A. D. Ivannikov, Dr. of Tech. Sc., e-mail: ADI@ippm.ru,
Institute for Design Problems in Microelectronics of Russian Academy of Sciences

The Theoretical Basis for the Selection of Design Debugging Tests Set for Digital Systems Based on the Alphabet of Functions Performed

The paper considers control digital systems, the functioning of which can be represented as a sequence of functions from the finite alphabet. The sequences of functions performed are presented as their products, it is shown that they form a partial semigroup. When debugging projects of digital systems using the simulation method, to verify the correctness of the project, project debugging tests are used, which should most fully verify the correctness of the performance of all functions by the designed system. The methods of compiling and modifying the list of digital system functions by the developer in the way, which is most convenient for verification, are presented. In addition, the breakdown of each digital system function into subfunctions is considered in order to verify the correct functioning of various hardware modes and program branches. The action sequence of a designer while digital systems debugging tests set developing is formally described.

Keywords: *digital systems modeling, design debugging, debugging tests, digital system functions alphabet*

Acknowledgements: This work was supported by the RFBR grant No. 17-07-00683.

DOI: 10.17587/it.25.657-662

References

1. **Ivannikov A. D., Stempkovsky A. L.** *Informacionnye Tekhnologii*. 2014, no. 9, pp. 3—10 (in Russian).
2. **Lin Yi-Li, Su Alvin W. Y.** *2011 IEEE International SOC Conference (SOCC)*, 2011, pp. 201—206.
3. **Shi Jin, Liu Weichao, Jiang Ming,** el al. *2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, 2013, pp. 71—74.
4. **Kasheev N. I., Ponomarev D. M., Podyablonsky F. M.** *Vestnik Nijegorodskogo Universiteta*, 2011, no. 3 (2), pp. 72—77 (in Russian).
5. **Ivannikov A. D., Stempkovsky A. L.** Complex Digital Systems and Microsystems Design Debugging Mathematic Model on the Basis of Stationary Dynamic System Family Presentation, *Problemi Rasrabotki Perspektivnih Mikro- I Nanoelectronnih Sistem*, 2014, part II, pp. 123—128 (in Russian).
6. **Ivannikov A. D.** *Mekhatronika, Avtomatizatsiya, Upravlenie*, 2017, vol. 18, no. 12, pp. 795—801 (in Russian).
7. **Ivannikov A. D.** *Mekhatronika, Avtomatizatsiya, Upravlenie*, 2018, vol. 19, no. 12, pp. 770—776 (in Russian).

УДК 004.85 DOI: 10.17587/it.25.662-669

М. Д. Ершов, аспирант, e-mail: ershov.m.d@rsreu.ru,
Рязанский государственный радиотехнический университет, г. Рязань

Методы оптимизации первого порядка в машинном обучении[1]

Рассмотрены проблемы, возникающие при обучении многослойных нейронных сетей прямого распространения из-за недостатков метода градиентного спуска. Выполнен обзор методов оптимизации первого порядка, которые нашли широкое применение в машинном обучении, и менее известных методов. Обзор включает стохастический градиентный спуск, быстрый градиентный метод Нестерова и различные методы с адаптацией скорости обучения. Описаны особенности каждого метода и проблемы их использования на практике.

Ключевые слова: *нейронные сети, машинное обучение, глубокое обучение, оптимизация, градиентный спуск, адаптивная скорость обучения*

Введение

Методы оптимизации являются востребованными при решении различных научных и инженерных задач, на практике возникают новые проблемы оптимизации, и их сложность растет. Одной из основных составляющих машинного обучения также является математическая оптимизация. С точки зрения машинного обучения основная цель оптимизации заключается в минимизации или максимизации значения математической функции (целевой функции, функции потерь или функции стоимости). Методы оптимизации составляют совершенно отдельную область исследований, которая почти столь же раз-

[1] Исследования выполнены при финансовой поддержке стипендии Президента Российской Федерации молодым ученым и аспирантам (СП-2578.2018.5).

нообразна, как и область самого машинного обучения.

Целью данной статьи является систематизация информации о различных методах оптимизации, которые нашли широкое применение в машинном обучении, и о новых менее известных подходах. Обзор включает краткое описание метода обратного распространения ошибки — одного из методов обучения нейронных сетей, метода оптимизации на основе градиентного спуска и возникающих при их использовании проблем сходимости. В последующих разделах статьи рассмотрены методы: градиентный спуск с импульсом, быстрый градиентный метод Нестерова, AdaGrad, RMSprop, AdaDelta, Adam, AdaMax, Nadam, AMSGrad, ND-Adam, NosAdam, Padam и Yogi.

1. Обратное распространение ошибки

Чтобы определить место и роль математической оптимизации в машинном обучении, сначала необходимо рассмотреть один из методов обучения нейронных сетей: метод обратного распространения ошибки (англ. backpropagation) — градиентный метод обновления весов многослойных нейронных сетей прямого распространения (многослойных персептронов) [1—3], представляющий собой итеративный процесс минимизации ошибки работы многослойного персептрона в целях получения выхода, близкого к желаемому.

При обучении нейронной сети методом обратного распространения ошибки предполагается, что для каждого входного вектора задан желаемый выходной вектор. Данные векторы образуют обучающую пару, а для обучения используется множество пар. Метод обратного распространения ошибки включает в себя прямой и обратный проходы по всем слоям сети. Прямой проход предполагает подачу на входной слой нейронной сети входного вектора сигналов и дальнейшее его распространение от слоя к слою. В результате формируется вектор выходных сигналов, который вычитается из желаемого результата. Полученная ошибка распространяется в направлении, обратном прямому распространению сигналов, т. е. от выходного слоя к входному, при этом весовые коэффициенты нейронов (параметры сети) корректируются. Для возможности применения метода обратного распространения ошибки передаточная (активационная) функция нейронов должна быть дифференцируема.

Следует отметить, что при использовании метода обратного распространения ошибки процесс обучения может требовать больших временных затрат, также процесс обучения может фактически остановиться. Так как в алгоритме обучения применяется разновидность градиентного спуска, то одной из проблем является наличие точек, в которых процесс обучения останавливается, хотя рядом может находиться оптимальное решение. Такой точкой может быть локальный минимум или седловая точка (стационарная для заданной функции, однако не являющаяся экстремумом, значение функции является максимальным в одном измерении и минимальным в другом).

Кроме того, необходимо уделять внимание длине шага обучения. Доказано, что метод обратного распространения ошибки обладает сходимостью при условии, что коррекция весовых коэффициентов бесконечно мала, однако при малом шаге обучения скорость сходимости будет также мала, а время обучения будет стремиться к бесконечности. При очень большой длине шага процесс обучения оказывается неустойчивым, сходимость может не достигаться, возникает паралич сети.

2. Метод градиентного спуска

Рассмотрев основные принципы метода обратного распространения ошибки, можно сделать вывод, что каждый параметр сети вносит свой вклад в ошибку. Процесс обратного распространения ошибки напрямую не корректирует весовые коэффициенты, а только перемещает информацию с выхода сети. За изменение весов отвечают другие методы, решающие задачу оптимизации. Базовым методом является градиентный спуск. Применяются разные подходы к реализации градиентного спуска для коррекции параметров сети:

1) стохастический градиентный спуск: параметры сети меняются после расчета ошибки для каждой обучающей пары;

2) пакетный (англ. batch) градиентный спуск: параметры сети корректируются после прохождения всех обучающих пар;

3) градиентный спуск с использованием мини-пакетов: множество обучающих пар разбивается на группы.

Для корректировки параметров используется одна из разновидностей метода градиентного спуска. Задачу обычного градиентного спуска можно сформулировать в следующем виде [4, 5]:

$$x_{i+1} = x_i - \lambda_i \nabla F(x_i), \quad (1)$$

где i — номер итерации обучения; x — веса (параметры) сети; $F(x)$ — целевая функция или функция потерь; λ — коэффициент, влияющий на значение шага обучения. Значение λ может быть задано константой (обучение с постоянным шагом) или может меняться на каждой итерации (например, в методе наискорейшего спуска [4]).

Представленные в явном виде формулы для обновления параметров современных многослойных сетей могут выглядеть крайне сложно, так как каждый нейрон связан с множеством других нейронов и зависит от них. Возникает ряд проблем:

1) сложная и неоднородная поверхность функции потерь, которая может зависеть от миллионов параметров. Имеется большое число локальных минимумов и седловых точек;

2) зависимость от длины шага обучения (долгое обучение, зависание в локальных минимумах или седловых точках либо постоянная неустойчивость);

3) проблема разреженных данных. Ожидаемые значения могут отсутствовать в наборе, а значимые признаки могут встречаться редко, но учет всех редких признаков может привести к переобучению.

В теории оптимизации существуют методы второго порядка (Ньютона [6] и др.), способные найти оптимальное решение при сложной поверхности функции потерь, однако число параметров значительно увеличивает вычислительную сложность. Чтобы воспользоваться методом второго порядка, необходимо для функции $F(x)$ вычислить матрицу Гессе и обратную ей (для метода Ньютона).Отдельно сто́ит выделить квазиньютоновские методы, которые основаны на приближенных выражениях для матрицы Гессе (например, метод Бройдена—Флетчера—Гольдфарба—Шанно (BFGS) [7]).

