

Занятие 4. Оптимизация параметров нейронных сетей. Инициализация параметров

Гирдюк Дмитрий Викторович

27 сентября 2025

СПбГУ, ПМ-ПУ, ДФС

Стартер [1]



Метод градиентного спуска

- Попробуем за час (академический) освежить в памяти метод градиентного спуска, а также бегло познакомиться с его вариациями в контексте оптимизации параметров нейронных сетей
- Кратко по обозначениям в этой части: θ – параметры, которые хотим оптимизировать, L – лосс-функция
- Метод градиентного спуска:

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

- Почему не актуально в контексте глубокого обучения?

Градиентный спуск II

- Выбрав начальное приближение для весов, наше финальное положение будет однозначно определено
- И путь этот может занять очень много времени – особенно для датасетов с сотнями миллионов примеров и миллиардами параметров
- Не говоря уже про "сложный" ландшафт лосс-функции и проблемы с выбором длины шага/скорости обучения λ

Стохастический градиентный спуск

- Стохастический градиентный спуск (Stochastic Gradient Descent, SGD):

$$\theta_{t+1} \leftarrow \theta_t - \lambda \frac{1}{|B_t|} \sum_{i \in B_t} \nabla_{\theta} l(\theta_t)$$

где B_t – это мини-батч из (перемешанных) примеров

- Далее для упрощения записи будем подразумевать под $\nabla_{\theta} L(\theta_t)$ именно стохастический градиент
- Вычислительно менее затратно (в смысле шага оптимизации), двигаемся криво-косо, но в нужном направлении (хотя размер мини-батча имеет существенное значение), имеет шанс выпрыгивать из седловых точек (близкий к 0 градиент) и избегать локальных минимумов

Стохастический градиентный спуск II

- Зачастую используется в паре с различными схемами изменения скорости обучения (learning rate scheduler), о них позже
- Есть набор оценок сходимости GD и SGD для выпуклого и невыпуклого случая, но там все достаточно печально: сильно зависит от начального приближения (квадратично или хуже от расстояния между начальной точкой и оптимумом) [2]

Метод импульса

- Идем дальше, как вам такая идея: попробуем на каждом шаге ориентироваться не только на антиградиент, но и на историю движения
- Сделаем это на основе экспоненциального сглаживания

$$v_{t+1} \leftarrow \beta v_t + \lambda \nabla_{\theta} L(\theta_t)$$

$$\theta_{t+1} \leftarrow \theta_t - v_{t+1}$$

- Интерпретация следующая: если некоторое время двигаемся в определенном направлении (накопили импульс/momentum), то его резкое изменение может быть не самой лучшей идеей
- Из проблем, у нас еще один гиперпараметр β . Впрочем, настраивают его достаточно редко и оставляют равным 0.9

- Немного скорректируем предыдущую идею, считаем градиент не в текущей точке, а в той, в которой мы окажемся с учетом импульса

$$v_{t+1} \leftarrow \beta v_t + \lambda \nabla_{\theta} L(\theta_t - \beta v_t)$$

$$\theta_{t+1} \leftarrow \theta_t - v_{t+1}$$

- Эта идея позволяет повысить устойчивость и скорость сходимости в определенных случаях
- На самом деле есть альтернативные варианты импульса/момента Нестерова (в англоязычной литературе Nesterov Accelerated Gradient, NAG), а также масса дополнительных модификаций. Подробно можно почитать вот тут [3]

Adagrad

- Идея Adagrad (adaptive gradient) состоит в том, чтобы корректировать скорость обучения отдельных параметров на основе исторической информации: шаг изменения должен быть меньше у тех параметров, которые в большей степени варьируются в данных, и больше у менее изменчивых

$$\theta_{t+1} \leftarrow \theta_t - \frac{\lambda}{\sqrt{G_t} + \epsilon} \nabla_{\theta} L(\theta_t)$$

тут $G_t \leftarrow G_{t-1} + g_t^2$, а $g_t = (\nabla_{\theta} L(\theta_t))^2$

- Можете выделить плюсы и минусы?

