In [1]:

```
import numpy
import urllib
import scipy.optimize
import random
from collections import defaultdict # Dictionaries with default values
import nltk
import string
from nltk.stem.porter import *
from sklearn import linear_model
import ast
import gzip
f = gzip.open("train_Interactions.csv.gz", 'rt', encoding="utf8")
header = f.readline()
header = header.strip().split(',')
datatrain = []
datavalid = []
count=0
lenAll = 200000
for line in f:
    fields = line.strip().split(',')
    d = dict(zip(header, fields))
    if count <lenAll*0.95 :
        datatrain.append(d)
    else:
        datavalid.append(d)
    count=count+1

userReadBook = defaultdict(set)
bookAllUser  = defaultdict(set)

for d in datatrain:
    user,book,r =d['userID'],d['bookID'],d['rating']
    userReadBook[user].add(book)
    bookAllUser[book].add(user)
ratingMean = sum([float(d['rating']) for d in datatrain]) / len(datatrain)

N = len(datatrain)
nUsers = len(userReadBook)
nBooks = len(bookAllUser)
users = list(userReadBook)
books = list(bookAllUser)

alpha = ratingMean

userBiases = defaultdict(float)
bookBiases = defaultdict(float)

def prediction(user, book):
    return alpha + userBiases[user] + bookBiases[book]

def unpack(theta):
    global alpha
    global userBiases
    global bookBiases
    alpha = theta[0]
    userBiases = dict(zip(users, theta[1:nUsers+1]))
    bookBiases = dict(zip(books, theta[1+nUsers:]))

def cost(theta, labels, lamb):
    unpack(theta)
    predictions = [prediction(d['userID'], d['bookID']) for d in datatrain]
    cost = MSE(predictions, labels)
    print("MSE = " + str(cost))
    for u in userBiases:
        cost += lamb*userBiases[u]**2
    for i in bookBiases:
        cost += lamb*bookBiases[i]**2
    return cost

def derivative(theta, labels, lamb):
    unpack(theta)
    N = len(datatrain)
```

```
    dalpha = 0
    dUserBiases = defaultdict(float)
    dbookBiases = defaultdict(float)
    for d in datatrain:
        u,i = d['userID'], d['bookID']
        pred = prediction(u, i)
        diff = pred - float(d['rating'])
        dalpha += 2/N*diff
        dUserBiases[u] += 2/N*diff
        dbookBiases[i] += 2/N*diff
    for u in userBiases:
        dUserBiases[u] += 2*lamb*userBiases[u]
    for i in bookBiases:
        dbookBiases[i] += 2*lamb*bookBiases[i]
    dtheta = [dalpha] + [dUserBiases[u] for u in users] + [dbookBiases[i] for i in books]
    return numpy.array(dtheta)

def MSE(predictions, labels):
        differences = [(x-y)**2 for x,y in zip(predictions,labels)]
        return sum(differences) / len(differences)

alwaysPredictMean = [ratingMean for d in datatrain]
labels = [float(d['rating']) for d in datatrain]

MSE(alwaysPredictMean, labels)
scipy.optimize.fmin_l_bfgs_b(cost, [alpha] + [0.0]*(nUsers+nBooks),derivative, args = (labels, 1))

user_largest_beta =-100
user_largest_id = ""
user_smallest_beta = 100
user_smallest_id =""
book_largest_beta =-100
book_largest_id = ""
book_smallest_beta = 100
book_smallest_id =""\
for user in userBiases:
    if userBiases[user]>user_largest_beta:
        user_largest_beta = userBiases[user]
        user_largest_id = user

for user in userBiases:
    if userBiases[user]<user_smallest_beta:
        user_smallest_beta = userBiases[user]
        user_smallest_id = user

for book in bookBiases:
    if bookBiases[book]>book_largest_beta:
        book_largest_beta = bookBiases[book]
        book_largest_id = book

for book in bookBiases:
    if bookBiases[book]<book_smallest_beta:
        book_smallest_beta = bookBiases[book]
        book_smallest_id = book
print("user_largest_beta is ", user_largest_beta,"user_largest_id is ",user_largest_id)
print("user_smallest_beta is",user_smallest_beta,"user_smallest_id",user_smallest_id)
print("book_largest_beta is",book_largest_beta,"book_largest_id is ",book_largest_id)
print("book_smallest_beta is ",book_smallest_beta,"book_smallest_id is ",book_smallest_id)
```

```
MSE = 1.4735475011336192
MSE = 1.4560931393014562
MSE = 1.473389955772163
MSE = 1.4733899534013817
user_largest_beta is  0.00040413237874470305 user_largest_id is  u92864068
user_smallest_beta is -0.0015796730337471908 user_smallest_id u11591742
book_largest_beta is 0.0008292191795822705 book_largest_id is  b76915592
book_smallest_beta is  -0.0002721486787445039 book_smallest_id is  b57299824
```

In [ ]: