# $\mathrm{math}_M uON_p roject$

Александр Клевайчук

April 2025

## 1 Introduction

В данной работе будет рассмотрен метод MuON, а также ряд других методов оптимизации нейронных сетей с целью сформировать обзорную работу по данному материалу.

MuON - относительно новый оптимизатор двумерных параметров скрытых слоев нейронной сети. С помощью него уже поставили рекорды в обучении LLM моделей. При этом показатели в обучении небольших нейронных сетей достигают лучших результатов, нежели в больших.

Сами авторы описывают метод, как основанный на работах, а не только на эмперических результатах или эвристике, как многие другие оптимизаторы, вроде ADAM (это не мои слова).

## 2 MuON. Algorithm

Предварительно мы имеем заданные lr и momentum параметры $= \eta$ и $\mu$ соответственно

**Algorithm 1** Moun

**Require:** Learning rate $\eta$, momentum $\mu$
  Initialize $B_0 \leftarrow 0$
  **for** $t = 1, \ldots :$
    Compute gradient $G_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta_{t-1})$
    $B_t \leftarrow \mu B_{t-1} + G_t$
    $O_t \leftarrow \mathrm{NewtonSchulz5}(B_t)$
    Update parameter $\theta_t \leftarrow \theta_{t-1} - \eta O_t$
  **return** $\theta_t$

Где NewtonSchulz5 - алгоритм поиска ближайшей ортогональной матрицы для заданной. Изначально, конечно, его суть в поиске обратной, но через последовательность итераций мы сможем приблизиться к ортогональной

(для которой обратная матрица равна транспонированной). Мы используем его чтобы обойти необходимость прямого вычисления SVD-разложения, из-за вычислительной дороговизны последнего.

Таким образом NS используется для замены стандартного обновления параметров в алгоритме SGD с моментом. Это позволяет адаптировать обновления таким образом, чтобы они были более близки к ортогональным матрицам, что может улучшить устойчивость и эффективность процесса обучения.

## 2.1 MuON. Newton-Schulz

$\nabla_W \mathcal{L} = U\Sigma V^\top \mapsto UV^\top$.

Мы бы хотели опустить $\Sigma$ переменную из уравнения выше, чтобы пропустить сложное вычисление и при этом ускорить нашу процедуру. При этом мы можем допустить такую процедуру, так как для любой матрицы её ортогональная аппроксимация с минимальной ошибкой в Фробениусовом норме даётся именно матрицей $UV^\top$ из $SVD$

Фробениусова норма - это мера ошибки при приближении матриц:

$$\|A\|_F = \sqrt{\sum_i \sum_j a_{ij}^2}$$

Основная идея, лежащая в основе алгоритма, заключается в том, что нечётные матричные многочлены «коммутируют» с сингулярным разложением (SVD).

$$p(X) := a \cdot X + b \cdot XX^\top X + c \cdot XX^\top XX^\top X + \cdots,$$

Применение полинома $p$ к матрице $A$ эквивалентно применению $p$ непосредственно к сингулярным значениям $A$, а затем сборке сингулярных векторов обратно вместе.

$$p(X) = p(U\Sigma V^\top) = Up(\Sigma)V^\top.$$

Для ортогонализации матрицы мы хотели бы установить все сингулярные значения равными единице. Поскольку все сингулярные значения положительны, мы можем рассматривать это как применение функции знака к сингулярным значениям. Хотя функция знака является нечетной функцией, к сожалению, она не является полиномом. Однако мы можем создать нечетные полиномы низкого порядка, которые при итеративном применении сходятся к функции знака. Простым примером (Коварик, 1970) является кубический полином:

$$p_3(\Sigma) := \frac{3}{2} \cdot \Sigma - \frac{1}{2} \cdot \Sigma\Sigma^\top\Sigma.$$

Для визуализации этого кубического многочлена построим график функции $y = p_3(x)$ слева. А справа — $y = p_3(p_3(p_3(p_3(p_3(x)))))$. Обратите внимание, что повторное применение кубической функции начинает всё больше напоминать функцию знака в окрестности нуля:

Картинка

На самом деле функция будет сходиться к sgn функции знака, если исходные сингулярные значения нормированы так, чтобы быть меньше $\sqrt{3}$. Авторы статей о MuON называют эту итерацию Newton-Schulz, следуя терминологии Николаса Хайама.

При этом если добавить в процедуру NS коэффициенты a,b,c:

$$G' := aG + b(GG^\top)G + c(GG^\top)^2G$$

$$= (aI + b(GG^\top) + c(GG^\top)^2)G$$

$$= (aI + bUS^2U^\top + cUS^4U^\top)USV^\top$$

$$= U(aS + bS^3 + cS^5)V^\top$$

То для того, чтобы убедиться, что у нас получается ортоганализация: Ortho(G) = $UV^\top$

Нам остаётся только:

1) Убедиться, что S лежит в интервале [0,1]

2) Выбрать коэффициенты так, чтобы $\varphi^N(x) \to 1$ при $N \to \infty$ для любого $x \in [0, 1]$

Чтобы удовлетворить 1 условие мы просто заменим G на $G/\|G\|_F$. Это работает поскольку $\mathrm{Ortho}(cG) = \mathrm{Ortho}(G)$

А чтобы удовлетворить 2 условие можно использовать разные подходы. Авторы отмечают здесь большую свободу для файн-тюнинга коэффициентов, но сами остановились на наборе $(a, b, c) = (3.4445, 4.7750, 2.0315)$.

Так как набор был получен в качестве ad-hoc задачи на основе вычисления градиентов, я не буду глубоко останавливаться здесь. Отмечу только, что решаемая оптимизационная задача действительно выглядит достаточно свободной для поиска других параметров.

# 3 SHAMPOO

Другим, близким по духу и прямо связанному по эволюции методов, алгоритмом оптимизации является Shampoo - онлайн-алгоритм стохастической оптимизации, относящийся к семейству методов AdaGrad, используемых для обучения нейронных сетей. Алгоритм строит блочно-диагональный предобуславливатель, где каждый блок представляет собой грубое приближение к полному матричному AdaGrad для каждого параметра нейронной сети с помощью приближения в виде тензорного произведения Кронекера

### 3.0.1 Диагональные адаптивные градиентные методы

Семейство адаптивных градиентных методов [Dozat 2016; Duchi et al. 2011; Kingma and Ba 2015; Reddi et al. 2018] предназначено как для стохастической оптимизации, так и для онлайн выпуклой оптимизации. Метод AdaGrad предварительно обусловливает (суб)градиент псевдообратным квадратным корнем суммы квадратов градиентов или произведений градиентов, т.е.

$$w_{t+1} = w_t - \alpha_t A_t^{+\frac{1}{2}} g_t \tag{1}$$

где $g_t \in \mathbb{R}^d$ — векторизованный (мини-батч) стохастический градиент, $\alpha_t > 0$ — скорость обучения или длина шага, и

$$A_t = \begin{cases} \sum_{s=0}^{t} g_s g_s^T & \text{(Full-Matrix AdaGrad)} \\ \sum_{s=0}^{t} \operatorname{diag}(g_s^2) = \sum_{s=0}^{t} \operatorname{matdiag}(g_s g_s^T) & \text{(Diagonal AdaGrad)} \end{cases} \tag{2}$$

для $\epsilon > 0$. В этом случае $p_t = A_t^{+\frac{1}{2}} g_t$ — это адаптивное направление градиентного поиска. Отметим, что полный матричный AdaGrad требует $O(d^2)$ памяти и $O(d^3)$ вычислений, в то время как диагональный AdaGrad требует $O(d)$ памяти и $O(d)$ вычислений. Связанные методы, такие как RMSProp и Adam, используют экспоненциальные скользящие средние вместо суммы, т.е.

$$A_t = \beta_2 A_{t-1} + (1 - \beta_2) \operatorname{diag}(g_t^2) \tag{3}$$

Ладно, полностью не разберу метод, очень большие статьи, не успеваю просто прочитать и осознать, постараюсь какое-то саммари выделить.

## 3.1 SUMMARY

Shampoo - это алгоритм оптимизации для нейронных сетей, который сочетает в себе преимущества диагональных и полно-матричных методов предобуславливания. Он предназначен для эффективного обучения глубоких нейронных сетей и использует специальную структуру градиентов нейронных сетей для уменьшения потребления памяти и вычислительных ресурсов.

1. **Градиентный Спуск**: основная формула градиентного спуска имеет вид:

   $w_{\text{new}} = w_{\text{old}} - \alpha \cdot \nabla L(w_{\text{old}})$

   где $w_{\text{old}}$ - текущее значение весов, $\alpha$ - шаг обучения, $\nabla L(w_{\text{old}})$ - градиент функции потерь в точке $w_{\text{old}}$.

2. **Предобуславливание**: предобуславливание градиента имеет вид:

   $\nabla L(w_{\text{old}}) = H^{-1} \cdot \nabla L(w_{\text{old}})$

   где $H$ - матрица Гесса функции потерь, $H^{-1}$ - обратная матрица Гесса.

3. **Блок-Диагональное Предобуславливание**: Shampoo использует блок-диагональное предобуславливание, где каждая диагональная матрица соответствует одному слою нейронной сети. Это можно записать как:

   $H = \text{diag}(H_1, H_2, \ldots, H_n)$

   где $H_1, H_2, \ldots, H_n$ - диагональные матрицы для каждого слоя.

4. **Кронекерское Произведение**: Shampoo использует кронекерское произведение для аппроксимации полно-матричной предобуславливания. Это можно записать как:

   $H_{\text{approx}} = \text{Kronecker}(H_1, H_2, \ldots, H_n)$

   где Kronecker - оператор кронекерского произведения.

### 3.1.1 Ортогонализация

Shampoo использует ортогонализацию для уменьшения размерности матрицы Гесса и ускорения вычислений. Ортогонализация заключается в том, что матрица Гесса разбивается на ортогональные матрицы, которые можно вычислить более эффективно.

В частности, Shampoo использует ортогонализацию по столбцам, которая заключается в том, что столбцы матрицы Гесса ортогонализируются друг другу. Это можно записать как:

$H = Q \cdot R$

где $Q$ - ортогональная матрица, $R$ - верхняя треугольная матрица.

Это позволяет уменьшить размерность матрицы Гесса и ускорить вычисления, что делает Shampoo более эффективным для обучения глубоких нейронных сетей.