Machine Learning Report of Pro2

Haorui Dong
UBIT number: 50291149
Department of Computer Science and Engineering
University at Buffalo
haoruido@buffalo.edu

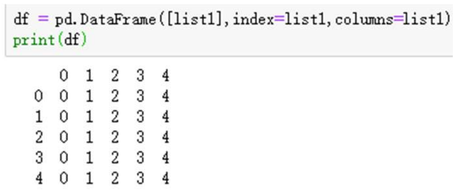
Abstract

In project2, we get 6 .csv files. We use them to do a linear regression to find similarity between the handwritten samples of the known and the questioned writer. It's the first time that I use a lot of pandas.DataFrame. So I do lots of works with these 6 csv files. After that I modify code of linear regression in project1.2 to train my data.

1. Process the origin data.

1.1 introduce the pandas and DataFrame

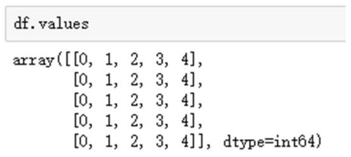
When I want to read data from csv files, I found that there is a method in pandas called pandas.read_csv(file). This function could read the data from csv files and make a DataFrame which is very flexible and easy to use in data processing. When we print a DataFrame, it looks like an excel form.



```
df = pd.DataFrame([list1],index=list1,columns=list1)
print(df)

   0  1  2  3  4
0  0  1  2  3  4
1  0  1  2  3  4
2  0  1  2  3  4
3  0  1  2  3  4
4  0  1  2  3  4
```

Figure1:initial a dataframe

It has a raw index and columns name. We can use df.loc or df.iloc to get each element we want. Or we can easily change the DataFrame to numpy.ndarray by df.values.



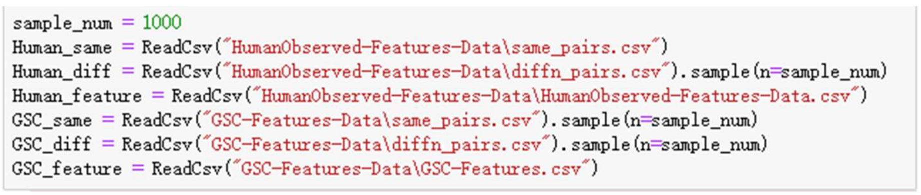
```
df.values

array([[0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4]], dtype=int64)
```

Figure2:change type from dataframe to np.array

Dataframe.iloc(i,j) can get the element which index is i and columns is j.

1.2 processing the csv data

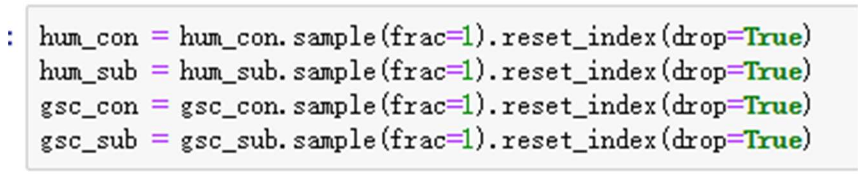
I use read_csv to get 6 dataframes.



```
sample_num = 1000
Human_same = ReadCsv("HumanObserved-Features-Data\same_pairs.csv")
Human_diff = ReadCsv("HumanObserved-Features-Data\diffn_pairs.csv").sample(n=sample_num)
Human_feature = ReadCsv("HumanObserved-Features-Data\HumanObserved-Features-Data.csv")
GSC_same = ReadCsv("GSC-Features-Data\same_pairs.csv").sample(n=sample_num)
GSC_diff = ReadCsv("GSC-Features-Data\diffn_pairs.csv").sample(n=sample_num)
GSC_feature = ReadCsv("GSC-Features-Data\GSC-Features.csv")
```

