УДК 004.42

**МЕТОД ГРАДИЕНТНОГО СПУСКА И ЕГО МОДИФИКАЦИИ**

**К. А. Сермягин,** старший full-stack разработчик в TNWOIH LLC (https://vbrl.ai), студент РГРТУ, Рязань, Россия;

orcid.org/0009-0002-6078-4655, e-mail: sermyagin.abcen7@gmail.com.

**К. А. Ципоркова,** к.ф.-м.н., доцент кафедры ВМ РГРТУ, Рязань, Россия;

orcid.org/, e-mail: -

На текущий момент ИИ (искусственный интеллект) быстро развивается и является одной из передовой технологий в отрасли автоматизации рабочих процессов. За счет большого количества данных и требуемых мощностей алгоритмы оптимизации в ИИ требуют особого внимания и понимания их использования при проектировании и создании нейронных сетей. ***Целью работы*** является сравнение градиентного спуска и его модификаций (стохастического градиентного спуска) при помощи математического анализа с доказательством условий сходимости и исследования влияния параметров алгоритма на обучение нейронной сети.

***Ключевые слова:*** *Метод градиентного спуска, стохастический градиентный спуск, модификации градиентного спуска, искусственный интеллект, сходимость, нейронные сети, оптимизации.*

**Введение**

Градиентный спуск является основным методом оптимизации при обучении нейронных сетей. При помощи алгоритма обратного распространения ошибка вычисляется градиент функции потерь по отношению к весам сети, а затем эти веса обновляются в направлении, противоположном градиенту. Благодаря такому подходу это позволяет постепенно снижать значение функции потерь (loss function) и улучшать точность модели (accuracy).

**Теоретическая часть**

**Определение градиента функции.**

Рассмотрим функцию , ее градиент может быть определен как:

где — это полное бесконечно малое изменение для бесконечно малого смещения , и оно становится максимальным, когда находится в направлении градиента . Символ набла , записанный в виде перевернутого треугольника обозначает векторный дифференциальный оператор.

Когда используется система координат, в которой базисные векторы не зависят от положения, градиент определяется как вектор, чьи компоненты являются частными производными функции в точке . То есть, для функции её градиент в точке в *n*-мерном пространстве представляется вектором:

Градиент функции с количеством переменных определяется оператором набла .

Где:

* — это скалярное произведение градиента и произвольного вектора .
* — направленная производная функции в направлении .

Пример градиента функции для двух переменных с проецированием векторного поля:

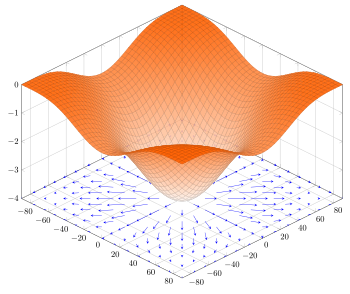


Рис. 1. Градиент функции

Градиент функции показывает направление подъема (в какую сторону требуется сделать шаг, чтобы функция возрастала быстрее), в случае же градиентного спуска решается задача нахождения вектора направления убывания функции, длина вектора градиента указывает на крутость спуска (см рис. 1).

**Градиентный спуск**

**Градиентный спуск (GD – gradient descent)** – это итеративный метод оптимизации, который используется для нахождения минимума функции . Естественным выбором направления поиска является отрицательный градиент [2].

**Алгоритм 1.0** *Метод градиентного спуска [2]*

Пусть дана начальная точка

1.

2. Линейный поиск. Выберите размер шага с помощью точного или возвратного линейного поиска.

3. Обновляем.

Пока не будет выполнен *критерий остановки.*

*Критерий остановки* обычно имеет вид , где η — малое положительное число. В большинстве реализаций это условие проверяется после шага 1, а не после обновления. Полученный алгоритм называется градиентным алгоритмом или методом градиентного спуска.

