HW5 Linear Regression

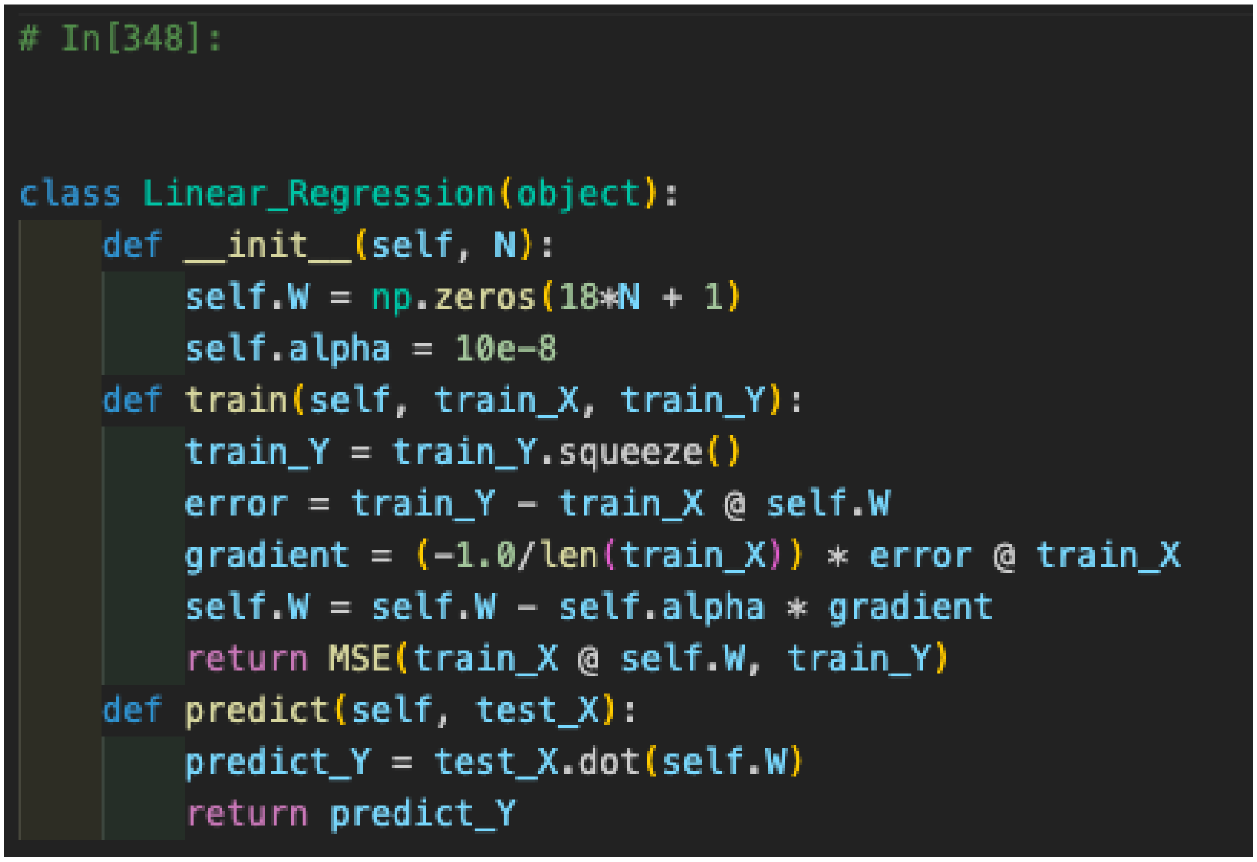
Brady Huang

06/12/2021

Abstract

This homework uses Linear Regression to find the best model for predicting the PM2.5 in the next hour, by updating the weight and features amount we can get the better model for precise prediction. However, the system of linear equations in multiple unknown has formula solution. We can simply find the best weight that fit in our training model.

1 Complete the TODO part in the Linear Regression class



```
# In[348]:


class Linear_Regression(object):
    def __init__(self, N):
        self.W = np.zeros(18*N + 1)
        self.alpha = 10e-8
    def train(self, train_X, train_Y):
        train_Y = train_Y.squeeze()
        error = train_Y - train_X @ self.W
        gradient = (-1.0/len(train_X)) * error @ train_X
        self.W = self.W - self.alpha * gradient
        return MSE(train_X @ self.W, train_Y)
    def predict(self, test_X):
        predict_Y = test_X.dot(self.W)
        return predict_Y
```

Figure 1: Linear Regression class with train and predict function

If we need to update the weight in Linear Regression class, we have to update the weight based on the first derivative function of Loss function, denoted as L(w). The derivative function of L(w) called gradient which indicates the slope in the solution domain, if gradient is close to 0 means it's close to solution. Thus, We take negative direction of the slope to find the global minimum solution. The old weight would be updated based on alpha times gradient.

First we calculate the gradient, and then update the old weight with alpha times gradient. Alpha means the learning rate as a constant to update the weight.

2 Plot the training and testing error N from 1 to 48

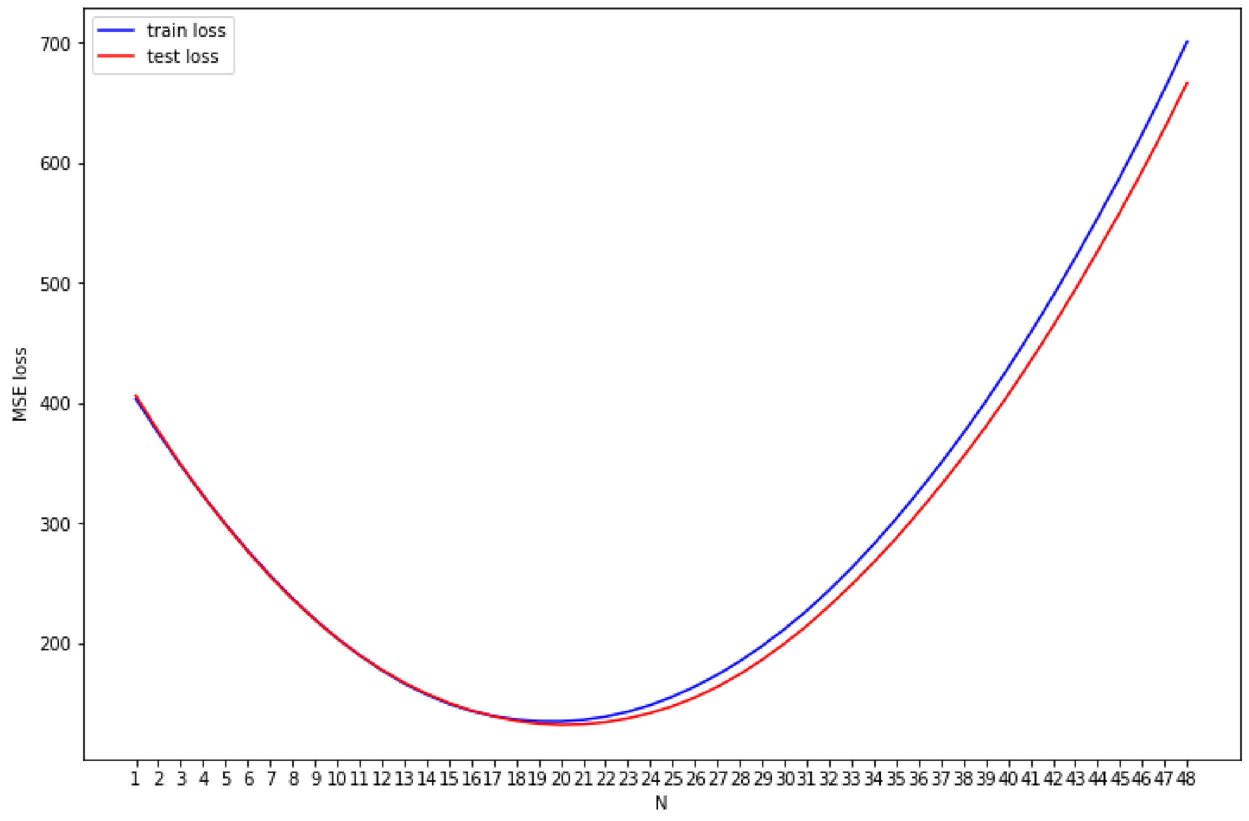


Figure 2: Loss graph with training and testing set

N indicates how many hours to include in your features, N equals to 1 means only use 1 hour to predict PM2.5 in the next our. N equals to 48 means include 48 hours to predict the next hour. The graph shows that with learning rate keeping in 10^{-7} getting a pretty result at the N with 20. Ideally, if we include more features, the prediction should be more reliable. But the graph shows that it's a curve with high MSE loss on the two sides. The reason for it is that I think low N means not enough feature to get a good prediction, and high N means overfitting might occur.

