STANFORD UNIVERSITY
Stanford Center for Professional Development

HOMEWORK ROUTE FORM

Stanford Center for Professional Development Student Information

Course No. CS246 | Faculty / Instructor Name: Jure Leskovec | Date: 2/7/2013

Student Name: Philip Scuderi | Phone:

Company: | Email: pscuderi@gmail.com

City. | State | Country

Check One: (●) Homework #: 2 ( ) Midterm (●) Other

The email address provided on this form will be used to return homework, exams, and other documents and correspondence that require routing.

Total number of pages faxed including cover sheet: 27

	For Stanford Use Only	
Date Received by the Stanford Center for Professional Development	Date Instructor returned graded project Score/Grade: (to be completed by instructor or by teaching assistant)	Date the Stanford Center for Professional Development returned graded project:

Please attach this route form to ALL MATERIALS and submit ALL to:

Stanford Center for Professional Development
496 Lomita Mall, Durand Building, Rm 410, Stanford, CA 94305-4036
Office 650.725.3015 | **Fax** 650.736.1266 or 650.725.4138
For homework confirmation, email scpd-distribution@lists.stanford.edu
http://scpd.stanford.edu

Last modified October 27, 2008

CS246: Mining Massive Datasets

Assignment number: Homework 2

Submission time: 1:45 AM EST **and date:** 2/7/2013

Fill in and include this cover sheet with each of your assignments. It is an honor code violation to write down the wrong time. Assignments are due at 9:30 am, either handed in at the beginning of class or left in the submission box on the 1^{st} floor of the Gates building, near the east entrance. Failure to include the coversheet with you assignment will be penalized by 2 points.
Each student will have a total of *two* free late days. *One late day expires at the start of each class.* (Assignments are usually due on Thursdays, which means the first late day expires on the following Tuesday at 9:30am.) Once these late days are exhausted, any assignments turned in late will be penalized 50% per late day. However, no assignment will be accepted more than *one* late day after its due date. (If an assignment is due to Thursday then we will not accept it after the following Tuesday.)

Your name: Philip Scuderi
Email: pscuderi@gmail.com **SUNet ID:** pscuderi

Collaborators: Olga Kapralova, Valentina Kroshilina, Peter Lepeska
I acknowledge and accept the Honor Code.

(Signed) /Philip Scuderi/

(For CS246 staff only)

Late days: 0 1

Section	Score
1	
2	
3	
4	
Total	

Comments:

1a

Let $X =$ the non-normalized item similarity matrix $= (R^T)R$

$$S_I = \left(Q^{-1/2}\right)\left((X)\left(Q^{-1/2}\right)\right) = \left(Q^{-1/2}\right)\left(\left((R^T)R\right)\left(Q^{-1/2}\right)\right)$$

$$\boldsymbol{S_I = \left(Q^{-1/2}\right)\left(\left((R^T)R\right)\left(Q^{-1/2}\right)\right)}$$

Computing S_I for the example given, we obtain the following.

```
R = [1 0 1; 1 0 0; 1 1 1; 0 0 1];

Q = [3 0 0; 0 1 0; 0 0 3];

Si = (Q^-.5)*(((R')*R)*(Q^-.5))
```

Si =

1.0000	0.5774	0.6667
0.5774	1.0000	0.5774
0.6667	0.5774	1.0000

Let $T =$ the non-normalized user similarity matrix $= R(R^T)$

$$S_U = \left(P^{-1/2}\right)\left((T)\left(P^{-1/2}\right)\right) = \left(P^{-1/2}\right)\left(\left(R(R^T)\right)\left(P^{-1/2}\right)\right)$$

$$\boldsymbol{S_U = \left(P^{-1/2}\right)\left(\left(R(R^T)\right)\left(P^{-1/2}\right)\right)}$$

Computing S_U for the example given, we obtain the following.

```
R = [1 0 1; 1 0 0; 1 1 1; 0 0 1];

P = [2 0 0 0; 0 1 0 0; 0 0 3 0; 0 0 0 1];

Su = (P^-.5)*((R*(R'))*(P^-.5))
```

