Homework 3
CSE 258
Spring 2017

Xiaowen Mao
PID: A53220159
x9mao@eng.ucsd.edu

1. Trivial Predictor

$$\alpha = 0.783737364232$$

```
import gzip
import numpy
from collections import defaultdict

def readGz(f):
    for l in gzip.open(f):
        yield eval(l)
```

```
Data = []
for l in readGz("assignment1/train.json.gz"):
    Data.append(l)
```

```
data_train = Data[:100000]
data_valid = Data[100000:]
```

```
# task 1
alpha = 0
count = 0
for d in data_train:
    if d['helpful']['outOf'] != 0:
        alpha += d['helpful']['nHelpful'] * 1.0 / d['helpful']['outOf']
        count += 1

alpha /= count
print alpha
```

```
0.783737364232
```

Figure 1: Code for Task 1

2. Performance of Trivial Predictor

$$MAE = 0.258702157422$$

```
# task 2
MAE = 0
for d in data_valid:
    predict = alpha * d['helpful']['outOf']
    MAE += abs(predict - d['helpful']['nHelpful'])

MAE /= len(data_valid)
print MAE
```

```
0.258702157422
```

Figure 2: Code for Task 2

3. Non-trivial Predictor

$$\theta = [5.62218966 \times 10^{-1}, 2.11835412 \times 10^{-4}, 5.07029148 \times 10^{-2}]$$

$$MAE = 0.240245808704$$

```
# task 3
def filter(data):
    feature= []
    label = []
    for d in data:
        if d['helpful']['outOf'] != 0:
            # feature
            feat = [1]
            feat.append(len(d['reviewText'].strip().split()))
            feat.append(d['rating'])
            feature.append(feat)
            # label
            l = d['helpful']['nHelpful'] * 1.0 / d['helpful']['outOf']
            label.append(l)
    return feature, label
```

```
feature_train, label_train = filter(data_train)
theta,residuals,rank,s = numpy.linalg.lstsq(feature_train, label_train)
```

```
print theta
```

```
[  5.62218966e-01   2.11835412e-04   5.07029148e-02]
```

Figure 3: Code for Theta in Task 3

```
# task 3 MAE
def inner(x, y):
    result = 0
    for a, b in zip(x, y):
        result += a * b
    return result

feature_valid, label_valid = filter(data_valid)
helpful_valid = []
for d in data_valid:
    if d['helpful']['outOf'] != 0:
        helpful_valid.append(d['helpful'])

MAE = 0
for i in range(len(helpful_valid)):
    predict = inner(theta, feature_valid[i]) * helpful_valid[i]['outOf']
    MAE += abs(helpful_valid[i]['nHelpful'] - predict)
MAE /= len(data_valid)
print MAE
```

```
0.240245808704
```

Figure 4: Code for MAE in Task 3

4. To Kaggle

My user name is **XiaowenMao**. Link.
The code are attached in Figure 5.

```
# task 4
reviews = {}
for l in readGz("assignment1/test_Helpful.json.gz"):
    user,item = l['reviewerID'], l['itemID']
    key = user + ' ' + item
    reviews[key] = {}
    reviews[key]["rating"] = l["rating"]
    reviews[key]["outOf"] = l["helpful"]["outOf"]
    reviews[key]["length"] = len(l['reviewText'].strip().split())
```

```
predictions = open("assignment1/predictions_Helpful.txt", 'w')

for l in open("assignment1/pairs_Helpful.txt"):
    if l.startswith("userID"):
        # first line
        predictions.write(l)
        continue

    u, i, outOf = l.strip().split('-')
    outOf = int(outOf)
    key = u + ' ' + i
    feature = [1, reviews[key]["length"], reviews[key]["rating"]]
    predictions.write(u + '-' + i + '-' + str(outOf) + ',' \
                      + str(outOf*(inner(theta , feature)))+ '\n')

predictions.close ()
```

Figure 5: Code for Task 4

5. Trivial Predictor

$$\alpha = 4.23198$$

$$MSE = 1.2264713284$$

```
# task 5
ratings_train = []
ratings_valid = []

for d in data_train:
    ratings_train.append(d['rating'])
for d in data_valid:
    ratings_valid.append(d['rating'])

alpha = sum(ratings_train) * 1.0 / len(ratings_train)
print alpha

MSE = 0
for i in range(len(data_valid)):
    MSE += (ratings_valid[i]-alpha) **2
MSE /= len(data_valid)
print MSE
```

```
4.23198
1.2264713284
```

Figure 6: Code for Task 5

6. Non-trivial Predictor

$$MSE = 1.28154916046$$

```
# task 6
userRatings_train = defaultdict(lambda: defaultdict(int))
itemRatings_train = defaultdict(lambda: defaultdict(int))
beta_u = defaultdict(float)
beta_i = defaultdict(float)

for d in data_train:
    user, item = d['reviewerID'], d['itemID']
    userRatings_train[user][item] = d['rating']
    itemRatings_train[item][user] = d['rating']
    beta_u[user] = 0
    beta_i[item] = 0
```

