

February 20, 2017

HOMEWORK THREE

Problem 1. Solution

Since the prediction is based on absolute error ($\sum |nHelpful - prediction|$), I choose to use $\text{sum}(nhelpful)/\text{sum}(outOf)$ to get α , the result is:

$$\alpha = 0.85040619252$$

Listing 1: Key code for Prob.1

```
1 data_train=data[:100000]
2 data_validation=data[100000:]
3 nHelpful=[t['helpful']['nHelpful'] for t in data_train]
4 outOf=[t['helpful']['outOf'] for t in data_train]
5 total_nHelpful=sum(nHelpful)
6 total_outOf=sum(outOf)
7 alpha=total_nHelpful*1.0/total_outOf
8 print 'alpha=',alpha
```

Problem 2. Solution

The performance of this trivial predictor on the validation set in terms of MAE is:

$$MAE = 0.21605420072$$

Listing 2: Key code for Prob.2

```
1 validation_outOf=[t['helpful']['outOf'] for t in data_validation]
2 validation_nHelpful=[t['helpful']['nHelpful'] for t in data_validation]
3 diff=[validation_outOf[i]*alpha-validation_nHelpful[i] for i in range(len(validation_outOf))]
4 error=[abs(i) for i in diff]
5 MAE=sum(error)/len(validation_outOf)
6 print 'MAE=',MAE
```

Problem 3. Solution

In the train dataset, we just ignore all the elements whose 'outOf' equals to 0 because it provides no information for us to predict the helpful rate. The fitted parameters and the MAE are shown as below:

$$\alpha = 5.62218966e - 01, \beta_1 = 2.11835412e - 04, \beta_2 = 5.07029148e - 02$$

$$MAE = 0.240245808704$$

Listing 3: Key code for Prob.3

```

1 data_train=data[:100000]
2 data_validation=data[100000:]
3 data_train_valid=[datum for datum in data_train if datum['helpful']['outOf']!=0]
4
5 def feature(datum):
6     feat=[1]
7     word=datum['reviewText'].split()
8     word_count=len(word)
9     feat.append(word_count)
10    feat.append(datum['rating'])
11    return feat
12
13 X_train=[feature(d) for d in data_train_valid]
14 y_train=[datum['helpful']['nHelpful']*1.0/datum['helpful']['outOf'] for datum in
15             data_train_valid]
16 ##fit a lst
17 theta,residuals,rank,s = np.linalg.lstsq(X_train,y_train)
18 print 'theta=',theta
19
20 X_validation=[feature(d) for d in data_validation]
21 theta=np.matrix(theta)
22 X_validation=np.matrix(X_validation)
23 rate_predict=X_validation*theta.T
24
25 outOf_validation=[datum['helpful']['outOf'] for datum in data_validation]
26 outOf_validation=np.matrix(outOf_validation)
27 y_predict=np.multiply(outOf_validation.T,rate_predict)
28
29 y_validation=[datum['helpful']['nHelpful'] for datum in data_validation]
30 y_predict=y_predict.T.tolist()[0]
31 diff=[y_validation[i]-y_predict[i] for i in range(len(y_predict))]
32 error=[abs(i) for i in diff]
33 MAE=sum(error)/len(y_predict)
34 print 'MAE=',MAE

```

Problem 4. Solution

My Kaggle username: YaphetS

Listing 4: Key code for Prob.4

```

1 data_train=data[:100000]
2 data_validation=data[100000:]
3
4 data_train_valid=[datum for datum in data_train if datum['helpful']['outOf']!=0]
5
6 def feature(datum):

```

```

7     feat=[1]
8     word=datum['reviewText'].split()
9     word_count=len(word)
10    feat.append(word_count)
11    feat.append(datum['rating'])
12    return feat
13
14 X_train=[feature(d) for d in data_train_valid]
15 y_train=[datum['helpful']['nHelpful']*1.0/datum['helpful']['outOf'] for datum in
16           data_train_valid]
17 ##fit a lst
18 theta,residuals,rank,s = np.linalg.lstsq(X_train,y_train)
19
20 data_test = [l for l in readGz("test_Helpful.json.gz")]
21 X_test=[feature(d) for d in data_test]
22 theta=np.matrix(theta)
23 X_test=np.matrix(X_test)
24 test_predict=X_test*theta.T
25
26 outOf_test=[datum['helpful']['outOf'] for datum in data_test]
27 outOf_test=np.matrix(outOf_test)
28 y_predict=np.multiply(outOf_test.T,test_predict)
29 y_predict=y_predict.T.tolist()[0]
30
31 predictions = open("predictions_Helpful.txt", 'w')
32 cur=0
33 for l in open("pairs_Helpful.txt"):
34     if l.startswith("userID"):
35         #header
36         predictions.write(l)
37         continue
38     u,i,prediction = l.strip().split('-')
39     predictions.write(u + '-' + i + '-' + prediction + ',' + str(y_predict[cur]) + '\n')
40     cur=cur+1
41 predictions.close()