Su =

1.0000	0.7071	0.8165	0.7071
0.7071	1.0000	0.5774	0
0.8165	0.5774	1.0000	0.5774

0.7071 0 0.5774 1.0000

1b

Recall that $S_U(i,j) = cos(i,j)$

Also, because S_U is symmetric, $S_U(i,j) = S_U(j,i)$

$$r_{u,s} = \sum_{x \in users} cos(x,u) * R_{xs} = \sum_{x=1}^{m} S_U(x,u) * R_{xs} = \sum_{x=1}^{m} S_U(u,x) * R_{xs}$$

$$r_{u,s} = S_U(u,1) * R_{1s} + S_U(u,2) * R_{2s} + \cdots + S_U(u,m) * R_{ms}$$

Let Γ_U = the user recommendation matrix s.t. $\Gamma_U(i,j) = r_{i,j}\ \forall i,j$

$$\Gamma_U = (S_U)R$$

$$\boldsymbol{\Gamma_U = \left[\left(P^{-1/2}\right)\left(\left(R\left(R^T\right)\right)\left(P^{-1/2}\right)\right)\right]R}$$

Recall that $S_I(i,j) = cos(i,j)$

Also, because S_I is symmetric, $S_I(i,j) = S_I(j,i)$

$$r_{i,s} = \sum_{x \in items} R_{is} * cos(x,s) = \sum_{x=1}^{n} R_{is} * S_I(x,s) = \sum_{x=1}^{n} R_{is} * S_I(s,x)$$

$$r_{i,s} = R_{1s} * S_I(s,1) + R_{2s} * S_I(s,2) + \cdots + R_{ns} * S_I(s,n)$$

Let Γ_I = the item recommendation matrix s.t. $\Gamma_I(i,j) = r_{i,j}\ \forall i,j$

$$\Gamma_I = R(S_I)$$

$$\boldsymbol{\Gamma_I = R\left[\left(Q^{-1/2}\right)\left(\left(\left(R^T\right)R\right)\left(Q^{-1/2}\right)\right)\right]}$$

1c

T_{ii} = the degree of user node *i, i.e.* the number of items that user *i* likes

$T_{ij}\ \forall(i \neq j)$ = the number of item nodes with an edge to both user node *i* and user node *j, i.e. the* number of items that user *i* and user *j* both like

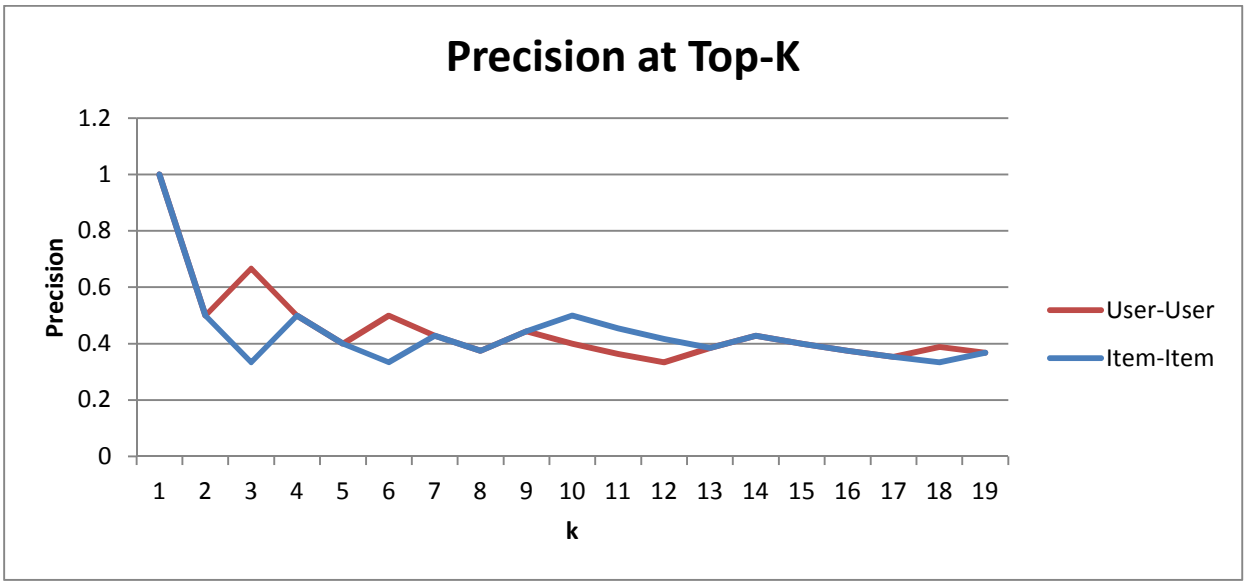
1d

User-User Collaborative Filtering:

