Yes, MM^T and M^TM are both symmetric, square and real.

(1) Symmetric: We know that $M_{i,j} = M^T_{j,i}$, so $MM^T_{i,j} = \sum_{k=1}^{q} M_{i,k}M^T_{k,j} = \sum_{k=1}^{q} M_{i,k}M_{j,k}$ and $M^TM_{i,j} = \sum_{k=1}^{p} M^T_{i,k}M_{k,j} = \sum_{k=1}^{p} M_{k,i}M_{k,j}$. We can see that $MM^T_{i,j} = \sum_{k=1}^{q} M_{i,k}M_{j,k} = \sum_{k=1}^{q} M_{j,k}M_{i,k} = MM^T_{j,i}$ and $M^TM_{i,j} = \sum_{k=1}^{p} M_{k,i}M_{k,j} = \sum_{k=1}^{p} M_{k,j}M_{k,i} = M^TM_{j,i}$. Hence, MM^T and M^TM are both symmetric.

(2) Square: Since M is a $p \times q$ matrix and M^T is a $q \times p$ matrix, MM^T is a $p \times p$ matrix and M^TM is a $q \times q$ matrix. Hence, MM^T and M^TM are both square.

(3) Real: Since M is a real matrix, the product of M and its transpose will also be real.

Let v be the eigenvector of MM^T and $\lambda \neq 0$ be the corresponding eigenvalue, then we have $MM^Tv = \lambda v$. Then we multiply M^T om both side of the equation, we get $M^TMM^Tv = M^T(\lambda v) = \lambda M^Tv$. Then we have $M^TM(M^Tv) = \lambda(M^Tv)$. Set $M^Tv = u$, then we get $M^TMu = \lambda u$. We see that u is the eigenvector of M^TM with corresponding eigenvalue $\lambda \neq 0$. Hence, MM^T and M^TM have the same eigenvalue, but with different corresponding eigenvectors, where the eigenvector of MM^T is v but the eigenvector of M^TM is M^Tv.

By the definition of eigenvalue decomposition of a real, symmetric and square matrix, B can be written as: $B = Q\Lambda Q^T$ where $\Lambda = diag(\lambda_1, \dots, \lambda_d)$ contains the eigenvalues of B along its main diagonal and Q is an orthogonal matrix containing the eigenvectors of B as its columns. From 1(a), we know that M^TM is a real, symmetric and square matrix. Then we can denote M^TM as: $M^TM = Q\Lambda Q^T$ where $\Lambda = diag(\lambda_1, \dots, \lambda_d)$ contains the eigenvalues of M^TM along its main diagonal and Q is an orthogonal matrix containing the eigenvectors of M^TM as its columns.

Since U and V are column-orthonormal, we have $U^TU = I$ and $V^TV = I$. Since Σ is a diagonal matrix, we have $\Sigma = \Sigma^T$. Hence,

$$M^TM = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^TU^TU\Sigma V^T = V\Sigma^T\Sigma V^T = V\Sigma^2V^T$$

- Compute the SVD of M (*Use scipy.linalg.svd function in Python and set the argument* `full_matrices` *to False*). The function returns values corresponding to U, Σ and V^T. What are the values returned for U, Σ and V^T?

  ans:

  $$U = \begin{bmatrix} -0.27854301 & 0.5 \\ -0.27854301 & -0.5 \\ -0.64993368 & 0.5 \\ -0.64993368 & -0.5 \end{bmatrix}$$

  $$\Sigma = \begin{bmatrix} 7.61577311 & 0 \\ 0 & 1.41421356 \end{bmatrix}$$

  $$V^T = \begin{bmatrix} -0.70710678 & -0.70710678 \\ -0.70710678 & 0.70710678 \end{bmatrix}$$

- Compute the eigenvalue decomposition of M^TM (*Use scipy.linalg.eigh function in Python*). The function returns two parameters: a list of eigenvalues (let us call this list $Evals$) and a matrix whose columns correspond to the eigenvectors of the respective eigenvalues (let us call this matrix $Evecs$). Sort the list $Evals$ in descending order such that the largest eigenvalue appears first in the list. Also, re-arrange the columns in $Evecs$ such that the eigenvector corresponding to the largest eigenvalue appears in the first column of $Evecs$. What are the values of $Evals$ and $Evecs$ (after the sorting and re-arranging process)?

  ans: $Evals = \begin{bmatrix} 58 & 2 \end{bmatrix}$, $Evecs = \begin{bmatrix} 0.70710678 & -0.70710678 \\ 0.70710678 & 0.70710678 \end{bmatrix}$

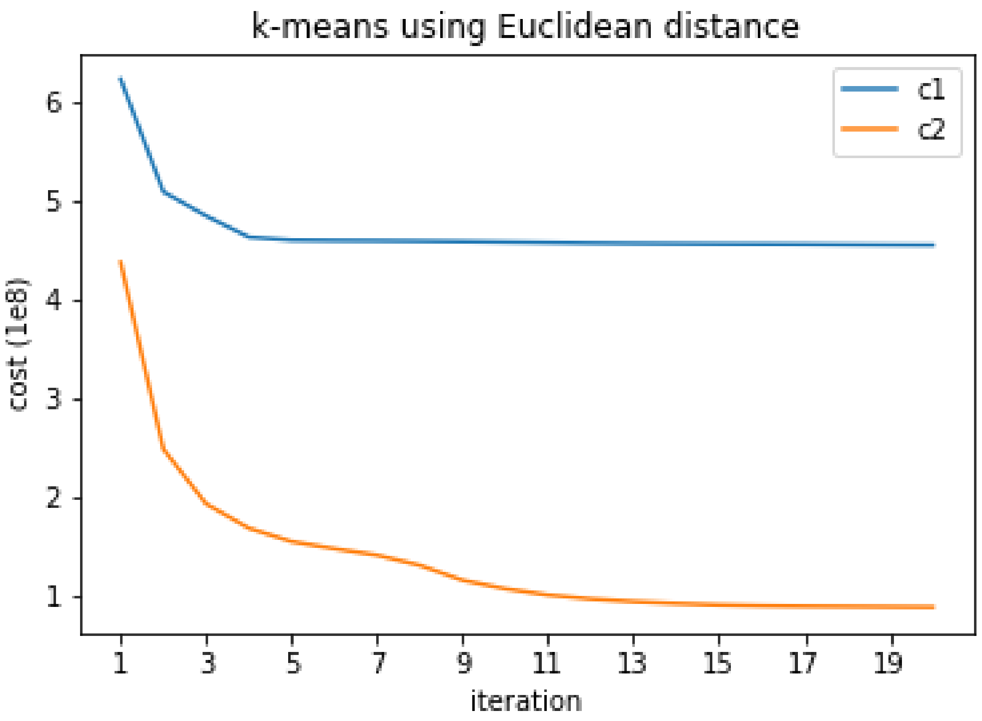
- Based on the experiment and your derivations in part (c) and (d), do you see any correspondence between V produced by SVD and the matrix of eigenvectors $Evecs$ (after the sorting and re-arranging process) produced by eigenvalue decomposition? If so, what is it?

  ans: The first column of V is -1 times the first eigenvector, and the second column of V is equal to the second eigenvector.

- Based on the experiment and the expressions obtained in part (c) and part (d) for M^TM, what is the relationship (if any) between the eigenvalues of M^TM and the singular values of M? Explain.

  ans: From (d), we have $M^TM = V\Sigma^2V^T$ which implies that $M^TMV = V\Sigma^2V^TV = V\Sigma^2$, so $diag(\Sigma^2)$ are eigenvalues of M^TM. Hence, the singular values of M are square roots of the eigenvalues of M^TM.

