CS547 Assignment 3
Collaborative Recommendation Algorithm

Tingjun Li

1. Implement the Memory-Based Collaborative Filtering Algorithm

I implemented both two versions of the memory-based collaborative filtering algorithm as the Pearson Coefficient method and the vector similarity method.

```
#Similarity Function
#  Vector Space Similarity
def vectorSpaceSimilarity(x,y):
    a=x.astype(float)
    b=y.astype(float)

    num = dot(a,b)
    if num == 0:
       return num

    num = dot(a, b)
    temp = ((a**2).sum()**0.5)*((b**2).sum()**0.5)
    if temp == 0:
       temp = 1
    return num/temp

#  Pearson Correlation Coeffieient Similarity
def pearsonSimilarity(a,b):
    a=a.astype(float)
    b=b.astype(float)
    return vectorSpaceSimilarity(a-a.mean(), b-b.mean())
```

Complete codes all attached in the end.

2. Implement your algorithm 1 [memory based extension - default voting and amplification]

Discussion will be included in session 4

```
def vectorSpaceSimilarity(x,y):
    a=x.astype(float)
    b=y.astype(float)

    num = dot(a,b)
    if num == 0:
        return num
    #plug in default voting
    a_mean = a[a!=0].mean()
    b_mean = b[b!=0].mean()

    for i in range(numMovie):
        if a[i]>0 or b[i]>0:
            if a[i] == 0:
                a[i] = 2   # or a_mean
            elif b[i] == 0:
                b[i] = 2  # or b_mean

    num = dot(a, b)
    temp = ((a**2).sum()**0.5)*((b**2).sum()**0.5)
    if temp == 0:
        temp = 1
    return num/temp
```

3. Implement your algorithm 2 [memory based extension - Inverse User Frequency]

I implemented Inverse User Frequency on top of my vector space similarity method. Detailed discussion will be discussed in session 4.

```
# ==== in main ===
# inverse user freq
iuf_rating = empty_like (rating)
iuf_rating[:] = rating

iuf = zeros(numMovie)
for i in range(numMovie):
    iuf[i] = log(numUser/(len(rating[:,i][rating[:,i]!=0])+1))

for i in range(numMovie):
    iuf_rating[:,i] = iuf_rating[:,i] * iuf[i]
# =================

# === in predict ==

# Inverse User Freq
iuf_user = empty_like (user)
iuf_user[:] = user
for k in range(numMovie):
    iuf_user[k] = iuf_user[k] * iuf[k]

num = num + vectorSpaceSimilarity(iuf_user, iuf_rating[i]) * (normRating)
den = den + abs(vectorSpaceSimilarity(iuf_user, iuf_rating[i]))
# =================
```

4. Result Discussion

[1]**Accuracy**

Naive memory based with Vector Space Similarity:

Vector space similarity can generate a positive similarity from 0 to 1. Therefore it will only have positive impact to the prediction unlike Pearson coefficient.

MAE of GIVEN 5 : 0.855695885957234
MAE of GIVEN 10 : 0.802166666666667
MAE of GIVEN 20 : 0.769171409279444
OVERALL MAE : 0.805696929896569

Naive memory based with Pearson Coefficient method and the vector similarity method:

Pearson Coefficient normalized the votes with the user's mean.
It has two benefits:
1) normalizing the bias in voting - too mean or too generous.
2) Generate a score center at 0. Negative score will have a negative impact to prediction: I will hate the

movies user A likes if we have opposite taste.
From the scores we get, this factor does not seems to be that significant in our cases.

MAE of GIVEN 5 : 0.869826184819307
MAE of GIVEN 10 : 0.808833333333333
MAE of GIVEN 20 : 0.776020063663548
OVERALL MAE : 0.814890822525037

Memory based with Default Voting and Amplification:

MAE of GIVEN 5 : 0.834813054895586
MAE of GIVEN 10 : 0.785166666666667
MAE of GIVEN 20 : 0.749107745731649
OVERALL MAE : 0.786118863897554

Memory based with Inverse User Frequency:

MAE of GIVEN 5 : 0.857196448668251
MAE of GIVEN 10 : 0.798833333333333
MAE of GIVEN 20 : 0.767145750940484
OVERALL MAE : 0.80450664915449

Memory based with Inverse User Frequency, Default Voting and Amplification:

MAE of GIVEN 5 : 0.815180692759785
MAE of GIVEN 10 : 0.783666666666667
MAE of GIVEN 20 : 0.736567956014276
OVERALL MAE : 0.773969791495649

[2] Efficiency

Since all my algorithm are memory based, therefore they shared the same complexity which is O(numMovie * numUser).
It took about 2 mins to make a full run.

