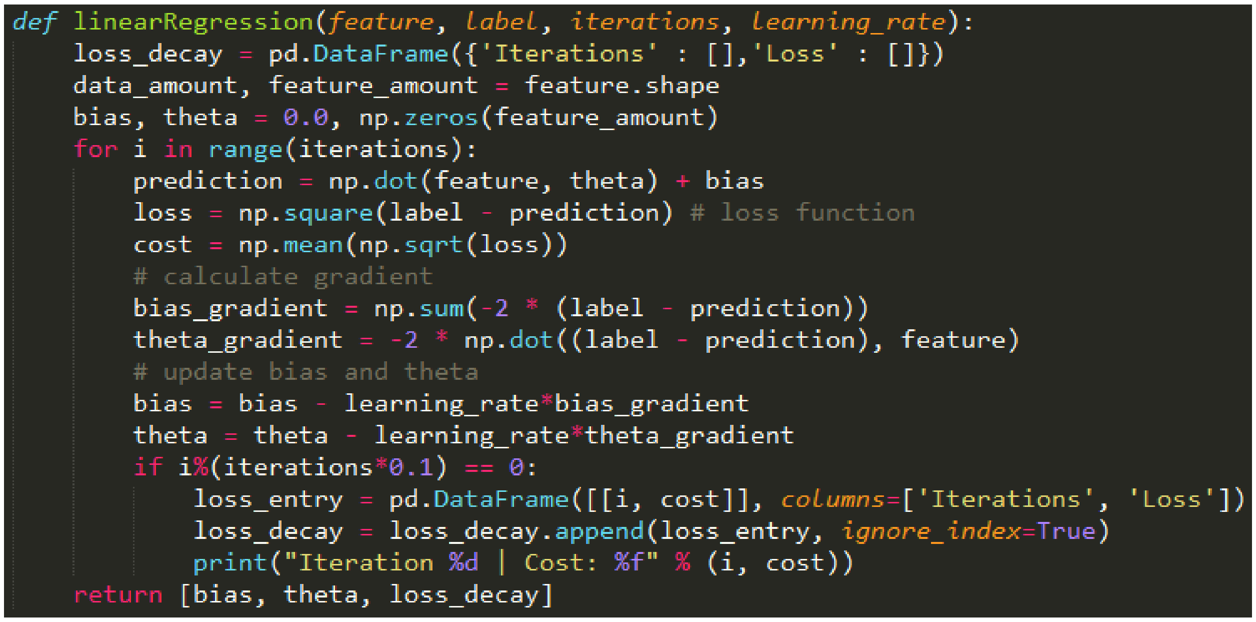
Linear Regression in Gradient Descent



```
def linearRegression(feature, label, iterations, learning_rate):
    loss_decay = pd.DataFrame({'Iterations' : [],'Loss' : []})
    data_amount, feature_amount = feature.shape
    bias, theta = 0.0, np.zeros(feature_amount)
    for i in range(iterations):
        prediction = np.dot(feature, theta) + bias
        loss = np.square(label - prediction) # loss function
        cost = np.mean(np.sqrt(loss))
        # calculate gradient
        bias_gradient = np.sum(-2 * (label - prediction))
        theta_gradient = -2 * np.dot((label - prediction), feature)
        # update bias and theta
        bias = bias - learning_rate*bias_gradient
        theta = theta - learning_rate*theta_gradient
        if i%(iterations*0.1) == 0:
            loss_entry = pd.DataFrame([[i, cost]], columns=['Iterations', 'Loss'])
            loss_decay = loss_decay.append(loss_entry, ignore_index=True)
            print("Iteration %d | Cost: %f" % (i, cost))
    return [bias, theta, loss_decay]
```