- Проблема Adagrad в том, что сумма квадратов градиентов некоторых параметров может достаточно быстро вырасти, что, в свою очередь, приведет к проблеме затуханию обновлений
- Для исправления предлагается не суммировать квадраты градиентов, а считать экспоненциальное среднее

$$E[g^2]_t \leftarrow \beta E[g^2]_{t-1} + (1 - \beta)g_t^2$$

- И получаем RMSProp

$$\theta_{t+1} = \theta_t - \frac{\lambda}{\sqrt{E[g^2]_t} + \epsilon} \nabla_{\theta} J(\theta_t)$$

- Adam (adaptive momentum estimation) сочетает в себе идеи накопления импульса и адаптивного обновления весов переменных признаков

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) \nabla_{\theta} L(\theta_t)$$

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_{\theta} L(\theta_t))^2$$

$$m_t \leftarrow \frac{m_t}{1 - \beta_1^t}, \quad v_t \leftarrow \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} \leftarrow \theta_t - \frac{\lambda}{\sqrt{v_t} + \epsilon} m_t$$

- Обратите внимание на коэффициенты в 3 строчке. Как вы думаете, для чего они добавляются?
- Его незначительная модификация до сих пор выбор по умолчанию вне зависимости от предметной области

- Обычно в реализациях Adam'a присутствует возможность для регуляризации, называемая weight decay. Она эквивалентна L_2 регуляризации в некоторых случаях, но не для Adam'a
- В Adam, прежде чем считать моменты m_t и v_t , предлагается скорректировать градиент:

$$g_t \leftarrow g_t + \lambda \theta_{t-1}$$

- В AdamW градиенты используются исходны, а регуляризация добавляется на последнем шаге при обновлении параметров

$$\theta_{t+1} \leftarrow \theta_t - \eta \left(\frac{1}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t + \lambda \theta_{t-1} \right)$$

Что, еще?

- *AMSgrad*, NAdam, RAdam, LookAhead, AggMo, YellowFin, LAMB, QHM, QHAdam, Demon, AdaPlus, Lion...
- По умолчанию используйте AdamW. Есть время и ресурсы на активный перебор гиперпараметров – пробуйте современные альтернативы, в первую очередь ориентируясь на размер батча
- Самое время задать себе вопрос, а сколько суммарно параметров будет задействовано при обучении нейронных сетей, скажем, используя AdamW?

Промежуточный итог по оптимизаторам

- Кратко познакомились с основными идеями, используемыми в оптимизаторах нейронных сетей
- Для интересующихся, вот тут несколько устаревший (2017), но все еще актуальный обзор с иллюстрациями поведения градиента и отдельных компонент оптимизаторов [1], уже упомянутый блогпост по вариантам импульса Нестерова [3], сравнение оптимизаторов в 2020 [4]
- Отдельного внимания заслуживает глава из учебника по МЛ Яндекса, в рамках которой разбираются стохастические оптимизаторы с точки зрения алгоритмов онлайн-обучения [5]

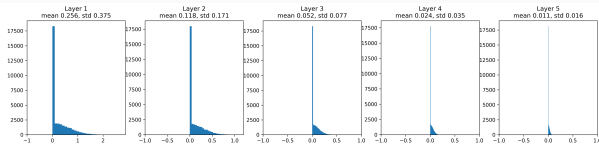
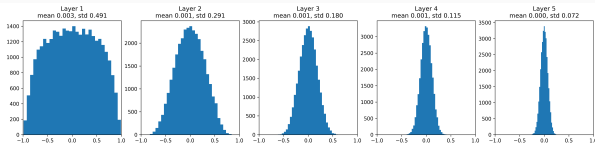
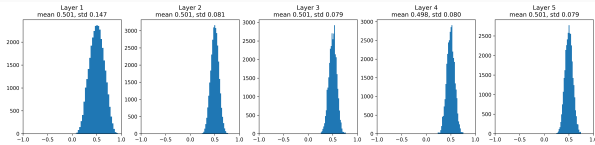
Инициализация параметров

- Мы рассмотрели различные способы настройки параметров весов, но опустили очень важный момент: как параметры должны быть инициализированы?
- Некорректная инициализация параметров может привести к проблемам исчезающих или взрывающихся градиентов. Это, в свою очередь, может свести на нет все попытки обучить нейронную сеть
- Начнем с исключения очевидных вариантов: почему в общем случае не следует инициализировать параметры нулями? А произвольной константой?

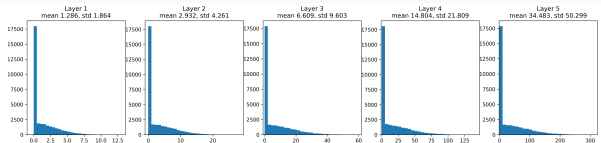
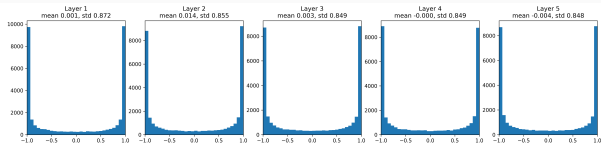
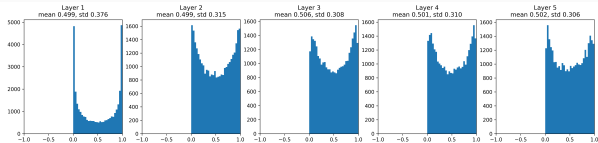
Рандомная инициализация

