Introduction:

1. Description

In the assignment, I used Concrete Compressive Strength dataset to perform linear regression.

The input includes first eight columns (Cement, Blast Furnace Slag, Fly Ash, Water, Superplasticizer, Coarse Aggregate, Fine Aggregate, Age) as input and the ninth column (Concrete compressive strength) as output.

From the given eight inputs, I use Uni-variate linear regression on each feature input to try to predict the outcome of the output. Then, I use Multi-variate linear regression on all eights feature to try to predict the output. Besides, I will use MSE and gradient descent as loss function/metric to measure the performance of linear regression.

Finally, I will compare 9 models’ performance on both train and test dataset to draw conclusions from them.

1. Details of algorithm

Uni-variate linear regression

Formula:

Loss function (MSE):

I choose several fixed learning rates (0.00001, 0.01…), stopping criterion is after 10000 iterations. The values of m and b will change when the algorithm is running.

mi = m - bi = b -

Multi-variate linear regression

Loss function is = \* ||2

pred = ( . ) (dot product between a and x)

= (a0, a1, …, an) – a0 is the intercept and = (1, x1, x2, …, xn) where xi is the input feature.

new = - = a – XT \* ()

1. Pseudocode

Input m , b, X\_train\_data, y\_train\_data, learning rate, epochs

Output MSE and m, b after iteration

Uni-variate linear regression

def uni\_linear(m,b,x,y,lr):

Db, Dm = 0

n = len(x)

for i in range(10000):

ypred = m\*x + b

Dm = (-2/n) \* sum(x \* (y - ypred))

Db = (-2/n) \* sum(y - ypred)

m =m- lr \* Dm

b =m- lr \* Db

return m, b

def MSE(m,b,x,y):

ypred=m\*x+b

n = len(y)

sum = 0

for i in range(n):

diff = y[i] - ypred[i]

sum += diff\*\*2

return sum/n

Multi-variate linear regression:

Input m , b, X\_train\_data, y\_train\_data, learning rate, epochs

Output MSE and m, b after iteration

def GradientDescent(x, y, m, b, learning\_rate, epochs):

for epoch in range(epochs):

ypred = x.dot(m) + b

loss = ypred - y

Dm = x.T.dot(loss) / len(y)

Db = np.sum(loss) / len(y)

m = m - learning\_rate\*Dm

b = b - learning\_rate\*Dbreturn w, b

def MSE(X, m, b, y):

y\_pred= X.dot(m) + b

n = len(y)

sum = 0

for i in range(n):

diff = y[i] - y\_pred[i]

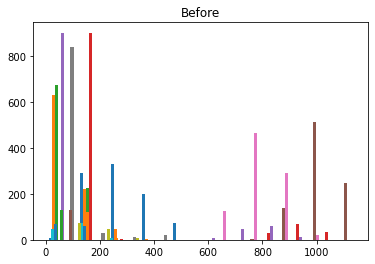
sum += diff\*\*2

return sum/n

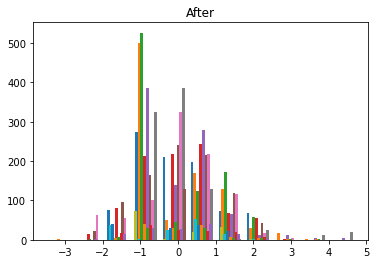
1. Normalize

In my code, I use sklearn.preprocessing.StandardScaler to normalize x, after normalization, the distribution of x value has change from 0-1000 to -3 - 5, which becomes more closer to 1-0, that is easier and more precise to compute the loss of the model

Before normalize, the scatter plot of adata



After normalize, the scatter plot of adata



Results:

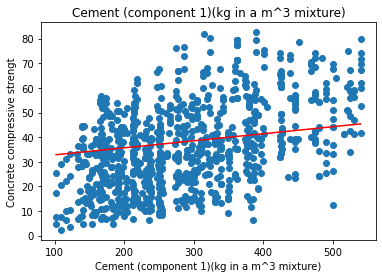
Before normalize:

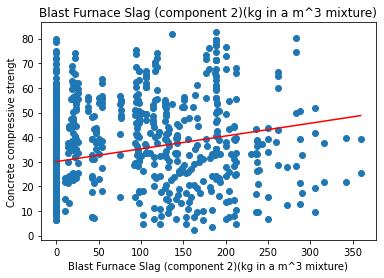


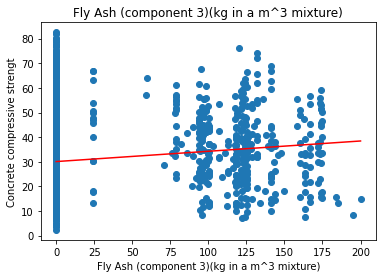
After normalize:

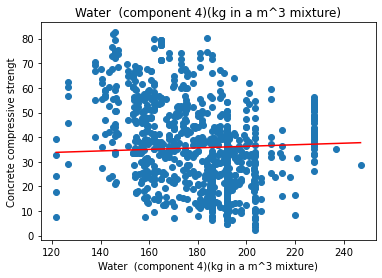


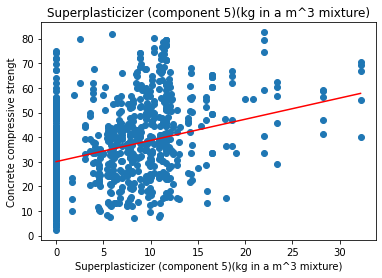
Plots:

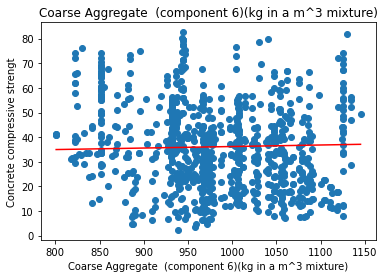


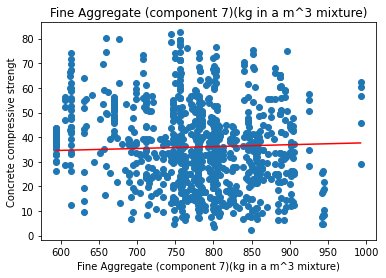


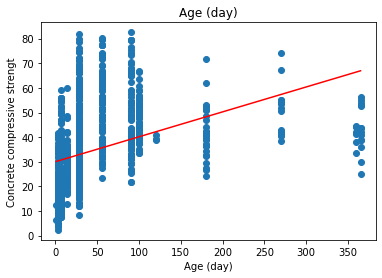












Discussion

1. **Describe how the different models compared in performance on the training data. Did the same models that performed well on the training data do well on the testing data?**

In my result, the results of uni-variate linear regression are not stable, it varies from 324-250 on training set, and from 188 to 129 on testing set. On the contrary, the multi-variate linear regression performs much better, its MSE on traing set is 115 and 58.8 on testing set, much smaller than MSE in the uni-variate linear regression.

As for uni-variate linear regression, it doesn’t perform well because it only has one feature to fit in, and the data is different among columns. But it can be found that if the model performs well in training set (low MSE), it will also do a good job in testing set as the chart shows blew.



1. **Describe how the coefficients of the uni-variate models predicted or failed to predict the coefficients in the multi-variate model**

Combining with the mse\_train results in the former charter, it shows that the value of coefficients has little relationship with the model’s performance. Sometimes the coefficient in uni-variate model is very high, but in multi model is quite low. The failure is because the importance of different features is not the same, and the multi-variate linear regression model doesn’t consider completely. Maybe the failed reason is that currently I used fixed epochs and learning rate, which can’t be enough to reach the optimal result.



1. **Draw some conclusions about what factors predict concrete compressive strength. What would you recommend to make the hardest possible concrete?**

From the result, I recommend using Cement, Water and Coarse Aggregate to make the hardest possible concrete. Because the coefficients of these components are similar, the MSE results are small.

1. **Extra: Comparisons to the results** 

The chart above is the MSE result of data before normalize and after normalize.

As we can see, the result in uni-variate linear regression performs much better after normalize (MSE becomes more smaller), but the result doesn’t change much in multi-variate linear regression. I think the reason why MSE becomes lower is because after normalization, the data get closer and standardized.