МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Исследование и реализация алгоритмов оптимизации»

Обучающийся С.А. Грибков
(Инициалы Фамилия) (личная подпись)

Руководитель к.ф.-м.н., доцент, Г.А. Тырыгина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Е.В. Косс
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Тема выпускной квалификационной работы – «Исследование и реализация алгоритмов оптимизации».

Область применения методов оптимизации быстро расширилась в течение последних нескольких десятилетий. Было предложено много новых теоретических, алгоритмических и вычислительных вкладов в оптимизацию для решения различных проблем проектирования и управления.

По мере развития алгоритмических и вычислительных методов оптимизация стала применяться к широкому спектру задач, включая экономику, финансы, инженерию, управление и др.

В этой связи исследование и реализация алгоритмов оптимизации представляют актуальность и научно-практический интерес.

Объектом исследования бакалаврской работы является оптимизация.

Предметом исследования бакалаврской работы являются алгоритмы оптимизации.

Цель бакалаврской работы – исследование и реализация алгоритмов оптимизации.

Методы исследования – методы оптимизации.

Практическая значимость бакалаврской работы заключается в разработке программы, реализующей алгоритмы оптимизации.

Результаты бакалаврской работы представляют научно-практический интерес и могут быть рекомендованы для исследования методов и алгоритмов оптимизации.

Бакалаврская работа состоит из 43 страниц текста, содержит 7 рисунков, 2 таблицы и 22 источника.

Abstract

The title of the graduation work is "Research and implementation of optimization algorithms".

The graduation work consists of an explanatory note on 42 pages, contains 7 figures, 2 tables and 22 references.

The field of application of optimization methods has expanded rapidly over the past few decades. Many new theoretical, algorithmic, and computational contributions to optimization have been proposed to solve various design and control problems.

As algorithmic and computational methods have evolved, optimization has been applied to a wide range of problems, including economics, finance, engineering, management, etc.

In this regard, the study and implementation of optimization algorithms are of relevance and scientific and practical interest.

The object of the graduation work is optimization.

The subject of the graduation work is optimization algorithms.

The aim of the work - research and implementation of optimization algorithms.

Methods of research - methods of optimization.

The practical significance of the bachelor's work is to develop a program that implements the optimization algorithms.

The results of the bachelor's work are of scientific and practical interest and can be recommended for research of optimization methods and algorithms.

Оглавление

Введение

Область применения методов оптимизации быстро расширилась в течение последних нескольких десятилетий. Было предложено много новых теоретических, алгоритмических и вычислительных вкладов в оптимизацию для решения различных проблем проектирования и управления.

Последние разработки методов оптимизации можно в основном разделить на детерминированные и эвристические подходы.

Под детерминированной оптимизацией понимаются все алгоритмы, которые следуют строгому математическому подходу. Строго говоря, это относится к математическому программированию.

По мере развития алгоритмических и вычислительных методов детерминированная оптимизация стала применяться к широкому спектру задач, включая экономику, финансы, инженерию, управление и др.

В этой связи исследование и реализация алгоритмов оптимизации представляют актуальность и научно-практический интерес.

Объектом исследования бакалаврской работы является оптимизация.

Предметом исследования бакалаврской работы являются алгоритмы оптимизации.

Цель бакалаврской работы – исследование и реализация алгоритмов оптимизации.

Для достижения данной цели необходимо выполнить следующие задачи:

- выполнить постановку задачи исследования и проанализировать методы детерминированной оптимизации;
- проанализировать алгоритмы детерминированной оптимизации;
- выполнить реализацию и тестирование алгоритмов детерминированной оптимизации.

Методы исследования – методы оптимизации.

Практическая значимость бакалаврской работы заключается в

разработке программы, реализующей алгоритмы оптимизации.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы.

Первая главе работы посвящена постановке задачи исследования и анализу методов оптимизации.

Вторая глава работы посвящена обзору и анализу алгоритмов оптимизации.

Третья глава посвящена реализации и тестированию алгоритмов й оптимизации.

В заключении описываются результаты выполнения выпускной квалификационной работы.

Бакалаврская работа состоит из 42 страниц текста, содержит 7 рисунков, 2 таблицы и 22 источника.

Глава 1 Постановка задачи и методы оптимизации

Оптимизация – это выбор наилучшего варианта из множества возможных по некоторому критерию. Если критерий выбора известен и вариантов немного, то решение может быть найдено путём простого перебора и сравнения всех вариантов [2].

В общем виде задачу оптимизации можно формализовать следующим образом.

В некотором пространстве $\mathfrak{H}$ задано конечное или счетное множество допустимых альтернатив $X \subseteq \mathfrak{H}$.

На множестве X задана функция F: $X \to \Re$ – критерий оптимизации.

Задача – найти оптимальную допустимую альтернативу (1):

$$x^* \in \operatorname{Arg}\ \max_{x \in X} F(x) \qquad (1)$$

или все множество оптимальных альтернатив (2):

$$X^* = \operatorname{Arg}\ \max_{x \in X} F(x) \qquad (2)$$

Детерминированная оптимизация или математическое программирование – это классический способ оптимизации с помощью математических моделей [8].

Рассмотрим математическую постановку задачи детерминированной оптимизации.

1.1 Постановка задачи детерминированной оптимизации

Детерминированной задачей оптимизации называется задача оптимизации, в которой критерий оптимальности Ф(X) и ограничивающие функции g(X), h(X) не зависят от случайного вектора внешних параметров Q

[10].

Детерминированная задача оптимизации формулируется следующим образом (3):

$$\min \Phi(X) = \Phi(X^*) = \Phi^*, \;\; X \in D \qquad (3)$$

где X^* – оптимальное значение вектора варьируемых параметров (переменных);

$\Phi(X^*) = \Phi^*$ – наименьшее (оптимальное) значение критерия оптимальности $\Phi(X)$.

Задача максимизации критерия оптимальности $\Phi(X)$ сводится к задаче минимизации критерия $(-\Phi(X))$ (4):

$$\max \Phi(X) = \min(-\Phi(X)), \;\; X \in D \qquad (4)$$

Имея в виду специфику задач оптимального проектирования, сделаем следующее допущение: если не оговорено противное, функция $\Phi(X)$ в своей области допустимых значений D определена, непрерывна и, за исключением, быть может отдельных точек, имеет в этой области конечные частные производные.

Рассмотрим два основных аспекта детерминированной оптимизации, а именно: оптимизацию без ограничений и оптимизацию с ограничениями [11].

Алгоритм безусловной оптимизации **(**оптимизации без ограничений**)** обычно начинается с точки $x^{(1)}$ и генерирует последовательность точек $\{x^{(n)}\}$ в пространстве оптимизации, сходящуюся к решению $x*$.

Назовем линией множество точек (5):

$$x(\alpha) = x' + \alpha s, \forall \alpha \in R \qquad (5)$$

где x′ – точка в пространстве дизайна, а s – направление.

Предположим, что переменная отклика y = *f*(x) достаточно гладкая (класс C^1 или C^2, независимо от того, нужно ли нам вычислять градиенты или гессианы).

Под функцией класса C^m понимается функция, непрерывная, выводимая и имеющая непрерывные производные до порядка m.

Матрица Гессе представляет собой квадратную матрицу частных производных второго порядка функции, поэтому для ее однозначного определения в каждой точке области требуется, чтобы функция относилась к классу C^2.

По цепному правилу производные (наклон и кривизна) переменной отклика вдоль любой линии, предполагая ||**s**|| = 1, равны (6), (7):

$$\frac{df(\mathrm{x})}{d\alpha} = \sum_{i=1}^{k} \frac{dx_i(\alpha)}{d\alpha} \frac{\partial f(x_i)}{\partial x_i} = \sum_{i=1}^{k} s_i \frac{\partial f(x_i)}{\partial x_i} = \mathrm{s}^T \nabla f(x) = f(\mathrm{x})^T \mathrm{s} = \mathrm{g}(\mathrm{x})^T \mathrm{s}, \quad (6)$$

$$\frac{d^2 f(\mathrm{x})}{d\alpha} = \frac{d}{d\alpha}\left(\mathrm{s}^T \nabla f(\mathrm{x})\right) = \mathrm{s}^T \nabla (\nabla f(\mathrm{x})^T \mathrm{s}) = \mathrm{s}^T \nabla^2 f(\mathrm{x}) \mathrm{s} = \mathrm{s}^T \mathrm{G}(\mathrm{x}) \mathrm{s}, \quad (7)$$

где k – количество переменных задачи оптимизации, **g** (**x**) градиент и **G** (**x**) гессиан переменной отклика, соответственно.