В данной статье будут рассмотрены различные вариации метода градиентного спуска, которые позволяют справиться с названными проблемами без значительного увеличения вычислительной сложности и не прибегать к использованию вторых производных.

3. Импульс и быстрый градиентный метод Нестерова

Быстрый градиентный метод Нестерова (англ. Nesterov Accelerated Gradient, NAG) является одним из методов, которые ускоряют работу градиентного спуска в соответствующем направлении и уменьшают колебания в областях с локальным минимумом [8].

В подобных методах (Momentum [9]) вводится понятие импульса. Идея накопления импульса проста: если некоторое время происходит движение в определенном направлении, то в будущем некоторое время также следует двигаться в этом направлении. Для реализации данного подхода можно хранить N последних изменений каждого параметра и на каждой итерации считать среднее. Для экономии памяти используется экспоненциальная фильтрация:

$$E[\nabla F(x)]_{i+1} = \alpha E[\nabla F(x)]_i + (1-\alpha)\nabla F(x_{i+1}), \quad (2)$$

где $0 \leqslant \alpha \leqslant 1$ — коэффициент сохранения экспоненциального скользящего среднего E последовательности величин (градиентов). Значение коэффициента α обычно берется близким к 1.

Задача оптимизации с учетом импульса (классический подход):

$$x_{i+1} = x_i + \mu_i v_i - \lambda_i \nabla F(x_i), \quad (3)$$

где v вычисляется по следующей формуле:

$$v_{i+1} = \mu_i v_i - \lambda_i \nabla F(x_i), \quad (4)$$

где $0 \leqslant \mu \leqslant 1$ — коэффициент сохранения импульса (эквивалент α). При этом коэффициент $(1 - \mu)$ может быть учтен в значении λ.

В методе Нестерова также используется идея экстраполяции (предсказания), и градиент вычисляется в другой точке:

$$x_i = y_i - \lambda_i \nabla F(y_i), \quad (5)$$

$$y_{i+1} = x_i + \mu_i (x_i - x_{i-1}). \quad (6)$$

Важно отметить, что минимизация выполняется относительно параметров, обозначенных y, но искомым вектором параметров нейронной сети также является x.

Коэффициент μ может быть константой, близкой к 1. Также встречаются варианты, использующие обновление μ:

$$\mu_i = (a_i - 1)/a_{i+1}, \quad a_{i+1} = (1 + \sqrt{4a_i^2 + 1})/2; \quad (7)$$

$$\mu_i = i/(i+3); \quad (8)$$

$$\mu_i = 1 - 3/(5+i). \quad (9)$$

4. Метод AdaGrad (Adaptive Gradient)

Метод AdaGrad предполагает подстройку шага обучения для каждого параметра сети [10]. Иначе говоря, скорость обучения адаптируется, выполняя значительные обновления для редких признаков и малые обновления для часто встречающихся. Считается сравнительно простым методом оптимизации, хорошо подходит для обучения на разреженных данных.

Обозначим параметр сети $x_{i,k}$, где k — номер параметра. Так как каждый параметр обновляется отдельно, то

$$x_{i+1,k} = x_{i,k} - \lambda \nabla F(x_{i,k}). \qquad (10)$$

Рассматриваемый метод использует предобусловливание. Вводится диагональная матрица предобусловливания, основанная на внешнем произведении векторов:

$$G_i = \operatorname{diag}\left(\sum_{j=1}^{i} \nabla F(x_j) \cdot \nabla F(x_j)^{\mathrm{T}} \right), \qquad (11)$$

где каждый диагональный kk-й элемент является суммой квадратов градиентов.

Запишем правило обновления k-го параметра:

$$x_{i+1,k} = x_{i,k} - \frac{\lambda}{\sqrt{G_{i,kk}}} \nabla F(x_{i,k}). \qquad (12)$$

На практике, чтобы избежать деления на ноль, в формулу добавляют коэффициент ε (равный, например, 10^{-8}):

$$x_{i+1,k} = x_{i,k} - \frac{\lambda}{\sqrt{G_{i,kk} + \varepsilon}} \nabla F(x_{i,k}). \qquad (13)$$

Учитывая, что введенная матрица содержит значения для всех параметров x_i, можно представить G_i в виде вектора, и общее правило обновления параметров сети записать с использованием поэлементного произведения ($\odot$) вектора скорректированных коэффициентов и вектора градиентов:

$$x_{i+1} = x_i - \frac{\lambda}{\sqrt{G_i + \varepsilon}} \odot \nabla F(x_i). \qquad (14)$$

Замечание: в ходе дальнейшего изложения при умножении векторов будет иметься в виду поэлементное произведение векторов.

