Новейшие методы оптимизации для нечетких сетей

Д. А. Михайлов

Санкт-Петербургский Федеральный исследовательский центр Российской академии наук

dam@dscs.pro

М. В. Абрамов

Санкт-Петербургский Федеральный исследовательский центр Российской академии наук

mva@dscs.pro

Аннотация. **В настоящее время в области искусственного интеллекта происходит активное развитие методов модификации оптимизаторов для параметров нейронных сетей, что обусловлено значительным влиянием этих алгоритмов на точность и скорость обучения моделей. В этой связи стоит отметить, что методы, такие как SGD, Adam и RMSProp, уже хорошо известны и получены исчерпывающие результаты их исследования. Однако наряду с этим существуют новые подходы, свойства и эффективность которых требуют более глубокого изучения. Данная работа фокусируется на исследовании влияния выбора метода оптимизации параметров на качество и скорость обучения нечетких нейронных сетей. Анализирование этих факторов имеет важное значение для оптимизации процесса обучения и повышения общей производительности математических моделей, использующих нечеткие нейронные сети, что открывает новые возможности для их применения в различных областях науки и техники.**

Ключевые слова: искусственный интеллект; нейронные сети; методы оптимизации; эффективность обучения; градиентные методы

I. Введение

В последнее время в компьютерных науках особое место уделяется нейронным сетям [1]. При грамотной настройке за счет большого количества параметров, они могут обеспечить высокую точность в задачах регрессии и классификации [2]. Однако, при увеличении параметров в какой-то момент прирост точности останавливается. Это объясняется тем, что модель уже является достаточно сложной для задачи, и усложнять ее дополнительно не имеет никакого смысла. В таких ситуациях повысить точность можно за счет оптимизации [3].

Веса нейронов, полученные во время обучения нейронной сети, подбираются исходя из значения функции потерь (loss function), которая задается перед обучением модели в зависимости от нашей задачи. Оптимальные веса должны достигаться в точке минимума этой функции, в которую оптимизатор спускается в обратном направлении ее градиента. Однако, вычисление значения функции может быть затратным по времени, ведь в подсчете значения, как правило, участвует большое количество наблюдений и признаков [3].

Работа выполнена в рамках проекта по государственному заданию СПБ ФИЦ РАН (СПИИРАН) №FFZF-2024-0003

Избежать этого можно, например, оптимизацией стохастическим градиентом, который будет использовать для подсчета случайные подвыборки меньшего размера.

Другая проблема заключается в «застревании» оптимизатора в точке локального минимума и трудность получения глобально оптимальных параметров. Для решения этой проблемы можно использовать модифицированные методы, которые сохраняют историю расчетов, применяют регуляризацию, используют адаптивный шаг [3, 4].

В настоящее время получены достаточно обширные результаты по применению «классических» оптимизаторов SGD, Momentum, RMSProp, Adagrad, Adam [4]. Однако, новейшие методы, которые появились в течение последних нескольких лет, исследованы не так подробно, особенно в области применения к нечетким сетям.

II. Рассматриваемые алгоритмы оптимизации

В данной статье рассматриваются наиболее цитируемые алгоритмы из современных методов оптимизации:

- **Lion** – это новый, быстрый оптимизатор от Google Research, созданный специально для больших моделей. Его главная особенность заключается в том, что он не использует градиенты напрямую, а двигается по знаку градиента, то есть только по направлению (в плюс или минус), а не по его величине, что делает его очень легким и стабильным по памяти [5].
- **Adafactor** – оптимальная по памяти модификация алгоритма Adam (за счет использования факторизации матриц, вместо хранения вторых моментов) [6].
- **AdamW** – модификация Adam с включенной регуляризацией. Исходный оптимизатор плохо справляется с L2-регуляризацией (weight decay), потому что применяет её внутри адаптивного шага, что математически неправильно. В этом алгоритме исправили эту проблему, при этом скорость работы осталась сопоставимой [7].