Как правило, градиент и гессиан неизвестны из эксперимента или моделирования. Таким образом, градиент аппроксимируется с использованием прямой или центральной конечной разности.

Задача оптимизации может быть записана в терминах минимизации целевой функции (8):

$$\text{minimize } f(\mathrm{x}), \quad \mathrm{x} \in \mathrm{R}^k \quad (8)$$

В тех случаях, когда целевая функция $f(\mathbf{x})$ должна быть максимизирована, это эквивалентно минимизации $-f(\mathbf{x})$. Точка минимума $\mathbf{x}*$ должна удовлетворять следующим условиям (9):

$$\begin{cases} s^T g^* = 0 \ \forall s \\ s^T G^* s \geq 0 \ \forall s \end{cases} \Rightarrow \begin{cases} g^* = 0 \\ G^* - \text{положительная} \\ \text{полуопределительная матрица,} \end{cases} \quad (9)$$

где $g^* = g(x^*)$ и $G^* = G(x^*)$.

Эти условия известны как необходимое условие первого порядка и необходимое условие второго порядка, соответственно. Порядок сходимости метода дает представление о том, насколько быстро итерации сходятся в окрестности решения.

Определяя $h^{(n)} = x^{(n)} - x^*$, мы говорим, что порядок сходимости метода равен p, если $\exists n_0$, $\exists a$ такие, что $\forall n > n_0$ (10):

$$\frac{||h^{(n+1)}||}{||h^{(n)}||^p} \leq a, h^{(n+1)} = O(||h^{(n)}||^p) \quad (10)$$

Алгоритмы безусловной оптимизации обычно основаны на аппроксимации общей целевой функции. Аппроксимацию можно сделать, например, с помощью разложения в ряд Тейлора до второго порядка.

Структура общей задачи оптимизации с ограничениями имеет следующий вид (11):

$$\text{minimize } f(x), \ x \in R^k \quad (11)$$

при ограничениях:

$$c_i(x) = 0, i \in E$$
$$c_i(x) \geq 0, i \in I,$$

где $f(x)$ – целевая функция, $c_i(x)$ – функции ограничений, E – множество ограничений равенства, I – множество ограничений неравенства.

Точка, удовлетворяющая всем ограничениям, называется допустимой точкой, а множество допустимых точек является допустимой областью R.

Предполагается, что $c_i(x)$ и $f(x)$ непрерывны, R замкнуто и не допускаются ограничения вида $c_i(x) > 0$. Если допустимая область не пуста и ограничена, решение $x*$ задачи оптимизации существует [12].

Определим множество активных ограничений в точке x (12):

$$A = A(x) = \{i : c_i(x) = 0\} \quad (12)$$

Самый простой подход к решению задач оптимизации с ограничениями в случае ограничений-равенств - это исключение, то есть уравнения ограничений используются для исключения некоторых переменных задачи (13):

$$c(x) = 0 \Rightarrow x_1 = \varphi(x_2) \Rightarrow \psi(x_2) = f(\varphi(x_2), x_2), \quad (13)$$

где x_1 и x_2 образуют область x.

Затем $\psi(x_2)$ минимизируется по x_2 без ограничений.

Рассмотрим методы оптимизации [7].

1.2 Метод градиентного спуска

Метод градиентного спуска представляет собой приближенный итеративный метод математической оптимизации.

Его можно использовать для приближения к минимуму любой дифференцируемой функции.

Рассмотрим постановку задачи [9].

Даны множество K и функция $f: K \to \mathbb{R}$.

Требуется найти точку $x^* \in K$, такую что $f(x) \geq f(x^*)$ для всех $x \in K$ (14):

$$f(x) \to \min, x \in K \quad (14)$$

где f – дифференцируемая и выпуклая функция;

K – выпуклое множество.

«Известно, что если x^* – минимум функции $f(x)$, то $f'(x^*) = 0$, где f' – производная функции f.

Этот критерий основан на линейном приближении (15):

$$f(x) \approx f(x^*) + f'(x^*)(x - x^*) \quad (15)$$

Чем ближе x к x^*, чем точнее это приближение.

В правой части – выражение, которое при $f'(x^*) \neq 0$ может быть как больше, так и меньше $f(x^*)$» [22].

Это основная суть критерия. В многомерном случае аналогично из линейного приближения (16):

$$f(x) \approx f(x^*) + \nabla f'(x^*)^T (x - x^*), \quad (16)$$

где $x^T y \sum x_i y_i \quad (i = 1,2, \dots, n)$ – стандартное скалярное произведение.

Форма записи обусловлена тем, что скалярное произведение – это то же самое, что матричное произведение вектор-строки на вектор-столбец.

Тогда получаем критерий (17):

$$\nabla f(x^*) = 0, \quad (17)$$

где $\nabla f(x^*)$ – градиент функции f в точке x^*.

Также равенство градиента нулю означает равенство всех частных производных нулю, поэтому в многомерном случае можно получить этот критерий, просто последовательно применив одномерный критерий по каждой переменной в отдельности.

Чтобы получить градиентный спуск строим последовательность вида (18):

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \tag{18}$$

где α_k – размер шага.

На практике размер шага часта подбирают эмпирически: если α_k - очень маленький, то последовательность медленно меняется, что делает метод не очень эффективным.

Если α_k - очень большой, то линейное приближение становится плохим, а может даже и неверным.

Для стохастического градиентного спуска единственное формальное отличие от классического градиентного спуска заключается в том, что вместо градиентной функции используется функция $g(x,\theta)$ такая, что (19):

$$E_\theta\, g(x,\theta) = \nabla f(x), \tag{19}$$

где E_θ – математическое ожидание случайной величины θ.

Таким образом стохастический градиентный спуск имеет вид (20):

$$x_{k+1} = x_k - \alpha_k g(x_k, \theta_k), \tag{20}$$

где θ_k – некоторый случайный параметр.

В качестве примера рассмотрим функции (21):

$$f(x) = \frac{1}{2m}\sum_{j=1}^{m}\|x - y_j\|^2, \nabla f(x) = \frac{1}{m}\sum_{j=1}^{m}(x - y_j) \quad (21)$$

и функцию (22):

$$g(x, i) = x - y_i \quad (22)$$

Если i принимает значения 1, …, m равновероятно, то можно утверждать, что в среднем g – это градиент f.

«Этот пример показателен еще и следующим: сложность вычисления градиента в m раз больше, чем сложность вычисления g. Это позволяет стохастическому градиентному спуску делать за одно и то же время в m раз больше итераций.

Несмотря на то, что стохастический градиентный спуск обычно сходится медленней обычного, за счет такого большого увеличения числа итераций получается улучшить скорость сходимости на единицу времени.

Стохастический градиентный спуск является базовым методом обучения большинства нейронных сетей» [9].

1.3 Метод стохастической аппроксимации

Методы стохастической аппроксимации (МСА) – это семейство итерационных методов, обычно используемых для поиска корней или задач оптимизации [9].

«Рекурсивные правила обновления методов стохастической аппроксимации могут быть использованы, среди прочего, для решения линейных систем, когда собранные данные искажены шумом, или для аппроксимации экстремальных значений функций, которые не могут быть вычислены напрямую, а только оцениваются с помощью зашумленных

наблюдений» [5].

Пусть $k \in \{1, 2,...\}$ – номер текущей итерации; напоминая метод простого градиентного спуска, МСА генерирует итеративную последовательность в соответствии с выражением (23) [19]:

$$\mathrm{u}^{k+1} = \mathrm{u}^k - \alpha^k \mathrm{G}^k, \quad (23)$$

где α^k – неотрицательный повышающий коэффициент;

G^k – аппроксимация градиента.