TV Show	Similarity Score
FOX 28 News at 10pm	908.4801
Family Guy	861.1760
2009 NCAA Basketball Tournament	827.6013
NBC 4 at Eleven	784.7820
Two and a Half Men	757.6011

Item-Item Collaborative Filtering:

TV Show	Similarity Score
FOX 28 News at 10pm	31.3647
Family Guy	30.0011
NBC 4 at Eleven	29.3968
2009 NCAA Basketball Tournament	29.2270
Access Hollywood	28.9713



As you can see above, the precision decreases as k increases

2a

2a.1

We know from class (Dimensionality Reduction Lecture, slide 36) that:

$AA^T = U\sum\sum^T U^T$

$A^T A = V\sum\sum^T V^T$

It follows that the magnitudes of the singular values of AA^T and $A^T A$ are the same. Accordingly, we can use either AA^T or $A^T A$.

However, remember that AA^T has size 100x10,000. Therefore AA^T has size 100x100 and $A^T A$ has size 10,000x10,000.

Matrix AA^T is far smaller than matrix $A^T A$. Smaller matrices are typically easier to work with, and as such I recommend using AA^T.

2a.2

We can use eigenvalue decomposition to represent AA^T and $A^T A$ as follows:

$AA^T = UDU^T$

$A^T A = V\mathrm{E}V^T$

Where D and E represent the eigenvalues (Λ) of AA^T and $A^T A$.

2a.3

We know from the problem definition that:

$A = USV^T$

$A^T = (USV^T)^T$

$A^T = (USV^T)^T$

$A^T = VS^T U^T$

$A^T U = (VS^T U^T)U$

$A^T U = VS^T U^T U$ (2.a.3.1)

U is orthogonal, and therefore:

$$U^T = U^{-1} \tag{2.a.3.2}$$

Using equation (2.a.3.2), we can reduce equation (2.a.3.1) as follows:

$$A^T = VS^T \tag{2.a.3.3}$$

S is a diagonal matrix, so it follows that $S = S^T$ (2.a.3.4)

Using equation (2.a.3.4), we can rewrite equation (2.a.3.3) as follows:

$$A^T U = VS$$

$$A^T[u_1 \ u_2 \ \dots \ u_m] = \begin{bmatrix} v_{11} & \cdots & v_{1m} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{nm} \end{bmatrix} S \tag{2.a.3.5}$$

Recall that S is a diagonal matrix with the singular values of A on its diagonal. Thus, we can rewrite equation (2.a.3.5) as follows:

$$A^T[u_1 \ u_2 \ \dots \ u_m] = \begin{bmatrix} v_{11} & \cdots & v_{1m} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{nm} \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_m \end{bmatrix} = \begin{bmatrix} \lambda_1 v_{11} & \cdots & \lambda_n v_{1m} \\ \vdots & \ddots & \vdots \\ \lambda_1 v_{n1} & \cdots & \lambda_n v_{nm} \end{bmatrix} = [\lambda_1 v_1 \ \lambda_1 v_2 \ \dots \ \lambda_m v_m]$$

$$A^T[u_1 \ u_2 \ \dots \ u_m] = [\lambda_1 v_1 \ \lambda_1 v_2 \ \dots \ \lambda_m v_m]$$

$$A^T u_i = \lambda_i v_i \tag{2.a.3.6}$$

<u>2a.4</u>

From part 3 above (2.a.3.5), we know:

$$A^T U = VS$$

$$A^T U(S^{-1}) = VS(S^{-1})$$

$$A^T USS^{-1} = V$$

$$V = A^T USS^{-1} \tag{2.a.4.1}$$

We are given the eigenvalue decomposition of AA^T as follows:

$$AA^T = UDU^T \tag{2.a.4.2}$$

We know from class that:

$$D = S^2$$

$$S = D^{1/2} \tag{2.a.4.3}$$

Remember from equation (2.a.4.1), we know the following:

$$V = A^T US S^{-1} \tag{2.a.4.5}$$

We are given A^T. We know U from the singular value decomposition (2.a.4.2). We know D from the singular value decomposition (2.a.4.2), which means we can use equation (2.a.4.4) to derive S and S^{-1}. It follows that we can solve for V using equation (2.a.4.5).