В рамках работы алгоритмы, перечисленные выше, будут сравниваться с базовым решением – алгоритмом оптимизации Adam [8].

III. Описание данных и структура сети

Для обучения использовались данные **winequality-red**. Этот датасет — часть проекта по прогнозированию качества красного вина сорта Vinho Verde, которые размещены на ресурсе UCI Machine Learning Repository. В нем содержится 11 химико-физических параметров и 1 целевая переменная качества вина. Всего в датасете 1599 наблюдения: 1119 использовались в качестве тренировочной выборки, 240 в качестве валидационной выборки и 240 в качестве тестовой выборки. Для нормализации данных был использован алгоритм min-max [9].

После инициализации датасета была создана нечеткая логическая система, чтобы предсказывать качество вина на основе двух параметров: *fixed acidity* и *volatile acidity*. В рамках нее использовались функции принадлежности *trimf*, для создания нечетких треугольных множеств. Исследуемое качество делилось на 3 уровня: низкое, среднее и высокое, на основе медианы и крайних значений [10].

После преобразования исходных признаков данные подавались на вход двум моделям нейронной сети с фиксированной структурой: базовой и продвинутой.

Layer (type)	Output Shape	Param #
dense_249 (Dense)	(None, 128)	384
batch_normalization_102 (BatchNormalization)	(None, 128)	512
dropout_51 (Dropout)	(None, 128)	0
dense_250 (Dense)	(None, 64)	8,256
dense_251 (Dense)	(None, 32)	2,080
batch_normalization_103 (BatchNormalization)	(None, 32)	128
dense_252 (Dense)	(None, 1)	33

Total params: 11,393 (44.50 KB)

Trainable params: 11,073 (43.25 KB)

Non-trainable params: 320 (1.25 KB)

Рис. 1. Структура базовой сети

На рис. 1 приведено послойное и общее количество параметров базовой нейронной сети. Ее компоненты:

1. Входной слой — принимает 2 числовых признака (fixed_acidity, volatile_acidity)
2. Полносвязный слой на 128 нейронов с активацией ReLU
3. Нормализирующий слой — нормализует выход предыдущего слоя, ускоряет обучение, повышает стабильность
4. Dropout слой — отключает случайные 30 % нейронов, борясь с переобучением
5. Скрытый слой на 64 нейрона
6. Скрытый слой на 32 нейрона
7. Еще один нормализирующий слой
8. Выходной слой — один нейрон без активации.

Всего общее количество параметров этой сети — 11, 393.

Продвинутая сеть использует подход известный как ResNet (Dense) [12]. Идея заключается в применении концепции остаточных соединений, что является ключевым элементом архитектуры. Этот подход был предложен для решения проблем, которые могут возникать при обучении глубоких нейронных сетей, таких как затухание градиента и переобучение [13]. Модель состоит из следующих компонентов: Input, Dense, 3 ResNet слоя, и выходной слой. Общее количество параметров: 29,697.

Для всех оптимизаторов использовались параметры «по-умолчанию». Число эпох обучения также для всех оптимизаторов и моделей было одинаковым и равнялось 500. Также задавался фиксированный размер батча — 128 и параметр случайности — 42.

IV. Результаты сравнения

Результаты сравнения оптимизаторов на базовой модели показаны на табл. 1.

ТАБЛИЦА I. Сравнение методов оптимизации на базовой модели

Оптимизатор	Test MAE	Test Loss	Epoch Time, сек
Adafactor	2.8350	8.0647	0.43
Lion	0.0418	0.0044	0.295
AdamW	0.0287	0.0084	0.385
Adam	0.0688	0.0091	0.350

A.

Видно, что наилучшим с точки зрения качества оказался метод AdamW, при этом по скорости и по значению функции потерь наилучшим оказался метод Lion, все еще оставаясь лучше по точности базового алгоритма Adam.

Хуже всего себя показал алгоритм Adafactor. Не смотря на то, что он является модификацией алгоритма Adam, он очень долго держался в окрестности начальной точки, что показано на рис. 2. Это не позволило ему продемонстрировать хороший результат.

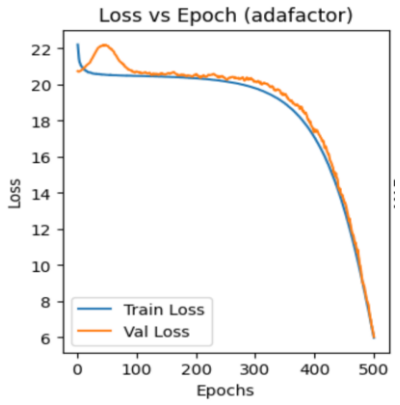

Рис. 2. График зависимости значения функции потерь от номера эпохи для оптимизатора Adafactor

Результаты сравнения оптимизаторов на продвинутой модели показаны на табл. 2.

ТАБЛИЦА II. Сравнение методов оптимизации на продвинутой модели

Оптимизатор	Test MAE	Test Loss	Epoch Time, сек
Adafactor	2.45	6.03	0.43
Lion	0.054	0.0110	0.290
AdamW	0.0352	0.0056	0.380
Adam	0.0482	0.0092	0.350

Исходя из результатов, можно сделать вывод, что базовая модель уже являлась оптимальной для исходных данных и ее усложнение не повлияло на итоговую точность. При этом темпы обучения для всех оптимизаторов не менялись под конец обучения, что позволяет сделать вывод, что ситуации «underfitting» не было, что показано на рис. 3 [14].

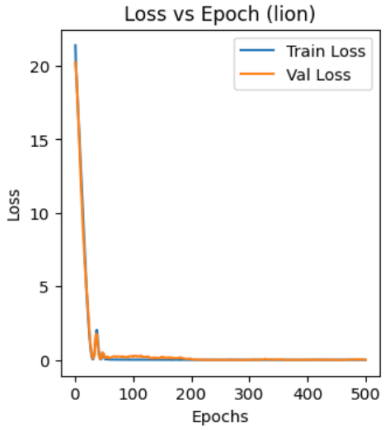

Рис. 3. График зависимости значения функции потерь от номера эпохи для оптимизатора Lion на продвинутой модели. Видно, что при приближении к 500-й эпохе изменения функции потерь несущественные

При обучении продвинутой модели, наилучшим с точки зрения качества также оказался метод AdamW, при этом он стал лучшим и по значению функции потерь. Алгоритм Lion, при этом, смог сохранить наилучшую скорость вычислений и точность лучше, чем Adam. Оптимизатор Adafactor сохранил наихудший результат в этой задаче.

V. Заключение

В данной работе было исследовано применение современных методов оптимизации к нейронным сетям, используемых для прогнозирования данных, предобработанных нечеткой моделью.

Было показано, что для подобных размерностей не имеет смысла усложнение модели сети, так как базовая модель уже достаточно оптимальная.

Алгоритм AdamW показал наилучший результат по точности, при этом сохраняя приемлемую скорость обучения. Существуют исследования, которые показывают, что для данных с большим количеством индивидов и признаков, а также на сложных больших моделях он может работать хуже [15, 16].

Алгоритм Lion показал результат лучше Adam в точности и скорости обучения, но оказался хуже AdamW для данной задачи. При этом в задаче классификации изображений со сложными моделями он, в среднем, показывает результаты лучше, чем AdamW [17]. Также этот оптимизатор сильно зависит от начальных гиперпараметров [16]. В рамках исследования они были зафиксированы по умолчанию для всех методов. При этом их настройка гипотетически сможет ощутимо улучшить результат для некоторых, чувствительных к их изменению, оптимизаторов.

