Information Retrieval

Assignment-3 : Recommender Systems

DOCUMENTATION

Date : 10/11/2017

Members : Rahul Polisetti - 2015AAPS0240H

Sai Aasrith - 2015A7PS0016H

Hitesh V. Bhagchandani - 2015A7PS0023H

## dataset.py

### get\_dataset():

Return:

ratings(user X items) : all the ratings given by users to movies

## utils.py

rmse(pred,y):

Parameter(s):

pred: The prediction made by the model

y: The actual value/rating given by the user

Return:

err : The root-mean-squered error

get\_means(ratings):

Parameter(s):

ratings(user X items) : matrix containing all the ratings

Return:

movie\_mean(row-vector) : mean of all movie ratings calculated separately for each movie.

user\_mean(row-vector) : mean of all the ratings given by each user

mean : mean of all the ratngs in the matrix

Note:

if x,y are 1-d vectors, np.where(condition,x,y) works like:

[xv if c else yv for (c,xv,yv) in zip(condition,x,y)]

get\_sim(ratings,f="users"):

Parameters(s):

ratings(user X items) : matrix containing all the ratings

f : selects whether to find nearest items or movies.

Return:

The normalized similarities of items/movies using Pearson correlation coefficient.

## predict\_CF.py

predict\_neighbours(user,movie,ratings,movie\_sim,k=15):

Parameter(s):

user : the user whose rating to the movie has to be predicted

movie : the movie

k : number of neighbours to consider

Return:

estimated\_rating : the rating estimated by taking a weighted average of neighbours

Note: implements only item-item C.F.

predict\_neighbours\_baseline(user,movie,ratings,movie\_mean,user\_mean,mean,movie\_sim,k=15):

Description:

Finds the weighted average of the ratings with respect to teh similarities of the movies. The calculation of the final rating can be done using the equation given in the design doc.

Parameter(s):

user : the user whose rating to the movie has to be predicted

movie : the movie

k : number of neighbours to consider

Return:

estimated\_rating : the rating estimated by taking a weighted average of neighbours

Note: implements only item-item C.F.

baseline = (bias-movie) + (bias-user) + all\_mean

= (movie\_mean - all\_mean) + (user\_mean - all\_mean) + all\_mean

= movie\_mean + user\_mean + all\_mean

test\_basic():

Description:

Predict using collaborative filtering without baseline model; using nearest neighbours only.

Return:

pred : prediction of ratings using only nearest neighbours

pred\_base : prediction of ratings using baseline along with basic C.F.

y : actual predictions

users\_items : users corresponding to the predictions made; useful for p@k

test\_base():

Description:

Predict using collaborative filtering using baseline model along with nearest neighbours.

Return:

pred : prediction of ratings using only nearest neighbours

pred\_base : prediction of ratings using baseline along with basic C.F.

y : a

evaluate(pred,y,users\_items,k):

Parameter(s):

pred : predicted ratings

y : actual ratings given

users\_items : users for whom ratings have been predicted

k : value for p@k

Return:

rmse : Root-mean squared error

p@k : precision at top-k

sman : Spearman's Correlation; rho = 1-6\*((\_sigma\_)(di)\*\*2)/n(n^2-1)

Note:

relevant items : items with actual rating greater or equal to 3.5.

Recommended item: has a predicted rating >= 3.5

p@k = (# of recommended items @k that are relevant) / (# of recommended items @k)

## predict\_cur.py

get\_prob():

Parameter(s):

Return:

prow,pcol : the probabilities of the corresponding rows and cols using the Frobenius

norm.

cur(k,no\_dup):

Parameter(s):

k : The number of columns and rows to sample based on the probability distribution.

no\_dup : 1 if duplications in the row and column matrices have to be avoided.

Return:

C,U,R : The factors obtained after CUR decomposition.

Note:

Replaces the missing values in the ratings matrix with the mean of the corresponding column.

predict\_cur(k,no\_dup=0):

Description:

Predicts movie ratings for users from the test set by reconstructing the ratings matrix.

Parameter(s):

k : The number of columns and rows to sample based on the probability distribution.

no\_dup : 1 if duplications in the row and column matrices have to be avoided.

Return:

reconstruct: The reconstructed ratings matrix

pred : A list of lists with each list of the form : [userid, prediction,itemid]

y : The actual ratings given by the users in the test set

evaluate(pred,y,k):

Description:

Evaluates the technique using for recommending along the following algorithms:

1) Root-mean-squared error

2) Precision at top-k

3) Spearman Coefficient

Parameter(s):

pred : predicted ratings

y : actual ratings given

users\_items : users for whom ratings have been predicted

k : value for p@k

Return:

rmse : Root-mean squared error

p@k : precision at top-k; this is a map which gives the precision at top-k for each user.

sman : Spearman's Correlation; rho = 1-6\*((\_sigma\_)(di)\*\*2)/n(n^2-1)

Note:

relevant items : items with actual rating greater or equal to 3.5.

Recommended item: has a predicted rating >= 3.5

p@k = (# of recommended items @k that are relevant) / (# of recommended items @k)

## predict\_svd.py

svd(r,flag):

Description:

Performs SVD decomposition of the given matrix using eigen-pairs.

Parameter(s):

r : The matrix to decompose

flag : 1 if 'r' is the ratings matrix

Return:

u,sigma,v : the decomposed elements after performing SVD

predict\_svd\_90():

Description :

Retains 90% of the energy in terms of singular values of the 'sigma' matrix obtained after

SVD decomposition and performs reconstruction of the original matrix using these singular

values.

Parameter(s):

Return:

pred : A list of lists with each list of the form : [userid, prediction,itemid]

y : The actual ratings given by the users in the test set

predict\_svd(U,S,Vt):

Description :

Reconstructs the rank matrix using the pieces obtained from SVD decomposition and predicts

the missing ratings

Parameter(s):

U,S,Vt : The decomposed pieces of the ratings matrix itself

Return:

pred : A list of lists with each list of the form : [userid, prediction,itemid]

y : The actual ratings given by the users in the test set

evaluate(pred,y,k):

Description:

Evaluates the technique using for recommending along the following algorithms:

1) Root-mean-squared error

2) Precision at top-k

3) Spearman Coefficient

Parameter(s):

pred : predicted ratings

y : actual ratings given

users\_items : users for whom ratings have been predicted

k : value for p@k

Return:

rmse : Root-mean squared error

p@k : precision at top-k; this is a map which gives the precision at top-k for each user.

sman : Spearman's Correlation; rho = 1-6\*((\_sigma\_)(di)\*\*2)/n(n^2-1)

Note:

relevant items : items with actual rating greater or equal to 3.5.

Recommended item: has a predicted rating >= 3.5

p@k = (# of recommended items @k that are relevant) / (# of recommended items @k)