В контексте нейронных сетей данный подход используется для вычисления функции распространения обратной ошибки, что позволяет корректировать и обновлять веса (аргументы) функции на каждом этапе обучения новой эпохи.

Пусть у нейросети есть функция обратного распространения ошибки и вектор весов :

Тогда вектор смещения для вектора :

Здесь вектор смещения выступает направлением сдвига весов, которой вызовет максимальное убывание функции ошибки

Обучение нейронной сети – **это минимизация функции ошибки.** При нахождении локального минимума, важно, чтобы эта функция была непрерывной.

*Замечание. Большую роль в обучении многослойных нейронных сетей играет параметр структурированности данных, задействованных при ее обучении. Если данные структурированы, то найти локальные минимумы гораздо легче [1]*

**Сходимость GD**

Рассмотрим функцию и докажем сходимость GD [3].

1) Пусть — выпуклая и гладкая (градиент Липшицев с константой *L*):

Таким образом, градиент не меняется слишком резко.

2) На каждой итерации мы обновляем точку:

где α — шаг обучения.

Наша цель: показать, что неуклонно уменьшается, пока мы не приблизимся к минимуму, где это точка на t-й итерации, то есть текущее приближение к оптимальному решению.

3) Используем свойство Липшицевости градиента и разложение в ряд Тейлора:

Подставляем (шаг градиентного спуска) и упрощаем:

Теперь выберем шаг ​, чтобы последний член уменьшался, и получаем:

Это доказывает, что значение функции **всегда уменьшается** (если градиент не нулевой).

Оценим скорость сходимости GD.

Если повторять этот шаг k раз, суммируем все убывания:

Где:

**— это приближённое решение на** k**-й итерации**. В градиентном спуске обозначает приближение к оптимальному **решению** после k шагов алгоритма. k — номер итерации.

**—** это глобальный минимум функции , то есть точка, где принимает наименьшее значение:

Это говорит о том, что градиентный спуск сходится со скоростью

1. Чем больше шагов k, тем ближе к минимуму.

2. Если функция сильно выпуклая, то сходимость ускоряется до где (экспоненциальная скорость).

Вывод:

Градиентный спуск **гарантированно сходится**, если:

1. Функция выпуклая и гладкая**.**

2. Шаг обучения α выбран правильно

3.Мы достаточно долго итеративно обновляем x

**Стохастический градиентный спуск (SGD – stochastic gradient descent)**

Стохастический градиентный спуск (SGD) — один из самых популярных методов оптимизации в машинном обучении [4]. Его активно улучшают для повышения эффективности и объяснения эмпирического успеха. Недавние достижения в глубоких нейросетях стали возможны благодаря способности SGD эффективно их обучать. Стохастический градиентный спуск является вариацией обычного градиентного спуска. Его отличие заключается в том, что вычисление значения градиента

производится не на всей обучающей выборке, а на одном случайном элементе, за счет этого скорость

обучения значительно возрастает.

Пусть дана задача минимизации функции:

где — случайная функция потерь, зависящая от случайной переменной γ, а — её математическое ожидание.

**Стохастический градиент** в этом случае определяется как приближённая оценка истинного градиента:

где γ — случайная переменная, выбранная из некоторого распределения. При этом выполняется условие:

то есть в среднем стохастический градиент является несмещённой оценкой истинного градиента .

Более простая обобщенная запись:

Здесь  — это некоторая аппроксимация градиента целевой функции  в точке ​, называемая стохастическим градиентом — это размер шага (stepsize, learning rate) на итерации *k*. Для простоты мы будем считать, что для всех . Обычно предполагается, что SGD является несмещённой оценкой  при фиксированном ​: .