3 Based on the experiment so far, give a complete method, result and observation in the report

3.1 Change the learning rate

3.1.1 10^{-6}

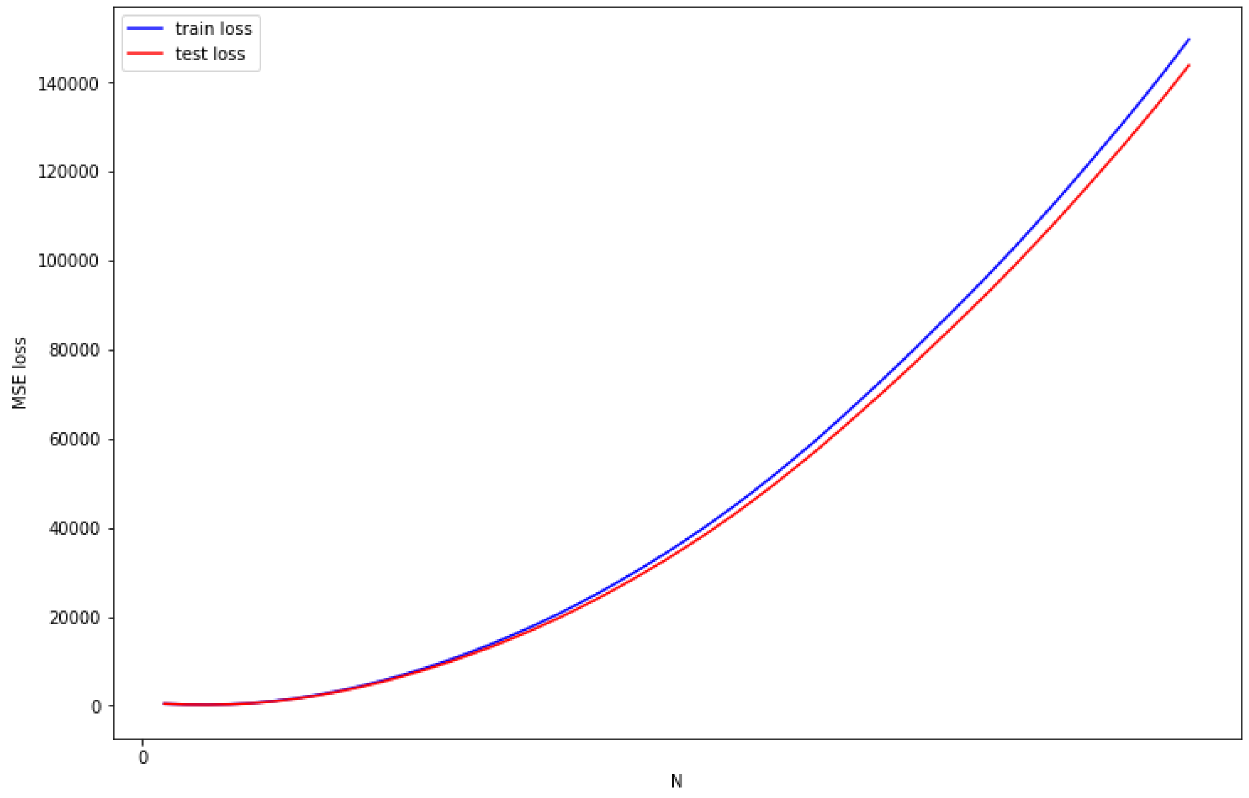


Figure 3: Increase the learning rate

