CS 532 Homework 9
Lab: Recommender Systems

Jinman Zhao
jzhao237@wisc.edu

December 14, 2015

1. I guess Lucy is similar to Eric. They all favor Forrest Gump and Wall-E and disfavor Matrix.

2. Apply `CosSimVecMatrix` for Eric's user vector $\boldsymbol{u}_{Eric}$ and $\mathbf{U}$. We have the similarity vector

$$\begin{bmatrix} 0.6535 \\ 0.8064 \\ 1.0000 \\ 0.6732 \end{bmatrix}.$$

Thus the second user, Lucy, is most similar to Eric.

3. Apply `PCSimVecMatrix` for Eric's user vector $\boldsymbol{u}_{Eric}$ and $\mathbf{U}$. We have the similarity vector

$$\begin{bmatrix} -0.8389 \\ 0.9218 \\ 1.0000 \\ -0.6592 \end{bmatrix}.$$

Thus the second user, Lucy, is most similar to Eric; and the first user, John, is similar to Eric but in the opposite direction.

4. Equation 4 does not make use of the similarity weights.

5. By looking at Table 3, I guess we should recommend Wall-E to Kim, since Kim has vary similar ratings to that of Lucy and Lucy rate Wall-E high.

`PCSimVecMatrix` has a second output which gives a matrix that normalize each row in the second input `i_M`. So I can use this to skip the normalization using mean-centering.

My code is listed as following (`E5.m`).

```
X = [5 1 0 2 2; 1 5 2 5 5; 2 0 3 5 4; 4 3 5 3 0; 1 4 0 5 0];
K = X(5,:);
Y = X(1:4,:);
[m,n] = size(X);
[PC, normX] = PCSimVecMatrix(K', X);
normK = normX(5,:);
normY = normX(1:4,:);
indSim = find(abs(PC(1:4))>0.8); % (1)

K_unrated = K == 0;
Y_rated = Y ~= 0;
wtsum = sum(Y_rated(indSim,:) .* repmat(abs(PC(indSim)),1,n), 1); % (2)
K_usergen = sum(normY(indSim,:) .* repmat(PC(indSim),1,n), 1) ./ wtsum;

K_reco = K_usergen .* K_unrated
[val, movie] = max(K_reco)
K_bestrating = val + K_ratedmean
```

And it gives the result:

```
K_reco =
         0         0   -1.0454         0    0.8078

val =
    0.8078

movie =
     5

K_bestrating =
    4.1411
```

I think there may be some problem in the code given in `UserBasedPredictionToy.m`. (1) It does not pick similar users in the opposite direction. (2) It does not take care of unrated movies of similar users.

6. I use `UserBasedPredictionReal_kn.m` to run with Pearson Correlation, Mean-Centering and varies value for kn (from 1 to 100 with step-size 10). It yields the minimal error 0.0243 when $kn = 21$.

For small value of kn, it simply take too few similar users into consideration. For large value of kn, too many not-so-similar users harm the accuracy of the prediction.

7. Because John and Susan share vary similar rates for Die Hard 1 and 2, and Susan and Jack share vary similar rates for Die Hard 3 and 4. (PS: You may want to resolve the confusing with "Jack" and "Eric".)

I run `Examples/ToyExampleModelSVDMotivation.m` and it gives the same row vector for John and Jack in `XLowRank`. I agree that this method is helping us with the sparsity problem. (PS: `X` in `Examples/ToyExampleModelSVDMotivation.m` seems not the same data as in Table 5.)

8. I run `Examples/ToyExampleModelSVDMotivation2.m` and observe no output. (PS: The file's name is `ToyExampleModeSVDMotivation2.m` in the folder.) The first and the third row in `XPrediction` is the same. Yes, Low Rank approximation is finding some sort of latent association. The predictions are all in `XPrediction`.

