Zhang, Jinhan PID A53211930
University of California, San Diego

February 20, 2017

HOMEWORK THREE

Problem 1. Solution

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

$$\alpha = 0.85040619252$$

Listing 1: Key code for Prob.1

```
data_train=data[:100000]
data_validation=data[100000:]
nHelpful=[t['helpful']['nHelpful'] for t in data_train]
outOf=[t['helpful']['outOf'] for t in data_train]
total_nHelpful=sum(nHelpful)
total_outOf=sum(outOf)
alpha=total_nHelpful*1.0/total_outOf
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

```
validation_outOf=[t['helpful']['outOf'] for t in data_validation]
validation_nHelpful=[t['helpful']['nHelpful'] for t in data_validation]
diff=[validation_outOf[i]*alpha-validation_nHelpful[i] for i in range(len(
                                                validation_outOf))]
error=[abs(i) for i in diff]
MAE=sum(error)/len(validation_outOf)
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

```
data_train=data[:100000]
data_validation=data[100000:]
data_train_valid=[datum for datum in data_train if datum['helpful']['outOf']!=0]

def feature(datum):
    feat=[1]
    word=datum['reviewText'].split()
    word_count=len(word)
    feat.append(word_count)
    feat.append(datum['rating'])
    return feat

X_train=[feature(d) for d in data_train_valid]
y_train=[datum['helpful']['nHelpful']*1.0/datum['helpful']['outOf'] for datum in data_train_valid]
##fit a lst
theta,residuals,rank,s = np.linalg.lstsq(X_train,y_train)
print 'theta=',theta

X_validation=[feature(d) for d in data_validation]
theta=np.matrix(theta)
X_validation=np.matrix(X_validation)
rate_predict=X_validation*theta.T

outOf_validation=[datum['helpful']['outOf'] for datum in data_validation]
outOf_validation=np.matrix(outOf_validation)
y_predict=np.multiply(outOf_validation.T,rate_predict)

y_validation=[datum['helpful']['nHelpful'] for datum in data_validation]
y_predict=y_predict.T.tolist()[0]

diff=[y_validation[i]-y_predict[i] for i in range(len(y_predict))]
error=[abs(i) for i in diff]
MAE=sum(error)/len(y_predict)
print 'MAE=',MAE
```

Problem 4. Solution

My Kaggle username: YaphetS

Listing 4: Key code for Prob.4

```
data_train=data[:100000]
data_validation=data[100000:]

data_train_valid=[datum for datum in data_train if datum['helpful']['outOf']!=0]

def feature(datum):
    feat=[1]
    word=datum[’reviewText’].split()
    word_count=len(word)
    feat.append(word_count)
    feat.append(datum[’rating’])
    return feat

X_train=[feature(d) for d in data_train_valid]
y_train=[datum[’helpful’][’nHelpful’]*1.0/datum[’helpful’][’outOf’] for datum in data_train_valid]
##fit a lst
theta,residuals,rank,s = np.linalg.lstsq(X_train,y_train)

data_test = [l for l in readGz("test_Helpful.json.gz")]
X_test=[feature(d) for d in data_test]
theta=np.matrix(theta)
X_test=np.matrix(X_test)
test_predict=X_test*theta.T

outOf_test=[datum[’helpful’][’outOf’] for datum in data_test]
outOf_test=np.matrix(outOf_test)
y_predict=np.multiply(outOf_test.T,test_predict)
y_predict=y_predict.T.tolist()[0]

predictions = open("predictions_Helpful.txt", ’w’)
cur=0
for l in open("pairs_Helpful.txt"):
  if l.startswith("userID"):
    #header
    predictions.write(l)
    continue
  u,i,prediction = l.strip().split(’-’)
  predictions.write(u + ’-’ + i + ’-’ + prediction + ’,’ + str(y_predict[cur]) + ’\n’)
  cur=cur+1
predictions.close()
```

Problem 5. Solution

$$\alpha = 4.23198, \; MSE = 1.2264713284$$

Listing 5: Key code for Prob.5

```
data_train=data[:100000]
data_validation=data[100000:]

rating_train=[datum[’rating’] for datum in data_train]
alpha=sum(rating_train)/len(data_train)
print "alpha=",alpha

rating_validation=[datum[’rating’] for datum in data_validation]
error=[(t-alpha)**2 for t in rating_validation]
MSE=sum(error)/len(rating_validation)
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

```
allRatings = []
userRating = defaultdict(list)
itemRating = defaultdict(list)
for l in data_train:
    user,item = l[’reviewerID’],l[’itemID’]
    allRatings.append(l[’rating’])
    userRating[user].append(l[’rating’])
    itemRating[item].append(l[’rating’])
alpha = sum(allRatings) / len(allRatings)
userAverage = {}
itemAverage = {}
lam=1
for u in userRating:
    userAverage[u] = sum(userRating[u]) / len(userRating[u])
for i in itemRating:
    itemAverage[i] = sum(itemRating[i]) / len(itemRating[i])
k=0
mse1,mse2,mse3,mse4,mse5 = 0,1000,0,1000,0
while(abs(mse2-mse1)+abs(mse3-mse2)+abs(mse4-mse3)+abs(mse5-mse4)>0.00001):
    s1=[datum[’rating’]-userAverage[datum[’reviewerID’]]-itemAverage[datum[’itemID’]]
                                              for datum in data_train]
    alpha=sum(s1)/len(data_train)
    userRating = defaultdict(list)
    for l in data_train:
        userRating[l[’reviewerID’]].append(l[’rating’]-alpha-itemAverage[l[’itemID’]])
    for u in userRating:
        userAverage[u] = sum(userRating[u]) / (len(userRating[u])+lam)
    itemRating = defaultdict(list)
    for l in data_train:
        itemRating[l[’itemID’]].append(l[’rating’]-alpha-userAverage[l[’reviewerID’]])
    for u in itemRating:
        itemAverage[u] = sum(itemRating[u]) / (len(itemRating[u])+lam)
    k = k + 1

    rating_validation = []
    for datum in data_validation:
        predict = alpha
        if(datum[’itemID’] in itemAverage):
            predict = predict + itemAverage[datum[’itemID’]]
        if(datum[’reviewerID’] in userAverage):
            predict = predict + userAverage[datum[’reviewerID’]]
        rating_validation.append(predict)
    error = [(rating_validation[i]-data_validation[i][’rating’])**2 for i in range(len(
                                              data_validation))]
    mse=sum(error)/len(data_validation)
    #print k," MSE= ",mse
    mse1, mse2, mse3, mse4, mse5 = mse, mse1, mse2, mse3, mse4
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

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

```
predictions = open("output_rating.txt", 'w')
for l in open("pairs_Rating.txt"):
  if l.startswith("userID"):
    #header
    predictions.write(l)
    continue
  u,i = l.strip().split('-')
  predict = alpha
  if (i in itemAverage):
      predict = predict + itemAverage[i]
  if (u in userAverage):
      predict = predict + userAverage[u]
  predictions.write(u + '-' + i + ',' + str(predict) + '\n')

predictions.close()
```

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