5. Code Attached: since all my algorithm are memory based therefore all the codes are in one file.

```
from numpy import *
import sys

#Similarity Function
#  Vector Space Similarity
def vectorSpaceSimilarity(x,y):
    a=x.astype(float)
    b=y.astype(float)

    num = dot(a,b)
    if num == 0:
       return num
    #plug in default voting
    a_mean = a[a!=0].mean()
    b_mean = b[b!=0].mean()

#    for i in range(numMovie):
#      if a[i]>0 or b[i]>0:
#          if a[i] == 0:
#              a[i] = 2
#          elif b[i] == 0:
#              b[i] = 2

    num = dot(a, b)
    temp = ((a**2).sum()**0.5)*((b**2).sum()**0.5)
    if temp == 0:
       temp = 1
    return num/temp


#  Pearson Correlation Coeffieient Similarity
def pearsonSimilarity(a,b):
    a=a.astype(float)
    b=b.astype(float)
    return vectorSpaceSimilarity(a-a.mean(), b-b.mean())

#    a_mod = empty_like (a)
#    b_mod = empty_like (b)
#    a_mod[:] = a
#    b_mod[:] = b
#    a_mean = a[a!=0].mean()
#    b_mean = b[b!=0].mean()
#    for j in range(numMovie):
#      if a[j]>0:
#          a_mod[j] = a_mod[j] - a_mean
#      if b[j]>0:
#          b_mod[j] = b_mod[j] - b_mean
#
    ##return vectorSpaceSimilarity(a_mod, b_mod)

def predict(user):
    pred = zeros(numMovie)
    pred = pred.astype(float)

    avg = user[user!=0].mean()

    user = user.astype(float)


    num = zeros(numMovie);
    den = 0.0;
```

```
    for i in range(numUser):
        normRating = empty_like (rating[i])
        normRating[:] = rating[i]

        for j in range(numMovie):
            if normRating[j]>0:
                normRating[j] = normRating[j] - trainMean[i]

        #=== Vector ===
        #num = num + vectorSpaceSimilarity(user, rating[i]) * (normRating)
        #den = den + abs(vectorSpaceSimilarity(user, rating[i]))

        #=== Inverse User Freq ===
        iuf_user = empty_like (user)
        iuf_user[:] = user
        for k in range(numMovie):
            iuf_user[k] = iuf_user[k] * iuf[k]

        #print iuf_rating
        #print iuf_user

        num = num + vectorSpaceSimilarity(iuf_user, iuf_rating[i]) * (normRating)
        den = den + abs(vectorSpaceSimilarity(iuf_user, iuf_rating[i]))

        #print vectorSpaceSimilarity(iuf_user, iuf_rating[i])

        #=== Pearson ===
        #num = num + pearsonSimilarity(user, rating[i]) * (normRating)
        #den = den + abs(pearsonSimilarity(user, rating[i]))

    if den==0:
        den = 1


    # amplication
    amp = 4
    pred = amp*num/den  + avg

    set_printoptions(precision=3, suppress=True)
    #print(num)
    #print(den)
    #print 10*num/den

    return pred

def generateResult(testfile):
    fin = open(testfile, "r")
    fout = open("result" + testfile[4:], "w")

    user = zeros(numMovie)
    isReading = True

    for line in fin:
        l = line.strip("\n").split(" ")
        if isReading == True and l[2] == '0':
            isReading = False
            pred = predict(user)
            pred
        if isReading == False and l[2] != '0':
            user = zeros(numMovie)
            isReading = True

        if isReading == True:
            user[int(l[1])-1] = int(l[2])
        if isReading == False:
            temp = int(round(pred[int(l[1])-1]))
```

```
            if temp > 5:
                #print "WRRRONNGGGG 5 "
                temp = 5
            if temp <= 0:
                #print "WRRRONNGGGG 0 "
                temp = 1

            result = l[0] + ' ' + l[1] + ' ' + str(temp) + '\n'
            fout.write(result)

trainFile = "train.txt"
testFile = ["test5.txt","test10.txt","test20.txt"]

rating = loadtxt(trainFile)
numUser,numMovie = shape(rating)


# inverse user freq
iuf_rating = empty_like (rating)
iuf_rating[:] = rating

iuf = zeros(numMovie)
for i in range(numMovie):
    iuf[i] = log(numUser/(len(rating[:,i][rating[:,i]!=0])+1))

for i in range(numMovie):
    iuf_rating[:,i] = iuf_rating[:,i] * iuf[i]
#==========


rating = rating.astype(float)
trainMean = zeros(numUser)
for i in range(numUser):
    trainMean[i] = rating[i][rating[i]!=0].mean()

#print rating
#print iuf_rating



for testfile in testFile:
    generateResult(testfile)
```