Преимуществом AdaGrad является автоматическая подстройка скорости обучения, нет необходимости в точном подборе λ. Недостаток заключается в увеличении суммы квадратов градиентов в процессе обучения. В итоге скорость обучения становится бесконечно малой.

5. Метод RMSProp (Root Mean Square Propogation)

Метод RMSProp был предложен Джеффом Хинтоном в его онлайн-лекции [11]. Работа метода направлена на устранение стремительного и монотонного уменьшения скорости обучения с использованием AdaGrad.

Для описания работы метода воспользуемся формулой экспоненциального фильтра (2), заменив градиент $g = \nabla F(x)$ квадратом градиента g^2. Знаменатель в формуле (14) меняется на вектор

$$RMS[g]_i = \sqrt{E[g^2]_i + \varepsilon}. \qquad (15)$$

При этом автор фиксировал значение коэффициента сохранения экспоненциального скользящего среднего $\alpha = 0{,}9$.

6. Метод AdaDelta

AdaDelta — модификация метода AdaGrad, которая также позволяет исправить недостаток этого метода [12]. RMSprop и AdaDelta были разработаны независимо друг от друга в одно время, что было связано с необходимостью радикально уменьшить время обучения при использовании AdaGrad.

Вместо накопления квадратов градиента за все время (по текущую i-ю итерацию) предложено ограничивать историю окном фиксированного размера w. Таким образом, за большое число итераций скорость обучения не замедлится.

AdaDelta отличается от RMSprop числителем в формуле (14). Запишем следующие выражения:

$$\Delta x_i = -\frac{\lambda}{RMS[\nabla F(x)]_i} \nabla F(x_i); \qquad (16)$$

$$E[\Delta x^2]_i = \alpha E[\Delta x^2]_{i-1} + (1-\alpha)\Delta x_i^2; \qquad (17)$$

$$RMS[\Delta x]_i = \sqrt{E[\Delta x^2]_i + \varepsilon}. \qquad (18)$$

Тогда правило обновления параметров сети (14) принимает вид

$$x_{i+1} = x_i - \frac{RMS[\Delta x]_{i-1}}{RMS[\nabla F(x)]_i} \odot \nabla F(x_i). \qquad (19)$$

Для AdaGrad, RMSProp и AdaDelta не нужно очень точно подбирать темп обучения λ. Для AdaDelta этот коэффициент не надо задавать ввиду его отсутствия в правиле обновления параметров сети, а для RMSProp рекомендуется $\lambda = 10^{-3}$. Коэффициент α обычно советуют оставить равным 0,9 или 0,95.

7. Метод Adam (Adaptive moment estimation)

Adam также является методом оптимизации, который адаптивно подбирает темп обучения для каждого параметра [13]. Как отмечают авторы, он сочетает в себе преимущества методов AdaGrad и RMSProp. Скорости обновления параметров адаптируются на основе экспоненциального скользящего среднего значений градиентов (первый момент градиента) $g_i = \nabla F(x_i)$:

$$m_i = \alpha_1 m_{i-1} + (1 - \alpha_1) g_i \tag{20}$$

и экспоненциального скользящего среднего квадратов градиентов (второй момент градиента):

$$v_i = \alpha_2 v_{i-1} + (1 - \alpha_2) g_i^2. \tag{21}$$

Авторы Adam отмечают, что оценки моментов могут быть смещены к нулю, так как изначально инициализируются нулями. Особенно это проявляется на начальных итерациях обучения и при гиперпараметрах α_1 и α_2, близких к 1 (рекомендованные авторами значения: $\alpha_1 = 0{,}9$, $\alpha_2 = 0{,}999$). Для решения проблемы вычисляются скорректированные оценки первого и второго моментов:

$$\widehat{m}_i = m_i / (1 - \alpha_1^i), \quad \hat{v}_i = v_i / (1 - \alpha_2^i). \tag{22}$$

Правило обновления параметров сети имеет следующий вид:

$$x_i = x_{i-1} - \lambda \widehat{m}_i / (\sqrt{\hat{v}_i} + \varepsilon), \tag{23}$$

где гиперпараметры $\lambda = 10^{-3}$, $\varepsilon = 10^{-8}$ (предложены авторами Adam).

Авторами эмпирически доказано, что данный метод устойчив и хорошо подходит для широкого круга задач оптимизации в машинном обучении.