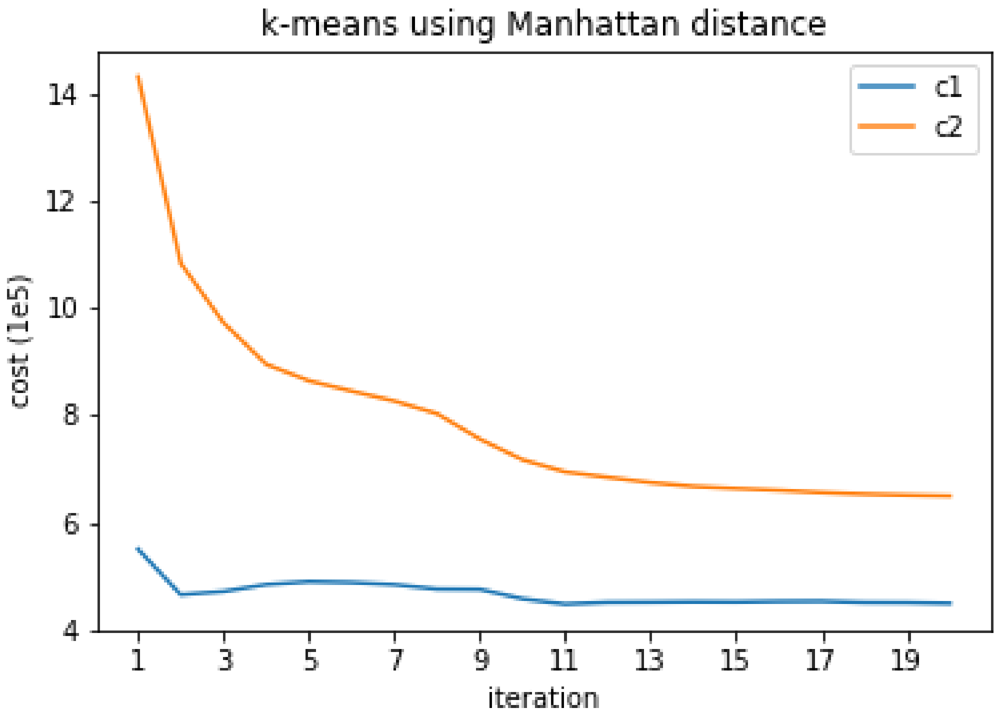
(1)



(2)

The percentage change in cost after 10 iterations of c1.txt is: 26.48%. The percentage change in cost after 10 iterations of c2.txt is: 76.70%. c2 is better than c1 since the initial clusters are far apart and less overlap. Hence, the true clusters will be split less often, leading to better final clusters results. Also, K-Means algorithm minimized the Euclidean distance between data and their centroids, so the cost of c2 less than c1 means that c2 is better.

(1)



(2)

The percentage change in cost after 10 iterations of c1.txt is: 18.65%. The percentage change in cost after 10 iterations of c2.txt is: 51.55%. c1 is better than c2 in terms of cost of Manhattan distance because c1 has lower final cost. Since the initial cluster centroids in c2 are as far as possible by using Euclidean distance, those centroids might not be the furthest in Manhattan distance.

Let's consider a case as follow: we have 5 data points $(-6, 0)$, $(-3, -4)$, $(\frac{-1}{2}, 1)$, $(3, 4)$, and $(6, 0)$. Then the initial centroids of c2 are $(-6, 0)$ and $(6, 0)$ with Euclidean distance 12 and Manhattan distance 12, but the initial centroids of c1 can be $(-3, -4)$ and $(3, 4)$ with Euclidean distance 10 and Manhattan distance 14. The results of K-Means with Manhattan distance metric are shown in the table. We can see that although c2 has the farthest seeds in Euclidean distance, its final cost is higher than c1 since the initial seeds are not the farthest seeds in Manhattan distance. Also, K-Means algorithm minimizes Euclidean distance but not Manhattan distance, so we can't guarantee that c2's final Manhattan distance cost is lower than c1's.

