УДК 004.032.26
DOI: 10.46548/21vek-2021-1056-0003

МЕТОДИКА ОПТИМИЗАЦИИ ОБУЧАЮЩЕГО АЛГОРИТМА НЕЙРОННЫХ СЕТЕЙ

Зоткина Алена Александровна, аспирант кафедры «Программирование»
Мартышкин Алексей Иванович, кандидат технических наук, доцент,
заведующий кафедрой «Программирование»
Пензенский государственный технологический университет
(440039, Россия, Пенза, проезд Байдукова/ул. Гагарина, д. 1а/11,
e-mails: alena.zotkina.97@mail.ru, alexey314@yandex.ru)
Новоселова Ольга Викторовна, доцент кафедры «Общей и прикладной физики »
Национальный исследовательский Московский государственный строительный университет
(129337, Россия, г. Москва, Ярославское шоссе, 26, e-mail: olganovoselova51@rambler.ru)

Аннотация. В статье рассматриваются обучающие алгоритмы нейронных сетей. Приводится математическая модель задачи машинного обучения, главной целью которой является минимизация функции потерь. Рассмотрен классический подход к задаче минимизации функции потерь на основе метода градиентного спуска. Приведен пример алгоритма классического градиентного спуска. В статье обосновано проектирование обучаемого оптимизатора и рассмотрены модификации правил изменения параметров. Рассмотрен пример модифицированного алгоритм градиентного спуска. Обосновано использование рекуррентной нейронной сети в качестве оптимизатора. Выявлены недостатки базовой имплементации рекуррентных нейронных сетей и приведен пример использования подкласса рекуррентных сетей, способных запоминать значения как на короткие, так и на длинные промежутки времени *LSTM (Long Short Term Memory network)*. Отмечено, что в реализации оптимизатора используется высокоуровневый язык программирования Python с динамической строгой типизацией и автоматическим управлением памятью, в синтаксисе которого содержатся библиотеки *NumPy, Theano, Lasagne*. Выявлено, что проектирование алгоритма оптимизации может быть представлено в качестве задачи машинного обучения, позволяющей алгоритму научиться использовать структуру интересующих задач в автоматическом режиме. Приведен пример обучения оптимизатора и его реализации. В заключении сформулированы основные выводы по проделанной работе.

Ключевые слова: нейронная сеть, машинное обучение, рекуррентная нейронная сеть, оптимизатор, *Python, Long Short Term Memory network*.

TECHNIQUE FOR OPTIMIZING THE LEARNING ALGORITHM OF NEURAL NETWORKS

Zotkina Alena Aleksandrovna, postgraduate of department «Programming»
Martyshkin Alexey Ivanovich, candidate of technical sciences, associate professor,
head of the Department «Programming»
Penza state technological University
(440039, Russia, Penza, BaydukovProyezd / Gagarin Street, 1a/11,
e-mails: alena.zotkina.97@mail.ru, alexey314@yandex.ru)
Novoselova Ol'ga Viktorovna, associate professor of the department of general and applied physics
National Research Moscow State University of Civil Engineering
(129337, Russia, Moscow, Yaroslavskoe highway, 26, e-mail: olganovoselova51@rambler.ru)

Abstract. The article discusses training algorithms for neural networks. A mathematical model of a machine learning problem is presented, the main goal of which is to minimize the loss function. A classical approach to the problem of minimizing the loss function based on the gradient descent method is considered. An example of a classical gradient descent algorithm is given. The article substantiates the design of a trainable optimizer and considers modifications of the rules for changing parameters. An example of a modified gradient descent algorithm is considered. The use of a recurrent neural network as an optimizer has been substantiated. The drawbacks of the basic implementation of recurrent neural networks are revealed and an example of using a subclass of recurrent networks capable of storing values for both short and long periods of time LSTM (Long Short Term Memory network), is given. It is noted that the implementation of the optimizer uses a high-level programming language Python with dynamic strong typing and automatic memory management, the syntax of which contains the NumPy, Theano, Lasagne libraries. It was revealed that the design of an optimization algorithm can be represented in machine learning problems, allowing the algorithm to learn how to use the tasks of interest in an automatic mode. An example of training the optimizer and its implementation is given. In the conclusion, the main conclusions on the work done are formulated.

Keywords: neural network, machine learning, recurrent neural network, optimizer, Python, Long Short Term Memory network.

Введение. Большинство задач из области машинного обучения можно свести к задаче минимизации функции потерь, численное значение которой является оценкой достоверности предсказания для текущего обучающего или тестового примера. При этом цель обучения можно определить как подбор таких параметров функции потерь, при которых последняя с некоторой погрешностью достигает своего минимума. Оптимальные параметры достигаются в результате модификации исходных параметров путем последовательного применения определенного правила обновления. Решение подобных задач предполагает анализ и моделирование правил обновления, адаптированных к конкретным классам задач, причем типы представляющих интерес задач разнятся в различных исследовательских областях. Адаптация выполняется путем нахождения оптимальных гиперпараметров для конкретного подкласса задач, а повышения эффективности правил добиваются сужением этого подкласса. При этом ценой адаптации к структуре задач данной предметной области становится снижение эффективности применения правил обновления на задачах из другой предметной области. Другой подход к оптимизации предполагает замену разработанных вручную правил обновления правилами, сформированными в результате обучения нейронной сети.

Целью данной работы является разработка методики построения обучающего алгоритма, который работает на конкретном классе задач. Интерпретация моделирования алгоритма как задачи машинного обучения позволяет конкретизировать класс актуальных задач, используя для обучения правила соответствующий набор тренировочных данных.

Материалы и результаты исследования. *Математическая модель задачи машинного обучения.* Задача машинного обучения сводится к минимизации функции потерь. Представим нейронную сеть в виде математической модели [1– 3].

$X = (X_1, ..., X_k)$, где X_i – вектор, подающийся на вход нейронной сети $X_i \in R_n$;

$A = (A_1, ..., A_k)$, где A_i – правильный вектор, $A_i \in R_m$;

(X, A) – обучающая выборка;

W – вектор весов нейронной сети;

$N(W, X)$ – функция, соответствующая нейронной сети;

$Y = N(W, X)$ – ответ нейронной сети, $Y \in R_m$;

$D(Y, A)$ – функция ошибки;

$Y = N(W, X)$ — ответ нейронной сети, $Y \in R_m$;

$D_i(Y) = D(Y, A_i)$ — функция ошибки на i–ом примере;

$E_i(W) = D_i(N(W, X_i))$ – ошибка сети на i–ом примере;

$E(W) = \sum_{i=1}^{m} E_i(W)$ – ошибка сети на всей обучающей выборке.

Тогда задачу обучения нейронной сети можно интерпретировать следующим образом: найти вектор W – такой, что $E(W) \rightarrow min$ (обучение на всей выборке [1]). Обозначим функцию ошибки сети на всей обучающейся выборке как $f(W)$. Тогда решением задачи будет подбор таких параметров W, при которых функция $f(W)$ достигает своего минимума. Приведение параметров к нужному виду осуществляется путем последовательных итераций с изменением параметров по определенному правилу $W_{t+1} = z(W_t)$, где z –некоторое преобразование.

Классический подход к задаче минимизации функции потерь. Для дифференцируемых функций существует стандартный подход к задаче оптимизации обучающего алгоритма нейронных сетей при помощи машинного обучения – применение метода градиентного спуска, суть которого заключается в нахождении локального экстремума функции с помощью движения вдоль градиента [4, 5]. Особенность метода состоит в выполнении нескольких итераций обновления параметров W. Правилом обновления параметров является некая параметризованная функция, определяющая порядок сопоставления значений параметров на i-ой итерации t с их значениями на предшествующей $(t–1)$-ой итерации. В соответствии с методом градиентного спуска находится локальный минимум функции потерь, по правилу обновления параметров при этом из параметра W_i вычитается градиент функции потери по этому параметру $\nabla f(W_i)$, умноженный на подобранное вручную число ε, которое отображает скорость обучения.

Алгоритм классического градиентного спуска имеет следующий вид [6].

1. Инициализация вектора параметров W_i случайным значением из множества R^n;

2. Присвоение значения номера итерации на первом шаге: $i=1$;

3. Применение правила обновления параметров: $W_{i+1} = W_i - \varepsilon \nabla f(W_i)$;

4. Инкрементирование значения номера итерации: i++;

5. Прерывание выполнения алгоритма в случае, если очередной шаг не вызвал существенного улучшения решения:

if $f(W_i) - f(W_{i+1}) > c$ *goto* 3.

Данное правило обновления использует вычисление значения градиентов, но игнорирует информацию второго порядка – кривизну целевой функции. В связи с этим эффективность данного правила нельзя считать удовлетворительной. Классические методы оптимизации градиентного спуска предлагают способ решения этой проблемы путем вычисления матрицы частных производных второго порядка.

Метод градиентного спуска позволяет найти один из локальных минимумов, который может не быть глобальным минимумом, но все же отчасти оптимизировать веса и уменьшить значение функции потери.

