РАЗДЕЛ IV. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Барвинский Д.А., Минеева Т.А.

Применение метода градиентного спуска в решении задач оптимизации

*Северный (Арктический) федеральный университет имени М. В. Ломоносова
(Россия, Архангельск)*

doi: 10.18411/lj-06-2021-56

Аннотация

В статье рассмотрено исследование градиентного спуска в задачах оптимизации с обозначением основных методов, показано нахождение минимального значения функции для повышения точности модели.

Ключевые слова: градиентный спуск, методы оптимизации, минимальное значение функции, точность модели.

Abstract

The article examines the study of gradient descent in optimization problems with the designation of the main methods, shows the finding of the minimum value of the function to improve the accuracy of the model.

Keywords: gradient descent, optimization methods, minimum function value, model accuracy.

Градиентный спуск — самый используемый алгоритм обучения, который применяется в математической модели для минимизации функции потерь. Актуальность выбранной темы объясняется тем, что метод градиентного спуска является непосредственной основой множества современных алгоритмов обучения, разрабатываемых в анализе данных. Градиентный спуск был выбран по нескольким причинам: во-первых, пожалуй, он самый простой, во-вторых, он лежит в основе большинства других методов, и если хорошо разобраться с тем или иным приемом на примере градиентного спуска, то это можно использовать при перенесении на более сложный метод, лучше подходящий для решения конкретной задачи [1].

Цель данной работы заключается в исследовании метода градиентного спуска в задачах оптимизации.

Существует три варианта градиентного спуска:

1. Мини-пакетный градиентный спуск – это разновидность алгоритма градиентного спуска, который разбивает обучающий набор данных на небольшие партии, которые используются для расчета ошибки модели и обновления коэффициентов модели.
2. Стохастический градиентный спуск – 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.
3. Пакетный градиентный спуск – это разновидность алгоритма градиентного спуска, который вычисляет ошибку для каждого примера в наборе обучающих данных, но обновляет модель только после того, как все обучающие примеры были оценены.

Основными методами оптимизации в градиентном спуске являются:

1. Исключение возможного попадания в локальный минимум.
2. Обеспечение наибольшей скорости и точности шага сходимости

- Подбор шага сходимости
 - Изменение шага сходимости по формуле
 - Нормализация градиента
3. Метод дробления шага — метод, основанный на пошаговой корректировке коэффициента.
 4. Метод наискорейшего спуска. Осуществляется поиск из заданной точки, параллельной одной из осей, до точки минимума. Затем поиск производится в направлении, параллельном другой оси, и т.д.
 5. Метод золотого сечения — поиск экстремума функции на заданном отрезке. В основе метода лежит принцип деления отрезка. [2]

Рассмотрим решение задачи методом золотого сечения. На рисунке 1 показан алгоритм данного метода:

Рисунок 1 – Алгоритм решения методом золотого сечения

Найдем точку минимума функции $f(x) = x^4 + 2x^2 + 4x + 1$ на отрезке $[-1,0]$ с точностью $E = 0,1$ по алгоритму золотого сечения. Согласно алгоритму, полученные данные представим в таблицу 1.

Таблица 1

Расчеты точки минимума методом золотого сечения

n	a_n	b_n	$b_n - a_n$	x	y	F(x)	F(y)
1	-1	0	1	-0.618	-0.382	-0.6623	-0.2149
2	-1	-0.382	0.618	-0.7639	-0.618	-0.548	-0.5623
3	-0.7639	-0.382	0.3819	-0.618	-0.5279	-0.5623	-0.4766
4	-0.7639	-0.5279	0.236	-0.6738	-0.618	-0.5811	-0.5623
5	-0.7639	-0.618	0.1459	-0.7082	-0.6738	-0.5782	-0.5811
6	-0.7082	-0.618	0.09018	-0.6738	-0.6524	-0.5811	-0.5772

Из данных таблицы найдем x_{min} как середину интервала $[-1,0]$:

$$x_{min} = (-0.618 - 0.7082)/2 = -0.6631$$

Ответ: $x_{min} = -0.6631$

Рисунок 2 – График исследуемой функции

Неточности в поиске минимального значения данной функции объясняются выбором шага сходимости и небольшим числом итераций.

Рассмотрим реализацию алгоритма на функции с одной переменной. Код на языке Python показан на рисунке 3.

```
import time
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return x*x*x*x-3*x+5

def df(x):
    return 4*x*x*x-3

Ni = 20 # число итераций
xx = 4 # начальное значение
lmd = 0.155 # шаг сходимости

x_plt = np.arange(-3, 5, 0.1)
f_plt = [f(x) for x in x_plt]

plt.ion() # включение интерактивного режима отображения графиков
fig, ax = plt.subplots() # создание окна и осей для графика
ax.grid(True) # отображение сетки на графе

ax.plot(x_plt, f_plt) # отображение графика функций
point = ax.scatter(xx, f(xx), c='black') # отображение точки черным цветом

for i in range (Ni):
    xx = xx - lmd*np.sign(df(xx))
    point.set_offsets([xx,f(xx)])

    plt.draw()
    plt.gcf().canvas.flush_events()
    # перерисовываем график
    time.sleep(0.02) # задержка 20мс

plt.ioff()
print(xx)
ax.scatter(xx, f(xx), c='black')
plt.show()
```

Рисунок 3 – Код работы алгоритма функции с одной переменной

Проблемой реализации данного алгоритма было обеспечение наибольшей скорости сходимости. В коде была использована нормализация градиента функции, при которой длина градиента свелась к единице и алгоритм перестал зависеть от величины градиента, используя только его направление, тем самым повысив точность окончательного минимального значения.