Если e_i – единичный вектор в направлении i-й координаты точки, то после возмущения текущей рабочей точки шумовой градиент можно вычислить следующим образом (24):

$$G^k = \frac{1}{2h^k}\begin{bmatrix} Y_{1+}^k - Y_{1-}^k \\ \dots \\ Y_{n+}^k - Y_{n-}^k \end{bmatrix}; \quad (24)$$

$$Y_{i+}^k = \tilde{\phi}_p\left(\mathrm{u}^k + h^k e_i, \xi_{i+}^k\right); Y_{i-}^k = \tilde{\phi}_p\left(\mathrm{u}^k - h^k e_i, \xi_{i-}^k\right)$$

При относительно слабых предположениях относительно липшицевой непрерывности и дифференцируемости функции стоимости, конечных вторых моментов случайных величин и ограниченности итераций можно доказать, что любые последовательности h^k и α^k, которые удовлетворяют условию (25):

$$\sum \alpha^k = \infty; \ \sum \left(\frac{\alpha^k}{h^k}\right) < \infty \quad (25)$$

сходятся к критической точке с вероятностью единица.

Рассмотрим пример последовательности (24), которая удовлетворяет

(26):

$$\alpha^k = \frac{a}{(k+1+A)^\alpha}; h^k = \frac{c}{(k+1)^\Upsilon} \qquad (26)$$

где a, c, α, γ строго положительны и A ≤ 0.

Теоретическая максимальная скорость стохастической сходимости может быть достигнута при α = 1 и γ = 1/6.

Однако такой выбор констант может привести к очень медленному продвижению алгоритма в практической реализации; поэтому выгоды обычно выбираются методом проб и ошибок.

Были предложены различные расширения метода, в зависимости от того, как оценивается шумовой градиент: конечно-разностная стохастическая аппроксимация (также известная как алгоритм Кифера-Вольфовица), стохастическая аппроксимация случайных направлений и стохастическая аппроксимация одновременного возмущения Спола (1992).

В отличие от конечной разницы, метод Спола требует только 2 вычислений функций на итерацию, независимо от размерности пространства поиска.

Оценка G_i^k определяется по формуле (27):

$$G_i^k \approx \frac{\tilde{\phi}_p(\mathrm{u}^k + h^k\delta^k, \xi_{i+}^k) - \tilde{\phi}_p(\mathrm{u}^k - h^k\delta^k, \xi_{i-}^k)}{2h^k\delta_i^k} \qquad (27)$$

где $\delta^k \in \mathrm{R}^n$ – вектор случайных возмущений с элементами δ_i^k, независимыми и одинаково распределенными из распределения с ограниченными обратными моментами, симметрично распределенными вокруг нуля и ограниченными для всех i и k (например, распределение Бернулли).

«В последнее время методы стохастической аппроксимации нашли

широкое применение в области статистики и машинного обучения, особенно в условиях работы с большими данными. Эти приложения варьируются от методов и алгоритмов стохастической оптимизации до онлайн-форм алгоритма EM, обучения с подкреплением через временные различия, глубокого обучения и других.

Алгоритмы стохастической аппроксимации также использовались в социальных науках для описания коллективной динамики: фиктивная игра в теории обучения и алгоритмы консенсуса могут быть изучены с использованием их теории» [4].

1.4 Метод Ньютона

Метод Ньютона — самый классический и известный алгоритм детерминированной оптимизации.

В методе Ньютона квадратичная модель целевой функции получается из усеченного разложения в ряд Тейлора (28):

$$f\left(\mathrm{x}^{(n)}+\delta\right) \approx q^{(n)}(\delta)=f^{(n)}+\mathrm{g}^{(n)T}\delta+\frac{1}{2}\,\delta^{T}\mathrm{G}^{(n)}\delta \qquad (28)$$

Тогда выбирается $\mathrm{x}^{(n+1)}=\mathrm{x}^{(n)}+\delta^{(n)}$, где $\delta^{(n)}$ минимизирует $q^{(n)}(\delta)$.

Метод требует вычисления первой и второй производных целевой функции, и он корректно определен, если $\mathrm{G}^{(n)}$ положительно определена.

Шаги метода на итерации n:

– решить $\mathrm{G}^{(n)}\delta=-\mathrm{g}^{(n)}$ для нахождения $\delta^{(n)}$;

– установить $\mathrm{x}^{(n+1)}=\mathrm{x}^{(n)}+\delta^{(n)}$.

Существует несколько вариантов алгоритма.

Например, матрица Гессе может обновляться каждые m итераций, хотя это снижает скорость сходимости метода, но это также снижает вычислительные затраты на одну итерацию.

Другая возможность состоит в том, чтобы использовать поправку в качестве направления поиска $s^{(n)}=\delta^{(n)}$ для использования с алгоритмом линейного поиска.

Несмотря на эти поправки, метод Ньютона не имеет общего применения, поскольку он может не сходится, когда $G^{(n)}$ не является положительно определенной.

Способ обеспечить сходимость метода к стационарной точке всякий раз, когда $G^{(n)}$ не является положительно определенной, состоит в том, чтобы вернуться к методу наискорейшего спуска $s^{(n)} = -g^{(n)}$ или придать смещение направлению поиска. в сторону самого крутого спуска (29):

$$(G^{(n)} + \nu I)s^{(n)} = -g^{(n)} \qquad (29)$$

где I — единичная матрица, а ν выбрано так, чтобы модифицированная матрица Гессе $(G^{(n)} + \nu I)$ была положительно определенной.

Подход доверительной области также можно использовать в сочетании с методом Ньютона.

Для анализа рассмотренных методов составлена таблица 1 их теоретических основ и областей применения.

Таблица 1 – Теоретические основы и области применения методов оптимизации

Метод	Описание	Область применения
Метод градиентного спуска	представляет собой приближенный итеративный метод математической оптимизации. Его можно использовать для приближения к минимуму любой дифференцируемой функции	широко применяется для обучения перцептрона и в теории искусственных нейронных сетей
«Метод стохастической аппроксимации	Рекуррентный метод построения состоятельной последовательности оценок решений уравнений регрессии и экстремумов функций регрессии в задачах непараметрического оценивания	применяется как средство решения задач распознавания, идентификации, обучения и адаптации

Продолжение таблицы 1

Метод	Описание	Область применения
Метод Ньютона	итерационный численный метод (второго порядка) решения оптимизационных задач, который позволяет определить экстремум (минимум или максимум) целевой функции	применяется для решения задач оптимизации в экономике, в которых требуется определить ноль первой производной либо градиента в случае многомерного пространства» [4]

Как показал анализ, эффективность того или иного метода оптимизации сильно зависит от условий конкретной задачи оптимизации.

Выводы по главе 1

Результаты проделанной работы позволили сделать следующие выводы:

- детерминированная оптимизация или математическое программирование – это классический способ оптимизации с помощью математических моделей;
- метод градиентного спуска представляет собой приближенный итеративный метод математической оптимизации. Его можно использовать для приближения к минимуму любой дифференцируемой функции;
- метод Ньютона – итерационный численный метод (второго порядка) решения оптимизационных задач, который позволяет определить экстремум целевой функции.

Как показал анализ, эффективность того или иного метода оптимизации зависит от условий конкретной задачи оптимизации

Глава 2 Обзор и анализ алгоритмов оптимизации

2.1 Алгоритм градиентного спуска

Алгоритм стохастического градиентного спуска (АСГС) является модификацией градиентного спуска.

«АСГС – оптимизационный алгоритм, отличающийся от обычного градиентного спуска тем, что градиент оптимизируемой функции считается на каждом шаге не как сумма градиентов от каждого элемента выборки, а как градиент от одного, случайно выбранного элемента» [22].

Псевдокод АСГС показан на рисунке 1.

```
def SGD(X, h, λ):  # где X – выборка, h – градиентный шаг, а λ – темп забывания
    w = initialize_weights()  # инициализировать веса
    Q̄ = (1/l) Σ_{i=1}^{l} L_i(w) # инициализировать оценку функционала
    while Q not converges or w not converges:
        i = rand() % l  # случайно выбрать элемент, по которому будет считаться градиент
        ε = L_i(w)  # вычислить потерю
        w = w − h∇L_i(w) # обновить вектор весов в направлении антиградиента
        Q̄ = λε + (1 − λ)Q̄ # оценить функционал
    return w
```

Рисунок 1 – Псевдокод АСГС

Существуют различные варианты АСГС, среди которых можно выделить следующие:

– АСГС онлайн. Это вариант стохастического градиентного спуска, в котором вы оцениваете градиент функции стоимости для каждого наблюдения и соответствующим образом обновляете переменные

решения. Это может помочь вам найти глобальный минимум, особенно если целевая функция выпуклая;

– пакетный АСГС. Находится где-то между обычным градиентным спуском и онлайн-методом. Градиенты рассчитываются, а переменные решения обновляются итеративно с подмножествами всех наблюдений, называемыми мини-пакетами. Этот вариант очень популярен для обучения нейронных сетей.

Можно представить онлайн-алгоритм как особый вид пакетного алгоритма, в котором каждый мини-пакет имеет только одно наблюдение.

Как и в случае обычного градиентного спуска, АСГС начинается с начального вектора переменных решения и обновляет его через несколько итераций. Разница между ними заключается в том, что происходит внутри итераций:

– АСГС случайным образом делит набор наблюдений на мини-пакеты;
– для каждого мини-пакета вычисляется градиент и перемещается вектор;
– как только все мини-пакеты использованы, считается, что итерация или эпоха завершена, и начинается следующая.

Так как этот АСГС случайным образом выбирает наблюдения для мини-пакетов, необходимо смоделировать это случайное (или псевдослучайное) поведение [1].

Это можно сделать с помощью генерации случайных чисел.

2.2 Алгоритм Роббинса-Монро

Алгоритм Роббинса-Монро относится к категории алгоритмов стохастической аппроксимации [16].

Рассмотрим работу данного алгоритма [13].

Предположим, мы хотим найти корень θ' функции $f: \mathrm{R} \rightarrow \mathrm{R}$.

Для этого мы можем использовать метод Ньютона, который генерирует

последовательность итераций (30):

$$\theta_{n+1} = \theta_n - \frac{f(\theta_n)}{f'(\theta_n)} \qquad (30)$$

Предположим, что мы известна область θ', где $f(\theta) < 0$ при $\theta < \theta'$, $f(\theta) > 0$ при $\theta > \theta'$ и f не убывает в этой области.

Тогда, если мы начнем с θ_0, достаточно близкого к θ', следующая более простая (но менее эффективная) схема также сходится к θ' и не требует производной от f (31):

$$\theta_{n+1} = \theta_n - \alpha f(\theta_n) \qquad (31)$$

для некоторого фиксированного и достаточно малого $\alpha > 0$.

Заметим, что если f сама является производной функции F, то эти схемы соответствуют методу Ньютона и процедуре градиентного спуска с фиксированным шагом для минимизации F соответственно (точнее, нахождению критической точки F или корень градиента F).

Очень часто в приложениях у нас нет доступа к математической модели f, но мы можем проводить эксперименты или моделировать выборку функции при определенных значениях θ.

Однако эти выборки обычно зашумлены, поэтому мы можем предположить, что в нашем распоряжении есть черный ящик (симулятор, лаборатория, где мы проводим эксперименты и т. д.), который на входе x_θ возвращает значение (32):

$$y = f(\theta) + d, \qquad (32)$$

где d — шум, который вскоре будет считаться случайным.

Дело в том, что у нас есть доступ только к значению y, и у нас нет

возможности убрать из него шум, т. е. выделить точное значение $f(\theta)$.

Теперь предположим, что мы все еще хотим найти корень f, как в задаче выше, с доступом только к этому шумному черному ящику.

Предположим пока, что мы знаем, что шум является IID и с нулевым средним.

Первый подход к проблеме может состоять в том, чтобы для заданного значения θ достаточно много раз произвести выборку в одной и той же точке θ и получить значения $y_1, \ldots, y_N$, а затем сформировать оценку $f(\theta)$ с помощью эмпирического среднего (33):

$$f(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} y_i \tag{33}$$

Имея достаточное количество выборок на каждой итерации θ_n уравнения (19), можно приблизительно найти корень f.

Проблема в том, что придется потратить много времени на взятие проб в точках θ', которые далеки от θ и на самом деле не имеют значения, за исключением указания, в каком направлении двигаться дальше.

Это может стать реальной проблемой, если получение каждого образца требует много времени или затрат.

Альтернативная процедура (34), предложенная Роббинсом и Монро, заключается в простом прямом использовании зашумленной версии f в слегка модифицированной версии алгоритма (19):

$$\theta_{n+1} = \theta_n - \gamma_n y_n, \tag{34}$$

где γ_n – последовательность положительных чисел, сходящаяся к 0 и такая, что $\Sigma_n \gamma_n = \infty$ (например, $\gamma_n = 1/(n + 1)$), а $y_n = f(\theta_n) + d_n$ – зашумленная версия $f(\theta_n)$).

Обратим внимание на то, что итерации θ_n теперь являются случайными величинами.

Идея увеличения размера шага γ_n, заключается в том, что он обеспечивает своего рода усреднение наблюдений.

Для аналогии в более простой ситуации предположим, что у нас есть IID наблюдения $\xi_1, \ldots, \xi_N$ случайной величины и необходимо сформировать их эмпирическое среднее, как в (23).

Рекурсивная альтернатива (23), чрезвычайно полезная в условиях, когда выборки становятся доступными постепенно с течением времени (например, фильтр Калмана), заключается в формировании (35):

$$\theta_1 = \xi_n,\ \theta_{n+1} = \theta_n - \gamma_n[\theta_n - \xi_{n+1}], \qquad (35)$$

где $\gamma_n = 1/(n+1)$.

Можно убедиться, что $\theta_n = \Sigma_i \xi_i\ (i = 1, \ldots, n)$ для всех n.

Здесь рассматриваются рекурренты, обобщающие (32), вида (36):

$$\theta_{n+1} = \theta_n + \gamma_n[f(\theta_n) + b_n + D_{n+1}], \qquad (36)$$

где $\theta_0 \in \mathrm{R}^d$ – возможно случайное значение;

f – функция $\mathrm{R}^d \to \mathrm{R}^d$;

b_n – небольшой систематический член возмущения, например, смещение в оценке $f(\theta_n)$;

D_{n+1} – случайный шум с нулевым средним.

Алгоритмы стохастической аппроксимации широко применяются на практике.

На рисунке 2 представлена блок-схема алгоритма работы адаптивной системы тестирования [3].

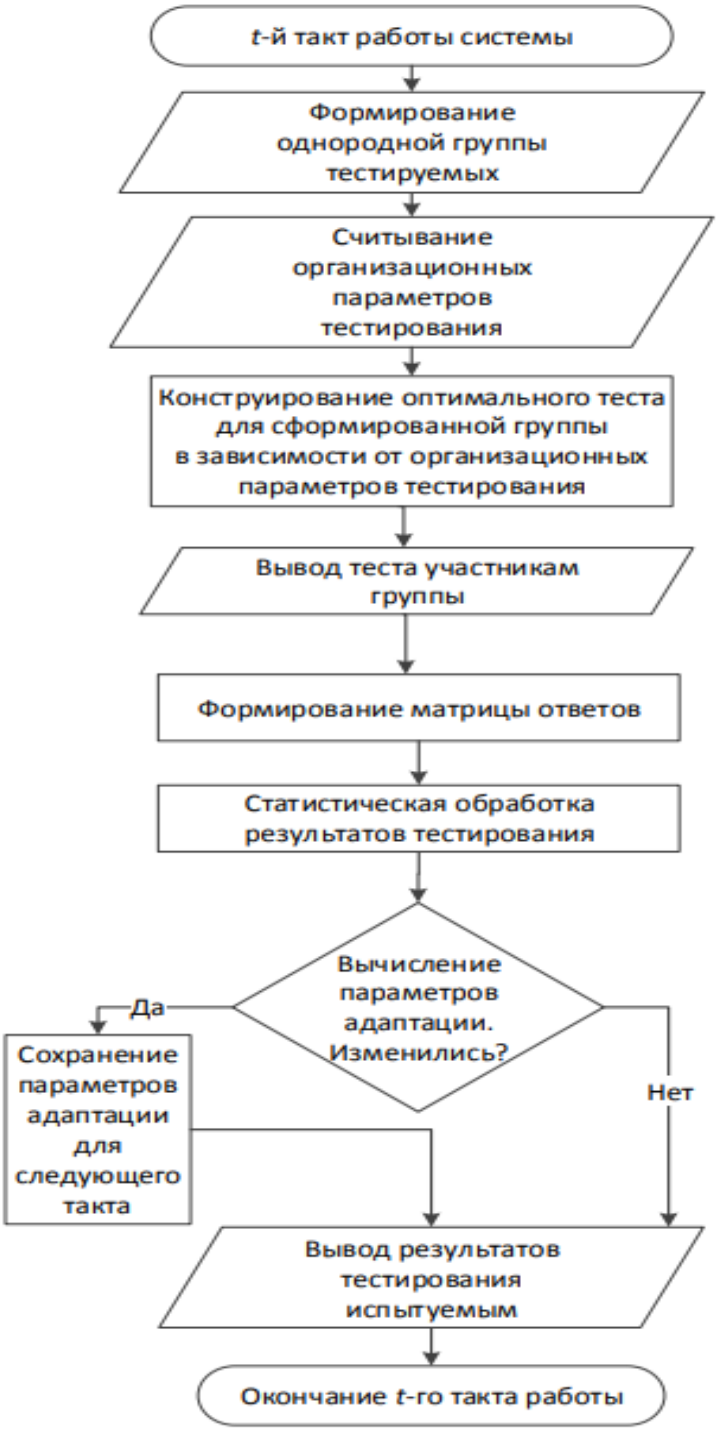

Рисунок 2 – Блок-схема алгоритма работы адаптивной системы тестирования

Оценки трудности и подготовленности в предложенной системе базируются на основе сходящейся процедуры стохастической аппроксимации.

