Using Gradient Descent Method to Predict House Price

Peilin Liu, kirby school

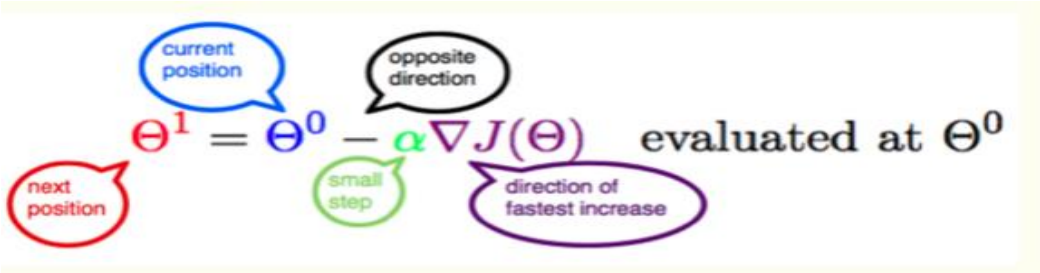
Abstract

The purpose of the project is to study the gradient descent algorithm in machine learning, choose linear regression models, obtain the linear regression function through the gradient descent algorithm to predict the house price, use Python programming to implement the algorithm.

1.Principle of gradient and gradient descent algorithm

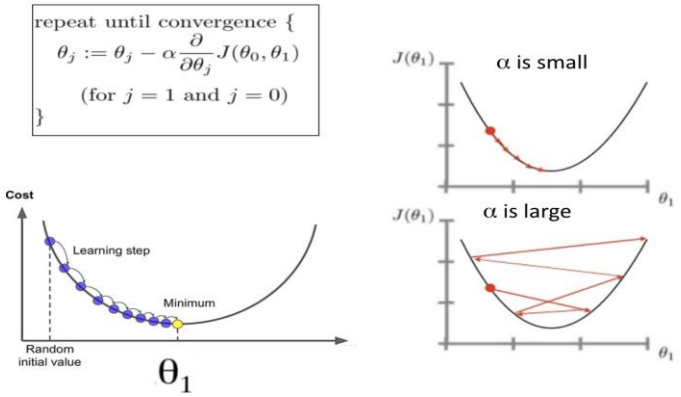
Gradient descent algorithm is widely used in machine learning, its principle is walking along the opposite direction of the function gradient to reach the local minimum of the function by taking repeated steps.

Gradient descent formula is：



In the formula , θ^0 is the current value， θ^1 is the value of next step， $J(\theta_i)$ is function of θ_i,θ_i can be multiple variables , $\nabla J(\theta_i)$ is the gradient of $J(\theta_i)$， $-\alpha \nabla J(\theta_i)$ is the opposite direction of the gradient， from θ^0 walks a distance of step length α to reach θ^1point.α is the gradient descent algorithm learning rate or step length, which means that we can through α control the distance of each step.α cannot be too large or too small. If it is too small, it may lead to delays in walking， if it is too large, it will cause the lowest point to be missed!

Following graph describes the gradient iterate algorithm in computer：



2.Using the gradient descent method to predict house price

There are lots of factors can affect house price, including area, house age， incoming level， population density， etc. Project adopts linear regression model， use gradient descent algorithm to learn the coefficient of house price the prediction linear regression function. The learning data can be from Zillow or house company history reports.

For machine learning ,it needs to choose a loss function , in general, regression models choose ***MSE*** ***(mean square error)*** as loss function. Set y_pred is the prediction function， y is the true value of a house，

then ***loss function = MSE(y_pred, y)*** = $\sum_{i=1}^{m}(y_{predi} - y_i)^2 / m$, m is the number of learning data samples. For MSE, the smaller is the better.

