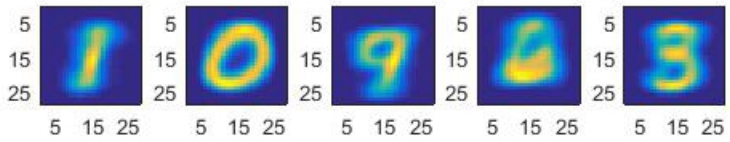
CS289 Homework 7

Jiaying Shi (SID: 24978491)

April 30, 2015

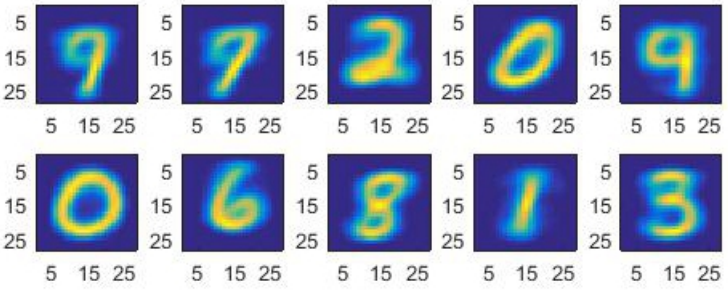
Problem 1

The K-means Clustering algorithm is implemented in MATLAB. The initial cluster centers are chosen randomly. When $k = 5$, the cluster centers are shown in the following figure:

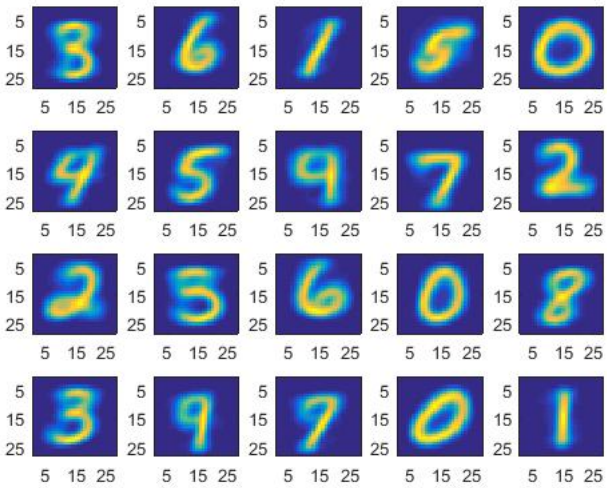


From the figure we can see that number 1 and number 0 has their own cluster. When there are 5 centers, number 4,7 and 9 may be clustered together. Number 3 and 8 may be clustered together. And the remaining numbers are clustered together. The results are pretty similar to the conclusion we get from the confusion matrix in the previous homework.

When $k = 10$, the cluster centers are shown as below:



When $k = 20$, the cluster centers are shown as below:



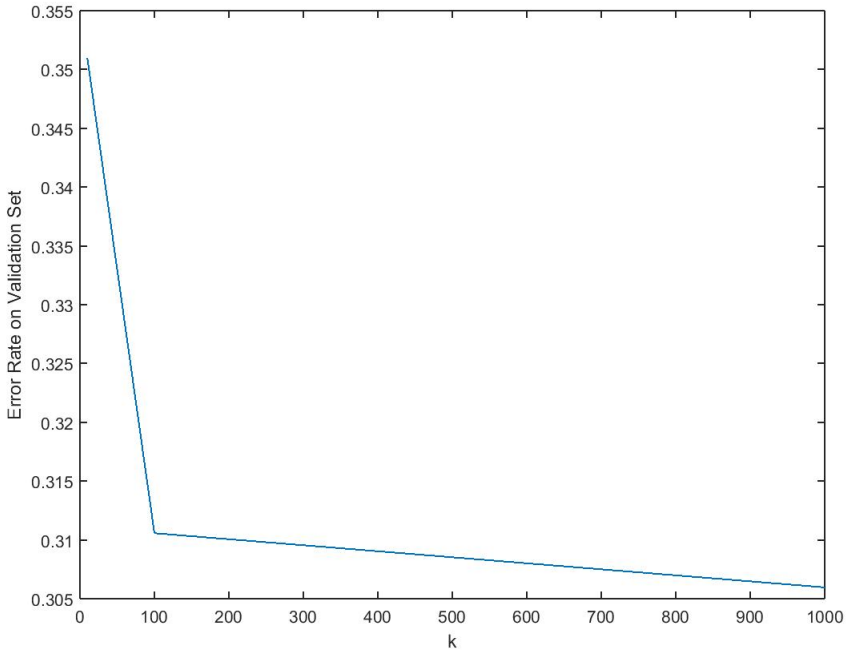
The algorithm is repeated for three times. The K-means loss varies in different runs. However the differences between different runs are not so large. The loss depends on the initialization of the algorithm but the influence of the initial centers on the loss is not so significant. The loss for each run is shown in the following table:

Loss	$k = 5$	$k = 10$	$k = 20$
Run 1	1.7682e+11	1.6259e+11	1.4901e+11
Run 2	1.7817e+11	1.6259e+11	1.4870e+11
Run 3	1.7682e+11	1.6259e+11	1.4894e+11

Problem 2

Part 1 Warm-up

The error rate on validation set when using the naive method by simply averaging the scores of all users is **0.3797**. When using K-nearest Neighborhood algorithm, when $k = 10, 10, 1000$, the error rate on validation set are **0.3509, 0.3106, 0.3060** ,respectively. The error rates are shown in the following graph:

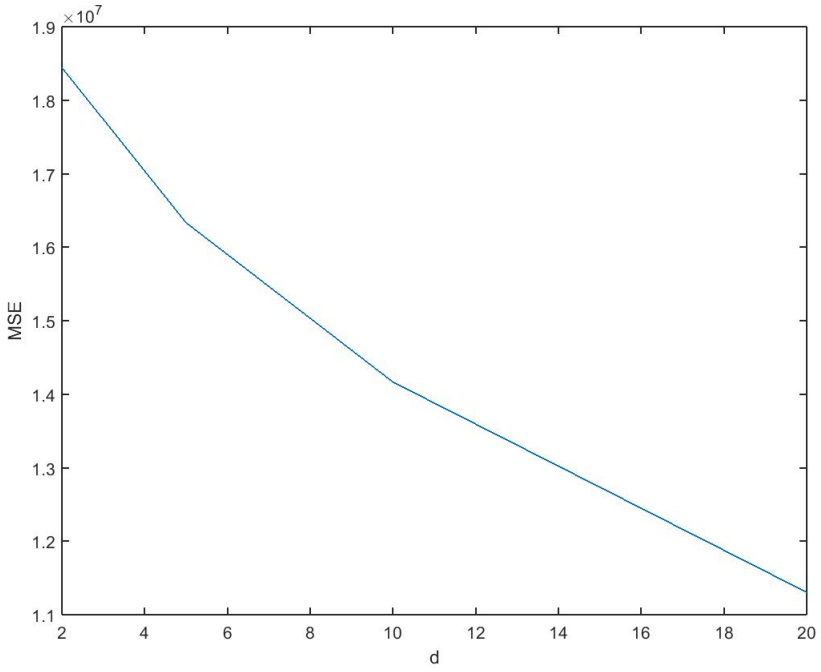


Comparing with the simple system, the accuracy rate has been improved.

Part 2: Latent Factor Model

The PCA algorithm is implemented in MATLAB. SVD decomposition is used to find vectors u_i and v_j. Since for matrix R, there are much fewer columns than rows, we can use $svd(R, 0)$ in MATLAB to do the decomposition so that the speed is much faster.