Проектирование обучаемого оптимизатора. Модификация правила изменения параметров. Модернизируем правило изменения параметров: в качестве способа вычисления таких параметров W^*, при которых функция $f(W^*)$ достигает минимума, определим последовательность итераций изменения параметров

W по смоделированному правилу, обученному на тренировочных данных, которое назовем оптимизатором g. Пусть t – номер шага оптимизации. Тогда правило обновления параметров оптимизируемой функции можно записать как $W_{t+1}=W_t+g_t(\nabla f(W_t), \phi)$ где ϕ – набор собственных параметров оптимизатора g. Входящим параметром для оптимизатора g является градиент оптимизируемой функции f по ее собственным параметрам W на шаге t. Оптимизатор возвращает обновление для параметров W (рис. 1).

Таким образом, модифицированный алгоритм градиентного спуска выглядит следующим образом:

1. Вычисление функции потери $f(W_t)$, где t – номер шага обновления параметров оптимизируемой функции;
2. Вычисление градиента $f(W_t)$ по параметрам W_t;
3. Обновление параметров с помощью оптимизатора g: $W_{t+1}=W_t+g_t(\nabla f(W_t), \phi)$;
4. *goto* 1.

Рисунок 1 – Получение оптимизатором данных о производительности функции и обновления оптимизируемой функции для повышения производительности

Использование рекуррентной нейронной сети в качестве оптимизатора. Необходимо, чтобы при вычислении актуального обновления параметров правило оптимизации использовало не только информацию первого порядка (текущий градиент по параметрам), но и полезную информацию, полученную в ходе предшествующих итераций. Такая возможность появляется при использовании рекуррентных нейронных сетей – сетей, которые могут использовать свою внутреннюю память для обработки последовательностей произвольной длины [7, 8]. Минусом базовой имплементации рекуррентных нейронных сетей является то, что они не способны обрабатывать долгосрочные зависимости, поэтому необходимо использовать подкласс рекуррентных сетей *LSTM*, способный запоминать значения как на короткие, так и на длинные промежутки времени [9]. Будем считать, что обновление g_t является выходом нейронной сети *LSTM* – параметризованной ϕ на шаге t. Состояние сети обозначим как h_t.

Обучение и реализация оптимизатора. Параметры оптимизируемой функции: $W^*=argmin_w f(W_t)$ можно выразить через функцию параметров оптимизатора ϕ и оптимизируемую функцию f: $W^*=W^*(f, \phi)$. Чтобы обучить оптимизатор, необходимо иметь возможность вычислять производные функции потерь $L(\phi)$ по параметрам оптимизатора ϕ. Для этого необходимо развернуть процесс оптимизации во времени. Развертка может быть представлена в виде вычислительного графа, который состоит из итеративного применения оптимизатора для оптимизации параметров целевой функции. Вычисление градиентов для параметров оптимизатора включает в себя обратное распространение ошибки по данному развернутому вычислительному графу [10, 11]. Далее с помощью метода усеченного обратного распространения ошибки во времени вычисляем внешние градиенты на более коротких временных отрезках [12]. Для точного вычисления градиента необходимо воспроизвести обратное распространение ошибки по всем ребрам вычислительного графа без исключений. В реализации оптимизатора была использована двухслойная *LSTM* сеть с 20 скрытыми узлами на каждом слое. В качестве входного параметра нейронная сеть получает градиент оптимизируемой функции по одному параметру и скрытое состояние, соответствующее данному параметру. На выходе сеть выдает обновление для соответствующего оптимизируемого параметра (рис. 2).

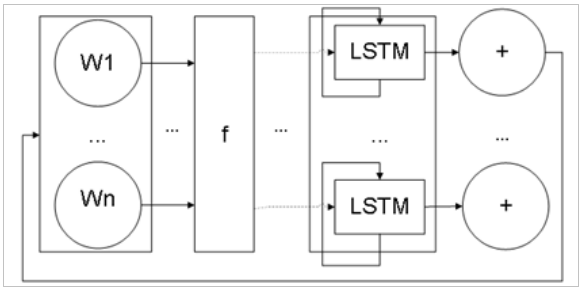

Рисунок 2 – Первая итерация используемого LSTM оптимизатора

В реализации оптимизатора используется высокоуровневый язык программирования *Python* с динамической строгой типизацией и автоматическим управлением памятью, в синтаксисе которого содержатся следующие библиотеки [13,14,]:

– *NumPy* – поддерживает многомерные массивы и матрицы вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами [16, 17];

– *Theano* – используется для быстрых численных вычислений, может быть запущена на *CPU* или *GPU*. Это ключевая базовая библиотека для глубокого обучения в *Python* [18];

– *Lasagne* – используется для создания и обучения нейронных сетей в *Theano*. Она поддерживает сети передачи данных, такие как сверточные нейронные сети (*CNN*), включая *LSTM* и любую их комбинацию [19, 20].