```

Problem 5. Solution

$$\alpha = 4.23198, \text{MSE} = 1.2264713284$$

Listing 5: Key code for Prob.5

```

1 data_train=data[:100000]
2 data_validation=data[100000:]
3
4 rating_train=[datum['rating'] for datum in data_train]
5 alpha=sum(rating_train)/len(data_train)
6 print "alpha=",alpha
7
8 rating_validation=[datum['rating'] for datum in data_validation]
9 error=[(t-alpha)**2 for t in rating_validation]
10 MSE=sum(error)/len(rating_validation)
11 print "MSE=",MSE

```

Problem 6. Solution

MSE on the validation set is (The convergence condition is the sum of the absolute difference among last 5 MSE on validation set is smaller than 10^{-6}):

$$MSE = 1.28154804347$$

Listing 6: Key code for Prob.6

```

1 allRatings = []
2 userRating = defaultdict(list)
3 itemRating = defaultdict(list)
4 for l in data_train:
5     user, item = l['reviewerID'], l['itemID']
6     allRatings.append(l['rating'])
7     userRating[user].append(l['rating'])
8     itemRating[item].append(l['rating'])
9 alpha = sum(allRatings) / len(allRatings)
10 userAverage = {}
11 itemAverage = {}
12 lam=1
13 for u in userRating:
14     userAverage[u] = sum(userRating[u]) / len(userRating[u])
15 for i in itemRating:
16     itemAverage[i] = sum(itemRating[i]) / len(itemRating[i])
17 k=0
18 mse1,mse2,mse3,mse4,mse5 = 0,1000,0,1000,0
19 while(abs(mse2-mse1)+abs(mse3-mse2)+abs(mse4-mse3)+abs(mse5-mse4)>0.00001):
20     s1=[datum['rating']-userAverage[datum['reviewerID']]-itemAverage[datum['itemID']]]
21         for datum in data_train]
22     alpha=sum(s1)/len(data_train)
23     userRating = defaultdict(list)
24     for l in data_train:
25         userRating[l['reviewerID']].append(l['rating']-alpha-itemAverage[l['itemID']])
26     userAverage[u] = sum(userRating[u]) / (len(userRating[u])+lam)
27     itemRating = defaultdict(list)
28     for l in data_train:
29         itemRating[l['itemID']].append(l['rating']-alpha-userAverage[l['reviewerID']])
30     for u in itemRating:
31         itemAverage[u] = sum(itemRating[u]) / (len(itemRating[u])+lam)
32     k = k + 1
33
34     rating_validation = []
35     for datum in data_validation:
36         predict = alpha
37         if(datum['itemID'] in itemAverage):
38             predict = predict + itemAverage[datum['itemID']]
39         if(datum['reviewerID'] in userAverage):
40             predict = predict + userAverage[datum['reviewerID']]
41         rating_validation.append(predict)
42     error = [(rating_validation[i]-data_validation[i]['rating'])**2 for i in range(len(
43         data_validation))]
44     mse=sum(error)/len(data_validation)
# print k, " MSE= ", mse

```

```

45     mse1, mse2, mse3, mse4, mse5 = mse, mse1, mse2, mse3, mse4
46 print "Final MSE= ",mse

```

Problem 7. Solution

The user and item IDs that have the largest and smallest values of β are shown as below:

Table 1: User and item with the largest and smallest β

Name	ID	β
Item with largest β	I558325415	1.2462415281242187
Item with smallest β	I071368828	-2.373051051351595
User with largest β	U816486110	1.5137627968220644
User with smallest β	U052814411	-2.5125477057820653

Listing 7: Key code for Prob.7

```

1 item_high=sorted(itemAverage.items(),key=lambda item:item[1],reverse=True)[0]
2 item_low=sorted(itemAverage.items(),key=lambda item:item[1])[0]
3 user_high=sorted(userAverage.items(),key=lambda item:item[1],reverse=True)[0]
4 user_low=sorted(userAverage.items(),key=lambda item:item[1])[0]

```

Problem 8. Solution

The λ I choose is 7, the MSE on validation set when $\lambda = 7$ is 1.1396028025.

Listing 8: Key code for Prob.8

```

1 predictions = open("output_rating.txt", 'w')
2 for l in open("pairs_Rating.txt"):
3     if l.startswith("userID"):
4         #header
5         predictions.write(l)
6         continue
7     u,i = l.strip().split(' - ')
8     predict = alpha
9     if (i in itemAverage):
10        predict = predict + itemAverage[i]
11    if (u in userAverage):
12        predict = predict + userAverage[u]
13    predictions.write(u + ' - ' + i + ' , ' + str(predict) + '\n')
14
15 predictions.close()

```

Submitted by Zhang, Jinhan PID A53211930 on February 20, 2017.