Указанные свойства адаптивной системы тестирования знаний

гарантируют повышение точности тестирования с течением времени и выравнивание оценок тестируемых пользователей на единой шкале.

2.3 Алгоритм имитации отжига

Алгоритм имитации отжига относится к классу пороговых алгоритмов локального поиска [21].

«Алгоритм основан на аналогии с процессом кристаллизации вещества, например, при отжиге металла.

В ходе этого процесса температура вещества понижается, оно отвердевает, при этом замедляется скорость движения частиц вещества. Кристаллическую решетку можно представить, как систему частиц, а ее энергетическое состояние – совокупностью состояний частиц.

Частицы переходят из одного энергетического состояния в другое произвольным образом, но вероятность переходов зависит от температуры системы. Вероятность перехода из высокоэнергетического состояния в низкоэнергетическое велика при любой температуре, также существует отличная от нуля вероятность перехода в состояние с более высоким значением энергии.

Эта вероятность тем выше, чем меньше разница между состояниями и чем выше температура системы.

Если в роли физической системы представить задачу оптимизации, в роли энергии системы – значение целевой функции $f(X)$, а в роли частиц – управляющие переменные X, то можно решать задачу оптимизации функции $f(X)$, используя механизмы и законы, которые определяют процесс отвердевания.

Необходимо задать закон, по которому будет меняться температура системы, закон, по которому будет случайным образом изменяться значение координат в пространстве решения, и правило определения перехода частицы в точку с новыми координатами» [6].

Кроме того, нужно выбрать начальную и конечную температуру: T_0 и T_f.

Блок-схема алгоритма имитации отжига показана на рисунке 3 [20].

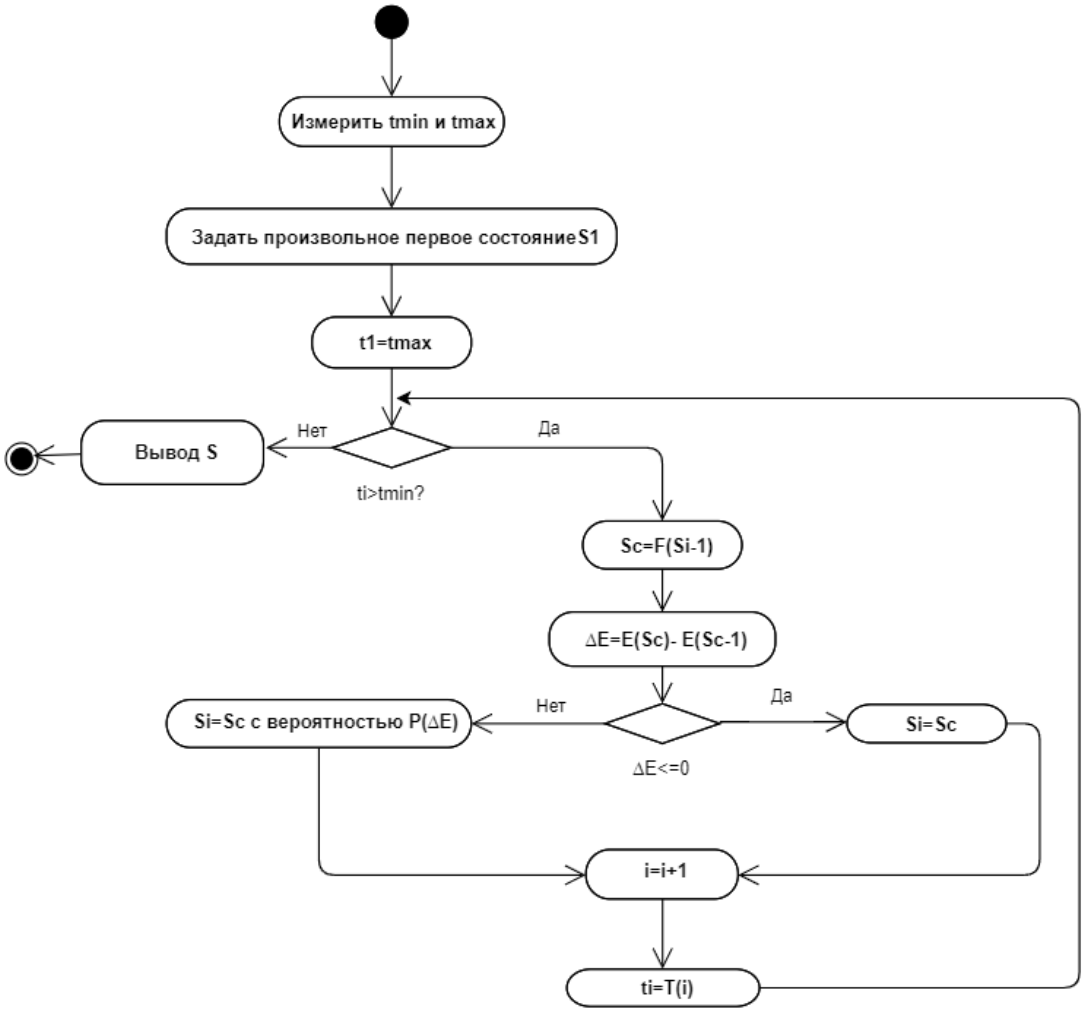

Рисунок 3 – Блок-схема алгоритма имитации отжига

Здесь:

S – множество всех состояний (решений) задачи;

$s_i \in$ S– состояние на i-м шаге алгоритма;

$t_i \in$ R – температура на i-м шаге алгоритма;

E – функция энергии, которую мы оптимизируем;

T – функция изменения температуры;

F – функция, создающая новое состояние.

Алгоритм имитации отжига состоит из следующих шагов:

Шаг 1. Генерация случайного начального состояния.

Шаг 2. Сравнение значения функции с наилучшим найденным, если текущее X лучше, то оно принимается в качестве лучшего.

Шаг 3. Вычисление случайного нового состояния (X_{new}) и определение значения функции для него. Закон распределения может быть любым, например, экспоненциальным (37):

$$x = x + r_1 M_x \ln(r_2) \quad (37)$$

где x – одна из координат состояния X;

r_1 – случайное число (–1 или +1);

r_2 – случайное число, равномерно распределенное от 0 до 1;

M_x – параметр алгоритма, от которого зависит, как далеко от текущей точки может переместиться процесс поиска за один шаг.

Шаг 4. Если новое состояние лучше текущего, система переходит в новое состояние ($X = X_{\text{new}}$), иначе случайным образом принимается решение о переходе или не переходе в новое состояние. При этом в случае максимизации может быть использовано следующее условие перехода (38):

$$\frac{\varphi(X_{\text{new}}) - \varphi(X)}{T} > r, \quad (38)$$

где T – текущая температура;

r – случайное число, равномерно распределенное от 0 до 1.

Шаг 5. Если температура достигла конечной, алгоритм завершается, иначе температура понижается и происходит переход на шаг 2. Новое значение температуры на i-м шаге может быть вычислено следующим образом (39):

$$T = \frac{T_0}{e^{ivt}} \quad (39)$$

где *vt* – коэффициент, влияющий на скорость снижения температуры.

При завершении алгоритма сохраненное в памяти наилучшее решение и будет результатом работы алгоритма.

«Алгоритм имитации отжига прост в реализации и показывает высокую эффективность в решении самых разных задач оптимизации.