Заключение. В данной статье показано, как проектирование алгоритма оптимизации может быть представлено в качестве задачи машинного обучения, позволяющей алгоритму научиться использовать

структуру интересующих задач в автоматическом режиме. Отмечено, что благодаря использованию нейронной сети *LSTM* возможно реализовать модели оптимизаторов, эффективно работающие на конкретном классе оптимизационных задач.

СПИСОК ЛИТЕРАТУРЫ:

1. Зоткина А.А. Характеристики и математическое описание нейрона / Д.Э. Ильичов, Н.А. Лысцов, А.А. Зоткина // Наука и образование в современном обществе: актуальные вопросы и инновационные исследования: сборник статей III Международной научно-практической конференции. Пенза, 2021. – С. 28-30.

2. Зоткина А.А. Персептрон как простейший вид искусственной нейронной сети на примере построения однослойной модели сети / А.А. Зоткина, А.И. Мартышкин // Современные методы и средства обработки пространственно-временных сигналов: сборник статей XIX Всероссийской научно-технической конференции, посвященной 60-летию первого полета в космос Юрия Алексеевича Гагарина. Под редакцией И.И. Сальникова. Пенза, 2021. – С. 33-38.

3. Машинное обучение [Электронный ресурс] – URL: http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение (дата обращения: 19.10.2021).

4. Е.С.Борисов. О методах обучения многослойных нейронных сетей прямого распространения. Часть 3: Градиентные методы второго порядка [Электронный ресурс] – URL: http://mechanoid.su/neural-net-backprop3.html (дата обращения: 19.10.2021).

5. Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In Advances in Neural Information Processing Systems, arXiv:1606.04474v2 [cs.NE], 30 Nov. 2016.

6. Sepp Hochreiter, Arthur Steven Younger and Peter R. Conwell. Learning to learn using gradient descent. International Conference on Artificial Neural Networks. ICANN 2001: Artificial Neural Networks — ICANN 2001 pp 87-94. DOI:10.1007/3-540-44668-0_13.

7. Зоткина А.А Методы разработки искусственных нейронных сетей / Е.И. Маркин, И.А. Подопригора, А.А. Зоткина, Е.Г. Бершадская // Вестник современных исследований. – 2018. – № 3.1 (18). – С. 49-52.

8. A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. [Электронный ресурс] – URL: https://proceedings.mlr.press/v48/santoro16.pdf (дата обращения: 19.10.2021).

9. Conrad Tiflin. LSTM Recurrent Neural Networks for Signature Verification, 2012. 104 p.

10. Зоткина А.А. Основные характеристики и алгоритм обучения нейронных сетей / Д.Э. Ильичов, Н.А. Лысцов, А.А. Зоткина // Наука и образование в современном обществе: актуальные вопросы и инновационные исследования: сборник статей III Международной научно-практической конференции. Пенза, 2021. – С. 25-27.

11. B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. arXiv Report 1604.00289, 2016.

12. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations, 2015.

13. Маккинни, У. Python и анализ данных / У. Маккинни; перевод с английского А. А. Слинкина. – 2-ое изд., испр. и доп. – Москва: ДМК Пресс, 2020. – 540 с.

14. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес / Хабр [Электронный ресурс]. – URL: https://habr.com/ru/company/ods/blog/324402/ (дата обращения: 19.10.2021).

15. GitHub – deepmipt/tdl: Course "Theories of Deep Learning" [Электронный ресурс]. – URL: https://github.com/deepmipt/tdl (дата обращения: 19.10.2021).

16. Учебник по библиотеке NumPy: учитесь на примерах [Электронный ресурс]. – URL: https://pythonist.ru/uchebnik-po-biblioteke-numpy-uchites-na-primerah/ (дата обращения: 21.10.2021).

17. Учебник по NumPy - Визуализация примеров для быстрого изучения [Электронный ресурс]. – URL: https://python-scripts.com/numpy (дата обращения: 19.10.2021).

18. GitHub - PacktPublishing/Deep-Learning-with-Theano: Deep Learning with Theano, published by Packt [Электронный ресурс]. – URL: https://github.com/PacktPublishing/Deep-Learning-with-Theano (дата обращения: 19.10.2021).

19. Библиотеки для глубокого обучения Theano/Lasagne / Хабр [Электронный ресурс]. – URL: https://habr.com/ru/company/ods/blog/323272/ (дата обращения: 19.10.2021).

20. Welcome to Lasagne – Lasagne 0.2. dev1 documentation [Электронный ресурс]. – URL: https://lasagne.readthedocs.io/en/latest/ (дата обращения: 19.10.2021).

Статья поступила в редакцию 28.10.2021
Статья принята к публикации 07.12.2021