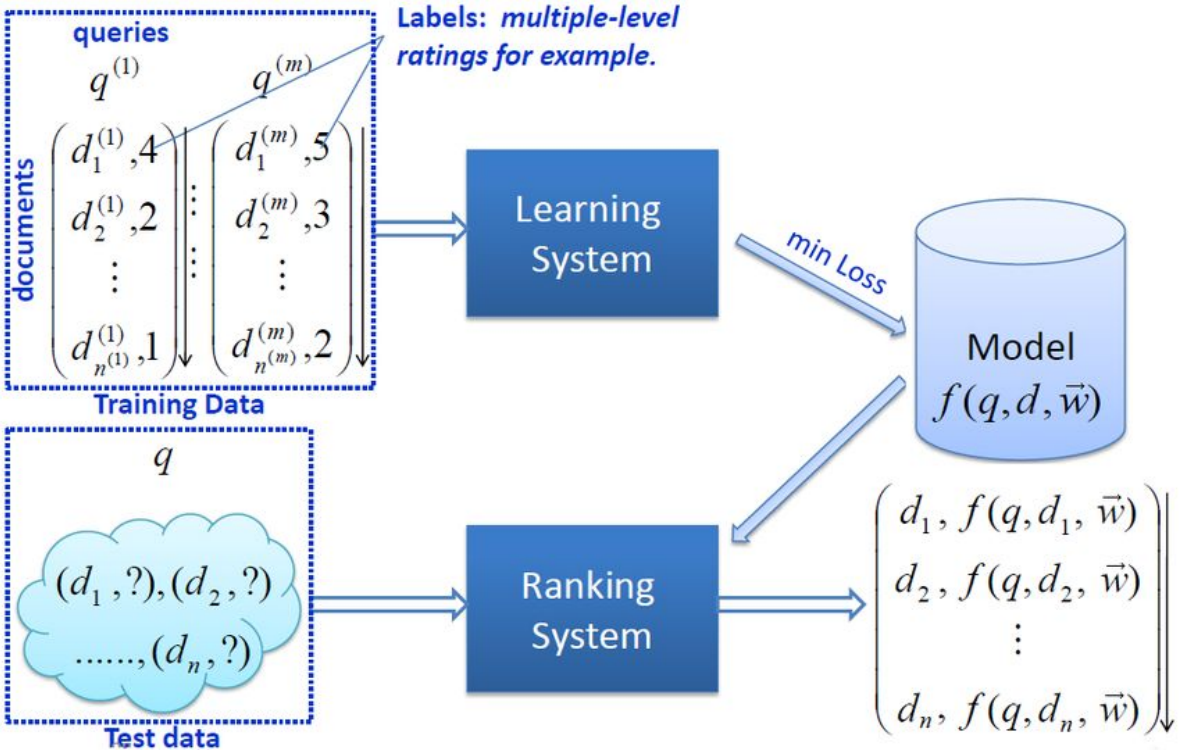
CSE 574 LEC001

Project 2

Handwriting Recognition using Mnist Data



Malini Anbazhagan

Person# 50289383

University at Buffalo

Buffalo, NY 14260

Contents

Introduction

The goal of this project is to use machine learning to identify handwriting match with each other.

There problem is solved using Regression. Linear regression and Logistic regression using gradient descent is used in project 2.

There are two types of feature data given for this project 2. Human observed feature data and GSC feature data. In human observed feature data has only 9 features while GSC feature data has 512 features

Four data set has to be created here. Concatenation of two csv files and subtraction of two csv files for human observed data and GSC data

Code explanation

```
# Import packages needed
import math
import csv
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot
from matplotlib import pyplot as plt

#Read files
df1=pd.read_csv('same_pairs.csv')
df2=pd.read_csv('HumanObserved-Features-Data.csv')
df3=pd.read_csv('diffn_pairs.csv')

#Merge same pair and human observed sata
df4=pd.merge(df1,df2,left_on="img_id_A",right_on="img_id")
df5=pd.merge(df4,df2,left_on="img_id_B",right_on="img_id")

#different pair and human observed data
df7=pd.merge(df3,df2,left_on="img_id_A",right_on="img_id")
df8=pd.merge(df7,df2,left_on="img_id_B",right_on="img_id")

# Append both the data frames to get one dataset, Human Observed Dataset with feature concatentation"
result=df5.append(df8)
```

In the first part, packages are imported. Then the csv files for human observed feature data is read into a data frame. Append operation is done to merge the two data frames horizontally.

```
# drop not needed columns obtained while concatenating the csv's
result.drop('img_id_x',axis=1,inplace=True)
result.drop('img_id_y',axis=1,inplace=True)
result.drop('index_x',axis=1,inplace=True)
result.drop('index_y',axis=1,inplace=True)
result.drop('img_id_A',axis=1,inplace=True)
result.drop('img_id_B',axis=1,inplace=True)
#result.to_csv('concat.csv', sep='\t', encoding='utf-8',header=None,index=False)
```

Unwanted columns are removed in the above code. Now one data set is ready. The following data frame is generated.