Figure 7: Code for task 6

```
lam = 1
i = 0
iteration = 100
while i < iteration:
    # update alpha
    alpha = 0
    for user in userRatings_train:
        for item in userRatings_train[user]:
            alpha += userRatings_train[user][item] \
                    - (beta_u[user] + beta_i[item])
    alpha /= len(data_train)
    # update beta_u
    for user in userRatings_train:
        beta_u[user] = 0
        for item in userRatings_train[user]:
            beta_u[user] += userRatings_train[user][item] \
                    - (alpha + beta_i[item])
        beta_u[user] /= (lam + len(userRatings_train[user]))
    # update beta_i
    for item in itemRatings_train:
        beta_i[item] = 0
        for user in itemRatings_train[item]:
            beta_i[item] += itemRatings_train[item][user] \
                    - (alpha + beta_u[user])
        beta_i[item] /= (lam + len(itemRatings_train[item]))
    # MSE
    MSE = 0
    for user in userRatings_train:
        for item in userRatings_train[user]:
            MSE += (alpha + beta_u[user] + beta_i[item]
                    - userRatings_train[user][item]) **2
    MSE /= len(data_train)
    if(i == 99):
        print MSE
    i += 1
```

```
0.510467637051
```

Figure 8: Code for task 6

```
print alpha
```

```
4.22632677668
```

```
userRatings_valid = defaultdict(lambda: defaultdict(int))
itemRatings_valid = defaultdict(lambda: defaultdict(int))

for d in data_valid:
    user, item = d['reviewerID'], d['itemID']
    userRatings_valid[user][item] = d['rating']
    itemRatings_valid[item][user] = d['rating']

MSE = 0
for user in userRatings_valid:
    for item in userRatings_valid[user]:
        MSE += ((alpha \
                 + (beta_u[user] if user in beta_u else 0) \
                 + (beta_i[item] if item in beta_i else 0) \
                 - userRatings_valid[user][item]) **2)
MSE /= len(data_train)
print MSE
```

```
1.28154916046
```

Figure 9: Result for task 6

7. Max/Min User/Item

user with largest β: U816486110

user with smallest β: U052814411

item with largest β: I558325415

item with smallest β: I071368828

```
# task 7
import sys

max_value = -sys.maxint
min_value = sys.maxint
for key, value in beta_u.iteritems():
    if value > max_value:
        max_value = value
        max_user = key

    if value < min_value:
        min_value = value
        min_user = key
print "user with largest beta: ", max_user
print "user with smallest beta: ", min_user

max_value = -sys.maxint
min_value = sys.maxint
for key, value in beta_i.iteritems():
    if value > max_value:
        max_value = value
        max_item = key

    if value < min_value:
        min_value = value
        min_item = key
print "item with largest beta: ", max_item
print "item with smallest beta: ", min_item
```

```
user with largest beta:  U816486110
user with smallest beta:  U052814411
item with largest beta:  I558325415
item with smallest beta:  I071368828
```

Figure 10: Code and result for task 7

8. To Kaggle

$$\lambda = 6.5$$

$$MSE = 1.13945170902$$

My user name is **XiaowenMao**. Link.

```
# task 8
lam = 6.5
i = 0
iteration = 100
while i < iteration:
    # update alpha
    alpha = 0
    for user in userRatings_train:
        for item in userRatings_train[user]:
            alpha += userRatings_train[user][item] \
                     - (beta_u[user] + beta_i[item])
    alpha /= len(data_train)
    # update beta_u
    for user in userRatings_train:
        beta_u[user] = 0
        for item in userRatings_train[user]:
            beta_u[user] += userRatings_train[user][item] \
                            - (alpha + beta_i[item])
        beta_u[user] /= (lam + len(userRatings_train[user]))
    # update beta_i
    for item in itemRatings_train:
        beta_i[item] = 0
        for user in itemRatings_train[item]:
            beta_i[item] += itemRatings_train[item][user] \
                            - (alpha + beta_u[user])
        beta_i[item] /= (lam + len(itemRatings_train[item]))
    # MSE
    MSE = 0
    for user in userRatings_train:
        for item in userRatings_train[user]:
            MSE += (alpha + beta_u[user] + beta_i[item] \
                    - userRatings_train[user][item]) **2
    MSE /= len(data_train)
    if(i == 99):
        print MSE
    i += 1
```

```
0.795455517619
```

Figure 11: Code for task 8

```
userRatings_valid = defaultdict(lambda: defaultdict(int))
itemRatings_valid = defaultdict(lambda: defaultdict(int))

for d in data_valid:
    user, item = d['reviewerID'], d['itemID']
    userRatings_valid[user][item] = d['rating']
    itemRatings_valid[item][user] = d['rating']

MSE = 0
for user in userRatings_valid:
    for item in userRatings_valid[user]:
        MSE += ((alpha \
                 + (beta_u[user] if user in beta_u else 0) \
                 + (beta_i[item] if item in beta_i else 0) \
                 - userRatings_valid[user][item]) **2)
MSE /= len(data_train)
print MSE
```

```
1.13945170902
```

```
predictions = open("assignment1/predictions_Rating.txt", 'w')
for l in open("assignment1/pairs_Rating.txt"):
    if l.startswith("userID"):
        #header
        predictions.write(l)
        continue
    u,i = l.strip().split('-')

    x = alpha
    if u in beta_u:
        x += beta_u[u]
    if i in beta_i:
        x += beta_i[i]
    predictions.write(u + '-' + i + ',' + str(x) + '\n')

predictions.close()
```

Figure 12: Result for task 8