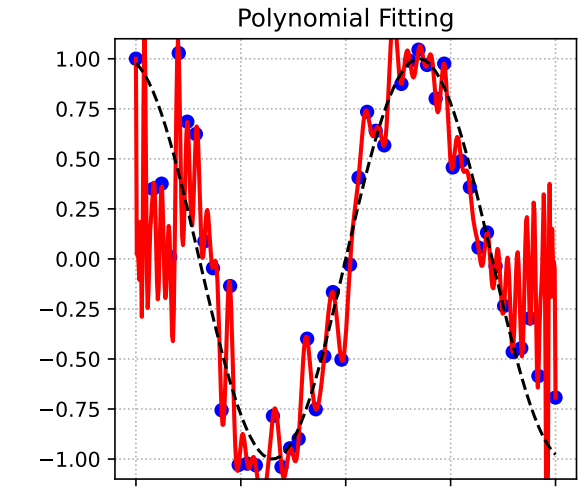
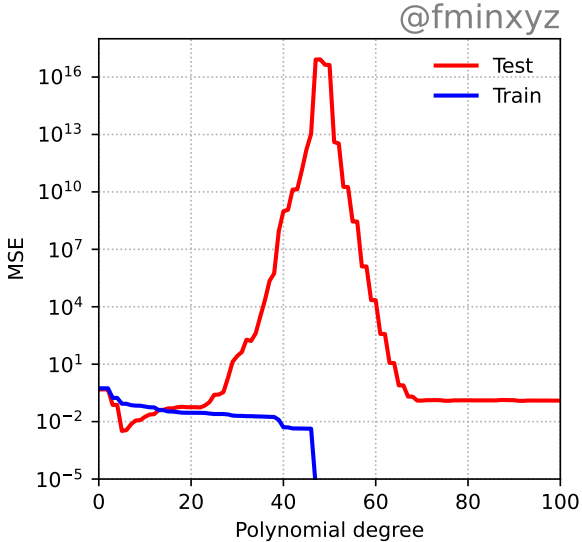


Рис. 7: Кривые риска обучения (пунктирная линия) и тестового риска (сплошная линия). (a) Классическая U-образная кривая риска, возникающая из компромисса смещения и дисперсии (bias-variance trade-off). (b) Кривая риска двойного спуска, которая объединяет U-образную кривую риска (т.е. «классический» режим) с наблюдаемым поведением на моделях высокой размерности (т.е. «современный» интерполяционный режим), разделенных порогом интерполяции. Предикторы справа от порога интерполяции имеют нулевой риск обучения.

[4] Belkin, Mikhail, et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off." (2019)

Двойной спуск (Double Descent)





Shampoo и Muon

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.
2. Обновить статистики $L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T$ и $R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k$.

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.
2. Обновить статистики $L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T$ и $R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k$.
3. Вычислить предобуславливатели $P_L = L_k^{-1/4}$ и $P_R = R_k^{-1/4}$. (Обратный матричный корень)

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.
2. Обновить статистики $L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T$ и $R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k$.
3. Вычислить предобуславливатели $P_L = L_k^{-1/4}$ и $P_R = R_k^{-1/4}$. (Обратный матричный корень)
4. Обновить: $W_{k+1} = W_k - \alpha P_L G_k P_R$.

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.
2. Обновить статистики $L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T$ и $R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k$.
3. Вычислить предобуславливатели $P_L = L_k^{-1/4}$ и $P_R = R_k^{-1/4}$. (Обратный матричный корень)
4. Обновить: $W_{k+1} = W_k - \alpha P_L G_k P_R$.

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.
2. Обновить статистики $L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T$ и $R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k$.
3. Вычислить предобуславливатели $P_L = L_k^{-1/4}$ и $P_R = R_k^{-1/4}$. (Обратный матричный корень)
4. Обновить: $W_{k+1} = W_k - \alpha P_L G_k P_R$.

Заметки:

- Нацелен на более эффективный учет информации о кривизне, чем методы первого порядка.

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.
2. Обновить статистики $L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T$ и $R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k$.
3. Вычислить предобуславливатели $P_L = L_k^{-1/4}$ и $P_R = R_k^{-1/4}$. (Обратный матричный корень)
4. Обновить: $W_{k+1} = W_k - \alpha P_L G_k P_R$.