	target	f1_x	f2_x	f3_x	f4_x	f5_x	f6_x	f7_x	f8_x	f9_x	f1_y	f2_y	f3_y	f4_y	f5_y	f6_y	f7_y	f8_y	f9_y
0	1	2	1	1	0	2	2	0	2	2	3	2	1	0	2	2	3	0	2
1	1	2	1	1	0	2	2	0	1	2	2	1	0	3	2	2	1	2	2
2	1	2	1	1	0	2	2	0	1	2	1	1	1	1	2	3	0	0	2
3	1	2	1	0	3	2	2	1	2	2	1	1	1	1	2	3	0	0	2
4	1	2	1	1	3	2	2	0	2	2	1	1	1	0	2	2	0	2	2
5	1	2	1	1	3	2	2	0	2	2	2	1	1	0	2	2	0	0	2
6	1	1	1	1	0	2	2	0	2	2	2	1	1	0	2	2	0	0	2
7	1	2	1	1	3	2	2	0	1	2	2	1	1	0	2	2	0	3	2
8	1	2	1	1	3	2	2	0	1	2	1	1	1	0	2	2	0	1	2
9	1	2	1	1	0	2	2	0	3	2	1	1	1	0	2	2	0	1	2
10	1	1	1	1	3	2	2	0	2	2	2	1	0	0	2	2	1	2	2
11	1	2	1	1	3	2	2	0	0	2	3	1	1	0	2	2	0	2	2
12	1	2	1	1	3	2	2	0	0	2	2	1	1	3	2	2	0	3	2
13	1	3	1	1	0	2	2	0	2	2	2	1	1	3	2	2	0	3	2
14	1	1	1	1	0	2	2	0	0	2	2	1	1	0	2	2	1	2	2
15	1	1	1	1	0	2	2	0	0	2	1	1	1	3	2	2	0	1	2
16	1	2	1	1	0	2	2	1	2	2	1	1	1	3	2	2	0	1	2
17	1	1	1	1	0	2	2	0	3	2	3	1	1	0	2	2	0	1	2

Figure

The following code is for normalizing the data and creating hyperparameters for the model

```
# Normalize the data
my_data = (result - result.mean())/result.std()
my_data

#create matrices and hyperparameter
X = my_data.iloc[:,0:18]
ones = np.ones([X.shape[0],1])
X = np.concatenate((ones,X),axis=1)
y = my_data.iloc[:,2:3].values #.values converts it from pandas.core.frame.DataFrame to numpy.ndarray
theta = np.zeros([1,19])
#set hyper parameters
alpha = 0.01
iters = 1000
```

The following code is for calculating gradient descent.

```
#Create cost function
def computeCost(X,y,theta):
    tobesummed = np.power(((X @ theta.T)-y),2)
    return np.sum(tobesummed)/(2 * len(X))

#gradient descent
def gradientDescent(X,y,theta,iters,alpha):
    cost = np.zeros(iters)
    for i in range(iters):
        theta = theta - (alpha/len(X)) * np.sum(X * (X @ theta.T - y), axis=0)
        cost[i] = computeCost(X, y, theta)

    return theta,cost
```

The following code is for calculating cost and plotting the graph.

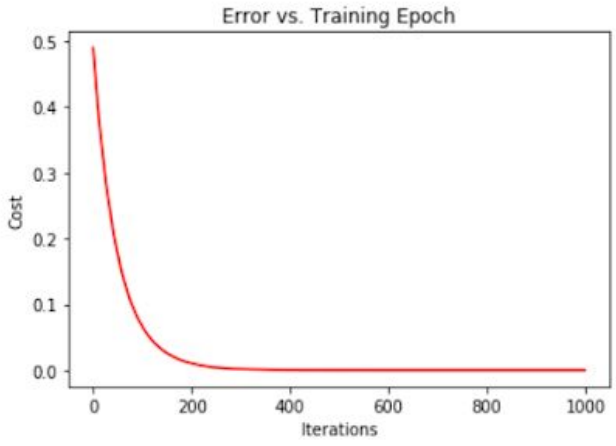
```
#running the gd and cost function
g,cost = gradientDescent(X,y,theta,iters,alpha)
print(g)
finalCost = computeCost(X,y,g)
print(finalCost)

#plot the cost
fig, ax = plt.subplots()
ax.plot(np.arange(iters), cost, 'r')
ax.set_xlabel('Iterations')
ax.set_ylabel('Cost')
ax.set_title('Error vs. Training Epoch')
```

The same code is used for subtraction after merging the dataframes. The columns are subtracted after merging the csv files as follow. And the same code for linear regression is applied.

```
#subtraction
result1=result
result1["f1"]=result["f1_x"]-result["f1_y"]
result1["f2"]=result["f2_x"]-result["f2_y"]
result1["f3"]=result["f3_x"]-result["f3_y"]
result1["f4"]=result["f4_x"]-result["f4_y"]
result1["f5"]=result["f5_x"]-result["f5_y"]
result1["f6"]=result["f6_x"]-result["f6_y"]
result1["f7"]=result["f7_x"]-result["f7_y"]
result1["f8"]=result["f8_x"]-result["f8_y"]
result1["f9"]=result["f9_x"]-result["f9_y"]
```

The graph between epoch VS Training epoch is as follows.



Reference

1. Pandas documentation

   https://pandas.pydata.org/pandas-docs/stable/

2. Tutorials point

   https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm