PHENIKAA UNIVERSITY
FACULTY OF ELECTRICAL AND ELECTRONICS

REPORT

BIG EXERCISE TUDTTN

PHENIKAA
UNIVERSITY

TOPIC :

Implementing gradient descent algorithm

Students perform:	Dương Ngọc Anh (23012040)
	Nguyễn Đình Dương (23010971)
Instructor guides:	Vũ Hoàng Diệu

Class: AI-ROBOT

Course: K17

Branch: Automation and control

HÀ NỘI , 16/10/2024

INTRODUCTION

In the field of Machine Learning and optimization, finding optimal parameters for a model is a key task to improve the model's prediction and performance. One of the most popular and powerful algorithms used to solve this problem is Gradient Descent. This is an iterative method that adjusts model parameters to effectively minimize the loss function.

Gradient Descent is widely applied in linear regression problems, logistic regression, and especially in deep neural network models. This algorithm uses the concept of gradient to guide the parameter update process in the direction of reducing the value of the loss function, thereby gradually approaching the optimal solution. With its power and flexibility, Gradient Descent has become a core tool in training machine learning models.

The Implementing Gradient Descent Algorithm problem not only helps us better understand how this algorithm works, but also provides an intuitive view of the optimization process through minimizing prediction errors. In the next section, we will explore how to install and implement the Gradient Descent algorithm specifically to optimize a linear regression model.

Contents

1 INTRODUCE

1.1 *Introduction to Gradient Descent Algorithm*

- Gradient Descent is an optimization algorithm used to find the minimum value of a function. In machine learning, it is used to minimize a loss function to optimize model parameters, such as weights and biases in linear regression models or neural networks.

1.2 *Objective of the problem*

- Find optimal values for the parameters θ (weights and biases) such that the value of the loss function is minimized.
- Gradient Descent works on the idea of moving the parameters in the opposite direction to the gradient of the loss function at the parameter's current position.

1.3 *Steps of the Gradient Descent algorithm*

- B1. Parameter initialization: Initialize the initial values of parameters, usually randomly.
- B2. Calculate the gradient: The gradient of the loss function is calculated at the current values of the parameters.
- B3.Update parameters: Parameters are updated by subtracting the gradient value multiplied by the learning rate.
- B4.Repeat: This process is repeated until the value of the loss function converges, that is, it almost does not change anymore.

1.4 *Formula of Gradient Descent*

- **Suppose we have a loss function that needs to be optimized as:**

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{1}$$

In there:

  - $J(\theta)$: Loss of jaw (Mean Squared Error - MSE).
  - m:Number of samples in the data set.
  - $h_\theta(x^{(i)})$: Prediction of the model at sample i.
  - $y^{(i)}$:Actual value at sample i.

- **Update the parameters (Gradient Descent) according to the formula:**

$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{2}$$

In there:

  - θ_j:Parameter to update (can be weight w or bias b).
  - α: Learning rate, which controls the level of adjustment per update.
  - $\frac{\partial J(\theta)}{\partial \theta_j}$: Gradient of loss function $J(\theta)$ with respect to parameter θ_j.

1.5 Update specific parameters in Linear Regression

- **With linear regression y=wX+b, update weight w and bias b based on Gradient Descent as follows:**

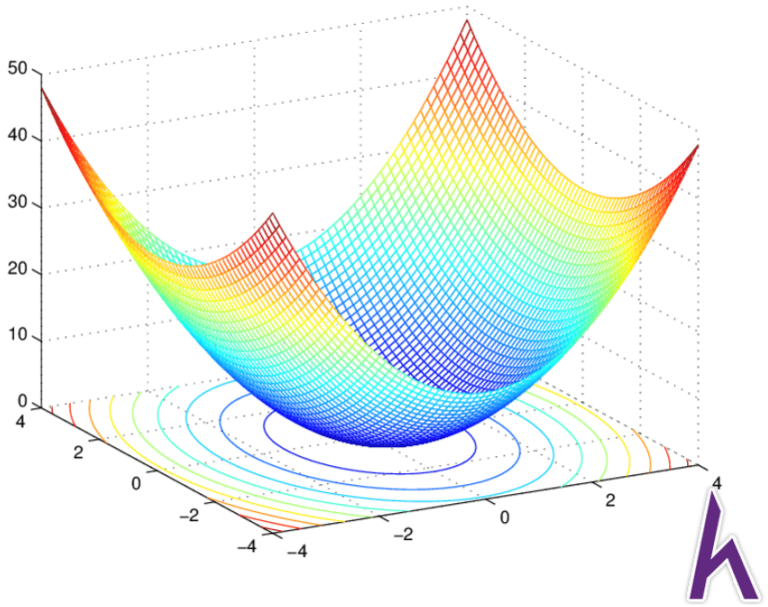
$$w = w - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} \left[(wX^{(i)} + b - y^{(i)}) X^{(i)} \right] \tag{3}$$