Заметки:

- Нацелен на более эффективный учет информации о кривизне, чем методы первого порядка.
- Вычислительно дороже, чем Adam, но может сходиться быстрее или к лучшим решениям с точки зрения количества шагов.

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.
2. Обновить статистики $L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T$ и $R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k$.
3. Вычислить предобуславливатели $P_L = L_k^{-1/4}$ и $P_R = R_k^{-1/4}$. (Обратный матричный корень)
4. Обновить: $W_{k+1} = W_k - \alpha P_L G_k P_R$.

Заметки:

- Нацелен на более эффективный учет информации о кривизне, чем методы первого порядка.
- Вычислительно дороже, чем Adam, но может сходиться быстрее или к лучшим решениям с точки зрения количества шагов.
- Требует тщательной реализации для эффективности (например, эффективное вычисление обратных матричных корней, работа с большими матрицами).

Shampoo [5]

Расшифровывается как **S**tochastic **H**essian-**A**pproximation **M**atrix **P**reconditioning for **O**ptimization **O**f deep networks. Это метод, вдохновленный оптимизацией второго порядка, разработанный для глубокого обучения больших моделей.

Основная идея: Аппроксимирует полноматричный предобуславливатель AdaGrad, используя эффективные матричные структуры, в частности произведения Кронекера.

Для матрицы весов $W \in \mathbb{R}^{m \times n}$ обновление включает предобуславливание с использованием аппроксимаций матриц статистики $L \approx \sum_k G_k G_k^T$ и $R \approx \sum_k G_k^T G_k$, где G_k — градиенты.

Упрощенная концепция:

1. Вычислить градиент G_k.
2. Обновить статистики $L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T$ и $R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k$.
3. Вычислить предобуславливатели $P_L = L_k^{-1/4}$ и $P_R = R_k^{-1/4}$. (Обратный матричный корень)
4. Обновить: $W_{k+1} = W_k - \alpha P_L G_k P_R$.

Заметки:

- Нацелен на более эффективный учет информации о кривизне, чем методы первого порядка.
- Вычислительно дороже, чем Adam, но может сходиться быстрее или к лучшим решениям с точки зрения количества шагов.
- Требует тщательной реализации для эффективности (например, эффективное вычисление обратных матричных корней, работа с большими матрицами).

Muon [6] [7] [8]

$$
\begin{aligned}
W_{t+1} &= W_t - \eta (G_t G_t^\top)^{-1/4} G_t (G_t^\top G_t)^{-1/4} \\
&= W_t - \eta (US^2U^\top)^{-1/4} (USV^\top)(VS^2V^\top)^{-1/4} \\
&= W_t - \eta (US^{-1/2}U^\top)(USV^\top)(VS^{-1/2}V^\top) \\
&= W_t - \eta US^{-1/2} SS^{-1/2} V^\top \\
&= W_t - \eta UV^\top
\end{aligned}
$$

[6] K. Jordan blogpost "Muon: An optimizer for hidden layers in neural networks". 2024.

[7] J. Bernstein blogpost "Deriving Muon". 2025.

[8] Kovalev, D. (2025). Understanding Gradient Orthogonalization for Deep Learning via Non-Euclidean Trust-Region Optimization. arXiv preprint arXiv:2503.12645.

Сравнение Muon с AdamW на LogReg

- Простое сравнение Muon и AdamW на небольшой задаче LogReg

Дополнительные материалы

Дополнительные материалы

- Д. Ветров "Удивительные свойства ландшафта функции потерь в перепараметризованных моделях"

Дополнительные материалы

- Д. Ветров "Удивительные свойства ландшафта функции потерь в перепараметризованных моделях"
- В. Голощапов "О чем не гроккинг"

Дополнительные материалы

- Д. Ветров "Удивительные свойства ландшафта функции потерь в перепараметризованных моделях"
- В. Голощапов "О чем не гроккинг"
- Д. Ковалев "Understanding Gradient Orthogonalization for Deep Learning via Non-Euclidean Trust-Region Optimization" (Теория за Muon)