2.1 single variable linear regression

Assume house price and the area is a linear relationship， then we can use single variable linear regression function ***f (x) = w*x + b*** to predict the price of a house， where f denotes the house price, x represents the house area, so ***Loss(w, b)***= $\sum_{i=1}^{m}(w * x_i + b - y_i)^2 / m$,where:

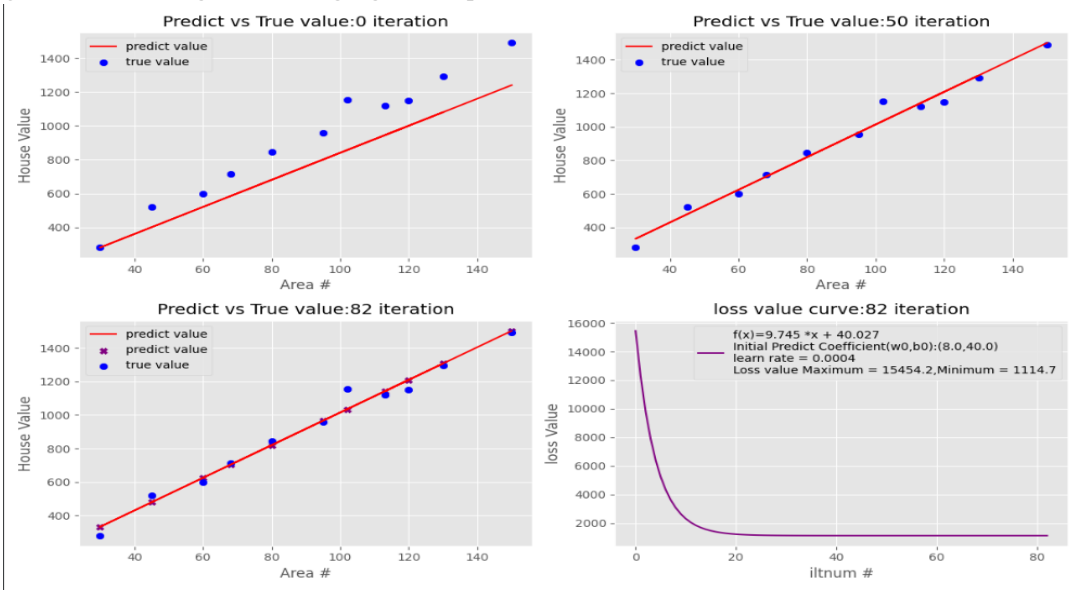
- m is the number of samples in house data set, e.g. m =11 means there are 11 samples in house data;
- y_i is the true price of each house in the data set;
- For every house area x_i, the prediction value is f(x_i)=w∗ x_i+b.

. In the loss function, there is two variables w and b, here try to use the gradient descent method to make *Loss*(w, b) to take the minimum value, then,

$\partial Loss/\partial w = \sum_{i=1}^{m}(w* x_i+b-y_i)\, x_i/m$ ①

$\partial Loss/\partial b = \sum_{i=1}^{m}(w* x_i+b-y_i)/m$ ②

Using Python to implement the gradient descent algorithm, give suitable ***initial value of*** $\theta^0(W^0, b^0)$ and ***learning rate*** α, we can get following figures of prediction and loss values:



A linear regression function learnt is: ***f(x) = 9.745*x+40.27.***

From the Predict-true value contrast graph, we can find that the predicted regression function can match true value closely, the loss value convergent to a small digit, that means gradient descent method is valid.

2.2 multiple variables linear regression

Single variable linear regression can be used to predict the price of house in the same district, but is too simple for the real world, so I try to use multiple variables linear regression. In three-variables linear regression, use area(x1), house age(x2), district(x3) as factors, linear regression $f(x_1, x_2, x_3) = w_1*x_1 + w_2*x_2 + w_3*x_3 + b$, still adopt MSE as loss function, then,

$\partial Loss/\partial w_1 = \sum_{i=1}^{m}(w_1 x_{1i} + w_2 x_{2i} + w_3 x_{3i} - y_i)\, x_{1i}/m$ ①

$\partial Loss/\partial w_2 = \sum_{i=1}^{m}(w_1 x_{1i} + w_2 x_{2i} + w_3 x_{3i} - y_i)\, x_{2i}/m$ ②