Figure 1 Linear Regression function by Gradient Descent

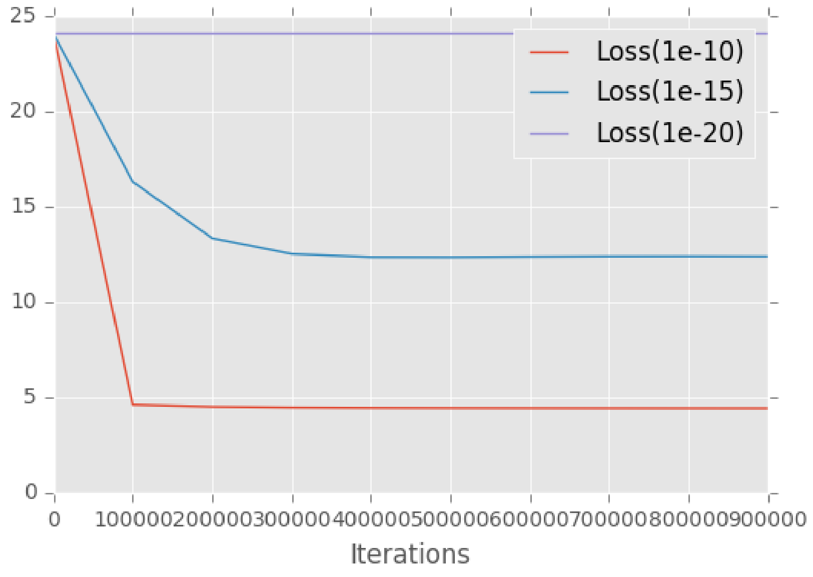


Figure 2 Loss function under different learning rate (with 1e6 iterations)

Window Sliding Method

In order to have more data for training, window sliding method is used in dataset parsing:

Using 9 hour records to predict 10th hour's PM2.5 value, dataset was divided by 10 for each group. However, it may only get 480 data points for training, so using the window sliding method with fixed window size = 10, 3360 data points are gathered for training.

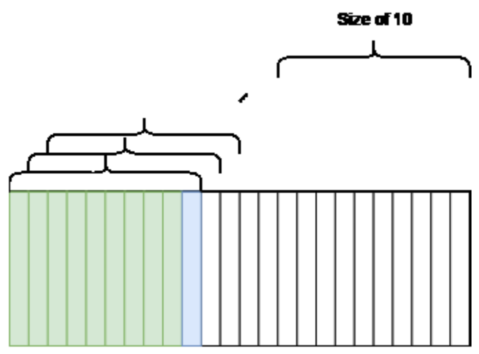


Figure 3 Sliding Window in daily record

Feature selections

Refer to the domain knowledge, PM2.5 is the average of serval previous hour's PM2.5 value, which only determined by its own value. However, the performance doesn't increase when other features are removed that the performance even get worst.

Stochastic Gradient Descent and Adagrad

Attempt to get higher score in Kaggle, SGD and Adagrad are introduced into kaggle_best.py , however, it only level off the loss even early and doesn't improve the performance (scored about 6.6x in public set in Kaggle)

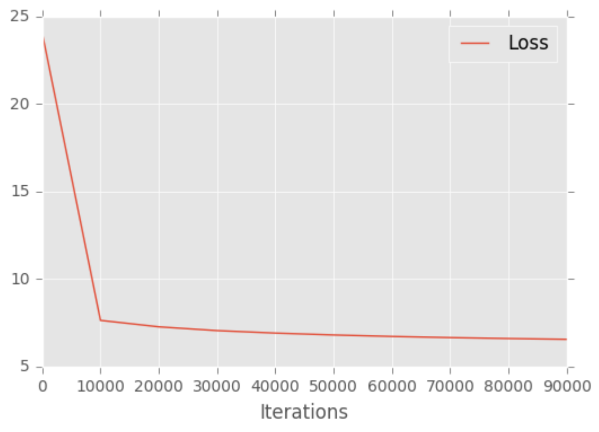


Figure 4 Loss value of SGD+Adahrad with (iterations: 1e5, learning rate: 5e-15)

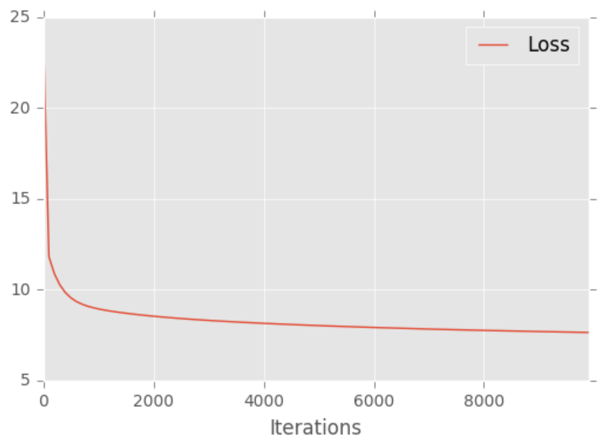


Figure 5 Loss value of SGD with model(mx^2+nx+b)

Conclusion

Neither rebuild Model (from mx+b to mx^2+nx+bias) nor other approach of gradient descent (e.g: SGD, Adagrad, regulation of features) can lower the loss.

Note: result on kaggle mostly just using linear_regression.py to reproduce and kaggle_best.py is just for the discussions