Однако для получения высокой эффективности может потребоваться тщательная настройка формулы понижения температуры как численных параметров, так и самого вида зависимости, кроме того, часто температуру меняют не на каждой итерации алгоритма, а через определенное число итераций» [18].

2.4 Алгоритм Левенберга-Марквардта

Алгоритм Левенберга-Марквардта (АЛМ) представляет собой метод, который использовался для извлечения параметров полупроводниковых устройств, и представляет собой гибридный метод, который использует методы Гаусса-Ньютона и метод наискорейшего спуска для сходимости к оптимальному решению [14].

Гибридный подход часто используется для компромисса между лучшими характеристиками различных алгоритмов для решения более широкого круга задач. Например, подход Гаусса-Ньютона быстрее, чем АЛМ, если начальное предположение относительно близко к оптимальному решению.

В случае, если это не так, АЛМ использует аспекты подхода наискорейшего спуска, чтобы пройти пространство проектирования и найти потенциальную область решения, а затем найти оптимум.

Этот метод особенно эффективен при решении систем нелинейных

уравнений, и это делает его полезным для полупроводниковых устройств, которые имеют как большие наборы нелинейных уравнений, так и большие наборы параметров.

Блок-схема алгоритма Левенберга-Марквардта показан на рисунке 4.

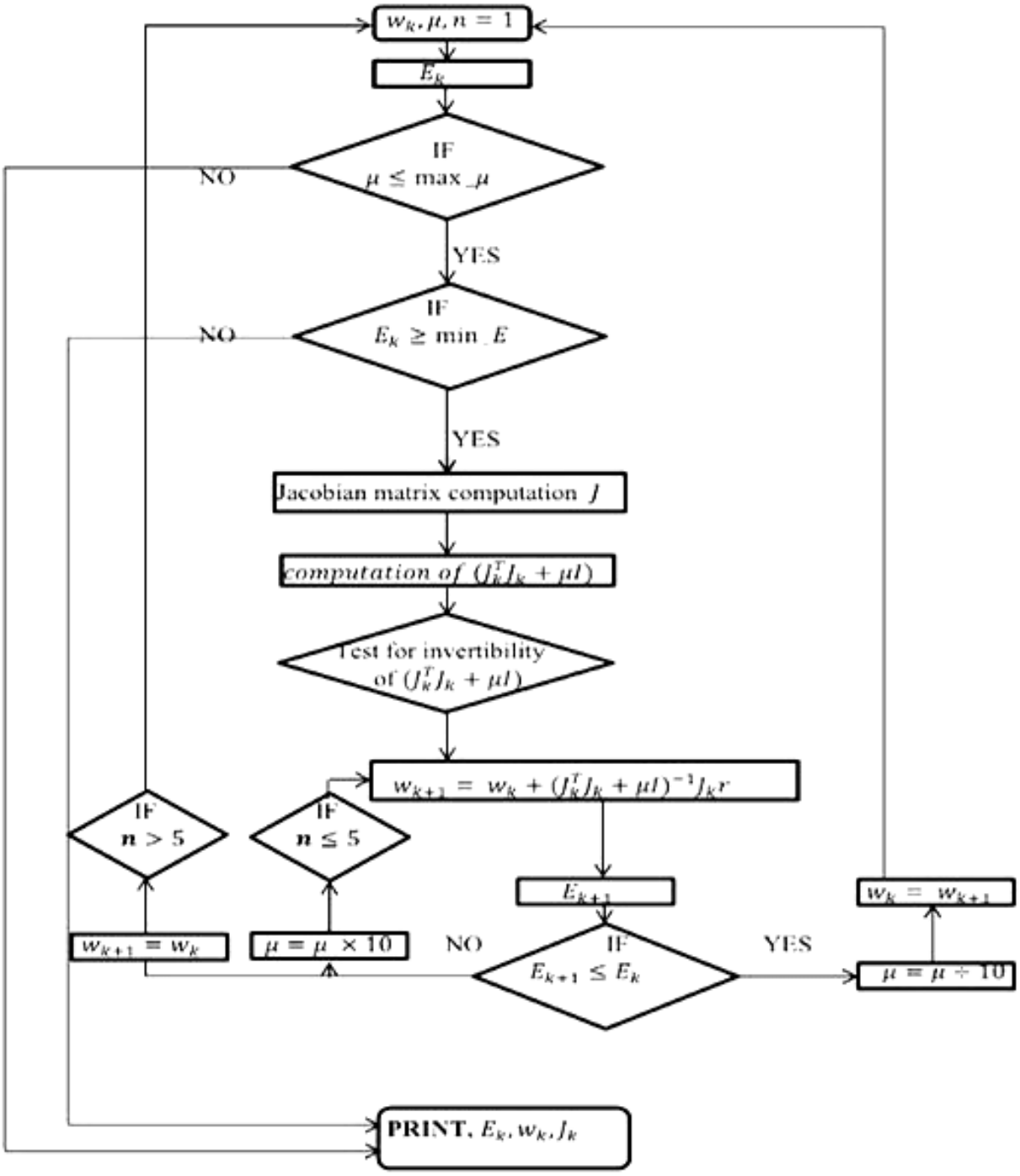

Рисунок 4 – Блок-схема алгоритма Левенберга-Марквардта

Для анализа рассмотренных алгоритмов составлена таблица 2 их

достоинств и недостатков.

Таблица 2 – Характеристики алгоритмов оптимизации

Алгоритм	Достоинства	Недостатки
АСГС	простота реализации; функция потерь и семейство алгоритмов могут быть любыми; легко добавить регуляризацию; возможно потоковое обучение; подходит для задач с большими данными	нет универсального набора эвристик, их нужно выбирать для конкретной задачи отдельно
Алгоритм стохастической аппроксимации	даёт хорошие прогнозы в среднесрочном аспекте, так как отсеивает шумы, ошибки и адаптируется только к существенным изменениям тенденций	величина интервала выбирается полностью на основе экспертных оценок; сложность реализации
Алгоритм имитации отжига	относительная простота реализации; высокая эффективность в решении самых разных задач оптимизации	результат работы сильно зависит от выбранных значений параметров; необходимость многократных тестов для верного подбора параметров алгоритма
Алгоритм Левенберга-Марквардта	наибольшие устойчивость и шансы найти глобальный экстремум; очень высокая скорость сходимости	налагает ограничения на вид целевой функции; сложность реализации

Как и в случае с методами, эффективность того или иного алгоритма оптимизации сильно зависит от условий конкретной задачи оптимизации.

Выводы по главе 2

Результаты проделанной работы позволили сделать следующие выводы:

- алгоритм имитации отжига основан на аналогии с процессом кристаллизации вещества, например, при отжиге металла;
- алгоритм Левенберга-Марквардта представляет собой гибридный метод, который использует методы Гаусса-Ньютона и метод наискорейшего спуска для сходимости к оптимальному решению.

Как и в случае с методами, эффективность того или иного алгоритма оптимизации сильно зависит от условий конкретной задачи оптимизации.

Глава 3 Реализация и тестирование алгоритмов оптимизации

3.1 Реализация алгоритмов оптимизации

В качестве примера для реализации использованы алгоритмы градиентного спуска.

Напомним, что АСГС представляет собой простую модификацию стандартного алгоритма градиентного спуска, который вычисляет градиент и обновляет весовую матрицу W для небольших пакетов обучающих данных, а не для всего обучающего набора.

Хотя эта модификация приводит к «более шумным» обновлениям, она также позволяет делать больше шагов по градиенту (один шаг на каждую партию по сравнению с одним шагом на эпоху), что в конечном итоге приводит к более быстрой сходимости и отсутствию негативных последствий для потерь и точности классификации.

Для реализации алгоритма выбран язык Python.

Язык Python – интерпретируемый, интерактивный, объектно-ориентированный язык программирования высокого уровня общего назначения.

Исходный код языка Python также доступен под лицензией GNU General Public License (GPL).

Язык Python широко применяется для реализации алгоритмов обработки больших данных и в машинном обучении [15].

В листинге 1 представлен код на языке Python для реализации мини-пакетного АСГС [17].

Листинг 1– Код АСПС

```
1.  # import the necessary packages
2.  from sklearn.model_selection import train_test_split
```

```
3.  from sklearn.metrics import classification_report
4.  from sklearn.datasets import make_blobs
5.  import matplotlib.pyplot as plt
6.  import numpy as np
7.  import argparse
8.  def sigmoid_activation(x):
9.  # compute the sigmoid activation value for a given input
10. return 1.0 / (1 + np.exp(-x))
11. def sigmoid_deriv(x):
12. # compute the derivative of the sigmoid function ASSUMING
13. # that the input "x" has already been passed through the sigmoid
14. # activation function
15. return x * (1 - x)
```

Строки 2–7 импортируют необходимые пакеты Python.