- Простейший вариант – инициализировать веса значениями нормального распределения с нулевым матожиданием и небольшим значением среднеквадратического отклонения (например, 0.01)
- Подобная инициализация нарушает симметрию, и вполне может работать для не очень глубоких сетей
- В зависимости от значения σ с ростом количества слоев будет наблюдаться постепенное уменьшение или увеличение абсолютных значений активации
- В этих двух ситуациях значения активаций могут стать настолько малыми или настолько большими, что их невозможно будет представить с помощью арифметики с плавающей точкой/запятой
- Проиллюстрируем это для 3 функций активации: сигмоида, гиперболический тангенс и ReLU

Рандомная инициализация, $\text{std} = 0.01$: σ , \tanh , ReLU



Рандомная инициализация, $\text{std} = 0.05$



Инициализация Xavier

- Хорошо, выходит, что все упирается в значение дисперсии

$$\begin{aligned}\text{Var}(y_i) &= \text{Var} \left(\sum_{j=1}^n w_{ij} x_j \right) = \sum_{j=1}^n \text{Var}(w_{ij} x_j) = \\ &= \sum_{j=1}^n (E[w_{ij}^2] E[x_j^2] - E[w_{ij}]^2 E[x_j]^2) = \\ &= \sum_{j=1}^n (E[w_{ij}]^2 \text{Var}(x_j) + \text{Var}(w_{ij}) \text{Var}(x_j) + \text{Var}(w_{ij}) E[x_j]^2) = \\ &= \sum_{j=1}^n \text{Var}(w_{ij}) \text{Var}(x_j) = n_{\text{in}} \text{Var}(w_{ij}) \text{Var}(x_i)\end{aligned}$$

- Тут предполагается, что веса инициализируются из симметричного и центрированного в нуле распределения. Аналогичные предположения для входов x_j (для каких функций активации этой справедливо?)

- Выходит, что если мы установим дисперсию $\text{Var}(w_{ij})$ равной $\frac{1}{n_{\text{in}}}$, то дисперсия $\text{Var}(y_i)$ будет равна дисперсии входов с предыдущего слоя
- Аналогичный анализ можно провести и для градиентов, что покажет необходимость инициализации параметров с дисперсией равной $\frac{1}{n_{\text{out}}}$

Инициализация Xavier III

- Тогда для случая нормального распределения предлагается следующий подход к инициализации

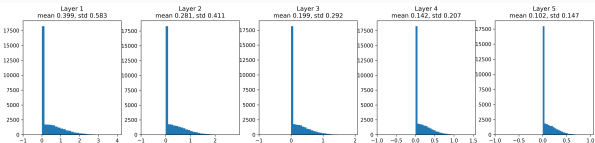
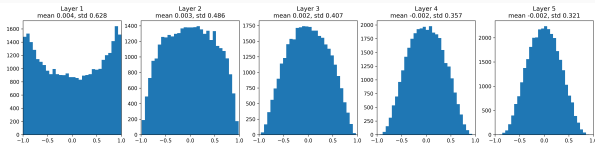
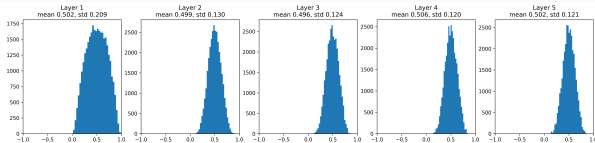
$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{in} + n_{out}}}\right)$$

- и для случая равномерного распределения

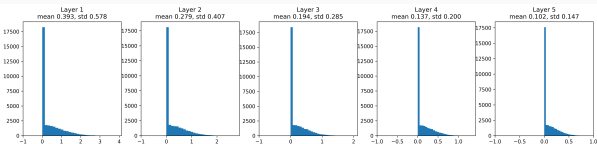
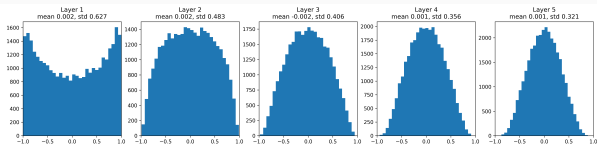
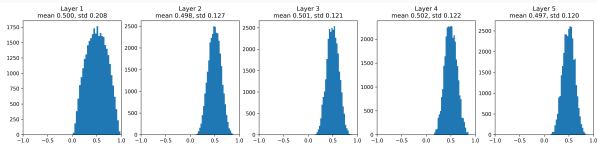
$$W \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{in} + n_{out}}}, \sqrt{\frac{6}{n_{in} + n_{out}}}\right)$$

- Данный подход принято называть в честь первого автора статьи [6] (Более 26500 цитирований!)