8. Метод AdaMax

AdaMax является модификацией метода Adam, предложенной в той же статье [13]. Отмечается, что правило обновления параметров в Adam основано на $L2$-норме текущего и прошлых значений градиентов и предлагается сформулировать правило на основе Lp-нормы. Формула для v_i принимает вид:

$$\begin{aligned} v_i &= \alpha_2^p v_{i-1} + (1 - \alpha_2^p) |g_i|^p = \\ &= (1 - \alpha_2^p) \sum_{j=1}^{i} \alpha_2^{p(i-j)} |g_i|^p . \end{aligned} \tag{24}$$

Теперь пусть p стремится к бесконечности:

$$\begin{aligned} u_i &= \lim_{p \to \infty} (v_i)^{1/p} = \\ &= \max(\alpha_2^{i-1} |g_1|, \alpha_2^{i-2} |g_2|, \ldots, \alpha_2 |g_{i-1}|, |g_i|) = \\ &= \max(\alpha_2 u_{i-1}, |g_i|). \end{aligned} \tag{25}$$

Вычисление второго момента v_i заменяется вычислением экспоненциально взвешенной бесконечной нормы u_i. Правило обновления параметров сети принимает следующий вид:

$$x_i = x_{i-1} - \lambda \widehat{m}_i / u_i, \tag{26}$$

где $\lambda = 2 \cdot 10^{-3}$.

Авторы утверждают, что хотя для больших p метод и становится нестабилен, но для значения p, стремящегося к бесконечности, метод дает положительные результаты.

9. Метод Nadam (Nesterov-accelerated Adam)

Метод Adam использует идеи RMSprop для адаптации скорости обучения и в то же время обладает импульсной компонентой. Классический подход на основе импульса уступает быстрому градиентному методу Нестерова (NAG). Исходя из этого был предложен метод Nadam, объединяющий в себе Adam и NAG [14]. Чтобы воспользоваться идеей NAG в методе Adam, необходимо изменить в нем импульсную компоненту m_i.

Во-первых, модифицируется NAG так, чтобы импульсная компонента напрямую участвовала в обновлении параметров сети. Слагаемые на основе импульса и градиента зависят от текущего градиента:

$$m_i = \alpha_{1,i} m_{i-1} + \lambda_i g_i; \tag{27}$$

$$x_i = x_{i-1} - (\alpha_{1,i+1} m_i + \lambda_i g_i). \tag{28}$$

Во-вторых, модифицируется метод Adam, и скорректированная оценка первого момента вычисляется следующим образом:

$$\widehat{m}_i = \frac{\alpha_{1,i+1} m_i}{1 - \prod_{j=1}^{i+1} \alpha_{1,j}} + \frac{(1-\alpha_{1,i}) g_i}{1 - \prod_{j=1}^{i} \alpha_{1,j}}. \qquad (29)$$

Заметим, что были учтены возможные изменения гиперпараметров α_1 и λ в зависимости от итерации, потому что их постепенное увеличение или уменьшение часто помогает работе метода. Однако в экспериментальных исследованиях использовались фиксированные значения: $\alpha_1 = 0{,}975$, $\lambda = 2 \cdot 10^{-3}$. Исходя из этого перепишем (29):

$$\widehat{m}_i = \frac{\alpha_1 m_i}{1 - \alpha_1^{i+1}} + \frac{(1-\alpha_1) g_i}{1 - \alpha_1^{i}}. \qquad (30)$$

Вычисления v_i и $\hat{v}_i$ не меняются и соответствуют формулам (21) и (22), $\alpha_2 = 0{,}999$. Правило обновления параметров сети соответствует виду (23).

Результаты исследований, представленные в работе [14], свидетельствуют о том, что Nadam повышает скорость сходимости и качество обучаемых сетей.

10. Методы AMSGrad, ND-Adam и NosAdam

Рассмотренные адаптивные методы оптимизации стали популярными при обучении нейронных сетей, но на практике было замечено, что в некоторых случаях, например, при обучении глубоких нейронных сетей, в задачах машинного перевода [15] и распознавания объектов [16] они не могут сходиться к оптимальному решению и проигрывают стохастическому градиентному спуску с импульсом.

В качестве причины плохой сходимости и ухудшения обобщающей способности нейронных сетей было названо экспоненциальное скользящее среднее квадрата градиента. С одной стороны, экспоненциальная фильтрация позволяла избежать падения скорости обучения, но с другой, такая краткосрочная память о градиентах оказалась проблемой, поскольку были приведены примеры, когда именно последние несколько значений градиента значительно ухудшали сходимость.

Для преодоления разрыва между стохастическим градиентным спуском и Adam с точки зрения обобщающей способности был предложен метод ND-Adam (Normalized Direction-preserving Adam) [17]. Во-первых, скорость обучения адаптируется для каждого вектора параметров сети, а не для каждого параметра, чтобы сохранить направление градиента. Во-вторых, каждый вектор нормализуется с помощью оптимизации параметров на основе сферы. Эмпирически показано, что производительность метода действительно может быть близка к стохастическому градиентному спуску с импульсом.

Авторы AMSGrad [18] предложили наделить метод долгосрочной памятью, чтобы повысить не только сходимость, но и производительность. Подобной идее следовали и авторы метода NosAdam (Nostalgic Adam) [19]. В NosAdam градиентам с прошлых итераций устанавливаются веса больше, чем для более новых градиентов. Ускорение сходимости объясняется тем, что оценка второго момента градиента v_i "забывается" медленнее, чем в Adam.

Наибольшее распространение получил метод AMSGrad, основное отличие которого от метода Adam в том, что для нормализации скользящего среднего значения градиента используется максимальное значение квадрата градиента v_i, а не его скользящее среднее (либо скорректированная оценка).

Значения m_i и v_i, как и в Adam, вычисляются согласно (20) и (21), но скорректированная оценка v_i вычисляется иначе:

$$\hat{v}_i = \max(\hat{v}_{i-1}, v_i). \qquad (31)$$

Запишем правило обновления параметров сети для AMSGrad:

$$x_{i+1} = x_i - \lambda_i m_i / \sqrt{\hat{v}_i}. \qquad (32)$$

В результате, даже если $v_{i-1,k} > g_{i,k}^2 > 0$, то скорость обучения не возрастает значительно, как это происходит у Adam (на практике это отрицательно сказывалось на работе метода). Эксперименты показали лучшую производительность AMSGrad на наборах CIFAR-10 и MNIST. В свою очередь, авторы NosAdam эмпирически показали превосходство своего метода для некоторых задач машинного обучения.

11. Методы Padam и Yogi

Одними из последних методов, предназначенных для улучшения работы адаптивных методов оптимизации при обучении глубоких нейронных сетей, являются методы Padam (Partially adaptive momentum estimation method) [20] и Yogi [21].

Авторы Padam рассматривают проблему ухудшения обобщающей способности при использовании адаптивных методов, причем речь идет не только о методе Adam, но и о AMSGrad. Фактически Padam модифицирует AMSGrad добавлением дополнительного гиперпараметра $\rho \in (0; 1/2]$ в правило обновления параметров сети (32):

$$x_{i+1} = x_i - \lambda_i m_i / \hat{v}_i^{\rho}. \quad (33)$$

Авторы называют ρ частично адаптивным гиперпараметром, причем при $\rho = 1/2$ метод повторяет AMSGrad, а при $\rho \to 0$ приближается к стохастическому градиентному спуску с импульсом. Рекомендованное значение: 1/8.

Метод Yogi является модификацией Adam. Авторы Yogi, во-первых, говорят о достижении лучшей сходимости при увеличении размера мини-пакетов с обучающими парами. Во-вторых, предлагают модификацию вычисления второго момента градиента:

$$v_i = v_{i-1} - (1 - \alpha_2)\text{sign}(v_{i-1} - g_i^2)g_i^2. \quad (34)$$

Оцениваемые моменты m_i и v_i не корректируются. В остальном метод повторяет оригинальный Adam:

$$x_{i+1} = x_i - \lambda_i m_i / (\sqrt{v_i} + \varepsilon). \quad (35)$$

Некоторое улучшение достигнуто методом Padam при обучении на наборах CIFAR-10 и CIFAR-100 моделей сетей ResNet и VGGNet. Для метода Yogi предоставлены обширные исследования для многих современных нейронных сетей (ResNet, DenseNet и др.). Показано, что Yogi достигает схожих с Adam или лучших результатов без особой настройки гиперпараметров.

Заключение

Были рассмотрены нашедшие применение в машинном обучении методы оптимизации первого порядка, такие как градиентный спуск с импульсом, быстрый градиентный метод Нестерова, AdaGrad, RMSprop, AdaDelta, Adam и его модификации, AMSGrad и др. Выбор наиболее подходящего метода нередко зависит от решаемой задачи и обучающего набора данных.

Исследования в данной области продолжаются, выявляются проблемы процесса обучения нейронных сетей. На основе анализа причин возникновения проблем предлагаются модификации методов оптимизации, а также другие методы первого и второго порядков. Можно отметить, что градиентный спуск, импульс и метод Нестерова являются базовыми для применяемых в обучении сетей методов AdaGrad, Adam и др., при этом подстройка скорости обучения на каждой итерации выполняется для каждого параметра отдельно. В более поздних работах отмечается ухудшение сходимости и обобщающей способности, связанное с использованием экспоненциального скользящего среднего (краткосрочная память о градиентах). Такие методы, как AMSGrad, NosAdam, Padam, направлены на решение данной проблемы и используют преимущества как Adam, так и стохастического градиентного спуска.

Список литературы

1. **Галушкин А. И.** Нейронные сети: основы теории. М.: Горячая линия-Телеком, 2010. 480 с.
2. **Werbos P. J.** Beyond regression: New tools for prediction and analysis in the behavioral sciences: Ph. D. thesis. Cambridge: Harvard University, 1974. 453 p.
3. **Rumelhart D. E., Hinton G. E., Williams R. J.** Learning Internal Representations by Error Propagation // Parallel Distributed Processing. 1986. Vol. 1. P. 318—362.
4. **Коршунов Ю. М.** Математические основы кибернетики: Учеб. пособие для вузов. 3-е изд. М.: Энергоатомиздат, 1987. 496 с.
5. **Глебов Н. И., Кочетов Ю. А., Плясунов А. В.** Методы оптимизации: Учеб. пособие. Новосибирск: Изд-во НГУ, 2000. 105 с.
6. **Mordecai A.** Nonlinear Programming: Analysis and Methods. New York: Dover Publications, 2003. 544 p.
7. **Fletcher R.** Practical methods of optimization. 2nd edition. New York: John Wiley & Sons, 1987. 456 p.
8. **Нестеров Ю. Е.** Алгоритмическая выпуклая оптимизация: дис. на соискание ученой степени доктора физ.-мат. наук. М.: МФТИ, 2013. 367 с.
9. **Qian N.** On the momentum term in gradient descent learning algorithms // Neural Networks: The Official Journal of the International Neural Network Society. 1999. Vol. 12 (1). P. 145—151.
10. **Duchi J., Hazan E., Singer Y.** Adaptive Subgradient Methods for Online Learning and Stochastic Optimization // Journal of Machine Learning Research. 2011. Vol. 12. P. 2121—2159.
11. **Hinton G. E.** Neural Networks for Machine Learning, lecture 6e // Department of Computer Science at the University of Toronto, 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (дата обращения: 15.04.19).
12. **Zeiler M. D.** ADADELTA: An Adaptive Learning Rate Method // arXiv:1212.5701. 2012. P. 1—6.
13. **Kingma D. P., Ba J. L.** Adam: A Method for Stochastic Optimization // Proc. International Conf. on Learning Representations. San Diego, 2015. P. 1—15.
14. **Dozat T.** Incorporating Nesterov Momentum into Adam // Proc. International Conf. on Learning Representations. San Juan, 2016. P. 1—4.
15. **Johnson M., Schuster M., Le Q. V., Krikun M. et al.** Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation // arXiv:1611.04558. 2016. P. 1—17.

16. **Huang G., Liu Z., Weinberger K. Q., Van der Maaten L.** Densely Connected Convolutional Networks // Proc. IEEE Conf. on Computer Vision and Pattern Recognition. Honolulu, 2017. P. 2261—2269.

17. **Zhang Z., Ma L., Li Z., Wu C.** Normalized Direction-preserving Adam // arXiv:1709.04546. 2017. P. 1—12.

18. **Reddi S. J., Kale S., Kumar S.** On the Convergence of Adam and Beyond // Proc. International Conf. on Learning Representations. Vancouver, 2018. P. 1—23.

19. **Huang H., Wang C., Dong B.** Nostalgic Adam: Weighing more of the past gradients when designing the adaptive learning rate // arXiv:1805.07557. 2018. P. 1—12.

20. **Chen J., Gu Q.** Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks // arXiv:1806.06763. 2018. P. 1—18.

21. **Zaheer M., Reddi S. J., Sachan D. et al.** Adaptive Methods for Nonconvex Optimization // Proc. 32nd Conf. on Neural Information Processing Systems. Montreal, 2018. P. 1—11.