Алгоритм Adafactor показал худшие результаты в рамках этой задачи. Исторически, он был создан для экономии памяти в больших трансформерах, например в T5, где миллионы параметров [18]. В нашей задаче параметров гораздо меньше, однако, он по-прежнему является современным методом оптимизации, поэтому для сравнения был добавлен в исследование. Он также оптимизирует память, используя факторизованные представления градиентов, что эффективно только для больших матриц, поэтому для нашей матрицы ранга 2, прироста производительности не будет [19].

Список литературы

[1] Ibomoiye Domor Mienye, Theo G. Swart. A Comprehensive Review of Deep Learning: Architectures, Recent Advances, and Applications. // Information: науч. журнал / MDPI (Multidisciplinary Digital Publishing Institute). 2024. Vol. 15, No. 12, pp. 755.

[2] You, Y., Li, Y., & Hsieh, C.-J. Why does Adam outperform SGD? A theoretical and empirical analysis. Proceedings of the 37th International Conference on Machine Learning (ICML 2020), 2020.

[3] Михайлов Д.А. Сравнение некоторых оптимизаторов нейронных сетей // Системы интеллектуального управления и искусственный интеллект: теория и практика // Сборник трудов II национальной научно-практической конференции. Санкт-Петербург: ГУМРФ, 2024. С. 83-86.

[4] Михайлов Д.А. Использование актуальных оптимизаторов нейронных сетей в задачах социокомпьютинга // Региональная информатика (РИ-2024) : Материалы XIX Санкт-Петербургской международной конференции, Санкт-Петербург: Санкт-Петербургское Общество информатики, вычислительной техники, систем связи и управления, 2024. С. 388-389.

[5] Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 404–413. PMLR, 10–15 Jul 2018.

[6] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 4596–4604. PMLR, 10–15 Jul 2018.

[7] Luo, Y. Xiong, Y. Liu, and X. Sun. Adaptive gradient methods with dynamic bound of learning rate. In International Conference on Learning Representations, 2018

[8] Kingma, Diederik & Ba, Jimmy. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations, 2014.

[9] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. "Wine Quality," UCI Machine Learning Repository, 2009. Available: https://doi.org/10.24432/C56S3T.

[10] Razak T. R. et al. Python scikit-fuzzy: developing a fuzzy expert system for diabetes diagnosis //Int J Artif Intell. T. 2252. №.8938. С. 1399.

[11] Shevchenko A. Landscape connectivity and dropout stability of SGD solutions for over-parameterized neural networks / A. Shevchenko, M. Mondelli // 37th International Conference on Machine Learning, ICML 2020: 37, Virtual, Online, 2020. P. 8732-8743.

[12] Shi, L. Evaluating Dropout Placements in Bayesian Regression Resnet / L. Shi, C. Copot, S. Vanlanduit // Journal of Artificial Intelligence and Soft Computing Research. 2022. Vol. 12, No. 1. P. 61-73. DOI 10.2478/jaiscr-2022-0005.

[13] Claudio Filipi Gonçalves Dos Santos and João Paulo Papa. 2022. Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks. ACM Comput. Surv. 54, 10s, Article 213 (January 2022), 25 pages. https://doi.org/10.1145/3510413

[14] Jabbar H., Khan R. Z. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study) //Computer science, communication and instrumentation devices. 2015. Т. 70. №. 10.3850. С. 978-981.

[15] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. Transactions on Machine Learning Research, 2022.

[16] Chen X. et al. Symbolic discovery of optimization algorithms. arXiv 2023 //arXiv preprint arXiv:2302.06675. 2023. Т. 5.

[17] Bhatnagar R. et al. Enhanced Vision Transformer with Lion Optimizer for Accurate Classification of Monkeypox Skin Images //2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS). IEEE, 2024. С. 1897-1902.

[18] Shazeer N., Stern M. Adafactor: Adaptive learning rates with sublinear memory cost //International Conference on Machine Learning. PMLR, 2018. С. 4596-4604.

[19] Zhao P. et al. Adapprox: Adaptive approximation in adam optimization via randomized low-rank matrices //arXiv preprint arXiv:2403.14958. 2024.