УДК 519.85

ИССЛЕДОВАНИЕ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ОПТИМИЗАЦИИ, ИСПОЛЬЗУЕМЫХ ПРИ ОБУЧЕНИИ НЕЙРОННЫХ СЕТЕЙ

И. Л. Каширина, М. В. Демченко

Воронежский государственный университет

Поступила в редакцию 30.10.2018 г.

Аннотация. Современные методы глубокого обучения нейронных сетей, по существу, заключаются в нахождении минимума некоторой непрерывной функции ошибки. В последние годы были предложены различные алгоритмы оптимизации, которые используют разные подходы для обновления параметров модели. Данная статья посвящена анализу наиболее распространенных методов оптимизации, применяющихся в задачах обучения нейронных сетей и формированию на основе выявленных свойств рекомендаций по выбору алгоритма для настройки нейронных сетей на различных наборах данных. В процессе анализа были рассмотрены различные реализации метода градиентного спуска, импульсные методы, адаптивные методы, квазиньютоновские методы, обобщены проблемы их использования, а также выявлены основные преимущества каждого из методов.
Ключевые слова: методы оптимизации, нейронные сети, метод градиентного спуска, стохастический градиент, квазиньютоновские методы, целевая функция ошибки.
Annotation. Modern methods of deep learning of neural networks consist in finding the minimum of some continuous error function. In recent years, various optimization algorithms have been proposed that use different approaches to update model parameters. This article is devoted to the analysis of the most common optimization methods used in the tasks of teaching neural networks and forming recommendations on the choice of an algorithm for setting up neural networks on different data sets based on the identified properties. In the process of analysis, various implementations of the gradient descent method, impulse methods, adaptive methods, quasi-Newtonian methods were considered, the problems of their use were generalized, and the main advantages of each method were identified.
Keywords: optimization methods, neural networks, gradient descent, stochastic gradient, quasi-Newton methods, global and local minimum, objective error function.

1. ВВЕДЕНИЕ

Современные нейросетевые методы относятся к числу наиболее востребованных и непрерывно развивающихся алгоритмов машинного обучения, применяемых в различных сферах практической деятельности. В связи с широкой областью их применения формируются разнообразные задачи, отличающиеся постановкой и типами входных данных: распознавание изображений, синтаксический анализ текстов, диагностика заболеваний [1] и др. В связи с постоянным совершенствованием существующих нейросетевых алгоритмов, отличающихся своими свойствами и особенностями реализации, зачастую возникает проблема определения наиболее эффективного метода минимизации функции ошибки, гарантирующего лучшие результаты при решении конкретной задачи.

Целью данной статьи является анализ основных особенностей данных алгоритмов и обобщение изученных результатов для дальнейшего обоснованного выбора в конкретных задачах машинного обучения.

Известно, что наиболее используемым методом обучения нейронных сетей является алгоритм обратного распространения ошибки, в котором минимизация целевой функции производится методом градиентного спуска

[2], т. е. на каждой итерации изменение весов происходит по формуле

$$w_{N+1} = w_N - \alpha \nabla_w E(w), \qquad (1)$$

где E – целевая функция ошибки, зависящая от параметров: w – весовых коэффициентов нейронной сети, α – скорости обучения. Таким образом, в данном методе веса нейронной сети обновляются в направлении, противоположном направлению градиента целевой функции с шагом, определяемым скоростью обучения. Подробная реализация алгоритма приведена в [3].

В качестве достоинств метода градиентного спуска можно выделить, в первую очередь, простоту реализации, а также тот факт, что метод гарантированно сходится к глобальному или локальному минимуму для выпуклых и невыпуклых функций соответственно. Однако существует множество недостатков данного метода, вследствие которых данный метод редко применяется в реальной практике.

1. Градиентный спуск может быть очень медленным на больших наборах данных, т. к. на каждой итерации вычисляется градиент для всех векторов обучающего набора.

2. Не позволяет обновлять модель «на лету» и добавлять в процессе обучения новые примеры обучающей выборки также по причине того, что обновление весов целевой функции производится сразу для всего исходного набора данных.

3. Для невыпуклых функций существует проблема попадания в локальные минимумы, т. к. метод гарантирует точное решение только для выпуклых целевых функций ошибки.

4. Выбор оптимальной скорости обучения может оказаться сложной проблемой. Слишком маленькая скорость обучения может привести к очень медленной сходимости, напротив, большая скорость обучения может препятствовать сходимости, и как следствие функция ошибок будет колебаться вокруг минимума и не достигнет его.

