1. **Introduction**
2. **Description/formulation of the problem**

I use the 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 nineth column as output (Concrete compressive strength)

From the given eight input, I will first try to do Uni-variate linear regression on each feature input to try to predict the outcome of the output. Then, I will do Multi-variate linear regression on all eights feature to try to predict the output. For both the Uni-variate linear regression and Multi-variate linear regression, I will use MSE as loss function/metric to measure the performance of linear regression. The lower the MSE, the better the model would be.

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

1. **Details of your algorithm**
2. Uni-variate linear regression

Model formula: pred = m \* x + b

Loss function (MSE): L = =

(the summation is performed on all data point)

Goal: minimize the MSE

Update rule: as the negative gradient direction is the steepest direction, I update both parameters b and m of its direction of negative gradient (with step size ):

mnew = m - = m -

bnew = b - = b - = b -

I keeps updating until the Loss function is improved insignificantly (at a chosen threshold of Lnew - Lold < threshold I will stop).

1. Multi-variate linear regression

The idea is same for uni-variate linear regression, only the formula changes.

Model formula: 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.

Loss function (MSE): L = = = \* ||2

(X is the whole dataset’s input – n \* (numFeature + 1) matrix, is the output – n \* 1 matrix)

new = - = a – XT \* ()

1. **Pseudo-code**
2. Uni-variate linear regression

prev\_loss = inf

n = .shape[0]

loss =

m = b = random()

While prev\_loss – loss > threshold:

y\_pred =

m =

b = b – \* sum()

prev\_loss = loss

loss =

return m, b

1. Multi-variate linear regression

prev\_loss = inf

X = append(1, X)

a = [random() for \_ in (numFeature + 1)]

loss = norm(X \* a - y)) ^ 2

While prev\_loss – loss > threshold:

a = a – \* X \* (X \* a - y)

prev\_loss = loss

loss = norm(X \* a - y)) ^ 2

return a

1. **Results**
2. **MSEs of models on the training/ testing dataset**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | All (multi regression) |
| Feature used | Cement | Blast Furnace Slag | Fly Ash | Water | Superplasticizer | Coarse Aggregate | Fine Aggregate | Age | All |
| MSE train | 511.5 | 1619 | 1619 | 1112.7 | 1619 | 348.1 | 374.5 | 1619 | 317.5 |
| MSE test | 193.2 | 582.7 | 582.7 | 353.9 | 582.7 | 84.9 | 85.5 | 582.7 | 148.8 |

1. **Plots of trained uni-variate models**

Helo

1. **Discussion**
2. **Different models compared in performance on the training data**

Helo

1. **Performance of the uni-variate models predicted or failed to predict which features were “more important”in the multi-variate model(s)**

Helo

1. **Draw some conclusions about what factors predict concrete conpressive strength.**

Helo