M. D. Ershov, Ph. D. Student, e-mail: ershov.m.d@rsreu.ru,
Ryazan State Radio Engineering University, Ryazan, 390005, Russian Federation

First-Order Optimization Methods in Machine Learning

The problems arising in the training of multilayer neural networks of direct distribution due to the disadvantages of the gradient descent method are considered. A review of first-order optimization methods which are widely used in machine learning and less well-known methods is performed. The review includes a brief description of one of training methods for neural networks: the backpropagation method (also known as the backward propagation of errors). A separate section is devoted to the gradient descent optimization method and convergence problems arising from the use of the backpropagation method with gradient descent. The review considers the following first-order optimization methods with adaptive learning rate: gradient descent with momentum, Nesterov accelerated gradient method (NAG), AdaGrad, RMSprop, AdaDelta, Adam, AdaMax, Nadam, AMSGrad, ND-Adam, NosAdam, Padam, and Yogi. The features of each method and the problems of their use in practice are described. It can be noted that gradient descent, momentum and NAG are basis for the AdaGrad, Adam and other methods used in machine learning. In addition, the learning rate adjustment is performed for each parameter separately at each iteration of the neural network training. In later works the deterioration of convergence and generalization ability is described, which is associated with the use of the exponential moving average (a short-term memory of gradients). Such methods as AMSGrad, NosAdam, Padam are aimed at solving this problem and take advantage of both Adam and the stochastic gradient descent.

Keywords: *neural networks, machine learning, deep learning, optimization, gradient descent, adaptive learning rate*

Acknowledgements: The studies were carried out with financial support from the scholarship of the President of the Russian Federation to young scientists and graduate students (SP-2578.2018.5).

DOI: 10.17587/it.25.662-669

References

1. **Galushkin A. I.** Neural networks: theoretical basis, Moscow, Goryachaya liniya-Telekom, 2010, 480 p. (in Russian).

2. **Werbos P. J.** Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph. D. thesis, Cambridge, Harvard University, 1974, 453 p.

3. **Rumelhart D. E., Hinton G. E., Williams R. J.** Learning Internal Representations by Error Propagation, *Parallel Distributed Processing*, 1986, vol. 1, pp. 318—362.

4. **Korshunov Yu. M.** Mathematical basics of cybernetics. Textbook for universities, 3rd edition, Moscow, Energoatomizdat, 1987, 496 p. (in Russian).

5. **Glebov N. I., Kochetov Yu. A., Plyasunov A. V.** Optimization methods. Study guide, Novosibirsk, Publishing House of NSU, 2000, 105 p. (in Russian).

6. **Mordecai A.** Nonlinear Programming: Analysis and Methods, New York, Dover Publications, 2003, 544 p.

7. **Fletcher R.** Practical methods of optimization6 2nd edition, New York, John Wiley & Sons, 1987, 456 p.

8. **Nesterov Yu. E.** Algorithms for convex optimization. Doctor of phys.-mat. sciences thesis, Moscow, MIPT, 2013, 367 p. (in Russian).

9. **Qian N.** On the momentum term in gradient descent learning algorithms, *Neural Networks: The Official Journal of the International Neural Network Society*, 1999, vol. 12, no. 1, pp. 145—151.

10. **Duchi J., Hazan E., Singer Y.** Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, 2011, vol. 12, pp. 2121—2159.

11. **Hinton G. E.** Neural Networks for Machine Learning, lecture 6e, Department of Computer Science at the University of Toronto, 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (date of access: 15.04.19).

12. **Zeiler M. D.** ADADELTA: An Adaptive Learning Rate Method, *arXiv*:1212.5701, 2012, pp. 1—6.

13. **Kingma D. P., Ba J. L.** Adam: A Method for Stochastic Optimization, *Proc. International Conf. on Learning Representations. San Diego*, 2015, pp. 1—15.

14. **Dozat T.** Incorporating Nesterov Momentum into Adam, *Proc. International Conf. on Learning Representations. San Juan*, 2016, pp. 1—4.

15. **Johnson M., Schuster M., Le Q. V., Krikun M. et al.** Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation, *arXiv*:1611.04558, 2016, pp. 1—17.

16. **Huang G., Liu Z., Weinberger K. Q., Van der Maaten L.** Densely Connected Convolutional Networks, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition. Honolulu*, 2017, pp. 2261—2269.

17. **Zhang Z., Ma L., Li Z., Wu C.** Normalized Direction-preserving Adam, *arXiv*:1709.04546, 2017, pp. 1—12.

18. **Reddi S. J., Kale S., Kumar S.** On the Convergence of Adam and Beyond, *Proc. International Conf. on Learning Representations. Vancouver*, 2018, pp. 1—23.

19. **Huang H., Wang C., Dong B.** Nostalgic Adam: Weighing more of the past gradients when designing the adaptive learning rate, *arXiv*:1805.07557, 2018, pp. 1—12.

20. **Chen J., Gu Q.** Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks, *arXiv*:1806.06763, 2018, pp. 1—18.

21. **Zaheer M., Reddi S. J., Sachan D. et al.** Adaptive Methods for Nonconvex Optimization, *Proc. 32nd Conf. on Neural Information Processing Systems. Montreal*, 2018, pp. 1—11.