CHAPTER 4: MINIMIZATION OF COST

Theory

In this chapter, let's consider the minimization methodology of the cost function from the previous chapter and solve the linear regression problems using cost minimization.

What is Gradient Descent Algorithm?

Let's remember the contents of the previous chapter.

We set the hypothesis of linear regression model like this equation using weights and bias.

$$H(x) = Wx + b$$

And the cost function, which shows how accurately the output of the model follows actual data, is defined as

$$\cos t(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left(H\left(x^{(i)}\right) - y^{(i)}\right)^2$$

The goal of linear regression problem was finding out the value of W and b which minimize the cost function.

Let's consider simple hypothesis for the explanation.

$$H(x) = Wx$$

In this case, we can think to remove b or $b = 0$.

Then the cost could be function of only W.

$$\cos t(W) = \frac{1}{m}\sum_{i=1}^{m}\left(Wx^{(i)} - y^{(i)}\right)^2$$

Let's look into the cost function in order to minimize the cost, then what can we know from there?

We saw this example table last chapter.

x	y
1	1
2	2
3	3

Let's calculate the value of cost when $W = 1$.

$$\text{cost}(1) = \frac{1}{3}\left((1\times 1-1)^2 + (1\times 2-2)^2 + (1\times 3-3)^2\right) = 0$$

In this way when $W = 0$, then the cost is

$$\text{cost}(0) = \frac{1}{3}\left((0\times 1-1)^2 + (0\times 2-2)^2 + (0\times 3-3)^2\right) = 4.67$$

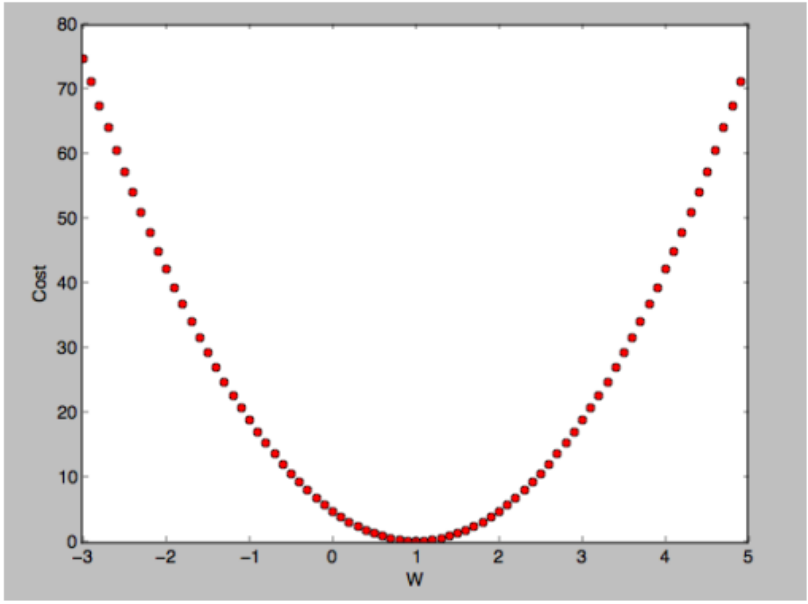
And when $W = 2$, then the cost is

$$\text{cost}(2) = \frac{1}{3}\left((2\times 1-1)^2 + (2\times 2-2)^2 + (2\times 3-3)^2\right) = 4.67$$

We can calculate the cost of several W value.

Then we can imagine the graph that x axis is W, and y axis is cost. Using Matlab and with above training data, we can get below graph.

$$\text{cost}(W) = \frac{1}{m}\sum_{i=1}^{m}\left(Wx^{(i)} - y^{(i)}\right)^2$$



Say again, our goal is to find the W that minimize the cost. From above graph, the value we want to get is 1, and the minimal cost value is 0.

This is the simplest case, so we can know it with our eyes, but we should know the method to find the minimal value with a mechanical method.

The most general method used here is the **gradient descent algorithm**.

We can know from the name of the algorithm, this is an algorithm that goes down the slope.