5. Равномерное обновление всех параметров с одинаковой скоростью обучения приводит к ухудшению качества обучения в случае, если исходный набор данных не является сбалансированным, т. е. в выборке существуют классы, представленные меньшим числом объектов.

Современные пакеты машинного обучения используют различные вариации классического метода градиентного спуска, обладающие более высокой производительностью и точностью в реальных практических задачах за счет реализованных в них механизмов решения вышеперечисленных проблем. Однако зачастую пользователь уже реализованных методов обучения нейронных сетей использует встроенные оптимизационные алгоритмы в режиме «черного ящика», т. е. не обладает достаточной информацией об особенностях поведения доступных для использования методов на рассматриваемом наборе данных. Так, например, классификатор MLPClassifier из популярной библиотеки Scikit-learn машинного обучения на языке Python предоставляет пользователю выбор из нескольких методов: SGD (стохастический градиент), Adam (метод адаптивной оценки моментов), L-BFGS (квазиньютоновский алгоритм Бройдена – Флетчера – Гольдфарба – Шанно с ограниченным использованием памяти) [4]. Наиболее востребованная сегодня при обучении нейронных сетей библиотека Keras, написанная на языке Python, включает алгоритмы SGD, RMSprop, Adagrad, Adadelta, Adam, Nadam, Adamax [5]. Причем пользователь должен не только выбрать алгоритм оптимизации, но и, в общем случае, отрегулировать значения настроечных параметров алгоритма, что затруднительно сделать без понимания особенностей этих методов.

В данной работе проведено исследование и анализ свойств методов градиентного спуска и квазиньютоновских методов, а также сформулированы условия их применения в различных практических задачах по обучению нейронных сетей.

2. МАТЕРИАЛЫ И МЕТОДЫ

2.1. Различные реализации метода градиентного спуска

2.1.1. Стохастический градиент (SGD)

Алгоритм стохастического градиентного спуска [6, 7] предполагает обновление весовых коэффициентов нейронной сети с использованием единственного примера i обучающей выборки на каждом шаге.

$$w_{N+1} = w_N - \alpha \nabla_w E(w; x^{(i)}; y^{(i)}), \qquad (2)$$

где $(x^{(i)}, y^{(i)})$ – i-й обучающий набор.

SGD не производит лишних вычислений, т. к., в отличие от классического градиентного спуска, функция ошибок алгоритма считается не по всей обучающей выборке, а только по одному примеру, а следовательно, алгоритм обучается значительно быстрее, а также допускает возможность обучения «на лету», т. е. новые примеры могут подаваться на вход сети непосредственно в процессе обучения.

Однако вследствие того, что на каждом шаге SGD вычисление градиента производится на основе различных примеров исходного набора данных, обновления весовых коэффициентов сопровождаются частыми колебаниями целевой функции, как показано на рис. 1. Таким образом, с одной стороны, SGD позволяет быстро перемещаться к потенциально лучшим локальным минимумам, а с другой стороны, большие колебания значительно замедляют сходимость. Однако было доказано, что если ввести в процесс обучения динамическое уменьшение скорости обучения, то SGD достигает решения с точностью, аналогичной классическому градиентному спуску [7].

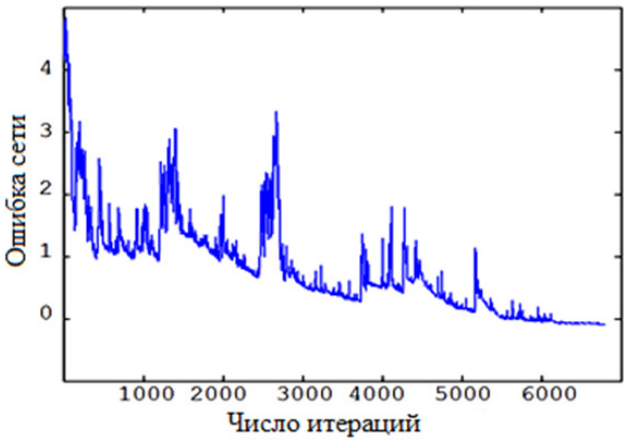

Рис. 1. Колебания целевой функции в зависимости от итерации обучения в методе SGD

2.1.2. Мини-пакетный градиентный спуск

Мини-пакетный градиентный спуск производит вычисление градиента для каждого мини-пакета, состоящего из n обучающих примеров, сочетая в себе преимущества классического и стохастического градиентных методов:

$$w_{N+1} = w_N - \alpha \nabla_w E(w; x^{(i:i+n)}; y^{(i:i+n)}). \qquad (3)$$

Вычисление градиента на мини-пакете размерностью от 50 до 256 обычно производится очень эффективно за счет используемых в современных библиотеках глубокого обучения высокопроизводительных матричных операций. Также, за счет использования не отдельных примеров, а частичных наборов векторов исходной выборки, данный метод обеспечивает существенно более стабильную сходимость, по сравнению с SGD.