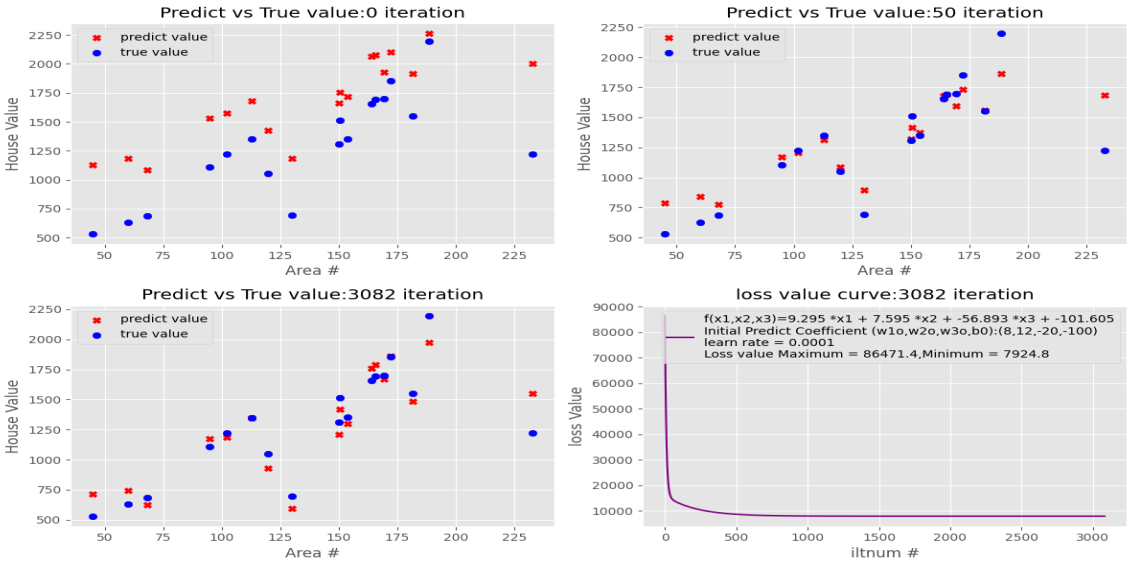
$\partial Loss/\partial w_3 = \sum_{i=1}^{m}(w_1 x_{1i} + w_2 x_{2i} + w_3 x_{3i} - y_i)\, x_{3i}/m$ ③

$\partial Loss/\partial b = \sum_{i=1}^{m} (w_1 x_{1i} + w_2 x_{2i} + w_3 x_{3i} - y_i)/m$ ④

gradient descent algorithm main Python code segment:

```
while True:
    y_pred = w1 * x1 + w2 * x2 + w3 * x3 + b  # function,linear regression
    loss = (y_pred - y) ** 2  # loss function ,a list store every y_pred(i)-y(i) loss value,i=[0:m]
    loss_value.append((0.5 * loss.sum() / m))  # MSE ,all the(y_pred(i)-y(i)) loss values summary
    grad_w1 = 0.5 * np.sum((y_pred - y) * x1) / m  # partial derivative, gradient for w1
    grad_w2 = 0.5 * np.sum((y_pred - y) * x2) / m  # partial derivative, gradient for w2
    grad_w3 = 0.5 * np.sum((y_pred - y) * x3) / m  # partial derivative, gradient for w3
    grad_b = 0.5 * np.sum(y_pred - y) / m  # partial derivative, gradient for b
    w1 -= learn_rate1 * grad_w1  # gradient descent for w1 , from gradient decent direction to  get the next w1
    w2 -= learn_rate2 * grad_w2  # gradient descent for w2 , from gradient decent direction to  get the next w2
    w3 -= learn_rate2 * grad_w3  # gradient descent for w2 , from gradient decent direction to  get the next w3
    b -= learn_rate2 * grad_b    # gradient descent for b , from gradient decent direction to  get the next b
    iltnum += 1
```

Give suitable ***initial value of*** $\theta^0(W_1^0, W_2^0, W_3^0, b^0)$ and ***learning rate*** α, after thousands of iteration computing, we can get figures of prediction and loss values:



A linear regression function learnt is: $f(x_1, x_2, x_3) = 9.295*x_1 + 7.595*x_2 - 56.893*x_3 - 101.605$.

3.Python Programming

In the Project, I use ***numpy*** to compute arrays and matrices, use ***Matplotlib*** for producing figures of regression functions and loss-value, and ***Pandas*** to read house data stored in Excel house_value.xlsx.

All Python codes and test data are released at *https://github.com/johnsonPcj/PythonProject* , gradient_descent1.py is a single-variable linear regression file , gradient_descent2.py is a two-variable linear regression file ,gradient_descent3.py is a three-variable linear regression file.

4.Conclusions

It is found in the project that in order to minimize the loss function value and the prediction accuracy, setting learning rate and coefficient initial value is a difficult problem. I output many figures of the predicted house price and loss values under different learning rates and coefficient initial values for further study of more complex gradient descent algorithms.