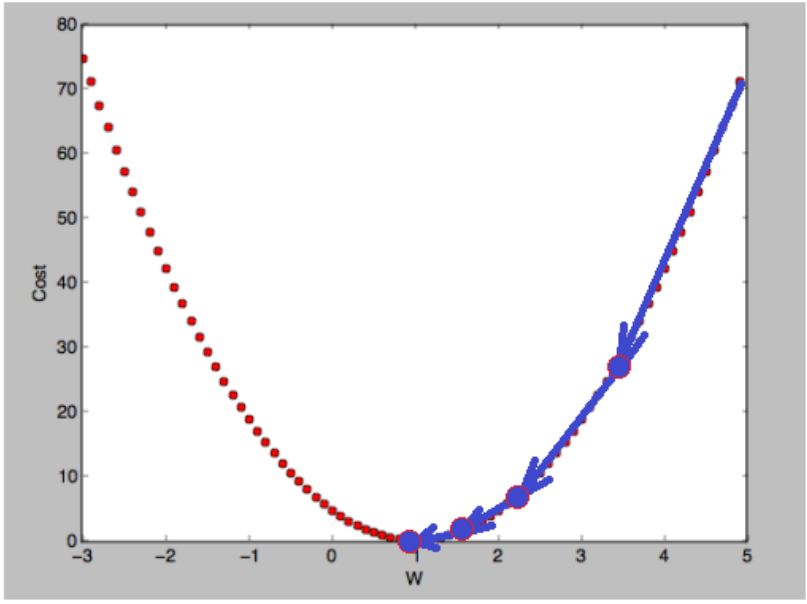
- Gradient descent algorithm minimizes the cost function.
- And gradient descent is used many minimization problems except for machine learning problems.
- For a given cost function, this algorithm will find the W and b to minimize the $\text{cost}(W, b)$.
- And it can be applied to more general function like as multi-variable function $\text{cost}(W_1, W_2, W_3, \cdots)$.

Then what is gradient descent algorithm?

Let's imagine we are on the top of a mountain and are going to down the bottom.

First, we will look around and go down the slope one step. Then we will look around and find the slope again and go down there. In this way, at last, we could arrive the bottom of the mountain.

This is the principal of gradient descent algorithm.



In above figure, let's start from W =5. First, go along the tangential direction. The movement distance is depends on the slope. Then regulate the direction and distance and then go down again. In this way, we can go to the bottom of the graph.

If we start from opposite site W <0, the result should be same.

How gradient descent algorithm works?

- Start with initial guesses
  - Start at 0, 0 (or any other value)
  - Keeping changing W and b a little bit to try and reduce $\text{cost}(W,b)$

- Each time you change the parameters, you select the gradient which reduces $\cos t(W,b)$ the most possible.
- Repeat.
- Do so until you converge to a local minimum

This algorithm has an interesting property. Where you start can determine which minimum you end up, this is the advantage of this algorithm.

In the above we said about slope, then how can we calculate slope?

As you know, the differential is used for calculation of slope.

Let's apply differential into gradient descent algorithm.

For the convenience of differential, we can change the cost function

$$\cos t(W) = \frac{1}{m}\sum_{i=1}^{m}\left(Wx^{(i)} - y^{(i)}\right)^2$$

to

$$\cos t(W) = \frac{1}{2m}\sum_{i=1}^{m}\left(Wx^{(i)} - y^{(i)}\right)^2$$

Minimizing the first expression and the second expression have the same meaning.

Then we can update the W for the minimization problem.

$$W := W - \alpha\frac{\partial}{\partial W}\cos t(W)$$

From now let's call α as the learning rate. The differential of cost function, $\frac{\partial}{\partial W}\cos t(W)$ means the slope at the W.

We can calculate the above equation as follows.

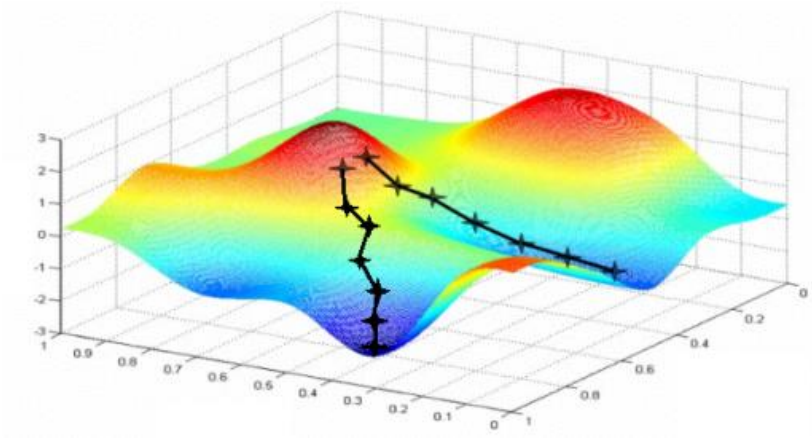
$$W := W - \alpha\frac{\partial}{\partial W}\frac{1}{2m}\sum_{i=1}^{m}\left(Wx^{(i)} - y^{(i)}\right)^2$$

$$W := W - \alpha \frac{1}{2m} \sum_{i=1}^{m} 2\left(Wx^{(i)} - y^{(i)}\right) x^{(i)}$$

