Introduction to Machine Learning Homework 3

# Environment used

For this homework we used python 3, pyCharm IDE and GitHub as a remote repository.

# Visualization of all the features with the target

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

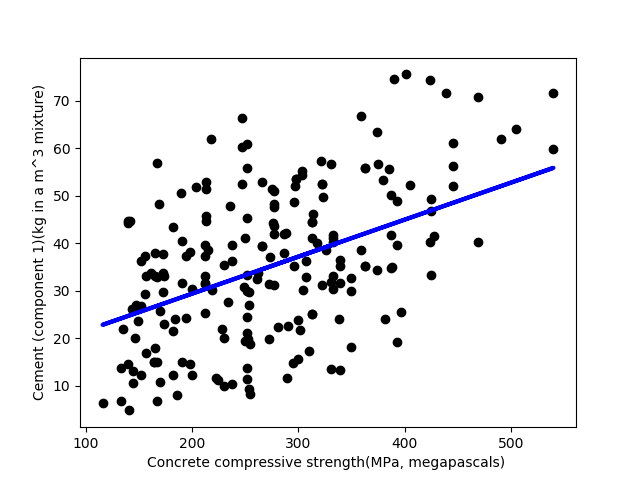
Plots with the feature on the y-axis and the target on the x-axis.

The code, graph, r2\_score, weight and bias for problem 1

We chose to use the feature cement to test our functions in problem 1 and 2.   
  
The code:

# Create linear regression object  
regr = linear\_model.LinearRegression()  
  
# Train the model using the training sets  
regr.fit(X, y)  
  
# Make predictions using the testing set  
y\_pred = regr.predict(X\_test)

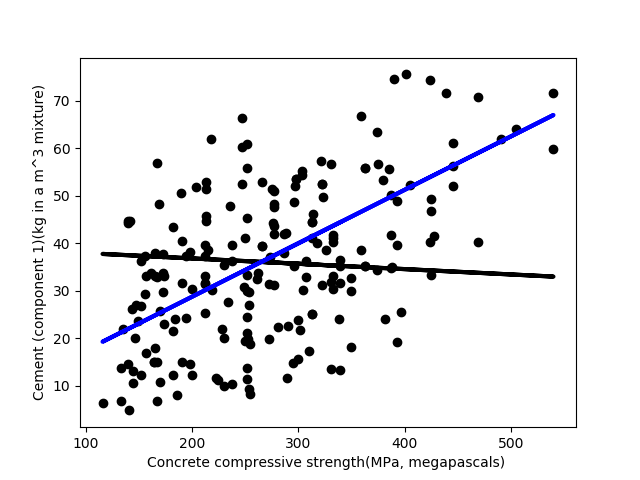
The graph:



Results:   
Weight: 0.07789607,   
Bias: 13.785171881915826  
Mean squared error: 192.7847985543255  
Regression score: 0.2518351362829808

The code, graph, r2\_score, weight and bias for problem 2The code for the gradient descent. We looped over this code to get closer to an accurate result:

**def gradient\_descent**(weight, bias, x\_vals, y\_vals, learning\_rate):  
 *"""  
 A function to calculate the gradient descent of two arrays of integers* ***:param*** *weight: the estimated weight as a float* ***:param*** *bias: the estimated bias as a float* ***:param*** *x\_vals, y\_vals: arrays of integers* ***:param*** *learning\_rate: the learning rate as a float* ***:return****: the new weight and bias  
 """* w\_grad = 0  
 b\_grad = 0  
 x\_vals = list(x\_vals)  
 y\_vals = list(y\_vals)  
 n = len(x\_vals)  
 **for** i **in** range(n):  
 x = x\_vals[i]  
 y = y\_vals[i]  
 w\_grad += -(2/n) \* (y - ((bias \* x) + weight))  
 b\_grad += -(2/n) \* x \* (y - ((bias \* x) + weight))  
 weight -= (learning\_rate \* w\_grad)  
 bias -= (learning\_rate \* b\_grad)  
 **return** weight, bias

  
The graph:

Results:   
Weight: 0.11249114   
Bias: 6.21681447   
Mean squared error: 196.92692335951128  
Regression score: 0.23576025764312047

# Compare Problem1 and Problem2, show what you got

The regression score for the two problems using the feature Cement are very similar, 0.2518351362829808 using the model from sklearn and 0.23576025764312047 using our own model. This shows that our model is fairly good!