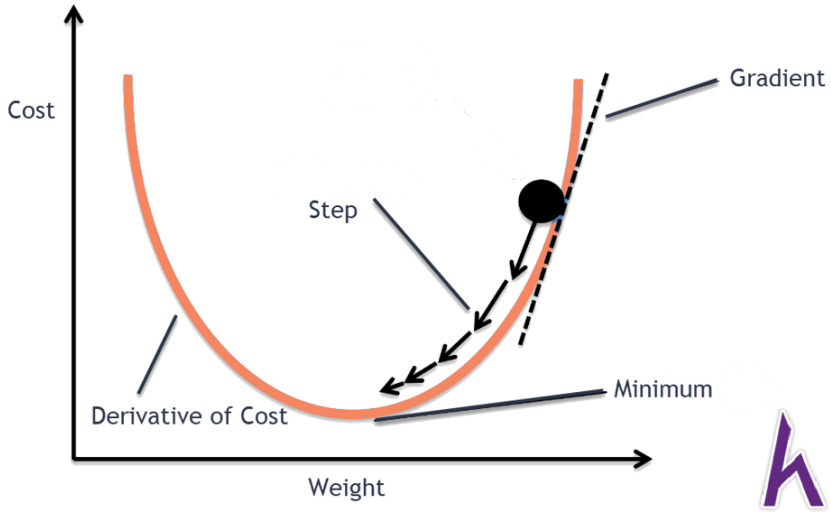
$$b = b - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (wX^{(i)} + b - y^{(i)}) \tag{4}$$

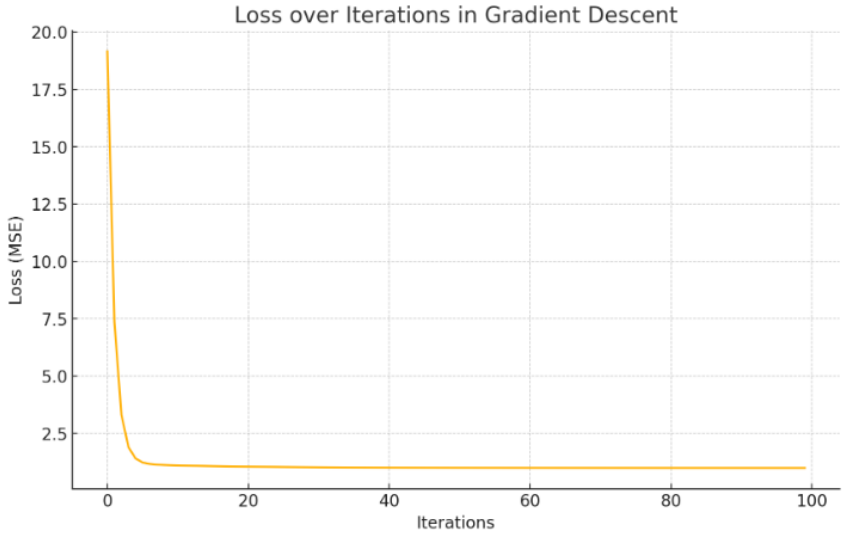
  In there:

  - w: Weights need to be optimal.
  - b: bias.
  - α: learning rate.