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(Wx^{(i)} - y^{(i)}\right) x^{(i)}$$

Last equation is the gradient descent algorithm. We update the value of W with iterative calculation of the last equation.

We can expand this algorithm to the multi-variable problem and could think convex function.

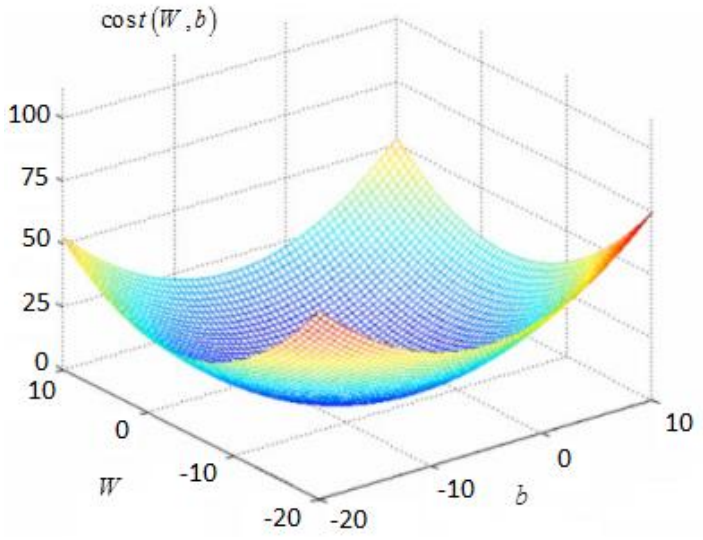


Let's think x axis is W , and y axis is b .

In both cases in above figure, we can see that the two results are different if the initial points are slightly different.

Fortunately, in our problem, the cost function graph has below the shape, and we call it convex function.

$$\cos t\left(W, b\right) = \frac{1}{m} \sum_{i=1}^{m} \left(H\left(x^{(i)}\right) - y^{(i)}\right)^2$$



In case of the convex function, unlike the previous case, the same result can be obtained starting from anywhere.

What is important in applying this gradient descent algorithm is to confirm that the cost function is a convex function. If yes, we could use this algorithm for linear regression problem.

AIM

The aim of the following lab exercise is to understand how to minimize the cost function using gradient descent algorithm in linear regression problem using Tensorflow.

Following steps are required.

Task 1: Build graph using TF operations

Task 2: Run and update graph and get results

Task 3: Using Placeholder

LAB EXERCISE 4: MINIMIZING COST FUNCTION

1. **Build graph using TF operations**
2. **Run graph and get results**

Task 1: Build graph using TF operations

We can implement the minimizing of the cost function here using Tensorflow.

STEP 1: Import tensorflow

First, import the tensorflow package in order to use tensorflow in Python script using below code.

```
import tensorflow as tf
```

STEP 2: Make training data

Let's make simple training data as follows same as the previous lab.

```
# X and Y data
x_train = [1, 2, 3]
y_train = [1, 2, 3]
```

STEP 3: Define model weights

And define W as tensorflow variable.

```
W = tf.Variable(tf.random_uniform([1], -100., 100.))
```

STEP 4: Define model input/output variables

Here we define the input and output variables using the placeholder.

```
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)
```

STEP 5: Define hypothesis

Next, let's define hypothesis.

```
# my hypothesis
hypothesis = W * X
```

We can know above hypothesis is the implementation of below equation we discussed before.

$$H(x) = Wx$$

STEP 6: Define cost function

And cost function could be defined as follows.

```
# Simplified cost function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
```

Of course, this is the implementation of the below cost function.

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left(H\left(x^{(i)}\right) - y^{(i)}\right)^2$$

As we see the previous chapter, this is the only definition of cost function operation, not means calculation. The value will be calculated and updated during training.

STEP 7: Update the weight

The weight update formula is below equation as we discussed before.

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(Wx^{(i)} - y^{(i)}\right)x^{(i)}$$

Let's implement this equation into below commands.

```
# minimize
descent = W - tf.multiply(0.1,
    tf.reduce_mean(tf.multiply((tf.multiply(W, X) - Y), X)))
update = W.assign(descent)
```

Task1 is completed.

Task 2: Run graph and get results

Let's complete the code and run it.

STEP 8: Initialize the tensorflow variables

Initialize the tensorflow variables such as W and b.

```
# launch
sess = tf.Session()

# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
```

STEP 9: Minimize the cost function and display result

Then let's minimize the cost function and print the results.

```
# Fit the line
for step in range(20):
  sess.run(update, feed_dict={X: x_data, Y: y_data})
  print(step, sess.run(cost,
            feed_dict={X: x_data, Y: y_data}), sess.run(W))

print(sess.run(hypothesis, feed_dict={X: 5}))
print(sess.run(hypothesis, feed_dict={X: 2.5}))
```

The coding is complete here. We print the value of W each step and predict the model output when X=5 and X=2.5 at last.

Using this we can evaluate the accuracy of trained model and result of cost minimization with gradient descent algorithm.

Then the full code is below.

```python
import tensorflow as tf

# data set
x_data = [1., 2., 3.]
y_data = [1., 2., 3.]

# try to find values for w and b that compute y_data = W * x_data
# range is -100 ~ 100
W = tf.Variable(tf.random_uniform([1], -100., 100.))

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

# my hypothesis
hypothesis = W * X

# Simplified cost function
cost = tf.reduce_mean(tf.square(hypothesis - Y))

# minimize
descent = W - tf.multiply(0.1, tf.reduce_mean(
    tf.multiply((tf.multiply(W, X) - Y), X)))
update = W.assign(descent)

# launch
sess = tf.Session()

# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# fit the line
for step in range(20):
    sess.run(update, feed_dict={X: x_data, Y: y_data})
    print(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}),
          sess.run(W))

print(sess.run(hypothesis, feed_dict={X: 5}))
print(sess.run(hypothesis, feed_dict={X: 2.5}))
```

Let’s run this code.

Then we can see the results such as below.

```
0 10679.2 [ 48.83728027]
1 3037.65 [ 26.51321602]
2 864.042 [ 14.60704803]
3 245.772 [ 8.25709152]
4 69.9084 [ 4.87044859]
5 19.885 [ 3.06423903]
6 5.65619 [ 2.10092759]
7 1.60887 [ 1.5871613]
8 0.457635 [ 1.31315267]
9 0.130172 [ 1.16701472]
10 0.0370266 [ 1.08907449]
11 0.010532 [ 1.04750645]
12 0.00299577 [ 1.02533674]
13 0.000852132 [ 1.01351297]
14 0.000242385 [ 1.00720692]
15 6.89433e-05 [ 1.00384367]
16 1.96102e-05 [ 1.00204992]
17 5.57751e-06 [ 1.00109327]
18 1.5863e-06 [ 1.00058305]
19 4.51488e-07 [ 1.00031102]
[ 5.00155497]
[ 2.50077748]

Process finished with exit code 0
```

As we can know from results, the cost value is decreased from 10679.2 to 4.5e-07, and the value of W is converged from 48.8 to 1.0003.

And when we take into the X as 5 and 2.5, the output of the model is 5.001 and 2.5007, from this we can know the model optimized accurately.

Task2 is completed.

LAB CHALLENGE

Challenge

What is important in understanding and applying gradient descent algorithm into linear regression problem is should check the cost function is the convex function.

If cost is the convex function then we could use gradient descent algorithm, but if not, unfortunately, we couldn't use this method.

SUMMARY

The gradient descent algorithm is the most used general method in linear regression problems.

- Minimize cost function.
- Gradient descent is used many minimization problems.
- For a given cost function, *cost (W, b)*, it will find *W, b* to minimize cost
- It can be applied to more general function: *cost (w1, w2, …)*

REFERENCES

- https://en.wikipedia.org/wiki/Gradient_descent
- https://en.wikipedia.org/wiki/TensorFlow
- https://en.wikipedia.org/wiki/Machine_learning

INDEX