2.2. Импульсные методы

Одним из известных недостатков метода градиентного спуска является проблема выбора локального минимума в качестве оптимальной точки в случае невыпуклой целевой функции. Данная проблема является довольно распространенной в задачах машинного обучения, т. к. зачастую функция ошибок является многоэкстремальной, что приводит к ошибочным результатам в результате работы классического или стохастического градиентного спуска.

Данная особенность учитывается в группе импульсных методов за счет накопления значений предыдущих градиентов, описываемых соотношением (4).

$$w_{N+1} = w_N + \mu \Delta w_{N-1} - \alpha \nabla_w E(w_N + \gamma \Delta w_{N-1}). \qquad (4)$$

При $\mu = \gamma$ формула (4) описывает метод Нестерова [8], а в случае $\gamma = 0$ – метод моментов, рассматриваемый далее в работе.

2.2.1. Метод моментов [9]

Стохастический градиент также зачастую не срабатывает в случае овражной целевой функции ошибки, т. е. такой функции, значение которой относительно одной из переменных изменяется существенно быстрее, чем в других направлениях. В таком случае значения выходов большинства нейронов задолго до конца обучения становятся близки к асимптотическим значениям функции активации, и возникает так называемый паралич сети, т. е. весовые коэффициенты практически перестают изменяться. В итоге обучение становится неприемлемо медленным. Простейшим усовершенствованием метода градиентного спуска является введение момента, когда влияние градиента на изменение весов накапливается со временем.

$$\begin{aligned} \Delta w_N &= \mu \Delta w_{N-1} - \alpha \nabla_w E(w), \\ w_{N+1} &= w_N + \Delta w_N. \end{aligned} \tag{5}$$

Таким образом, момент увеличивает скорость обновления, если градиент указывает в одном и том же направлении, и уменьшает, если градиент меняет направление. В результате такая модификация приводит к ускорению сходимости и сглаживанию колебаний.

2.3. Адаптивные методы

Несмотря на существенные улучшения производительности, импульсные методы не включают в себя встроенных механизмов оптимизации на несбалансированных выборках, т. е. таких наборах данных, в которых присутствуют редко встречающиеся признаки. Следствием данной проблемы является склонность таких алгоритмов к сходимости к локальным минимумам в случае невыпуклой целевой функции. Одним из возможных вариантов решения данной проблемы является динамическая модификация скорости обучения, реализованная в группе адаптивных оптимизационных алгоритмов (Adagrad, RMSProp, Adadelta, Adam). Проблема обучения нейронных сетей для решения задач классификации на несбалансированных выборках, в частности, рассматривалась в источниках [10,11]. Целью приведенных далее алгоритмов является повышение производительности процессов оптимизации путем устранения данной проблемы.

2.3.1. Адаптивный градиент (Adagrad [12])

Это алгоритм, в котором скорость обновления весовых коэффициентов нейронной сети адаптируется динамически, т. е. значимые обновления производятся для значений признаков, представленных в меньшинстве, а более слабые обновления – для часто встречаемых значений. Этот принцип реализуется за счет того, что скорость обучения в алгоритме Adagrad фактически вычисляется отдельно для каждого из параметров w_i на каждом шаге N. Обозначим

$$g_{N,i} = \nabla_{w_N} E(w_{N,i}). \tag{6}$$

Изменение весовых коэффициентов осуществляется по правилу (7):

$$w_{N+1,i} = w_{N,i} - \frac{\alpha}{\sqrt{G_{N,ii} + \varepsilon}} g_{N,i}, \tag{7}$$

где G_N – диагональная матрица, каждый диагональный элемент которой с индексами (i,i) представляет собой сумму квадратов частных производных по переменной w_i, вычисленных от начала работы алгоритма до шага N включительно, ε – сглаживающий фактор, позволяющий избежать деления на 0.

Запишем правило (7) в векторной форме:

$$w_{N+1} = w_N - \frac{\alpha}{\sqrt{G_N + \varepsilon}} g_N. \tag{8}$$

Один из главных преимуществ Adagrad является то, что он исключает необходимость настройки изменения скорости обучения вручную. Однако среди его недостатков можно выделить тот факт, что в процессе обучения знаменатель из правила (8) достаточно быстро возрастает, накапливая сумму квадратов градиентов. Как следствие, скорость обучения может уменьшаться и становиться бесконечно малой, т. е. алгоритм теряет свои обучающие свойства.

2.3.2. Adadelta, RMSProp

Adadelta [13] – это расширение Adagrad, в котором решается проблема стремительного уменьшения скорости обучения. Сумма квадратов градиентов в данном алгоритме заме-

няется на экспоненциально затухающее среднее всех предыдущих квадратов градиентов, то есть в большей степени учитываются последние значения частных производных.

$$E\left[g^2\right]_N = \gamma E\left[g^2\right]_{N-1} + (1-\gamma) g_N^2. \qquad (9)$$

Тогда правило обновления весовых коэффициентов принимает вид:

$$w^{N+1} = w^N - \frac{\eta}{\sqrt{E\left[g^2\right]_N + \varepsilon}} g_N. \qquad (10)$$

Так как знаменатель представляет собой корень из среднего квадратов градиентов (root mean square), обозначим знаменатель в формуле (10) следующим образом:

$$w^{N+1} = w^N - \frac{\eta}{RMS[g]_N} g_N. \qquad (11)$$

Задав значение $\gamma = 0.9$, получим правило обновления для алгоритма RMSProp (Hinton, 2012, [14]). Алгоритмы RMSProp и Adadelta были разработаны практически одновременно и независимо друг от друга, и являются похожими, за исключением того, что коэффициент скорости обучения в Adadelta заменяется на корень из среднего квадратов изменений веса. Правило обновления весов для Adadelta имеет вид:

$$w^{N+1} = w^N - \frac{RMS[\Delta w]_{N-1}}{RMS[g]_N} g_N. \qquad (12)$$

2.3.3. Метод адаптивной оценки моментов (Adam, [15])

Правило обновления весов для Adam определяется на основе использования оценок двух различных моментов (формулы (13) и (14)), в первом из которых используются вычисленные ранее значения частных производных (как в методе моментов), а во втором их квадраты (как в RMSProp). Метод Adam считается довольно устойчивым к выбору значений гиперпараметров β_1, β_2, и поэтому часто предлагается в качестве метода по умолчанию.

$$m_N = \beta_1 m_{N-1} + (1-\beta_1) g_N, \qquad (13)$$

$$v_N = \beta_2 v_{N-1} + (1-\beta_2) g^2{}_N. \qquad (14)$$

Вычисленные моменты корректируются по формулам (15), а затем производится пересчет весов по формуле (16):

$$\overline{m}_N = \frac{m_N}{1-\beta_1^N}, \quad \overline{v}_N = \frac{v_N}{1-\beta_2^N}; \qquad (15)$$

$$w^{N+1} = w^N - \frac{\eta}{\sqrt{\overline{v}_N + \varepsilon}} \overline{m}_N. \qquad (16)$$

Таким образом, существующие современные градиентные методы обладают множеством механизмов, позволяющих исключить такие проблемы алгоритмов оптимизации, как сходимость к локальным минимумам, неспособность распознавания редких признаков, а также адаптивную настройку оптимальной скорости обучения. Адаптивные методы подробно рассмотрены в источниках [16, 17].

2.4. Квазиньютоновские методы. BFGS, L-BFGS

Однако существует также группа методов второго порядка, основанных на вычислении вторых частных производных целевой функции ошибок. Такие методы обладают значительно более точной и быстрой сходимостью, однако являются более сложными в реализации и требуют больших затрат памяти. Рассмотрим класс квазиньютоновских методов, в которых обновление весовых коэффициентов происходит за счет вычисления оценки гессиана целевой функции ошибок, при этом формально они остаются методами первого порядка, так как прямого вычисления и обращения матрицы вторых частных производных они не производят.