Figure3: 6 dataframes of origin data

Dataframe.sample could do sampling. N is sample number. That's really convenient.
We will use Human_same and Human_diff pairs img_id to get feature of each id. And make them do concatenation and subtraction. Then we let GSC_same and GSC_diff do the same process. And we set target that same has 1 target and diff has 0 target. Then we can get 4 dataframe. They are:



```
hum_con = hum_con.sample(frac=1).reset_index(drop=True)
hum_sub = hum_sub.sample(frac=1).reset_index(drop=True)
gsc_con = gsc_con.sample(frac=1).reset_index(drop=True)
gsc_sub = gsc_sub.sample(frac=1).reset_index(drop=True)
```

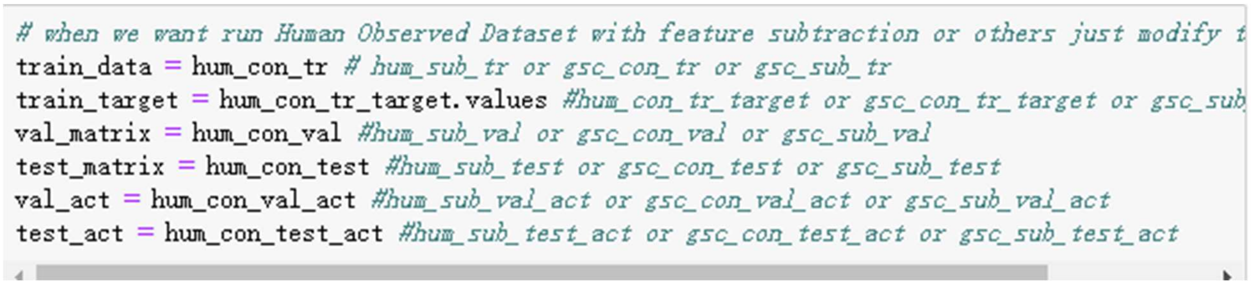
Figure4: sampling

We use df.sample to mess up the order. Frac = 1 means do not modify the size of matrix. And reset_index will make the new index instead of old index.

Then we use slice to get test validation and train matrix. And we get the target array of each matrix. After this we are ready to do linear regression.

2. Linear regression

First, we define many variables at beginning. It will make some changes more convenient.



```
# when we want run Human Observed Dataset with feature subtraction or others just modify t
train_data = hum_con_tr # hum_sub_tr or gsc_con_tr or gsc_sub_tr
train_target = hum_con_tr_target.values #hum_con_tr_target or gsc_con_tr_target or gsc_sub
val_matrix = hum_con_val #hum_sub_val or gsc_con_val or gsc_sub_val
test_matrix = hum_con_test #hum_sub_test or gsc_con_test or gsc_sub_test
val_act = hum_con_val_act #hum_sub_val_act or gsc_con_val_act or gsc_sub_val_act
test_act = hum_con_test_act #hum_sub_test_act or gsc_con_test_act or gsc_sub_test_act
```

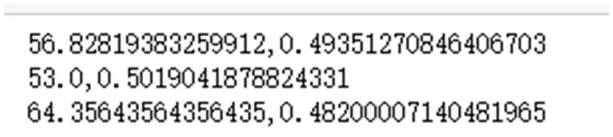
Figure5: modify variables to different groups of training

When we want to change the matrix to do another train. Just change these variables and be sure that change the same group of variables.

Then I use the code in project1.2, and there are lots of problems if we do not modify some code.
I do some transpose of my matrix. And finally, I got my close form result and the gradient descent for linear regression.