If tune up the learning rate one order, the loss would increase with larger N, we will never find the optimistic point.

3.1.2 10^{-8}

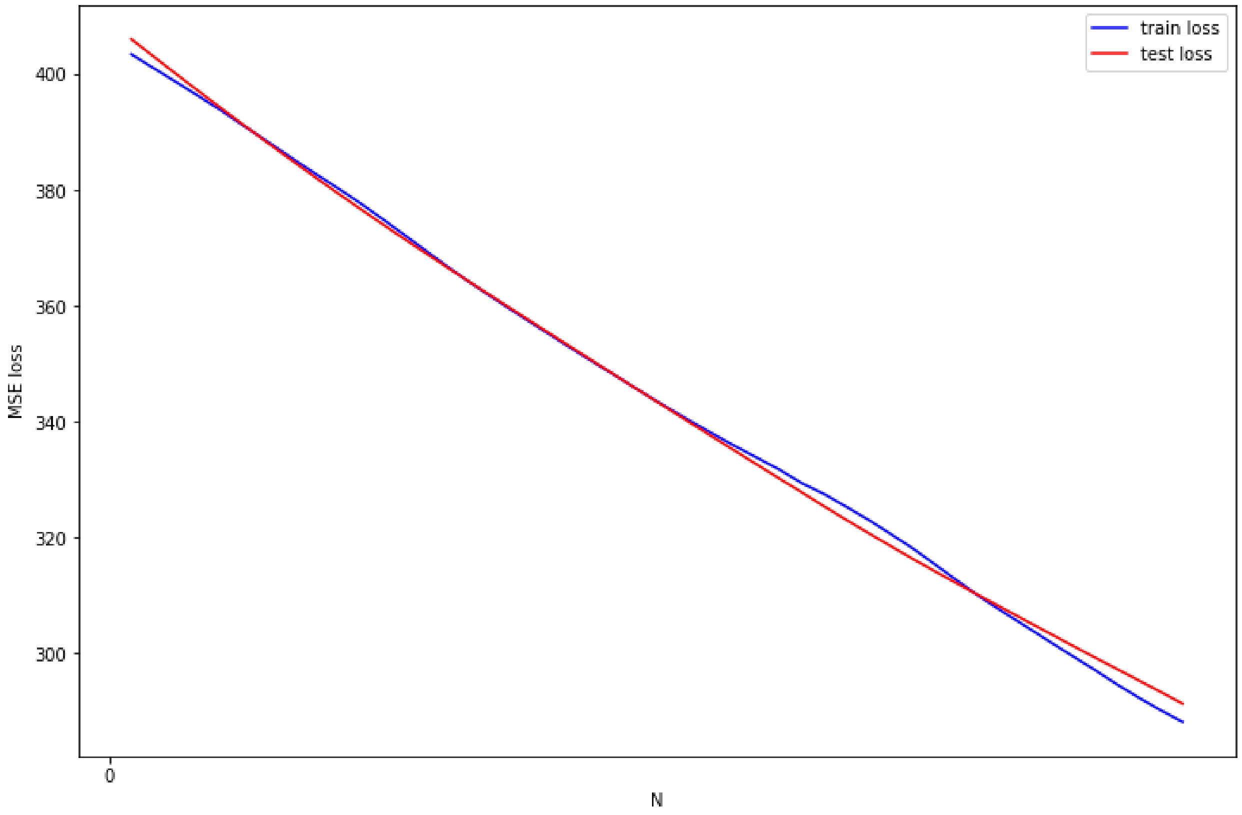


Figure 4: Decrease the learning rate

However, if we tune down an order, the learning seems to be very slow.