*Замечание.* Сходимость стохастического градиентного спуска доказывается значительно объемнее, ознакомится с ним можно по [хенд-буку от Яндекса](https://education.yandex.ru/handbook/ml/article/shodimost-sgd)

**Практическая часть. Сравнение GD & SGD**

В ходе практической части было обучено две нейросети с одним скрытым слоем для решения задачи бинарной классификации на датасете, состоящем из 100000 примеров и 2 признаков соответственно. Нейронные сети обучались на GPU (Графический процессор) с использованием ядер CUDA (Compute Unified Device Architecture – программно-аппаратная архитектура параллельных вычислений). Было задействовано 20 эпох обучения. Первая нейросеть обучая на простом градиентном спуске. Вторая же на стохастическом.

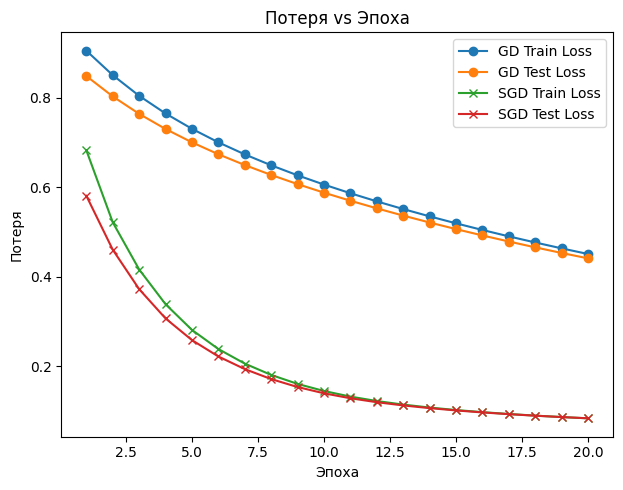


Рис. 2. График сравнения метрики loss и эпохи SGD & GD.

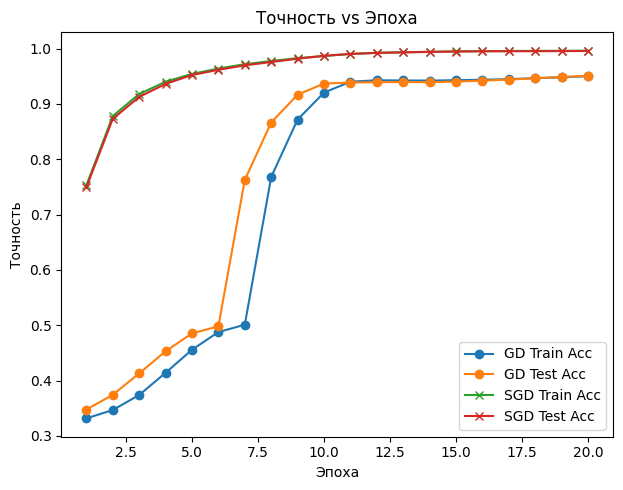


Рис. 3. График сравнения метрики Accuracy и эпохи обучения SGD & GD.

Из графиков (рис. 2 и рис. 3) обучения нейронной сети можно сделать вывод, что SGD будет эффективнее по времени обучения и при этом обладает преимуществами за счёт более частых обновлений параметров, что положительно сказывается на сходимости.

**Библиографический список**

1. The Loss Surfaces of Multilayer Networks. Courant Institute of Mathematical Sciences. [Электронный ресурс]. — URL: https://arxiv.org/pdf/1412.0233 — Дата обращения: 01.03.2025.

2. Convex Optimization. Stanford University. [Электронный ресурс]. — URL: https://web.stanford.edu/~boyd/cvxbook/bv\_cvxbook.pdf — Дата обращения: 01.03.2025.

3. Convex Optimization, notes from lectures. Carnegie Mellon University. [Электронный ресурс]. — URL: https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec6.pdf — Дата обращения: 05.11.2024.

4. Stochastic Gradient Descent in Theory and Practice. Stanford University. [Электронный ресурс]. — URL: https://ai.stanford.edu/~optas/data/stanford\_qual\_exams.pdf — Дата обращения: 01.03.2025.