3. result

3.1 human concatenation matrix



```
56.82819383259912,0.49351270846406703
53.0,0.5019041878824331
64.35643564356435,0.48200007140481965
```

Figure6: close form result of human concatenation

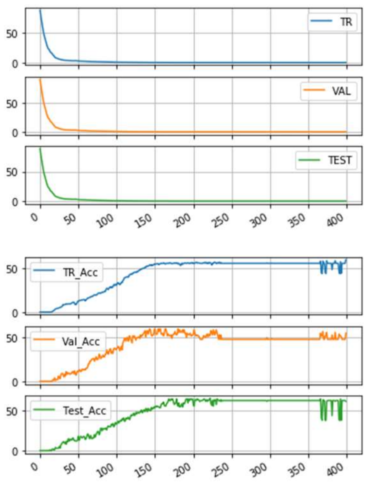
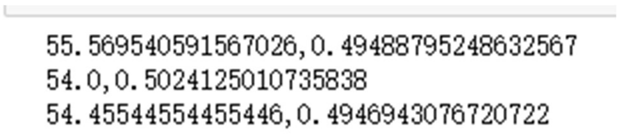


Figure7: gradient descent result

3.2 human subtraction matrix



```
55.569540591567026,0.49488795248632567
54.0,0.5024125010735838
54.45544554455446,0.4946943076720722
```

Figure8: close form result of human concatenation

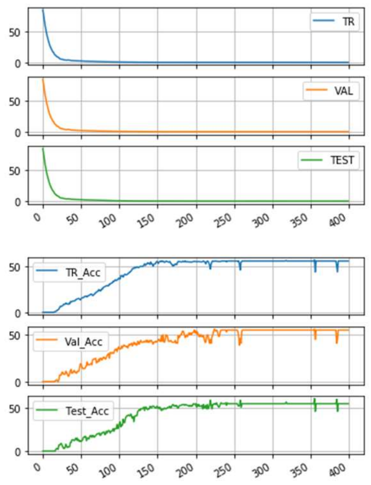
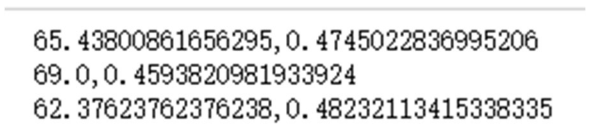


Figure9: gradient descent result

3.3 gsc concatenation matrix



```
65.43800861656295,0.4745022836995206
69.0,0.4593820981933924
62.37623762376238,0.48232113415338335
```

Figure10: close form result of human concatenation

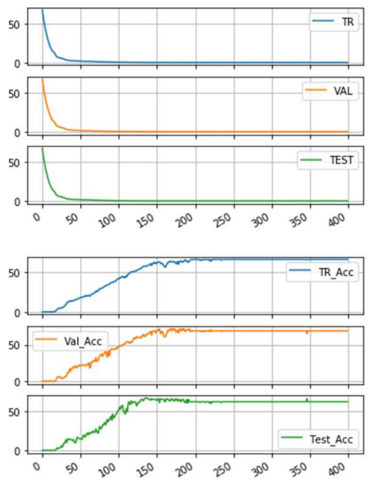


Figure11: close form result of human concatenation

3.4 gsc subtraction matrix

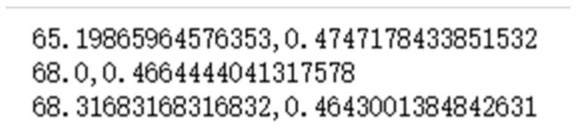
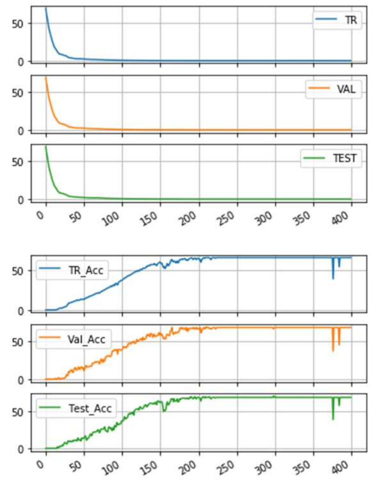

```
65.19865964576353,0.4747178433851532
68.0,0.4664444041317578
68.31683168316832,0.4643001384842631
```

Figure12: close form result of human concatenation



Figur13: close form result of human concatenation