COEN 240 Machine Learning

Homework #1

Name: Jinhao Wang ID: 4302178

Problem1:

$t(x, w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_N x^N = \sum_{j=0}^{N} w_j x^j$

$w = [w_0, w_1, \cdots, w_N]^T$

As we know, $t(x, w)$ is linear in w at each certain point of x

Therefore, for the nth data sample $x_n = [1, x_n^1, x_n^2, \cdots, x_n^N]^T$

Linear Regression Model: $t(x_n, w) = w^T x_n$

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{w^T x_n - t_n\}^2$$

$$= \frac{1}{2} \| w^T x_n - t_n \|_2^2 \quad \text{reverse the 2-norm of vector}$$

$$= \frac{1}{2} (\vec{w} x_n - \vec{t_n})^T \cdot (\vec{w} x_n - \vec{t_n})$$

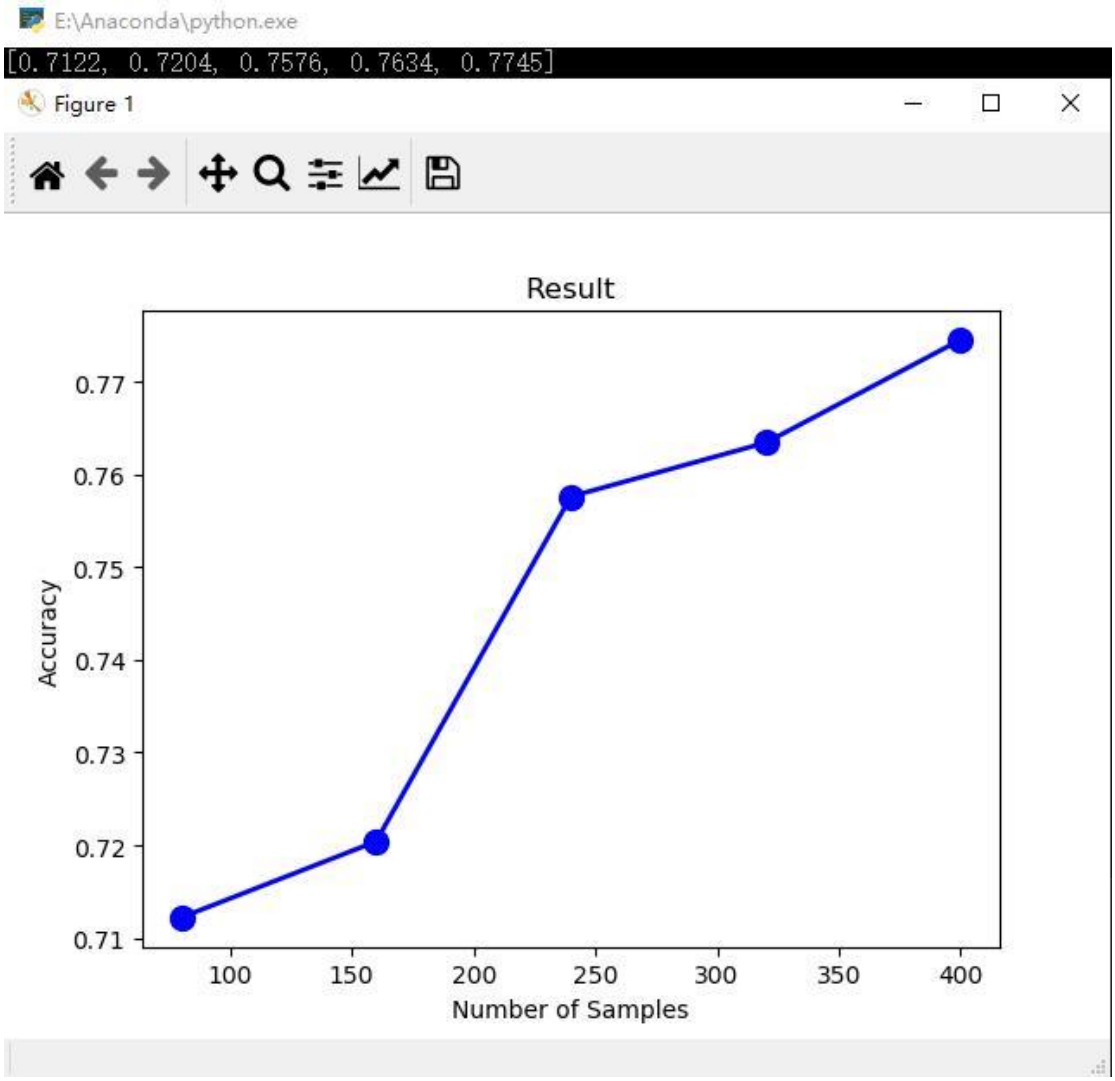
$$= \frac{1}{2} \cdot (\vec{w}^T x_n^T x_n \vec{w} - \vec{w}^T x_n^T \vec{t_n} - \vec{t_n}^T x_n \vec{w} + \vec{t_n}^T \vec{t_n})$$