	Initial centroids	Final centroids	Cluster 1	Cluster 2	Cost
c1	(-3, -4), (3, 4)	($\frac{-9}{2}$, -2), ($\frac{17}{6}$, $\frac{5}{3}$)	(-6, 0), (-3, -4)	($\frac{-1}{2}$, 1), (3, 4), (6, 0)	$\frac{53}{3}$
c2	(-6, 0), (6, 0)	($\frac{-19}{6}$, -1), ($\frac{9}{2}$, 2)	(-6, 0), (-3, -4), ($\frac{-1}{2}$, 1)	(3, 4), (6, 0)	$\frac{56}{3}$

By computing the derivative of E to R_{iu}, q_i and p_u, we get:

$$\frac{\partial E}{\partial R_{iu}} = \varepsilon_{iu} = 2(R_{iu} - q_i \cdot p_u^T) \tag{1}$$

$$\frac{\partial E}{\partial q_i} = \nabla q_i = -2 * (R_{iu} - q_i \cdot p_u^T) * p_u + 2 * \lambda * q_i = -\varepsilon_{iu} * p_u + 2 * \lambda * q_i \tag{2}$$

$$\frac{\partial E}{\partial p_u} = \nabla p_u = -2 * (R_{iu} - q_i \cdot p_u^T) * q_i + 2 * \lambda * p_u = -\varepsilon_{iu} * q_i + 2 * \lambda * p_u \tag{3}$$

$$q_i := q_i - \eta * \nabla q_i \tag{4}$$

$$p_u := p_u - \eta * \nabla p_u \tag{5}$$

Therefore, the expression of ε_{iu} is:

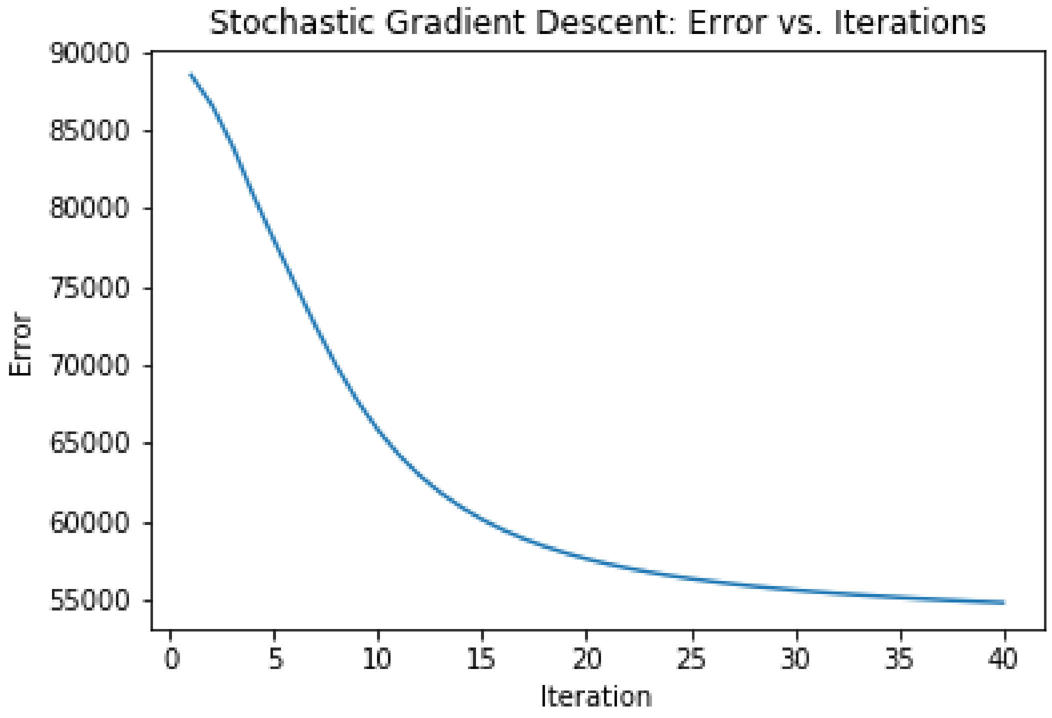
$$\varepsilon_{iu} = 2 * (R_{iu} - q_i \cdot p_u^T) \tag{6}$$

Then we can combine eq 2, 3, 4, and 5 to get the update equations for q_i and p_u as:

$$q_i := q_i + \eta * (\varepsilon_{iu} * p_u - 2 * \lambda * q_i) \tag{7}$$

$$p_u := p_u + \eta * (\varepsilon_{iu} * q_i - 2 * \lambda * p_u) \tag{8}$$

$\eta = 0.02.$


Stochastic Gradient Descent: Error vs. Iterations
Error
Iteration
90000
85000
80000
75000
70000
65000
60000
55000
0
5
10
15
20
25
30
35
40