2.4.1. Алгоритм Бройдена-Флетчера-Гольдфарба-Шанно (BFGS, [18])

Рассмотрим полную форму классического алгоритма BFGS. В алгоритме производится обновление оценки H_N обратного гессиана целевой функции, параметр η_N определяется с помощью процедуры одномерного поиска. Оценка сложности алгоритма BFGS составляет $O(n^2)$

Алгоритм классического метода BFGS

Шаг 1. $N := 0$;

Шаг 2. $H_0 = I$

Шаг 3. Пока $\left\|\nabla E(w_N)\right\| > \varepsilon$

1. $p_N = -H_N \nabla E(w_N)$

2. $\eta_N = \arg\min_{\eta} E(w_N + \eta p_N)$
3. $s_N = \eta_N p_N$
4. $w_{N+1} = w_N + s_N$
5. $y_N = \nabla E(w_{N+1}) - \nabla E(w_N)$
6. *если* $N = 0$: $H_N := \frac{s_N^T y_N}{y_N^T y_N} I$
7. $\sigma_N = \left(s_N^T y_N\right)^{-1}$
8. $H_{N+1} = \left(I - \sigma_N s_N y_N^T\right) H_N \left(I - \sigma_N y_N s_N^T\right) + \sigma_N s_N s_N^T$
9. $N := N + 1$

Шаг 4. w_N – *решение*

2.4.2. Алгоритм Бройдена – Флетчера – Гольдфарба – Шанно с ограниченным использованием памяти (L-BFGS [19])

Данный алгоритм является вариацией BFGS, разработанной специально для решения оптимизационных задач на больших объемах данных, в случае, когда сложность $O(n^2)$ является неприемлемой. Для L-BFGS оценка обратного гессиана производится только на основании данных последних m итераций. В данном методе движение в квазиньютоновском направлении осуществляется без использования матриц, путем формирования кольцевого буфера, содержащего m последних векторов s и y.

Для реализации данного метода требуется модифицировать алгоритм 1 следующим образом:

– исключить шаги 2 и 3.6 - 3.8;

– заменить шаг 3.1 алгоритмом изменения направлений L-BFGS.

Таким образом, сложность алгоритма уменьшается с $O(n^2)$ до $O(m \cdot n)$.

Алгоритм изменения направлений метода L-BFGS

Пусть $\forall i = 1, 2, ..., \min(t, m)$ s_{N-i}, y_{N-i} – из алгоритма BFGS.

1. $p_N = -\nabla E(w_N)$
2. Для $i = 1, 2, ... \min(t, m)$:
   a) $\alpha_i = \frac{s_{N-i} p_N}{s_{N-i} y_{N-i}}$
   b) $p_N := p_N - \alpha_N y_{N-i}$
3. если $N > 0$: $p_N := \frac{s_{N-1}^T y_{N-1}}{y_{N-1}^T y_{N-1}} p_N$
4. Для $i = \min(t, m), ..., 2, 1$
   a) $\beta = \frac{y_{N-i}^T p_N}{y_{N-i}^T s_{N-i}}$
   b) $p_N := p_N + (\alpha_i - \beta) s_{N-i}$

3. РЕЗУЛЬТАТЫ И ИХ ОБСУЖДЕНИЕ

3.1. Оценка способности к обобщению полученных решений

Оптимальное решение (набор весовых коэффициентов), полученное в ходе обучения нейронной сети, напрямую влияет на ее способность к обобщению, т. е. качество, с которым сеть выдает верные ответы на наборе примеров тестового множества, на которых не обучалась ранее. Однако в настоящее время точные выводы о способности нейронных сетей к обобщению находятся на стадии разработки и изучения [20]. Также многие современные исследования доказывают, что количество локальных минимумов функции ошибки обучения возрастает с числом независимых параметров, что в свою очередь, негативно сказывается на способности сети к обобщению (см. [21, 22]).

Для многих задач с большим числом параметров, адаптивные методы могут сходиться к решениям, кардинально отличающимся от решений, полученных в результате работы SGD. В настоящее время адаптивные алгоритмы часто являются методами по умолчанию во многих пакетах (например, Scikit-learn). Однако существуют исследования [23], подтверждающие низкую обобщаемость данных методов на наборах данных с большим количеством параметров (Adam, RMSProp, Adagrad), по сравнению с методом стохастического градиентного спуска. Данный недостаток объясняется тем, что алгоритмы с адаптивной скоростью обучения более склонны к переобучению, вследствие чего их погрешность на тестовой выборке может существенно повышаться.

3.2. Оценка производительности и точности рассматриваемых алгоритмов

Исходя из свойств и выявленных особенностей рассмотренных методов, сформируем следующие выводы о целях и рациональности использования алгоритмов SGD, Adam, L-BFGS, реализованных в библиотеке Scikit-learn, в реальных задачах на различных наборах данных.

1. Метод SGD стоит использовать на небольших сбалансированных наборах данных, в которых достаточно равномерно представлены элементы каждого класса. В случае несбалансированности исходной выборки, стохастический градиент не производит качественного распознавания редких значений признаков, также низкая скорость сходимости проявляется на большом объеме данных. Однако метод является наиболее простым для реализации, и может быть использован для разработки пользовательских оптимизаторов.