$$E'(\vec{w}) = x^T x \vec{w} - \frac{1}{2} x^T \vec{t} - \frac{1}{2} x^T \vec{t} + 0$$

$$= x^T x \vec{w} - x^T \vec{t}$$

This is the derivation of the error function. To minimize the error function, let the derivation equal $\vec{0}$

Problem2:



This plot shows the prediction accuracy rate under different number of samples.

We observe that the general trend of accuracy rate increases with more samples categorized into training set. The reason of this could be, with more training data, the model is trained to be more accurate for prediction.

Actually, the accuracy rate fluctuates when we run the program multiple times. Because for different number of N, we shuffle the datasets, which may lead to randomness during the training process. Maybe the data shuffled into the training set is not as good as the previous training set. Therefore, the accuracy rate depends on what samples are shuffled into the training set. However, the general scope of accuracy rate is always between 0.7 ~0.8, and so is the general trend, which tends to grow with the increase of training set.

Attachment:

Problem2 Code:

```
import pandas as pd
from numpy import *
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

input = pd.read_csv('pima-indians-diabetes.csv')
numberN = [40, 80, 120, 160, 200]
#dataframe contains only diabetes patients
diabetes = input[input["result"]==1]
#dataframe contains only nondiabetes patients
nondiabetes = input[input["result"]==0]
number_total = len(input.index)
number_diabetes = len(diabetes.index)
number_nondiabetes = len(nondiabetes.index)
res = []
#dataframe contains only 8 parameter values for diabetes patients
x_diabetes = pd.DataFrame(diabetes, columns = diabetes.columns[:-1])
#dataframe contains only 8 parameter values for nondiabetes patients
x_nondiabetes = pd.DataFrame(nondiabetes, columns = nondiabetes.columns[:-1])
y_diabetes = pd.DataFrame(diabetes, columns = ["result"])
y_nondiabetes = pd.DataFrame(nondiabetes, columns = ["result"])

for n in numberN:
    #counter for times of training
    i = 0
    accuracy = 0
    model = LinearRegression()
    #shuffle dataframes for a new train set
    x_diabetes = x_diabetes.sample(frac=1).reset_index(drop=True)
    x_nondiabetes = x_nondiabetes.sample(frac=1).reset_index(drop=True)
    #split train set and test set
    x_diabetes_train = x_diabetes[:n]
    x_nondiabetes_train = x_nondiabetes[:n]
    y_diabetes_train = y_diabetes[:n]
    y_nondiabetes_train = y_nondiabetes[:n]
    x_diabetes_test = x_diabetes[n:]
    x_nondiabetes_test = x_nondiabetes[n:]
    y_diabetes_test = y_diabetes[n:]
    y_nondiabetes_test = y_nondiabetes[n:]
    #combine diabetes train set and nondiabetes train set, do the same for test set
```

```
    x_train = pd.concat([x_diabetes_train, x_nondiabetes_train])
    x_test = pd.concat([x_diabetes_test, x_nondiabetes_test])
    y_train = pd.concat([y_diabetes_train, y_nondiabetes_train])
    y_test = pd.concat([y_diabetes_test, y_nondiabetes_test])

    while i < 1000:
        #train the model
        model.fit(x_train, y_train)
        i += 1

    #test the model
    y_predict = model.predict(x_test)
    correct = 0
    j = 0
    #calculate accuracy
    while j < (number_total - 2*n):
        if y_predict[j] >= 0.5 and j < (number_diabetes - n):
            correct += 1
        if y_predict[j] < 0.5 and j >= (number_diabetes - n):
            correct += 1
        j += 1

    accuracy = round(correct/(number_total-2*n), 4)
    #save accuracy rate for current 2*n data samples
    res.append(accuracy)

print(res)
#draw the plot
X = [80, 160, 240, 320, 400]
Y = res

plt.plot(X, Y, color = 'blue', linestyle = 'solid', linewidth = 2, marker = 'o',
markerfacecolor = 'blue', markersize = 10)

plt.xlabel('Number of Samples')
plt.ylabel('Accuracy')
plt.title('Result')
plt.show()
```