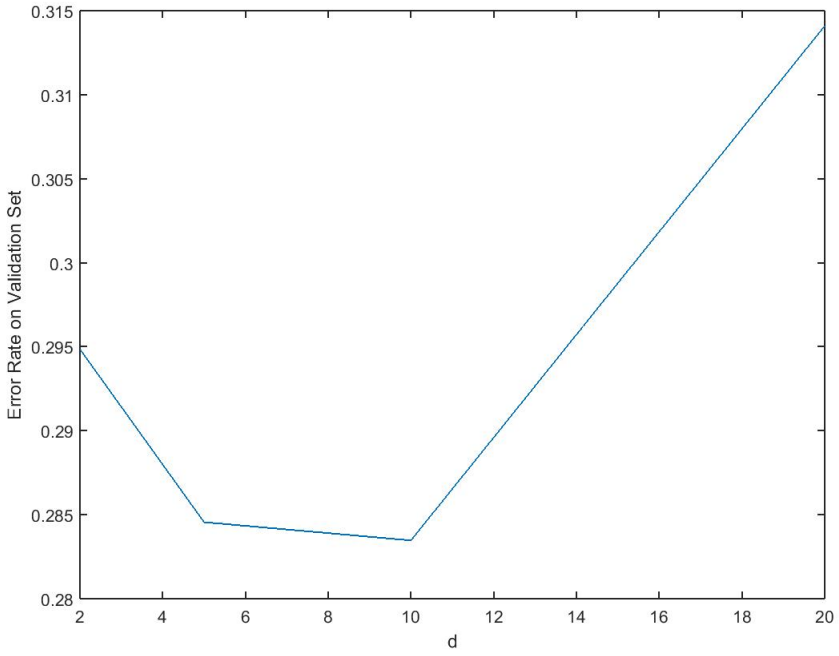
When evaluating the learnt vector represention by mean squared error (MSE), the MSE will decrease as d increases. The results can be seen in the following figure:



The learnt vector representations of users and jokes can be used to predict the response of a user to a certain joke. The R matrix is approximated as

$$\hat{R} = \begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_n^T \end{bmatrix} \begin{bmatrix} v_1 & v_2 & \dots & v_m \end{bmatrix}$$

The results for the validation data set, the prediction error rate is shown in the following graph:



From the above figure we can see that when $d = 10$, the validation error is small. When $d = 20$, there may be over fitting problem the variance is large. Hence, in the next part of the problem, I use $d = 10$. The prediction error rate on validation set is shown in the following table:

d	2	5	10	20
Error Rate	0.2949	0.2846	0.2835	0.3141

When the loss function is defined as

$$L\left(\{u_i\},\{v_j\}\right) = \sum_{(i,j)\in S} \left(\langle u_i, v_j\rangle - R_{ij}\right)^2 + \lambda \sum_{i=1}^{n} \|u_i\|^2 + \lambda \sum_{i=1}^{n} \|v_j\|^2, S = \{(i,j)\,|R_{ij} \neq NaN\}$$

to minimize the loss, we can apply an alternating minimization scheme. First, we can minimize the loss function with respective to $\{u_i\}$ by treating $\{v_j\}$ as constant vectors. Then we can treat $\{u_i\}$ as constant vectors and minimize the loss function with respect to $\{v_j\}$. When minimizing the loss function with respect to $\{u_i\}$, the optimal u_i^*can be expressed in closed form by taking derivative since the loss function is convex in each u_i:

$$u_i^* = \left(\sum_{(i,j)\in S} v_j v_j^T + \lambda I\right)^{-1} \left(\sum_{(i,j)\in S} v_j R_{ij}\right),\ i = 1, 2, ...n$$

Similarly, when updating $\{v_j\}$, we can use:

$$v_j^* = \left(\sum_{(i,j)\in S} u_i u_i^T + \lambda I \right)^{-1} \left(\sum_{(i,j)\in S} u_i R_{ij} \right), \ j = 1, 2, ...m$$

When the improvement in loss function between two iteration is smaller than some value , for example 0.01%, the algorithm should terminate. The algorithm is implemented in MATLAB. λ has been tuned to achieve a better performance. For different λ, the prediction error on validation data is shown in the following table.

λ	0.1	1	10
Error Rate	0.2835	0.2780	0.2835

The performance of the algorithm with this loss function is slightly better than the previtioous one with MSE.

Test is also conducted on the Kaggle data. When I do the test on Kaggle data, after obtain the representation vectors, we also use KNN after we estimate R. **The Kaggle score is 0.73042.**