9. I implement in `AverageValueBasedSVDCompletion.m`. Run the following code for the matrix in Table 1.

```
X = [5 1 0 2 2; 1 5 2 5 5; 2 0 3 5 4; 4 3 5 3 0];
k = 2;
AverageValueBasedSVDCompletion(X,k)
```

We have

```
ans =

    4.5762    0.9204    3.7139    1.8363    1.7539
    0.8183    4.9825    2.0864    5.0015    4.6704
    2.1134    3.0153    2.9259    5.0179    4.1469
    4.6148    3.1592    4.2453    3.4267    3.4328
```

The result of `Examples/ToyExampleAverageValueSVD.m` is

```
completion1 =

    4.4893    1.4752    3.3508    1.4523    1.7324
    0.7817    5.0519    2.3947    5.0843    4.6875
    2.0614    4.2411    2.8848    4.2577    4.0551
    4.5854    3.3197    4.1073    3.3100    3.4277
```

The two results are similar. (PS: The description in the pdf says step 2 "fill the empty cells with column average", but in `AverageValueBasedMatrixCompletion.m` you first call `FillZeroEntryWithAverageInArow`.)

The predictions make sense. They give similar ratings to movies for similar users. For example, it predicts Eric (row 3) will give good rate to Titanic (column 2) as Eric is similar to Lucy (row 2) who loves Titanic.

10. I implement in `IterativeSVDCompletion.m`. Run the following code for the matrix in Table 1.

```
X = [5 1 0 2 2; 1 5 2 5 5; 2 0 3 5 4; 4 3 5 3 0];
n_iter = 1000;
k = 2;
IterativeSVDCompletion(X, n_iter, k)
```

We have

```
ans =

    5.0000    1.0000    5.9773    2.0000    2.0000
    1.0000    5.0000    2.0000    5.0000    5.0000
    2.0000    4.4844    3.0000    5.0000    4.0000
    4.0000    3.0000    5.0000    3.0000    3.1283
```

The result of `Examples/ToyExampleIterativeSVD.m` is

```
completion2 =

    5.0000    1.0000    2.0455    2.0000    2.0000
    1.0000    5.0000    2.0000    5.0000    5.0000
    2.0000    3.4154    3.0000    5.0000    4.0000
    4.0000    3.0000    5.0000    3.0000    4.5010
```

The two results are similar.

The predictions make sense, as similarly above. My implementation gives better prediction than the previous, except for the first user with movie 3, it gives prediction out of 5. But your implementation seems give more like average prediction in the previous method.

11. I write in `E11.m`.

My results (Fig. 1) of running `ModelBasedPrediction/EvaluateModelBasedSVD.m` with the parameters $i_singularValueThreshold = 0.03, i_iterCount = 100, i_NumNearestNeighbor = 1$.

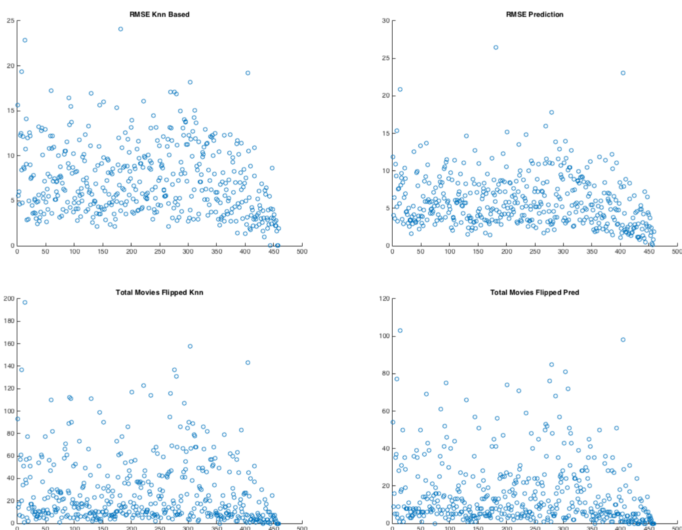
RMSE Knn Based
RMSE Prediction
Total Movies Flipped Knn
Total Movies Flipped Pred

Figure 1