2. Оба метода Adam и L-BFGS являются эффективными методами, обладающими достаточно высокой скоростью и точностью решения. Однако производительность метода Adam зачастую является более высокой по сравнению с L-BFGS в случае решения задач оптимизации на больших наборах данных. В основе метода L-BFGS используется оценка гессиана целевой функции ошибок, при этом данный алгоритм обладает способностью оценки ландшафта поверхности целевой функции ошибок, характерной для большинства методов второго порядка, что, безусловно, делает L-BFGS наиболее эффективным в оптимизационных задачах, когда, например, поверхность целевой функции содержит множество оврагов. Но на больших наборах данных метод L-BFGS производит значительно больший объем затрат при обновлении оценки Гессиана, следовательно, часто в таких случаях Adam сходится к точному решению значительно быстрее.

3. Несмотря на то, что Adam является методом первого порядка, на каждой итерации метода в некотором смысле конструируется диагональ гессиана, однако оценка получается значительно более грубой по сравнению с L-BFGS, за счет невозможности учета внедиагональных элементов. Вследствие данной особенности алгоритм Adam может проигрывать L-BFGS в точности решения, однако, цена потери точности зависит непосредственно от условий и данных конкретной рассматриваемой задачи.

3.3. Сравнение практических результатов оптимизации с использованием различных алгоритмов

Рассмотрим теперь результаты решения задачи нейросетевой классификации изображений рукописных цифр из набора данных MNIST с использованием алгоритмов Momentum, Adam, Adadelta, Adagrad, SGD, реализованных в библиотеке Keras. Набор MNIST включает 60000 обучающих и 10000 тестовых примеров [24] и часто используется для тестирования и отладки различных нейросетевых алгоритмов. Данный вычислительный эксперимент позволил произвести оценку эффективности рассматриваемых оптимизационных алгоритмов на обучающей и тестовой выборках для нейронных сетей различных конфигураций, отличающихся количеством скрытых слоев и видом функций активации. Функции активации каждого скрытого слоя для сетей соответствующих конфигураций, разработанных с целью проведения эксперимента, указаны в табл. 1. В последнем слое для всех сетей использовалась функция Softmax,

Таблица 1

Конфигурации сетей, участвующих в вычислительном эксперименте

№ сети	Скрытый слой 1	Скрытый слой 2	Скрытый слой 3	Скрытый слой 4	Выходной слой
1	Логистическая	Линейная	–	–	Softmax
2	Гиперболический тангенс	Линейная	–	–	Softmax
3	Логистическая	ReLU	Логистическая	Линейная	Softmax

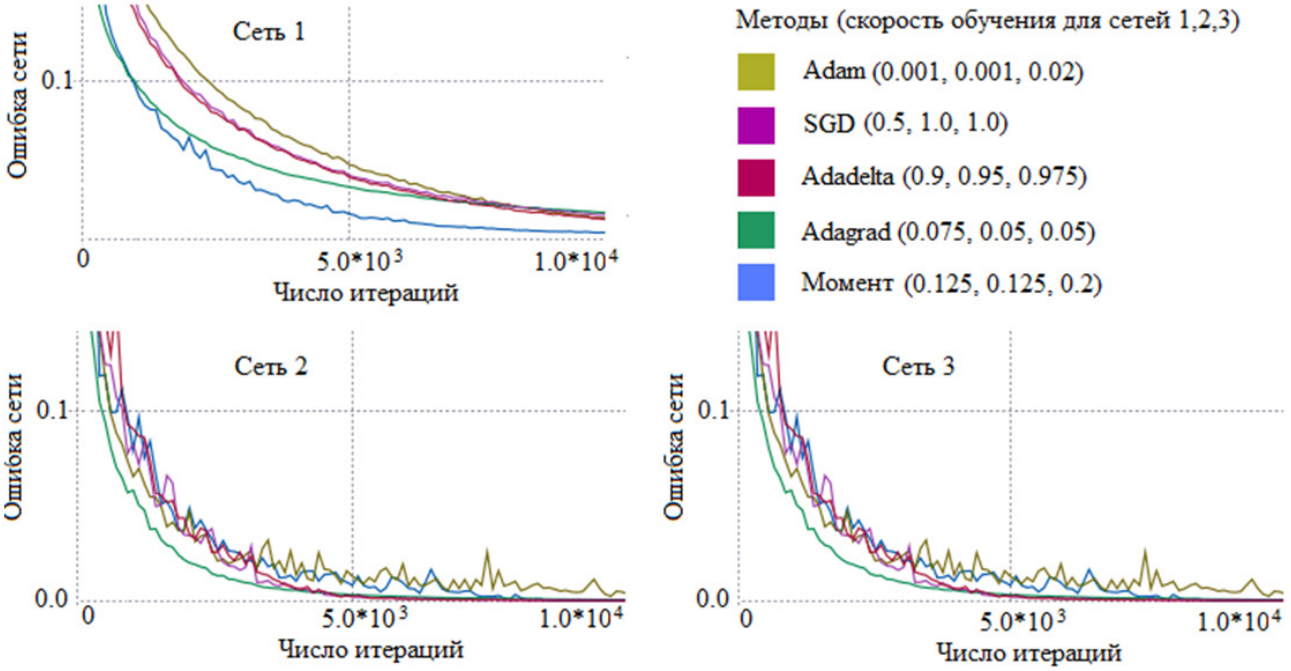

Рис. 2. Изменение ошибки обучения для рассмотренных сетей 1-3 в зависимости от выбранного алгоритма оптимизации

Таблица 2

Точность классификации на тестовой выборке рассматриваемых алгоритмов

№ сети	Momentum	Adam	Adadelta	Adagrad	SGD
1	**0,971**	0,9671	0,9698	*0,9664*	0,9701
2	0,966	0,9622	0,965	*0,9618*	**0,9672**
3	**0,9818**	0,9776	0,9787	*0,9747*	0,9787

так как это стандартная функция активации выходного слоя в задачах классификации.

Рис. 2 отражает значения функции ошибки сетей № 1-3 соответственно. На рис. 2 также приведены начальные значения скорости обучения, с которыми алгоритмы показали наилучшие результаты. В табл. 2 приведены значения точности классификации сетей № 1-3, обученных с использованием рассматриваемых методов, на тестовой выборке.

Результаты вычислительного эксперимента показали, что метод моментов и SGD проявили наилучшие результаты на данном наборе данных для сетей различных конфигураций. Adam и Adadelta, в свою очередь, являются стабильно эффективными, однако, не выигрывая, но и не сильно уступая в производительности другим методам.

Точность результатов, достигнутых в ходе работы SGD и метода моментов объясняется, в частности, тем, что исходная выборка является сбалансированной, что, в свою очередь, положительно сказывается на производительности этих методов.

Таким образом, исходя из данных наблюдений, следует сделать вывод о том, что результаты проведенных теоретических исследований подтверждаются данными эмпирических расчетов и могут быть использованы в качестве рекомендаций к применению рассмотренных оптимизационных методах в задачах машинного обучения.

4. ЗАКЛЮЧЕНИЕ

В ходе данной работы были рассмотрены основные методы оптимизации, применяемые в современных методах обучения нейронных сетей. В процессе исследования был проведен анализ свойств и особенностей рассматриваемых методов, а также сформулированы условия и обоснования их наиболее оптимального

применения в различных практических задачах с точки зрения их производительности и точности на обучающей и тестовой выборках. Данные анализа сопровождаются практическими результатами, подтверждающими сформулированные рекомендации по использованию рассмотренных методов классического и стохастического градиентного спуска, импульсных, адаптивных, а также квазиньютоновских алгоритмов оптимизации в задачах обучения нейронных сетей.

СПИСОК ЛИТЕРАТУРЫ

1. *Демченко, М. В.* Сравнительный анализ и оценка эффективности маркёров атеросклероза магистральных артерий / М. В. Демченко, И. Л. Каширина // Актуальные проблемы прикладной математики, информатики и механики : Сб. тр. Международ. науч.-тех. конференции, Воронеж, 18-20 декабря 2017 г. – Воронеж. : Изд-во «Научно-исследовательские публикации», 2017. – С. 636–643.

2. *Jordan, J.* Intro to optimization in deep learning: Gradient Descent/ J. Jordan // Paperspace. Series: Optimization. – 2018. – URL: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

3. *Каширина, И. Л.* Нейросетевые и гибридные системы: учебно-методическое пособие для вузов / И. Л. Каширина, Т. В. Азарнова. – Воронеж : Издательский дом ВГУ, 2014. – 80 с.

4. Scikit-learn – машинное обучение на Python. – URL: http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

5. Keras documentation: optimizers. – URL: https://keras.io/optimizers

6. *Ruder, S.* An overview of gradient descent optimization algorithms / S. Ruder // Cornell University Library. – 2016. – URL: https://arxiv.org/abs/1609.04747

7. *Robbins, H.* A stochastic approximation method / H. Robbins, S. Monro // The annals of mathematical statistics. – 1951. – Vol. 22. – P. 400–407.

8. *Нестеров, Ю. Е.* Метод минимизации выпуклых функций со скоростью сходимости O(1/k2) / Ю.Е. Нестеров // Докл. АН СССР. – 1983. – Т. 269, № 3. – С. 543–547.

9. *Поляк, Б. Т.* О некоторых способах ускорения сходимости итерационных методов / Б. Т. Поляк // Ж. вычисл. матем. и матем. физ. – 1964. – Т. 4, № 5. – С. 1–17.

10. *Kukar, M.* Cost-Sensitive Learning with Neural Networks / M. Kukar, I. Kononenko // Machine Learning and Data Mining : proceedings of the 13th European Conference on Artificial Intelligence. – 1998. – P. 445–449.

11. *Демченко, М. В.* Построение нейросетевого классификатора для выявления риска атеросклероза магистральных артерий / М. В. Демченко // Оптимизация и моделирование в автоматизированных системах : материалы всероссийской молодежной науч. школы, Воронеж, 13 декабря 2017 г. – Воронеж. : Изд-во Воронежский государств. технический университет, 2017. – С. 29–36.

12. *Duchi, J.* Adaptive Subgradient Methods for Online Learning and Stochastic Optimization / J. Duchi, E. Hazan, Y. Singer // The Journal of Machine Learning Research. – 2011. – Vol. 12. – P. 2121–2159.

13. *Zeiler, M. D.* ADADELTA: An Adaptive Learning Rate Method / Cornell University Library. – 2012. – URL: https://arxiv.org/abs/1212.5701

14. *Николенко, С.* Глубокое обучение / С. Николенко, А. Кадурин, Е. Архангельская. – СПб. : Питер, 2018. – 480 с.

15. *Kingma, D. P.* Adam: A Method for Stochastic Optimization / D. P. Kingma, J. Ba // Cornell University Library. – 2014. – URL: https://arxiv.org/abs/1412.6980

16. *Гудфеллоу, Я.* Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль. – М. : ДМК Пресс, 2018. – 652 с.

17. *Поляк, Б. Т.* Введение в оптимизацию / Б. Т. Поляк. – М. : Наука. Главная редакция физико-математической литературы, 1983. – 384 с.

18. *Fletcher, R.* Practical methods of optimization / R. Fletcher. – Wiley, 2000. – 450 p.

19. *Schraudolph, N. N.* A Stochastic Quasi-Newton Method for Online Convex Optimization / N.N. Schraudolph, J. Yu, S. Gunter // Statistical Machine Learning. – 2017. – URL: http://

proceedings.mlr.press/v2/schraudolph07a/schraudolph07a.pdf

20. *Ruder, S.* Optimization for Deep Learning Highlights in 2017 / S. Ruder // Optimization for Deep Learning Highlights in 2017. – 2017. – URL: http://ruder.io/deep-learning-optimization-2017.

21. *Kawaguchi, K.* Deep Learning without Poor Local Minima / K. Kawaguchi // Advances in Neural Information Processing Systems. – 2016. – URL: http://arxiv.org/abs/1605.07110

22. *Zhang, C.* Understanding deep learning requires rethinking generalization / C. Zhang, S. Bengio, S. Bengio, M. Hardt, B. Recht, O. Vinyals // Cornell University Library. – 2016. – URL: https://arxiv.org/abs/1611.03530

23. *Wilson, A. C.* The Marginal Value of Adaptive Gradient Methods in Machine Learning / A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, B. Recht // Cornell University Library. – 2017. – URL: https://arxiv.org/abs/1705.08292

24. The MNIST database. – URL: http://yann.lecun.com/exdb/mnist/

Каширина Ирина Леонидовна – д-р техн. наук, профессор кафедры математических методов исследования операций факультета ПММ Воронежского государственного университета.
E-mail: kash.irina@mail.ru

Демченко Мария Владимировна – аспирант факультета ПММ Воронежского государственного университета.
E-mail: masha-vrn@yandex.ru

Kashirina Irina Leonidovna – Doctor of Technical Sciences, Professor, Department of Mathematical Methods Operations Research, Faculty of Applied mathematics, Informatics and mechanics, Voronezh State University.
E-mail: kash.irina@mail.ru

Demchenko Maria Vladimirovna – postgraduate student of the Faculty of Applied mathematics, Informatics and mechanics, Voronezh State University.
E-mail: masha-vrn@yandex.ru