3.2 Update the weight multiple times

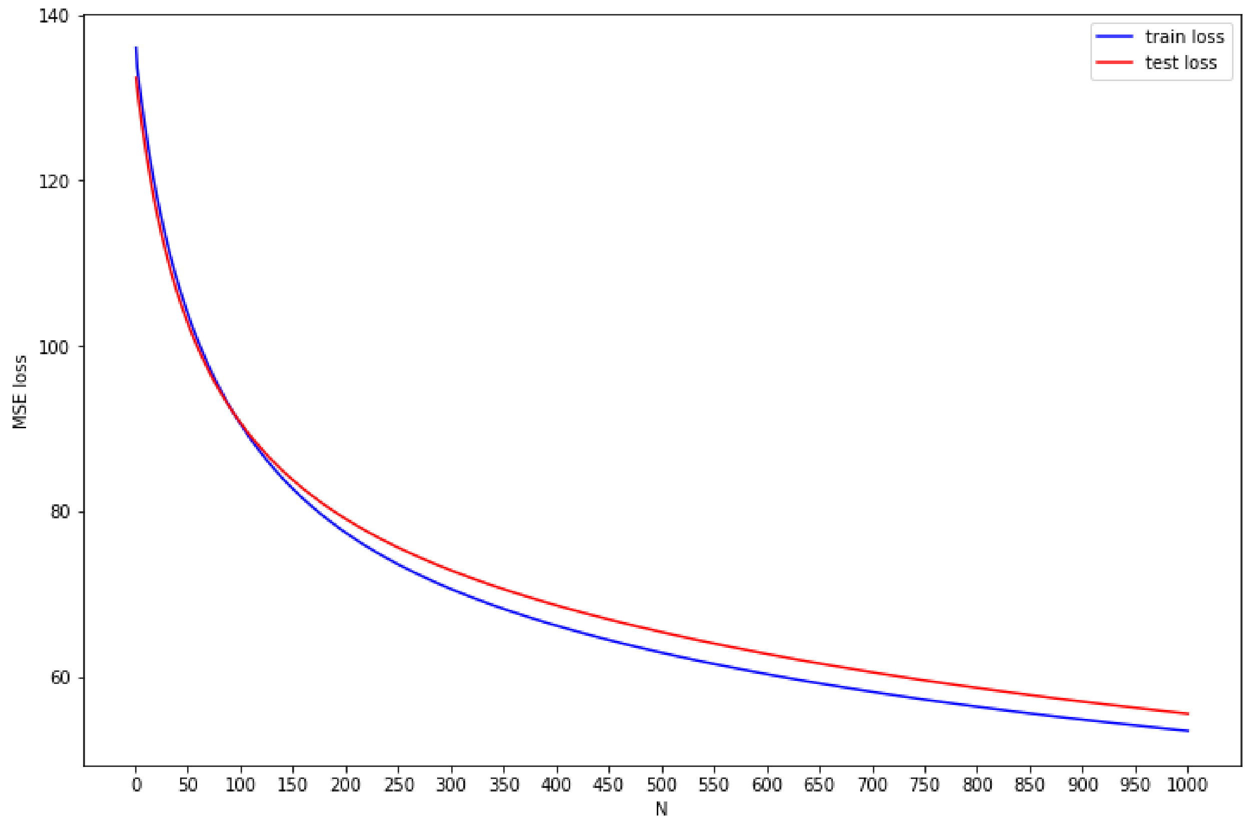


Figure 5: Update the weight

We keep the N number with 20, and keep updating the weight, what would be the loss in the end? I train the Linear Regression model 1000 times with N equals 20. Training loss attain 53.45 and testing loss 55.53. Very excited!

3.3 What is the optimal result of the weight

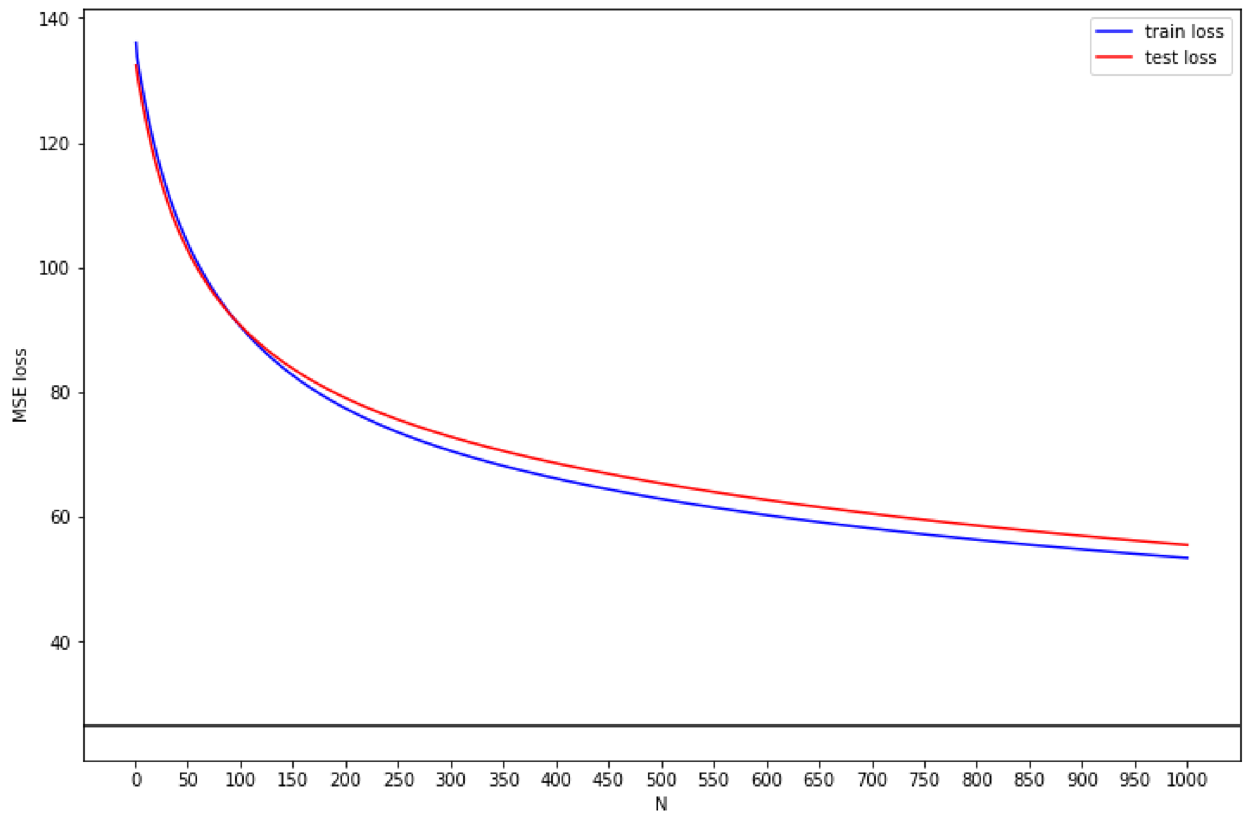


Figure 6: Update the weight

In the course, the professor had taught we can find the P_w projection function on the y vector. Then we can get the optimal function. By applying this weight to our training set, we can get the optimal MSE 26.53, and training set 26.20. So if we look back to our model, we still have room to improve. I think we have to add a momentum or gradually lower the learning rate would benefit. Look the figure surprise me there is still huge distance there.