 # Homework 3

Wenjun Zhang

A53218995

Problems:

1. Fitting the $\frac{\text{nHelpful}}{\text{outOf}}$, where nHelpful is the sum of all 'nHelpful' in the training set and outOf is the sum of all 'outOf' in training set. We define:

$$\frac{\text{nHelpful}}{\text{outOf}} \simeq \alpha$$

   The value of α is 0.85040619252

2. The performance of this trivial predictor in terms of mean absolute error (MAE) is:

```
MAE =  0.214373893977
```

3. To train a predictor of the form:

$$\frac{\text{nHelpful}}{\text{outOf}} \simeq \alpha + \beta_1(\#\text{ words in review}) + \beta_2(\text{review's rating in stars})$$

   The fitted parameters an the MAE on the validation set are:

```
alpha =  0.562218966266  beta1 =  0.000211835412408 beta2 =  0.050702914784
MAE =  [[ 0.24024581]]
```

4. Running the model on the test set, the submission score in Kaggle is 0.23919. (My user name is WenjunZhang)

5. The value of α and the performance of the trivial predictor

$$\text{rating(user, item)} = \alpha$$

   on the validation set is:

```
alpha = 4.23198
The performance of MSE =  1.2264713284
```

6. Fit a predictor of the form

$$\text{rating(user, item)} \simeq \alpha + \beta_{\text{user}} + \beta_{\text{item}}$$

   Use a regularization parameter of λ = 1, the MSE on the validation set is:

```
When lamda= 1 MSE= 1.28154916045
```

7. The user and item IDs that have the largest and smallest values of β are:

```
user who has the smallest beta is  U052814411
user who has the largest beta is  U816486110
item which has the smallest beta is  I071368828
item which has the largest beta is  I558325415
```

8. We try λ from 0 to 9, and find the minimum MSE on the validation set. The results are as below:

```
When lamda =  0 MSE =  1.99256194458
When lamda =  1 MSE =  1.28154916045
When lamda =  2 MSE =  1.19022331203
When lamda =  3 MSE =  1.15936726738
When lamda =  4 MSE =  1.14669784246
When lamda =  5 MSE =  1.14141577111
When lamda =  6 MSE =  1.13961701463
When lamda =  7 MSE =  1.13960376779
When lamda =  8 MSE =  1.14054718532
When lamda =  9 MSE =  1.14200977074
```

We can see that when λ = 7, we have the minmum MSE. So we choose λ = 7, the MSE on the validation set is 1.13960376779. Then we upload the solution to Kaggle. The score is 1.19218.

Appendix (Some important parts)

Q1:

```
data = list(readGz("train.json.gz"))

train_data = data[:len(data)/2]

validation_data = data[len(data)/2:]
```

```
### Question 1

allHelpful = []userHelpful = defaultdict(list)

for l in train_data:

  allHelpful.append(l['helpful'])alpha = sum([x['nHelpful'] for x in allHelpful]) * 1.0 / sum([x['outOf'] for

x in allHelpful])

print alpha
```

Q2:

```
### Question 2

allHelpful = []

for l in validation_data:

  allHelpful.append(l['helpful'])

averageRate = sum([x['nHelpful'] for x in allHelpful]) * 1.0 / sum([x['outOf'] for x in allHelpful])

mae_temp = 0for u in range(len(allHelpful)):

    mae_temp = mae_temp + abs(allHelpful[u]['nHelpful'] - averageRate*allHelpful[u]['outOf'])maeresult =

mae_temp / len(allHelpful)

print "MAE = ", maeresult
```

Q3:

```
### Question 3

for u in train_data:

    if u['helpful']['outOf'] > 0:

        trainHelpful.append(u['helpful']['nHelpful']*1.0/u['helpful']['outOf'])

        trainWords.append(len(u['reviewText'].split()))

        trainRatings.append(u['rating'])
```

```
def feature(words, ratings):
    feat = [[1] for row in range(len(words))]
    feat = numpy.column_stack((feat, words, ratings))
    return feat
def maecomp(theta, X, y, outof):
  theta = numpy.matrix(theta).T
  X = numpy.matrix(X)
  mae_temp = 0
  for i in range(len(X)):
    mae_temp = mae_temp + abs(X[i]*theta*outof[i] - y[i])
    maeresult = mae_temp / len(y)
  return maeresult
```

```
for u in validation_data:
    if u['helpful']['outOf'] > 0:
        validationHelpful.append(u['helpful']['nHelpful']*1.0/u['helpful']['outOf'])
        validationWords.append(len(u['reviewText'].split()))
        validationRatings.append(u['rating'])
        X_train = feature(trainWords, trainRatings)X_validation = feature(validationWords, validationRatin
gs)
theta,residuals,rank,s = numpy.linalg.lstsq(X_train, trainHelpful)
print "alpha = ", theta[0], " beta1 = ", theta[1], "beta2 = ", theta[2]
```

```
for u in validation_data:
    validationHelpful.append(u['helpful']['nHelpful'])
    validationOutof.append(u['helpful']['outOf'])
    validationWords.append(len(u['reviewText'].split()))
    validationRatings.append(u['rating'])
X_validation = feature(validationWords, validationRatings)maeresult = maecomp(theta, X_validation, validati
onHelpful, validationOutof)
print "MAE = ", maeresult
```