The code, MSE, and the r2\_score for problem 3

**for** i **in** range(no\_iterations):  
 **for** j **in** range(no\_variables):  
 w[j] = w[j] - (1 / (2 \* N)) \* learning\_rate \* (  
 (y\_train - (X\_train.dot(w))).T.dot(np.negative(X\_train[:, j])))  
  
y\_pred = X\_test.dot(w)

Weights get updated on each iteration, you can choose the number of variables and number of iterations. Then weights are multiplied with the input to get the predictions.

Training:

MSE using 3 variables (cement, water and superplasticizer): 183.61980343136517

R2 Score: 0.3536480143598838

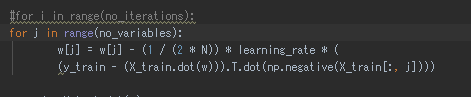
Testing:

MSE using 3 variables (cement, water and superplasticizer): 195.5273982258517

R2 Score: 0.2411915755620896

# Compare the performance between two different update method

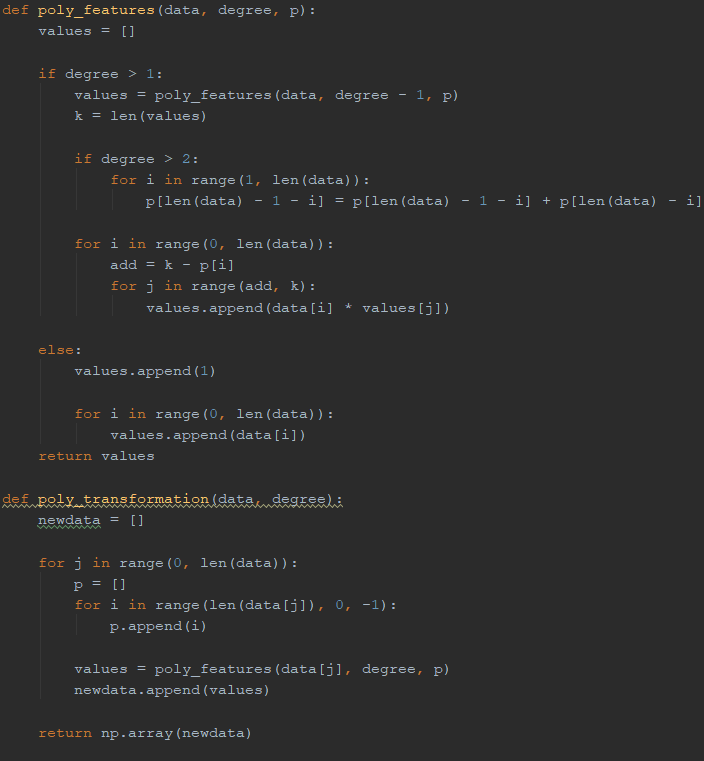
Code for updating wj only :



MSE for this method: 505.8533525014217

R2 Score: -0.9631304302676924

The code, MSE, and the r2\_score for problem 4

****

poly\_transformation is used to generate a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. For example, if an input sample is two dimensional and of the form [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2]. Then generated features are used in linear regression model from problem3.

Training:

MSE using 3 variables (cement, water and superplasticizer): 159.68174503430103

R2 Score: 0.23747370784473054

Testing: 0.4379113197777814

MSE using 3 variables (cement, water and superplasticizer): 149.61184783098173

R2 Score: 0.41938198145132544

# Answer the question

## 1. What is overfitting?

Overfitting occurs when the model only performs good when testing on the training set of the data, because it captures the noise of training data. It means that model is too specific for training data and is not good with new data.

## 2. Stochastic gradient descent is also a kind of gradient descent, what is the benefit of using SGD?

Since stochastic gradient descent samples a subset of summand functions at every step it is very effective for large-scale machine learning problems.

## 3. Why the different initial value to GD model may cause different result?

Because there might be more than one local maximum so a different initial value might make the model converge to a different local maximum.

## 4. What is the bad learning rate? What problem will happen if we use it?

Bad learning rate is when the learning rate is too small or too big, if its too small it might take too much time to train the model and if it is too big the weights would change too fast which might cause it to never converge and even go to infinite values.

## 5. After finishing this homework, what have you learned, what problems you encountered, and how the problems were solved?

We have learned how to use linear regression models to make predictions from data. We learned how the linear regression model works and how to code the weight updating by ourselves. We learned to use multiple variables in linear regression and to use this to make polynomial models. One of the problems we had was that the weight was going to infinite values we solved it by making the learning rate much smaller. Another problem was getting negative r2 scores to solve it we had to try different combinations of variables.

**10. Bonus(10%)**