First of all, we know that $R_{ij} = 0$ or 1 and $R^T_{ij} = R_{ji}$. Then we can compute T_{ii} and T_{ij} as follow:

$T_{ii} = \sum_{k=1}^{n} R_{ik} \times R^T_{ki} = \sum_{k=1}^{n} R^2_{ik}$. We know that $R^2_{ik} = 1$ if $R_{ik} = 1$, otherwise $R^2_{ik} = 0$. Hence, $T_{ii} = \sum_{k=1}^{n} R^2_{ik} = \sum_{k=1}^{n} R_{ik}$, so T_{ii} means the number of items that user i likes, also equals to the node degree of user i.

$T_{ij,i\neq j} = \sum_{k=1}^{n} R_{ik} \times R^T_{kj} = \sum_{k=1}^{n} R_{ik} \times R_{jk}$. $R_{ik} \times R_{jk} = 1$ if $R_{ik} = R_{jk} = 1$, otherwise $R_{ik} \times R_{jk} = 0$. It implies that $R_{ik} \times R_{jk} = 1$ if user i and user j both like item k. Thus, $T_{ij,i\neq j}$ means the number of items that both user i and user j like, also means the number of paths between user i and user j.

Let R_i^T be the i'th row vector in R^T, which means item i's ratings. Then we set R'^T such that $R'^T_{ij} = \frac{R^T_{ij}}{\|R^T_i\|}$, which means that the element R'^T_{ij} is equal to R^T_{ij} in R^T be divided by the length of i's row vector. And we know that $\|R_i^T\| = \sqrt{\sum_{j=1}^m (R^T_{ij})^2}$, which is equal to the square root of the number of users that like item i. Since the number of users that like item i is equal to the node degree of item i, we can rewrite $\|R_i^T\|$ as: $\|R_i^T\| = \sqrt{\sum_{j=1}^m (R^T_{ij})^2} = \sqrt{Q_{ii}}$. Let's define $Q^{-1/2}$ as $Q_{rc}^{-1/2} = 1/\sqrt{Q_{rc}}$ for all nonzero entries of the matrix, and 0 at all other positions. Thus, we can denote R'^T as: $R'^T = Q^{-1/2}R^T$. We know that the cosine similarity of *item* i and *item* j is:

$$\frac{\sum_{k=1}^m R^T_{ik}R^T_{jk}}{\|R^T_i\|\|R^T_j\|} = \frac{\sum_{k=1}^m R^T_{ik}R^T_{jk}}{\sqrt{Q_{ii}}\sqrt{Q_{jj}}} = R'^T_i \cdot (R'^T_j)^T$$

Thus, we can define the item similarity matrix S_I as:

$$S_I = R'^T \times (R'^T)^T$$

And we have $(R'^T)^T = (Q^{-1/2}R^T)^T = R(Q^{-1/2})^T = RQ^{-1/2}$ since $Q^{-1/2}$ is symmetric.
In conclusion, $S_I = R'^T \times (R'^T)^T = Q^{-1/2}R^TRQ^{-1/2}$.

Let R_i be the i'th row vector in R, which means user i's ratings. Then we set R' such that $R'_{ij} = \frac{R_{ij}}{\|R_i\|}$, which means that the element R'_{ij} is equal to R_{ij} in R be divided by the length of i's row vector. And we know that $\|R_i\| = \sqrt{\sum_{j=1}^n (R_{ij})^2}$, which is equal to the square root of the number of items that user i likes. Since the number of items that user i likes is equal to the node degree of user i, we can rewrite $\|R_i\|$ as: $\|R_i\| = \sqrt{\sum_{j=1}^n (R_{ij})^2} = \sqrt{P_{ii}}$. Let's define $P^{-1/2}$ as $P_{rc}^{-1/2} = 1/\sqrt{P_{rc}}$ for all nonzero entries of the matrix, and 0 at all other positions. Thus, we can denote R' as: $R' = P^{-1/2}R$. We know that the cosine similarity of *user* i and *user* j is:

$$\frac{\sum_{k=1}^n R_{ik}R_{jk}}{\|R_i\|\|R_j\|} = \frac{\sum_{k=1}^n R_{ik}R_{jk}}{\sqrt{P_{ii}}\sqrt{P_{jj}}} = R'_i \cdot (R'_j)^T$$

Thus, we can define the user similarity matrix S_U as:

$$S_U = R' \times (R')^T$$

And we have $(R')^T = (P^{-1/2}R)^T = R^T(P^{-1/2})^T = R^TP^{-1/2}$ since $P^{-1/2}$ is symmetric.
In conclusion, $S_U = R' \times (R')^T = P^{-1/2}RR^TP^{-1/2}$.

(1) user-user collaborative filtering:

$$r_{u,s} = \sum_{x \in users} \text{cos-sim}(x,u) * R_{x,s} = \sum_{x \in users} (P^{-1/2}RR^TP^{-1/2})_{u,x} * R_{x,s} = (P^{-1/2}RR^TP^{-1/2})_u \cdot R_s$$

where $(P^{-1/2}RR^TP^{-1/2})_u$ is the u'th row vector of $(P^{-1/2}RR^TP^{-1/2})$, and R_s is the s'th column vector of R. Thus, the recommendation matrix $\Gamma = P^{-1/2}RR^TP^{-1/2}R$.

(2) item-item collaborative filtering:

$$r_{u,s} = \sum_{x \in items} R_{u,x} * \text{cos-sim}(x,s) = \sum_{x \in items} R_{u,x} * (Q^{-1/2}R^TRQ^{-1/2})_{x,s} = R_u \cdot (Q^{-1/2}R^TRQ^{1-/2})_s$$

where $(Q^{-1/2}R^TRQ^{-1/2})_s$ is the s'th column vector of $(Q^{-1/2}R^TRQ^{-1/2})$, and R_u is the u'th row vector of R. Thus, the recommendation matrix $\Gamma = RQ^{-1/2}R^TRQ^{-1/2}$.

The names of five TV shows that have the highest similarity scores for Alex for the user-user collaborative filtering are:

1. FOX 28 News at 10pm
2. Family Guy
3. 2009 NCAA Basketball Tournament
4. NBC 4 at Eleven
5. Two and a Half Men

The names of five TV shows that have the highest similarity scores for Alex for the movie-movie collaborative filtering are:

1. FOX 28 News at 10pm
2. Family Guy
3. NBC 4 at Eleven
4. 2009 NCAA Basketball Tournament
5. Access Hollywood

Information sheet
CS246: Mining Massive Data Sets

Assignment Submission Fill in and include this information sheet with each of your assignments. This page should be the last page of your submission. Assignments are due at 11:59pm and are always due on a Thursday. All students (SCPD and non-SCPD) must submit their homework via Gradescope (http://www.gradescope.com). Students can typeset or scan their homework. Make sure that you answer each (sub-)question on a separate page. That is, one answer per page regardless of the answer length. Students also need to upload their code on Gradescope. Put all the code for a single question into a single file and upload it.

Late Homework Policy Each student will have a total of *two* late periods. *Homework are due on Thursdays at 11:59pm PT and one late period expires on the following Monday at 11:59pm PT.* Only one late period may be used for an assignment. Any homework received after 11:59pm PT on the Monday following the homework due date will receive no credit. Once these late periods are exhausted, any assignments turned in late will receive no credit.

Honor Code We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down their solutions independently, i.e., each student must understand the solution well enough in order to reconstruct it by him/herself. Students should clearly mention the names of all the other students who were part of their discussion group. Using code or solutions obtained from the web (GitHub/Google/previous year's solutions etc.) is considered an honor code violation. We check all the submissions for plagiarism. We take the honor code very seriously and expect students to do the same.

Your name: Yen-Yu Chang

Email: yenyu@stanford.edu **SUID:** 006350488

Discussion Group: Cheng-Min Chiang, Fang-I Hsiao, Alvin Hou

I acknowledge and accept the Honor Code.

(Signed) Y.Y.Chang