Инициализация Xavier, нормальное распределение



Инициализация Xavier, равномерное распределение



Ограничения инициализации Xavier

- Анализ в статье [6] был проведен для сигмоиды и гиперболического тангенса
- Рекомендация состояла в использовании последнего с указанным подходом к инициализации ввиду его симметричности относительно нуля (мы использовали это свойство при вычислении дисперсии активаций)
- Что делать в случае несимметричной относительно нуля ReLU?

Случай ненулевого матожидания активаций

- Как изменится формула для дисперсии, если x_j не симметричны относительно нуля?

$$\begin{aligned}\text{Var}(y_i) &= \sum_{j=1}^n (E[w_{ij}^2]E[x_j^2] - E[w_{ij}]^2 E[x_j]^2) = \\ &= \sum_{j=1}^n \text{Var}(w_{ij})E[x_j^2] = n_{\text{in}}\text{Var}(w_{ij})E[x_i^2]\end{aligned}$$

Случай ненулевого матожидания активаций II

- Если на предыдущем слое веса инициализированы из симметричного относительно нуля распределения, а смещения нулевые, то y'_i с предыдущего слоя (важно, до применения ReLU) имеют симметричные и центрированные в нуле распределения, причем

$$E[x_i^2] = \frac{1}{2}E[(y'_i)^2] = \frac{1}{2}\text{Var}(y'_i)$$

- Что дает

$$\text{Var}(y_i) = \frac{1}{2}n_{\text{in}}\text{Var}(w_{ij})\text{Var}(y'_i)$$

Инициализация Kaiming

- Все то, что мы проделали на предыдущих слайдах, было сделано в работе [7] (тоже 25000+ цитирований!)
- Аналогичным образом инициализация параметров названа в честь первого автора работы
- Для случая нормального распределения

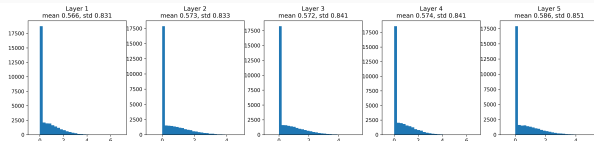
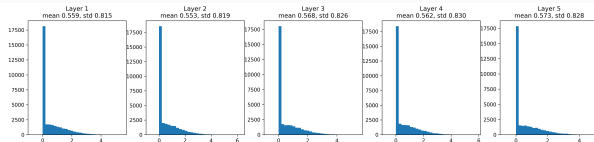
$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{in}}}\right)$$

- и для случая равномерного распределения

$$W \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{in}}}, \sqrt{\frac{6}{n_{in}}}\right)$$

- Еще раз отметим, что все смещения предполагается инициализировать нулями (во всех подходах)

Инициализация Kaiming, ReLU



Подытожим

- Рассмотрели вопросы оптимизации параметров нейронных сетей, разобрали основные алгоритмы оптимизации, используемые в современных библиотеках глубокого обучения
- Изучили основные способы инициализации параметров нейронных сетей (инициализация параметров в `pytorch` [8])
- На следующем занятии познакомимся с новым для нас типом слоев, свертками или сверточными сетями, проявившими себя в задачах компьютерного зрения

1. *Siarshai*. **Методы оптимизации нейронных сетей**. URL: <https://habr.com/ru/articles/318970/1>.
2. *Stich S. U.* **Unified Optimal Analysis of the (Stochastic) Gradient Method**. // arXiv e-prints. 2019. DOI: 10.48550/arXiv.1907.04232. arXiv: 1907.04232 [cs.LG].
3. *Melville J.* **Nesterov Accelerated Gradient and Momentum**. URL: <https://jlmelville.github.io/mize/nesterov.html>.
4. *Chen J.* **An updated overview of recent gradient descent algorithms**. URL: <https://johnchenresearch.github.io/demon/>.
5. *Морозов А.* **Онлайн-обучение и стохастическая оптимизация**. URL: <https://education.yandex.ru/handbook/ml/article/metody-optimizacii-v-deep-learning>.

6. *Glorot X., Bengio Y. Understanding the difficulty of training deep feedforward neural networks.* // Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Т. 9. PMLR, 2010. С. 249—256. (Proceedings of Machine Learning Research). URL: <https://proceedings.mlr.press/v9/glorot10a.html>.
7. **Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.** /. К. Хе [и др.] // arXiv e-prints. 2015. Февр. DOI: 10.48550/arXiv.1502.01852. arXiv: 1502.01852 [cs.CV].
8. *PyTorch. Parameters initialization module.* URL: <https://pytorch.org/docs/stable/nn.init.html>.