На рисунке 4 представлено начальное положение точки и результат работы кода.

Рисунок 4 – Работа алгоритма функции с одной переменной

Рассмотрим реализацию алгоритма градиентного спуска для функции двумя переменными. Код на языке Python показан на рисунке 5.

```
import time
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D, axes3d

def E(y, a, b):
    ff = np.array([a * z + b for z in range(N)])
    return np.dot((y-ff).T, (y-ff))
def dEda(y, a, b):
    ff = np.array([a * z + b for z in range(N)])
    return -2*np.dot((y-ff).T, range(N))
def dEdb(y, a, b):
    ff = np.array([a * z + b for z in range(N)])
    return -2*(y-ff).sum()

N = 100 # число экспериментов
Niter = 50 # число итераций
sigma = 3 # стандартное отклонение наблюдаемых значений
at = 1 # теоретическое значение параметра k
bt = 1.5 # теоретическое значение параметра b
aa = 0
bb = 0
lmd1 = 0.000001
lmd2 = 0.0005

f = np.array([at*z+bt for z in range(N)])
y = np.array(f + np.random.normal(0, sigma, N))

a_plt = np.arange(-1, 2, 0.1)
b_plt = np.arange(0, 3, 0.1)
E_plt = np.array([E(y, a, b) for a in a_plt for b in b_plt])

plt.ion() # включение интерактивного режима отображения графиков
fig = plt.figure()
ax = Axes3D(fig)

a, b = np.meshgrid(a_plt, b_plt)
ax.plot_surface(a, b, E_plt, color='y', alpha=0.5)
ax.set_xlabel('a')
ax.set_ylabel('b')
ax.set_zlabel('E')
point = ax.scatter(aa, bb, E(y, aa, bb), c='black')

for n in range(Niter):
    aa = aa - lmd1 * dEda(y, aa, bb)
    bb = bb - lmd2 * dEdb(y, aa, bb)
    ax.scatter(aa, bb, E(y, aa, bb), c='black')
    plt.draw()
    plt.gcf().canvas.flush_events() # перерисовываем график
    time.sleep(0.01) # задержка 10мс
    print(aa, bb)

plt.ioff()
plt.show()

ff = np.array([aa*z+bb for z in range(N)])

plt.scatter(range(N), y, s=2, c='black')
plt.plot(f)
plt.plot(ff, c='black')
plt.grid(True)
plt.show()
```

Рисунок 5 – Код работы алгоритма функции с двумя переменными

Функция $f(x) = ax + b$ представлена на рисунке 5. Целью работы данного алгоритма является нахождение оптимальных значений a и b .

На рисунке 6 показано начальное положение точки и результат работы кода.

Рисунок 6 – Работа алгоритма функции с двумя переменными

Рисунок 7 – График аппроксимации

График аппроксимации экспериментальных данных, представленный на рисунке 7 позволяет разобрать числовые характеристики и качественные свойства объекта. Исследование функции переходит к использованию более упрощенных методов.

В исследовании рассмотрены основные теоретические основы и методы оптимизации, которые используются для повышения точности работы математической модели. Функция используется, чтобы контролировать ошибку в модели обучения. Поиск минимума означает получение наименьшей возможной ошибки – повышение точности модели. Определение размера шага является главной проблемой в использовании градиентного спуска. Лучшим решением является предоставление нескольких различных размеров шага на выбор с последующим увеличением шага.

1. Гасников, А.В. Современные численные методы оптимизации. Метод универсального градиентного спуска: [Текст]/учебное пособие / А.В. Гасников– М. : МФТИ, 2018. – 291 с.
2. Аббасов, М. Э. Методы оптимизации: [Текст]/ учебное пособие / М.Э. Аббасов — СПб.: Издательство “БВМ”, 2014. — 64 с.
3. Глебов Н.И., Кочетов Ю. А., Плясунов А.В. Методы оптимизации. 2000.
4. Neurohive. [Электронный ресурс]/ Базовый курс. Градиентный спуск: всё, что нужно знать / Станислав Исаков – Режим доступа: <https://neurohive.io/ru/osnovy-data-science/gradient-descent>, свободный (Дата обращения: 01.06.2021)

Гамова Н.А., Абдуллин М.Ш.

История развития компьютерных технологий

Оренбургский государственный университет

(Россия, Оренбург)

doi: 10.18411/lj-06-2021-57

Аннотация

В данной статье рассматривается путь развития компьютерных технологий и их роль в нашей истории. Целью работы является установление роли человека в развитии компьютерных технологий и его вклад в практическое применение и создание технологий будущего. В статье показана взаимосвязь между развитием компьютерных технологий и увеличением их роли в жизни человека и государства, рассмотрены и изучены особенности строения, технического оснащения и спецификации компьютеров. На основе изучения и анализа данной темы, составлены прогнозы развития технологий будущего. В статье показан пример написания кода простейшей программы, на применение файлов.

Ключевые слова: компьютер, компьютерные технологии, ЭВМ.

Abstract

This article will discuss the development of computer technologies and their role in our history. The aim of the work is also to determine the role of man in the development of computer technologies and his contribution to the practical application and creation of future technologies. The article shows the relationship between the development of computer technologies and the increase in their role in the life of a person and the state, examines and studies the features of the structure, technical equipment and specifications of computers. Based on the study and analysis of the material, forecasts of the development of future technologies are made. The example code of the simplest program is shown.

Keywords: computer, computer technology, computer.

Огромную роль в жизни человека играет компьютер. Таблицы, формы, сложнейшие вычисления, на которые у человека ушли бы недели, он проделывает за