Строки 9-17 определяют функции sigmoid_activation и sigmoid_deriv.

В листинге 2 представлен код метода прогнозирования.

Листинг 2 – Код метода прогнозирования

```
16. «def predict(X, W):
17. # take the dot product between our features and weight matrix
18. preds = sigmoid_activation(X.dot(W))
19. # apply a step function to threshold the outputs to binary
20. # class labels
21. preds[preds <= 0.5] = 0
22. preds[preds > 0] = 1
23. # return the predictions
24. return preds» [17]
```

Следующим этапом является тестирование алгоритмов оптимизации.

3.2 Тестирование алгоритмов оптимизации

В листинге 3 представлены тестовые данные для проверки работы АСГС.

Листинг 3 – Тестовые данные

«[INFO] training...
[INFO] epoch=1, loss=0.1317637
[INFO] epoch=5, loss=0.0162487
[INFO] epoch=10, loss=0.0112798
[INFO] epoch=15, loss=0.0100234
[INFO] epoch=20, loss=0.0094581
[INFO] epoch=25, loss=0.0091053
[INFO] epoch=30, loss=0.0088366
[INFO] epoch=35, loss=0.0086082
[INFO] epoch=40, loss=0.0084031
[INFO] epoch=45, loss=0.0082138
[INFO] epoch=50, loss=0.0080364
[INFO] epoch=55, loss=0.0078690
[INFO] epoch=60, loss=0.0077102
[INFO] epoch=65, loss=0.0075593
[INFO] epoch=70, loss=0.0074153
[INFO] epoch=75, loss=0.0072779
[INFO] epoch=80, loss=0.0071465
[INFO] epoch=85, loss=0.0070207
[INFO] epoch=90, loss=0.0069001
[INFO] epoch=95, loss=0.0067843
[INFO] epoch=100, loss=0.0066731

[INFO] evaluating...

precision recall f1-score support» [17].

0 1.00 1.00 1.00 250

1 1.00 1.00 1.00 250

avg / total 1.00 1.00 1.00 500

Графически тестовые данные представляют собой наборы черных и красных точек (рисунок 5).

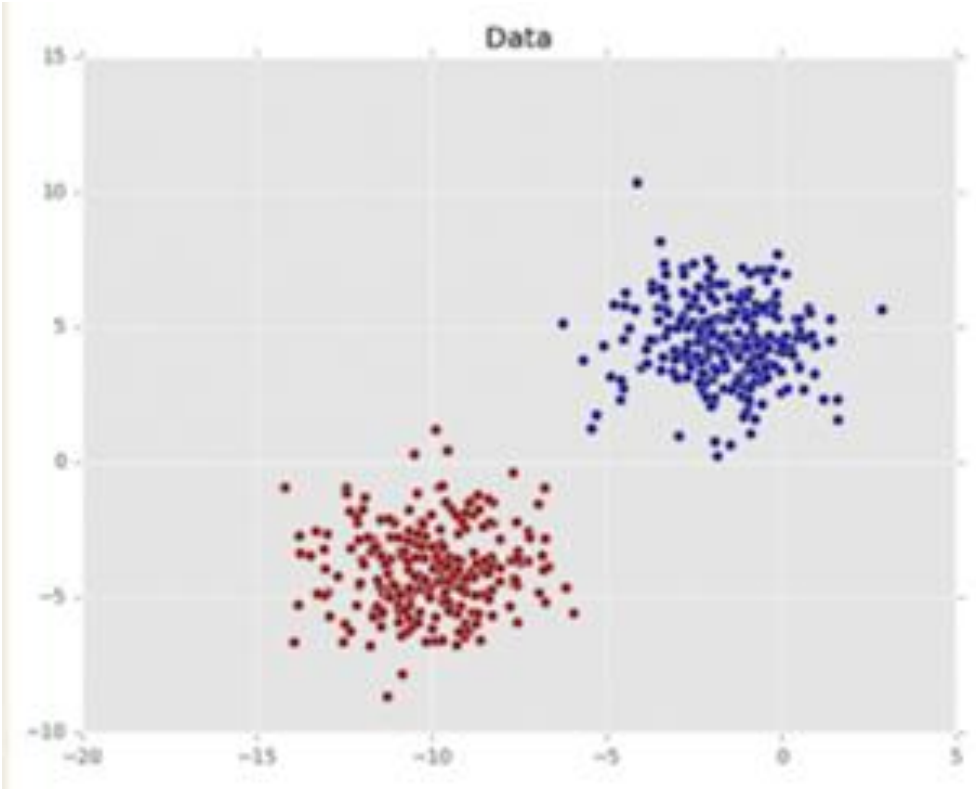

Рисунок 5 – Графическое представление тестовых данных

В примере используется скорость обучения 0,1 и количество эпох 100. Результат выполнения АСГС показан на рисунке 6.

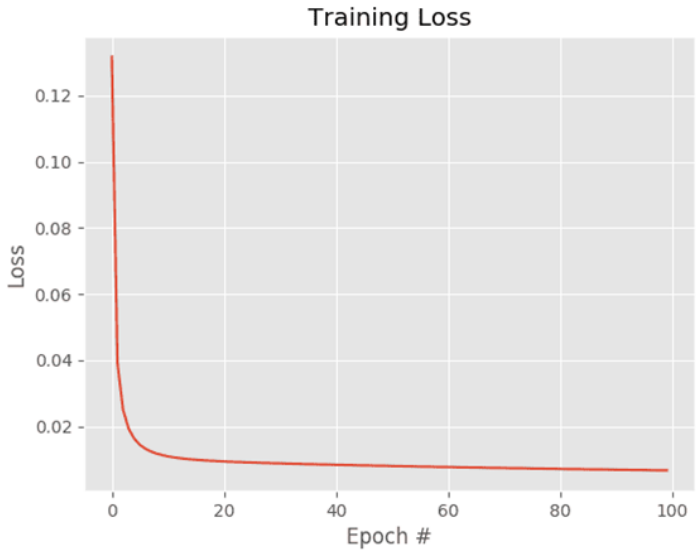

Рисунок 6 – График результатов тестирования АСГС

Исследуя фактические значения потерь в конце 100-й эпохи, отметим, что потери, полученные с помощью АСГС, почти на два порядка ниже, чем при классическом градиентном спуске (0,006 против 0,447 соответственно) (рисунок 7).

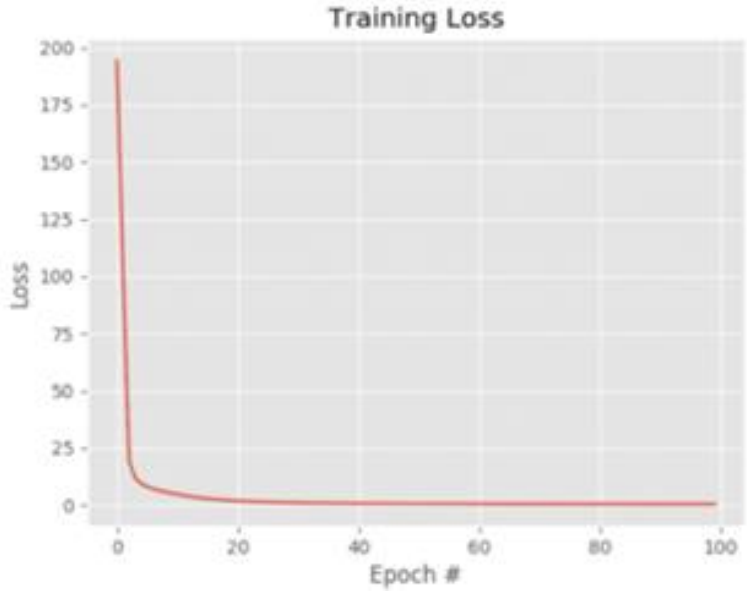

Рисунок 7 – График результатов тестирования алгоритма классического градиентного спуска

Эта разница связана с несколькими обновлениями весов за эпоху, что дает представленной модели больше шансов учиться на обновлениях, внесенных в матрицу весов.

Этот эффект еще более заметен на больших наборах данных, таких как ImageNet, где у есть миллионы обучающих примеров, а небольшие добавочные обновления наших параметров могут привести к решению с низкими потерями (но не обязательно оптимальному).

Выводы по главе 3

Результаты проделанной работы позволили сделать следующие выводы:

- Python – интерпретируемый, интерактивный, объектно-ориентированный язык программирования высокого уровня общего назначения. Язык Python широко применяется для реализации алгоритмов обработки больших данных и в машинном обучении;
- для проведения вычислительного эксперимента использованы тестовые данные, которые графически представляют собой наборы черных и красных точек;
- исследование фактических значений потерь в конце 100-й эпохи показали, что потери, полученные с помощью АСГС, почти на два порядка ниже, чем при классическом градиентном спуске (0,006 против 0,447 соответственно).

Эта разница связана с несколькими обновлениями весов за эпоху, что дает представленной модели больше шансов учиться на обновлениях, внесенных в матрицу весов.

Заключение

Выпускная квалификационная работа посвящена актуальной проблеме исследования и реализации алгоритмов оптимизации.

По мере развития алгоритмических и вычислительных методов оптимизация стала применяться к широкому спектру задач, включая финансовое планирование, производство, управление цепочками поставок, смягчение последствий изменения климата и контроль загрязнения.

В этой связи исследование и реализация алгоритмов оптимизации представляют актуальность и научно-практический интерес.

Для достижения данной цели в процессе работы над бакалаврской работой решены следующие задачи:

- выполнена постановка задачи исследования и проанализированы методы моделирования оптимизации. Детерминированная оптимизация или математическое программирование – это классический способ оптимизации с помощью математических моделей. Рассмотрены некоторые метолы оптимизации. Метод градиентного спуска представляет собой приближенный итеративный метод математической оптимизации. Его можно использовать для приближения к минимуму любой дифференцируемой функции. Методы стохастической аппроксимации – это семейство итерационных алгоритмов, обычно используемых для поиска корней или задач оптимизации. Метод Ньютона — самый классический и известный алгоритм детерминированной оптимизации, итерационный численный метод решения оптимизационных задач;
- проанализированы алгоритмы оптимизации. АСГС отличается от обычного градиентного спуска тем, что градиент оптимизируемой функции считается на каждом шаге не как сумма градиентов от каждого элемента выборки, а как градиент от одного, случайно

выбранного элемента. Целью стохастических алгоритмов является нахождение экстремумов функций, которые неизвестны, но где доступны только зашумленные наблюдения. Алгоритм имитации отжига основан на аналогии с процессом кристаллизации вещества, например, при отжиге металла. Как и в случае с методами, эффективность того или иного алгоритма оптимизации сильно зависит от условий конкретной задачи оптимизации.;

– выбран язык программирования Python и выполнена реализация АСГС. Язык Python широко применяется для реализации алгоритмов обработки больших данных и в машинном обучении. Для проведения вычислительного эксперимента использованы тестовые данные, которые графически представляют собой наборы черных и красных точек. Исследование фактических значений потерь в конце 100-й эпохи показали, что потери, полученные с помощью АСГС, почти на два порядка ниже, чем при классическом градиентном спуске (0,006 против 0,447 соответственно). Эта разница связана с несколькими обновлениями весов за эпоху, что дает представленной модели больше шансов учиться на обновлениях, внесенных в матрицу весов.

Результаты бакалаврской работы представляют научно-практический интерес и могут быть рекомендованы для исследования методов и алгоритмов оптимизации.

Список используемой литературы

1. Алгоритм градиентного спуска учебного процесса [Электронный ресурс]. URL: https://russianblogs.com/article/1969333121/ (дата обращения: 05.05.2022).

2. Балдин К. В., Башлыков В. Н., Рукосуев А. В. Методы оптимальных решений [Электронный ресурс] : учебник. Москва : ФЛИНТА, 2020. 323 с. URL: https://znanium.com/catalog/product/1145336 (дата обращения: 12.05.2022).

3. Бессарабов Н.А., Бондаренко А.В., Кондратенко Т.Н., Тимофеев Д.С. Алгоритмическое обеспечение адаптивной системы тестирования знаний // Программные продукты и системы. 2016. № 1 (113). С. 68-73.

4. Бородин А.И., Сорочайкин А.Н. Особенности методов стохастической оптимизации в социально-экономических системах // Математические и инструментальные методы экономики. 2013. № 4(101). С. 151-156.

5. Вахитов А.Т., Граничин О.Н., Гуревич Л. С. Алгоритм стохастической аппроксимации с пробным возмущением на входе в нестационарной задаче оптимизации // Автомат. и телемех. 2009. Выпуск 11. С. 70–79.

6. Введение в оптимизацию. Имитация отжига [Электронный ресурс]. URL: https://habr.com/ru/post/209610/ (дата обращения: 05.05.2022).

7. Лемешко Б.Ю. Л 442 Методы оптимизации: Конспект лекций. Новосибирск: Изд-во НГТУ, 2009. 26 с.

8. Новожилова М.В., Штань И.В. Решение детерминированной задачи оптимизации трехуровневой сети поставок одного товара // АСУ и приборы автоматики. 2014. №167. URL: https://cyberleninka.ru/article/n/reshenie-determinirovannoy-zadachi-optimizatsii-trehurovnevoy-seti-postavok-odnogo-tovara (дата обращения: 27.05.2022).

9. Обзор градиентных методов в задачах математической оптимизации

[Электронный ресурс]. URL: https://habr.com/ru/post/413853 (дата обращения: 05.05.2022).

10. Постановка детерминированной задачи оптимизации [Электронный ресурс]. URL: http://bigor.bmstu.ru/?cnt/?doc=MO/ch0103.mod/?cou=MO/base.cou (дата обращения: 05.05.2022).

11. Deterministic Optimization [Электронный ресурс]. URL: https://link.springer.com/chapter/10.1007/978-3-642-31187-1_4 (дата обращения: 05.05.2022).

12. Deterministic Optimization [Электронный ресурс]. URL: https://www.taylorfrancis.com/chapters/mono/10.1201/9781315155739-2/deterministic-optimization-juan-gabriel-segovia-hern%C3%A1ndez-fernando-israel-g%C3%B3mez-castro (дата обращения: 05.05.2022).

13. Introduction to Stochastic Approximation Algorithms [Электронный ресурс]. URL: https://www.professeurs.polymtl.ca/jerome.le-ny/teaching/DP_fall09/notes/lec11_SA.pdf (дата обращения: 05.05.2022).

14. Levenberg–Marquardt Algorithm [Электронный ресурс]. URL: https://www.sciencedirect.com/topics/engineering/levenberg-marquardt-algorithm (дата обращения: 05.05.2022).

15. Python Tutorial [Электронный ресурс]. URL: https://www.tutorialspoint.com/current_affairs.htm (дата обращения: 03.05.2022).

16. Robbins H., Monro S. A Stochastic Approximation Method. Ann. Math. Statist. 22 (3), pp. 400 - 407, 1951.

17. Rosebrock A. Stochastic Gradient Descent (SGD) with Python [Электронный ресурс]. URL: https://pyimagesearch.com/2016/10/17/stochastic-gradient-descent-sgd-with-python/ (дата обращения: 05.05.2022).

18. Simulated Annealing Algorithm [Электронный ресурс]. URL: https://www.sciencedirect.com/topics/engineering/simulated-annealing-algorithm#:~:text=The%20simulated%20annealing%20algorithm%20is,state%20is

%20reached%20%5B143%5D (дата обращения: 05.05.2022).

19. Stochastic Approximation in Online Steady State Optimization Under Noisy Measurements [Электронный ресурс]. URL: https://www.sciencedirect.com/topics/computer-science/stochastic-approximation (дата обращения: 05.05.2022).

20. Stochastic Model / Process: Definition and Examples [Электронный ресурс]. URL: https://www.statisticshowto.com/stochastic-model/ (дата обращения: 05.05.2022).

21. Stochastic Modeling [Электронный ресурс]. URL: https://corporatefinanceinstitute.com/resources/knowledge/other/stochastic-modeling/ (дата обращения: 05.05.2022).

22. Zhenhuan Yang et al. Simple Stochastic and Online Gradient Descent Algorithms for Pairwise Learning, 35th Conference on Neural Information Processing Systems (NeurIPS 2021), 2021.