In [1]:

```
import gzip
import numpy as np
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
```

In [2]:

```
def parse(path):
    g = gzip.open(path, 'rb')
    for l in g:
        yield eval(l)

def readJson(path):
    i = 0
    df = {}
    for d in parse(path):
        df[i] = d
        i += 1
    return df

df = readJson('assignment1/train.json.gz')
```

In [3]:

```
Data_train = [df[x] for x in range(0,len(df)/2)]
Data_valid = [df[x] for x in range(len(df)/2,len(df))]
```

Task 1

The value of alpha is : 0.783737364232

In [4]:

```
division = [x['helpful']['nHelpful']*1.0 / x['helpful']['outOf'] for x in Data_train if x['
alpha = sum(division) / len(division)
print 'The value of alpha is : ' + str(alpha)
```

```
The value of alpha is : 0.783737364232
```

Task 2

The Mean Absolute Error is : 0.258702157422

In [5]:

```
prediction2 = [x['helpful']['outOf'] * alpha for x in Data_valid]
y_valid = [x['helpful']['nHelpful'] for x in Data_valid]
MAE2 = mean_absolute_error(prediction2, y_valid)
print 'The Mean Absolute Error is : ' + str(MAE2)
```

```
The Mean Absolute Error is : 0.258702157422
```

Task 3

The fitted parameters are: [5.62218966e-01 2.11835412e-04 5.07029148e-02]

The Mean Absolute Error is: 0.240245808704

In [6]:

```
def feat(data):
    feat = [1]
    feat.append(len(data['reviewText'].split( )))
    feat.append(data['rating'])
    return feat
x3_train = [feat(x) for x in Data_train if x['helpful']['outOf'] != 0]

result3 = np.linalg.lstsq(x3_train, division)
theta3 = result3[0]
print 'The fitted parameters are: ' + str(theta3)
```

```
The fitted parameters are: [  5.62218966e-01   2.11835412e-04   5.07029148e-0
2]
```

In [7]:

```
x3_valid = [feat(x) for x in Data_valid]
prediction3 = np.dot(np.array(x3_valid), theta3) * np.array([x['helpful']['outOf'] for x in
MAE3 = mean_absolute_error(prediction3, y_valid)
print 'The Mean Absolute Error is: ' + str(MAE3)
```

```
The Mean Absolute Error is: 0.240245808704
```

Task 4

My user name is : mud G Jonh

In [8]:

```
df2 = readJson('assignment1/test_Helpful.json.gz')
Data_test = [df2[x] for x in range(0,len(df2))]
```

In [9]:

```
x4_test = [feat(x) for x in Data_test]
pred_rate = np.dot(np.array(x4_test), theta3)
```

In [10]:

```
index = 0
predictions = open("assignment1/predictions_Helpful.txt", 'w')
for l in open("assignment1/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*pred_rate[index]) +
    index += 1

predictions.close()
```

Task 5

The value of alpha is: 4.23198

The Mean Squared Error in validation set is: 1.2264713284

In [11]:

```
rating = [x['rating'] for x in Data_train]
alpha2 = sum(rating) / len(rating)
print 'The value of alpha is: ' + str(alpha2)
```

```
The value of alpha is: 4.23198
```

In [12]:

```
MSE5 = mean_squared_error([alpha2] * len(Data_valid), [x['rating'] for x in Data_valid])
print 'The Mean Squared Error is: ' + str(MSE5)
```

```
The Mean Squared Error is: 1.2264713284
```

Task 6

The MSE on the validation set is: 1.2815276734

In [13]:

```
rating_valid = [x['rating'] for x in Data_valid]
```

In [14]:

```
def regression (lmd):

    I_u = {} # number of items each user bought
    U_i = {} # nuber of buyer of each item
    for x in Data_train:
        if x['reviewerID'] not in I_u:
            I_u[x['reviewerID']] = 1
        else:
            I_u[x['reviewerID']] += 1
        if x['itemID'] not in U_i:
            U_i[x['itemID']] = 1
        else:
            U_i[x['itemID']] += 1
#     initialize
    a = 3
    beta_u = {}
    for x in I_u:
        beta_u[x] = 1
    beta_i = {}
    for x in U_i:
        beta_i[x] = 1

    count = 0
    MSE_list = []
    a_list = []
    beta_u_list = []
    beta_i_list = []
    while count < 5:
        a = sum([x['rating'] - beta_u[x['reviewerID']] - beta_i[x['itemID']] for x in Data_
        a_list.append(a)
        for x in I_u:
            beta_u[x] = 0
        for x in Data_train:
            beta_u[x['reviewerID']] += (x['rating'] - a - beta_i[x['itemID']]) / (lmd + I_u
        beta_u_list.append(beta_u)
        for x in U_i:
            beta_i[x] = 0
        for x in Data_train:
            beta_i[x['itemID']] += (x['rating'] - a - beta_u[x['reviewerID']]) / (lmd + U_i
        beta_i_list.append(beta_i)

        pred_valid = []
        for x in Data_valid:
            r = a
            if (x['reviewerID'] in I_u):
                r += beta_u[x['reviewerID']]
            if (x['itemID'] in U_i):
                r += beta_i[x['itemID']]
            pred_valid.append(r)
        MSE = mean_squared_error(pred_valid, rating_valid)
        MSE_list.append(MSE)
        if len(MSE_list) > 1:
            if MSE >= MSE_list[-2]:
                count += 1
            else:
                count = 0
    return MSE_list, a_list, beta_u_list, beta_i_list
```

In [15]:

```
MSE_list, a_list, beta_u_list, beta_i_list = regression(1)
print 'The MSE on the validation set is: ' + str(MSE_list[-6])
```

```
The MSE on the validation set is: 1.2815276734
```

Task 7

The user ID U816486110 has the largest value of beta: 1.51355433808

The user ID U052814411 has the smallest value of beta: -2.51275725845

The item ID I558325415 has the largest value of beta: 1.25102777231

The item ID I071368828 has the smallest value of beta: -2.36805972968

In [16]:

```
user_sorted = list(sorted(beta_u_list[-6], key = lambda x : beta_u_list[-6][x], reverse = F
item_sorted = list(sorted(beta_i_list[-6], key = lambda x : beta_i_list[-6][x], reverse = F
print 'The user ID ' + str(user_sorted[-1]) + ' has the largest value of beta: ' + str(beta
print 'The user ID ' + str(user_sorted[0]) + ' has the smallest value of beta: ' + str(beta
print 'The item ID ' + str(item_sorted[-1]) + ' has the largest value of beta: ' + str(beta
print 'The item ID ' + str(item_sorted[0]) + ' has the smallest value of beta: ' + str(beta
```

```
The user ID U816486110 has the largest value of beta: 1.51355433808
The user ID U052814411 has the smallest value of beta: -2.51275725845
The item ID I558325415 has the largest value of beta: 1.25102777231
The item ID I071368828 has the smallest value of beta: -2.36805972968
```

Task 8

I have a minimum MSE = 1.13959921647 with $\lambda = 7$

In [17]:

```
lmd = [0.01, 0.1, 1, 10, 100]
for x in lmd:
    result = regression(x)
    print 'The MSE with lambda = ' + str(x) + ' is: ' + str(result[0][-6])
```

```
The MSE with lambda = 0.01 is: 1.70419100323
The MSE with lambda = 0.1 is: 1.64228706141
The MSE with lambda = 1 is: 1.2815276734
The MSE with lambda = 10 is: 1.14374550024
The MSE with lambda = 100 is: 1.20329459617
```

In [18]:

```
for x in range(1,15):
    result = regression(x)
    print 'The MSE with lambda = ' + str(x) + ' is: ' + str(result[0][-6])
```

```
The MSE with lambda = 1 is: 1.2815276734
The MSE with lambda = 2 is: 1.19019963635
The MSE with lambda = 3 is: 1.15934870657
The MSE with lambda = 4 is: 1.14668337568
The MSE with lambda = 5 is: 1.14140518773
The MSE with lambda = 6 is: 1.13960918671
The MSE with lambda = 7 is: 1.13959783101
The MSE with lambda = 8 is: 1.14054328308
The MSE with lambda = 9 is: 1.1420066437
The MSE with lambda = 10 is: 1.14374550024
The MSE with lambda = 11 is: 1.14561826509
The MSE with lambda = 12 is: 1.14754102817
The MSE with lambda = 13 is: 1.14946350106
The MSE with lambda = 14 is: 1.15135556168
```

In [19]:

```
MSE_list8, a_list8, beta_u_list8, beta_i_list8 = regression(7)
```

In [20]:

```
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('-')
    r = a_list8[-6]
    if u in beta_u_list8[-6]:
        r += beta_u_list8[-6][u]
    if i in beta_i_list8[-6]:
        r += beta_i_list8[-6][i]
    predictions.write(u + '-' + i + ',' + str(r) + '\n')

predictions.close()
```