2 CHART


50
40
30
20
10
0
4
2
0
−2
−4
−4
−2
0
2
4


Cost
Gradient
Step
Minimum
Derivative of Cost
Weight


Loss over Iterations in Gradient Descent
Loss (MSE)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
20
40
60
80
100
Iterations

3 DATAPASE

3.1 Code

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.datasets import fetch_california_housing


def predict(X, weight, bias): # hồi quy tuyến tính nên hàm chỉ là y=wx+b
    return weight*X + bias


def loss(X,y,weight,bias):   #chọn MSE làm loss function
    samples = len(y)
    sum_squared_error = 0
    mse = 0

    for i in range(samples):
        sum_squared_error += (y[i] - (weight*X[i] + bias))**2

    mse = sum_squared_error/samples
    return mse


def update_weight(X, y, weight, bias, learning_rate):   #Gradient Descent
    samples = len(y)
    weight_temp = 0.0
    bias_temp = 0.0

    for i in range(samples):
        # tính gradient của hàm MSE theo weight và theo bias
        weight_temp += -2*X[i] * (y[i] - (X[i]*weight+bias))
        bias_temp += -2*(y[i]-(X[i]*weight+bias))

    # cập nhật weight và bias bằng cách chỉnh ngược hướng (là cộng ngược dấu) trung bình gradient nhân với learning
    weight -= (weight_temp/samples)*learning_rate
    bias -= (bias/samples)*learning_rate
    return weight, bias


def train(X, y, weight, bias, learning_rate, loop):
    list_loss = []

    for i in range(loop):
        # tìm weight và bias của mỗi lượt bằng hàm update
        weight_temp, bias_temp = update_weight(X, y, weight, bias, learning_rate)
        # tìm loss của lượt bằng hàm loss
        loss_temp = loss(X, y, weight_temp, bias_temp)

        # một list các loss để tiện theo dõi nếu print ra (không print cũng được)
        list_loss.append(loss_temp)

        weight = weight_temp
        bias = bias_temp

    return weight, bias, list_loss
```

3.2 *Data*

```
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# dữ liệu đầu vào
X = df.iloc[:,-3].values
# đầu ra (target) -> cho cặp x y để học, khi chạy test cho x và đoán y
y = df.iloc[:,-1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=1)
```

```
[ ] weight, bias, list_loss = train(X_train, y_train, 0.03, 0.0014, 0.01, 30)

    print(weight)
    print(bias)
    # print(list_loss)

    y_predict = predict(X_test,weight,bias)
    print(mean_absolute_error(y_test,y_predict))
```

3.3 *Result*

- **Final loss function value:**

$$-1.2895586268393123 \times 10^{-40} \tag{5}$$

- **Learning speed:**

$$0.0013999745641768517 \tag{6}$$

- **List of loss function values through loops:**

```
[590.9621834506249, 353440.74501324014, 211881918.76742885,
127020226976.10063, 76146838109255.42, 4.5648957588395064e+16,
2.736590751039907e+19, 1.6405476344504101e+22,
9.834852141768592e+24, 5.895855421647386e+27,
3.534482334039163e+30, 2.1188724071771897e+33,
1.2702341824316136e+36, 7.61487512297706e+38,
4.5650104477214135e+41, 2.736659505410147e+44,
1.6405888517274907e+47, 9.835099233542547e+49,
5.896003549687826e+52, 3.5345711347141853e+55,
2.1189256419319294e+58, 1.2702660959176022e+61,
7.61506643983336e+63, 4.565125139484129e+66,
2.7367282615074135e+69, 1.640630070040128e+72,
9.835346331525294e+74, 5.896151681450254e+77,
3.5346599376208873e+80, 2.1189788780244109e+83]
```

(7)

- **Final parameter value:**

$$4.594766113905676 \times 10^{41} \tag{8}$$

4 CONCLUDE

In this article, we have explored the Gradient Descent algorithm in detail, a powerful tool in the field of machine learning and optimization. We have understood how the algorithm works, from calculating the gradient to updating the model parameters to minimize the loss function.

Gradient Descent is not only an effective method to improve model accuracy, but is also highly flexible, allowing it to be applied to many different types of models, from linear regression to neural networks. deep. Adjusting the learning rate as well as the number of iterations is very important in the optimization process, directly affecting the performance and convergence of the algorithm.

The results obtained from installing and implementing Gradient Descent demonstrated the algorithm's effectiveness in improving model predictions. Thereby, we can see that understanding and applying the Gradient Descent algorithm properly is a decisive factor in building effective machine learning systems.

In the future, extending and improving optimization methods, such as Stochastic Gradient Descent (SGD) or Adam, will be necessary to further enhance the performance and applicability of the models. machine geometry.

5 REFERENCES

- https://howkteam.vn/course/machine-learning-co-ban-voi-numpy/thuat-toan-gradient-descent-cho-linear-regression-3997
- chatgpt.com
- https://www.youtube.com/watch?v=GVCkD927KMo