Q4:

```
### Question 4
for l in readGz("test_Helpful.json.gz"):
    user,item = l['reviewerID'],l['itemID']
    userReviews[user + item].append(l['reviewText'])
    wordCount = defaultdict(int)
    reviewWords[user + item] = (len(l['reviewText'].split()))
    helpfulRatings[user + item] = (l['rating'])
userRate = defaultdict(list)for u in reviewWords:
        X_feature = [1.0, reviewWords[u], helpfulRatings[u]]
        userRate[u] = sum(theta[i] * X_feature[i] for i in range(len(theta)))
```

```
predictions = open("predictions_Helpful.txt", 'w')for l in open("pairs_Helpful.txt"):
    if l.startswith("userID"):
        #header
        predictions.write(l)
        continue
    u,i,outOf = l.strip().split('-')
    outOf = int(outOf)
    predictions.write(u + '-' + i + '-' + str(outOf) + ',' + str(outOf * userRate[u + i]) + '\n')
predictions.close()
```

Q5:

```
### Question 5
for l in train_data:
        train_Ratings.append(l['rating'])
alpha = sum(train_Ratings) / len(train_Ratings)
for l in validation_data:
        allRatings.append(l['rating'])
mse_temp = 0
for i in range(len(allRatings)):
        mse_temp = mse_temp + abs(allRatings[i] - alpha)**2
mseresult = mse_temp / len(allRatings)
```

```
print "alpha =", alpha
print "The performance of MSE = ", mseresult
```

Q6:

```
### Question 6
def iterative(lamda):
    alpha = 0
    for iteration in range(1000):
        original_alpha=alpha
        total = 0
        for u in userRatings:
            for i in userRatings[u]:
                temp = userRatings[u][i] - (beta_u[u] + beta_i[i])
                total += temp
        alpha = total/100000

        diff = abs(original_alpha-alpha)
        if diff < 1e-100:
            break

        for u in userRatings:
            total = 0
            for i in userRatings[u]:
                total += userRatings[u][i] - alpha - beta_i[i]
            beta_u[u] = total / (lamda + len(userRatings[u].keys()))

        for i in itemRatings:
            total = 0
            for u in itemRatings[i]:
                total += itemRatings[i][u] - alpha - beta_u[u]
            beta_i[i] = total/(lamda+len(itemRatings[i].keys()))

    total = 0
```

```
    for l in validation_data:
        total += (l['rating'] - (alpha + beta_u[l['reviewerID']] + beta_i[l['itemID']])) ** 2
    mse = total / 100000
    return alpha, beta_u, beta_i, mse
```

```
lamda = 1
alpha, beta_u, beta_i, mse = iterative(lamda)
print"When lamda=", lamda, "MSE=", mse
```

Q7:

```
### Question 7
print"user who has the smallest beta is ", min(beta_u, key=beta_u.get)
print"user who has the largest beta is ", max(beta_u, key=beta_u.get)
print"item which has the smallest beta is ", min(beta_i, key=beta_i.get)
print"item which has the largest beta is ", max(beta_i, key=beta_i.get)
```

Q8:

```
### Question 8
for i in range(10):
    alpha, beta_u, beta_i, mse = iterative(i)
    print "When lamda = ", i, "MSE = ", mse
```

```
alpha, beta_u, beta_i, mse = iterative(7)
predictions = open("predictions_Rating.txt", 'w')for l in open("pairs_Rating.txt"):
  if l.startswith("userID"):
    #header
    predictions.write(l)
    continue
  u,i = l.strip().split('-')
  predictions.write(u + '-' + i + ',' + str(alpha + beta_u[u] + beta_i[i]) + '\n')
predictions.close()
```