2a.5

$$AV = US$$

$$A[v_1 \ v_2 \ \dots \ v_n] = \begin{bmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mm} \end{bmatrix} S \tag{2.a.5.1}$$

Recall that S is a diagonal matrix with the singular values of A on its diagonal. Thus, we can rewrite equation (2.a.5.1) as follows:

$$A[v_1 \ v_2 \ \dots \ v_n] = \begin{bmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mm} \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix} = [\lambda_1 v_1 \ \lambda_1 v_2 \ \dots \ \lambda_n v_n]$$

$$A[v_1 \ v_2 \ \dots \ v_n] = [\lambda_1 v_1 \ \lambda_1 v_2 \ \dots \ \lambda_n v_n]$$

$$Av_i = \lambda_i u_i$$

$$\frac{1}{\lambda_i} Av_i = u_i$$

$$u_i = \frac{1}{\lambda_i} A v_i \qquad \text{(2.a.5.2)}$$

We are given A. We know the λ_i values because they are on the diagonal of S, which we are given. We know each v_i because the eigenvalue decomposition of $A^T A = VEV^T$. Therefore, we can use equation (2.a.5.2) to construct U.

2b

2b.1

$M = M^T$

2b.2

$$M = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$$

$$M \begin{bmatrix} v_i \\ -u_i \end{bmatrix} = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} v_i \\ -u_i \end{bmatrix} = \begin{bmatrix} -A^T u_i \\ A v_i \end{bmatrix} \qquad \text{(2.b.2.1)}$$

Recall that in equation (2.a.3.6), we demonstrated the following:

$$A^T u_i = \lambda_i v_i \qquad \text{(2.b.2.2)}$$

We know from the problem definition that:

$A = USV^T$

$$AV = USV^T V \qquad \text{(2.b.2.3)}$$

V is orthogonal, and therefore:

$$V^T = V^{-1} \qquad \text{(2.b.2.4)}$$

Using equation (2.b.2.4), we can reduce equation (2.b.2.3) as follows:

$AV = US$

$$A[v_1 \ v_2 \ \dots \ v_n] = \begin{bmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mm} \end{bmatrix} S \qquad (2.b.2.5)$$

Recall that S is a diagonal matrix with the singular values of A on its diagonal. Thus, we can rewrite equation (2.b.2.5) as follows:

$$A[v_1 \ v_2 \ \dots \ v_n] = \begin{bmatrix} u_{11} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mm} \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix} = [\lambda_1 v_1 \ \lambda_1 v_2 \ \dots \ \lambda_n v_n]$$

$$A[v_1 \ v_2 \ \dots \ v_n] = [\lambda_1 v_1 \ \lambda_1 v_2 \ \dots \ \lambda_n v_n]$$

$$Av_i = \lambda_i u_i \qquad (2.b.2.6)$$

Using equations (2.b.2.2) and (2.b.2.b), we can rewrite equation (2.b.2.1) as follows:

$$M \begin{bmatrix} v_i \\ -u_i \end{bmatrix} = \begin{bmatrix} -\lambda_i v_i \\ \lambda_i u_i \end{bmatrix} = -\lambda_i \begin{bmatrix} v_i \\ -u_i \end{bmatrix}$$

$$\boldsymbol{M} \begin{bmatrix} \boldsymbol{v_i} \\ \boldsymbol{-u_i} \end{bmatrix} = -\boldsymbol{\lambda_i} \begin{bmatrix} \boldsymbol{v_i} \\ \boldsymbol{-u_i} \end{bmatrix} \qquad (2.b.2.7)$$

Therefore, $\begin{bmatrix} v_i \\ -u_i \end{bmatrix}$ is an eigenvector of $\boldsymbol{M}$ with eigenvalue $-\boldsymbol{\lambda_i}$.

$$M \begin{bmatrix} v_i \\ u_i \end{bmatrix} = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} v_i \\ u_i \end{bmatrix} = \begin{bmatrix} A^T u_i \\ A v_i \end{bmatrix} \qquad (2.b.2.8)$$

Using equations (2.b.2.2) and (2.b.2.b), we can rewrite equation (2.b.2.8) as follows:

$$M \begin{bmatrix} v_i \\ u_i \end{bmatrix} = \begin{bmatrix} \lambda_i v_i \\ \lambda_i u_i \end{bmatrix}$$

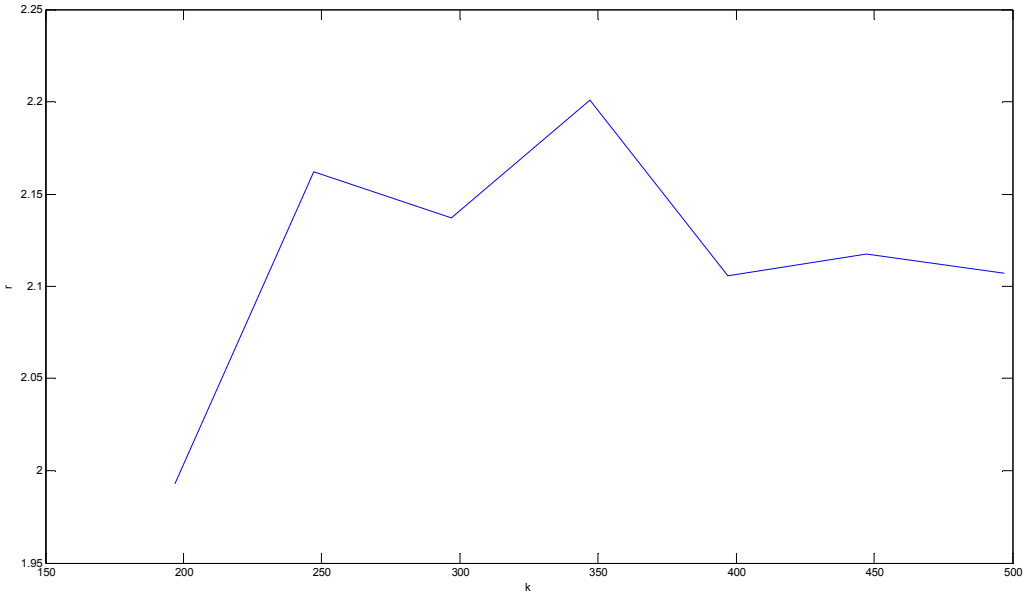
$$\boldsymbol{M} \begin{bmatrix} \boldsymbol{v_i} \\ \boldsymbol{u_i} \end{bmatrix} = \boldsymbol{\lambda_i} \begin{bmatrix} \boldsymbol{v_i} \\ \boldsymbol{u_i} \end{bmatrix} \qquad (2.b.2.9)$$

Therefore, $\begin{bmatrix} v_i \\ u_i \end{bmatrix}$ is an eigenvector of $\boldsymbol{M}$ with eigenvalue $\boldsymbol{\lambda_i}$.

2b.3

We showed that $-\lambda_i$ and λ_i are eigenvalues of respective eigenvectors $\begin{bmatrix} v_i \\ -u_i \end{bmatrix}$ and $\begin{bmatrix} v_i \\ u_i \end{bmatrix}$ in equations (2.b.2.7) and (2.b.2.9) above.

2c



3a

$$\sum_{x \in S} \|x - z\|^2 \; - \sum_{x \in S} \|x - c(S)\|^2$$

$$\sum_{x \in S} (\|x - z\|^2 - \; \|x - c(S)\|^2) \tag{3.a.1}$$

$$\forall v \; \in \; \mathbb{R}^d, v(v^T) = \|v\|^2 \tag{3.a.2}$$

Using equation (3.a.2), we can rewrite equation (3.a.1) as follows:

$$\sum_{x \in S} \left[(x - z)(x - z)^T - \; \big(x - c(S)\big)\big(x - c(S)\big)^T\right]$$

$$\sum_{x \in S} \left[(x - z)(x^T - z^T) - \; \big(x - c(S)\big)(x^T - c(S)^T)\right]$$

$$\sum_{x \in S}[(xx^T - x^Tz - xz^T + zz^T) - (xx^T - x\,c(S)^T - x^Tc(S) + c(S)c(S)^T)]$$

$$\sum_{x \in S}[xx^T - x^Tz - xz^T + zz^T - xx^T + x\,c(S)^T + x^Tc(S) - c(S)c(S)^T]$$

$$\textstyle\sum_{x \in S}[- x^Tz - xz^T + zz^T + x\,c(S)^T + x^Tc(S) - c(S)c(S)^T] \quad (3.a.3)$$

Because x and z are both scalars their transposes are equal, i. e, $x^Tz = (x^Tz)^T = xz^T$ (3.a.4)

Using the principle from equation (3.a.4), we can rewrite equation (3.a.3) as follows:

$$\sum_{x \in S}[-2xz^T + zz^T + 2x\,c(S)^T - c(S)c(S)^T]$$

$$\sum_{x \in S}[-2xz^T + 2x\,c(S)^T] + \sum_{x \in S}[zz^T - c(S)c(S)^T]$$

$$\sum_{x \in S}[2x(c(S)^T - z^T)] + |S|[zz^T - c(S)c(S)^T]$$

$$2(c(S)^T - z^T)\textstyle\sum_{x \in S}[x] + |S|[zz^T - c(S)c(S)^T] \quad (3.a.5)$$

Because $c(S)$ is the center of mass $\forall x \in S$, it follows that:

$$c(S) = \frac{\sum_{x \in S} x}{|S|}$$

$$\textstyle\sum_{x \in S} x = |S|\,c(S) \quad (3.a.6)$$

Using equation (3.a.6), we can rewrite equation (3.a.5) as follows:

$$2(c(S)^T - z^T)\big(|S|\,c(S)\big) + |S|[zz^T - c(S)c(S)^T]$$

$$2|S|\,c(S)(c(S)^T - z^T) + |S|[zz^T - c(S)c(S)^T]$$

$$|S|[2\,c(S)(c(S)^T - z^T) + (zz^T - c(S)c(S)^T)]$$

$$|S|\big[\big(2\,c(S)c(S)^T - 2\,z^Tc(S)\big) + (zz^T - c(S)c(S)^T)\big]$$

$$|S|[zz^T - 2\,z^Tc(S) + 2\,c(S)c(S)^T - c(S)c(S)^T]$$

$$|S|[zz^T - 2\,z^T c(S) + c(S)c(S)^T]$$

$$|S|[c(S)c(S)^T - 2c(S)\,z^T + zz^T] \quad \text{(3.a.7)}$$

Using the principle from equation (3.a.4), we can rewrite equation (3.a.7) as follows:

$$|S|[c(S)c(S)^T - c(S)\,z^T - c(S)^T z + zz^T]$$

$$|S|[(c(S) - z)(c(S)^T - z^T)]$$

$$|S|[(c(S) - z)(c(S) - z)^T] \quad \text{(3.a.8)}$$

Using the principle from equation (3.a.2), we can rewrite equation (3.a.8) as follows:

$$|S| \cdot \|c(S) - z\|^2$$

Thus, we have proved that:

$$\sum_{x \in S} \|x - z\|^2 - \sum_{x \in S} \|x - c(S)\|^2 = |S| \cdot \|c(S) - z\|^2$$

<u>3b</u>

At each iteration of the k-means algorithm the operations performed can be segmented into the following logical steps:

1) <u>Centroid Calculation</u>: The centroid of each cluster is recalculated by taking the average of all items in the cluster.
2) <u>Item Assignment</u>: All the elements are reassigned to the cluster with the closest centroid (calculated in step 1).

Cost function $\phi = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$

Let $\phi_i =$ the cost function ϕ at step i of the $k-$means algorithm

Let $\phi^x = \min_{c \in C} \|x - c\|^2$

$=$ the cost contributed to the cost funciton by an artibtary element x in the summation of the cost function ϕ

1) Centroid Calculation's Effect on the Cost Function ϕ

A new centroid is calculated by taking the average across all items in a cluster *C*. This necessarily minimizes $\sum_{x \in C} \phi^x$.

In other words, the cost contributed to the cost function by all the elements in an arbitrary cluster C cannot increase during recalculation of cluster C's centroid.

2) Item Assignment's Effect on the Cost Function ϕ

In this logical step, each element $x \epsilon X$ is assigned to the cluster that element *x* is closest to. The following two cases are possible for each element $x \epsilon X$:

2.1) *x* is closest to the centroid of the cluster that it was closest to in the previous k-means iteration: In this case, *x* is reassigned to the same cluster as it was in the previous iteration. This has no effect on ϕ^x. In other words, reassigning *x* to the same cluster does not increase *x*'s contribution to the cost function.

2.2) *x* is closest to the centroid of a different cluster than it was closest to in the previous k-means iteration: In this case, *x* is assigned to a new cluster. This has necessarily decreases ϕ^x. In other words, assigning *x* to a new cluster necessarily decreases increase *x*'s contribution to the cost function.

We have shown above that neither logical step (1) and (2) can decrease the cost function ϕ_i during any iteration of k-means. It necessarily follows that $\phi_i \geq \phi_{i+1}$.

3c

We showed above in 3b that the cost ϕ in the k-means algorithm monotonically decreases. According to Piazza thread #330, the only other fact we need to show is that the cost ϕ doesn't decrease all the way to negative infinity.

Remember that the cost function is defined as follows:

$$\text{Cost function } \phi = \sum_{x \epsilon X} \min_{c \in C} \|x - c\|^2$$

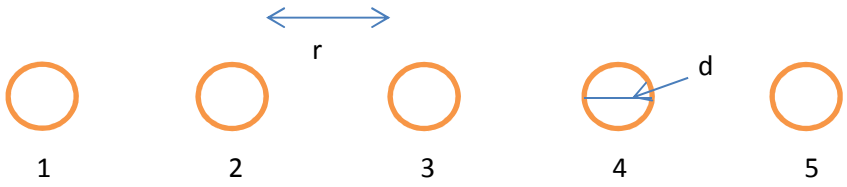
The cost function ϕ is necessarily a positive value because $\|\cdot\|$ denotes the L_2 norm, which is always greater than or equal to zero. The sum of any values greater than or equal to zero is necessarily greater than or equal to zero.

3d

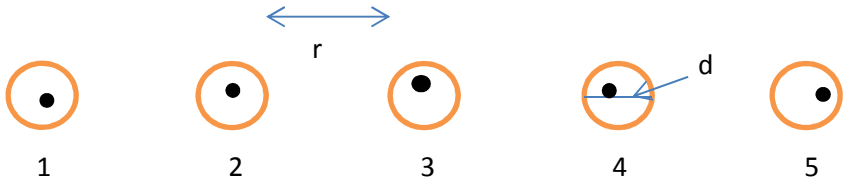
Consider a data set including n points in 5 clusters arranged in a row where:

1) r = the distance between each of the clusters that are adjacent (see figure blow); and
2) d = the diameter of each of the clusters, where d << r

The following figure demonstrates this setup. Note that the figure is not necessarily drawn to scale.

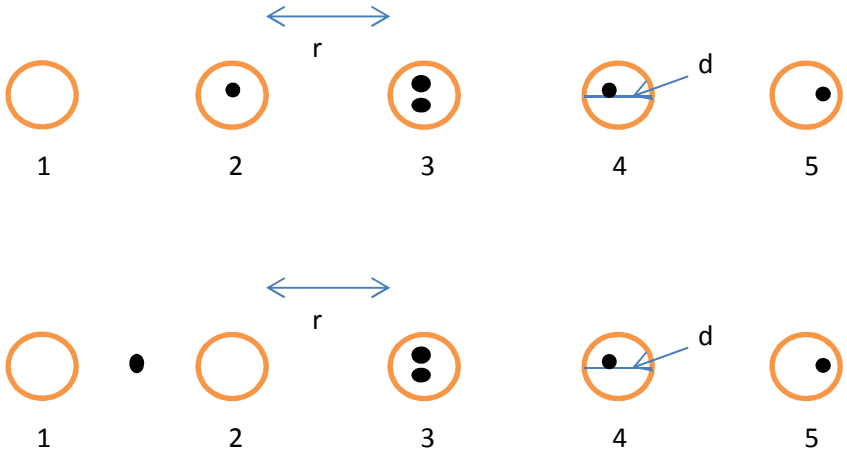


The optimal clustering for k = 5 occurs when one initial centroid is chosen in each cluster as follows.



The optimal clustering for k = 5 results in a cost $\phi = O(nd)$

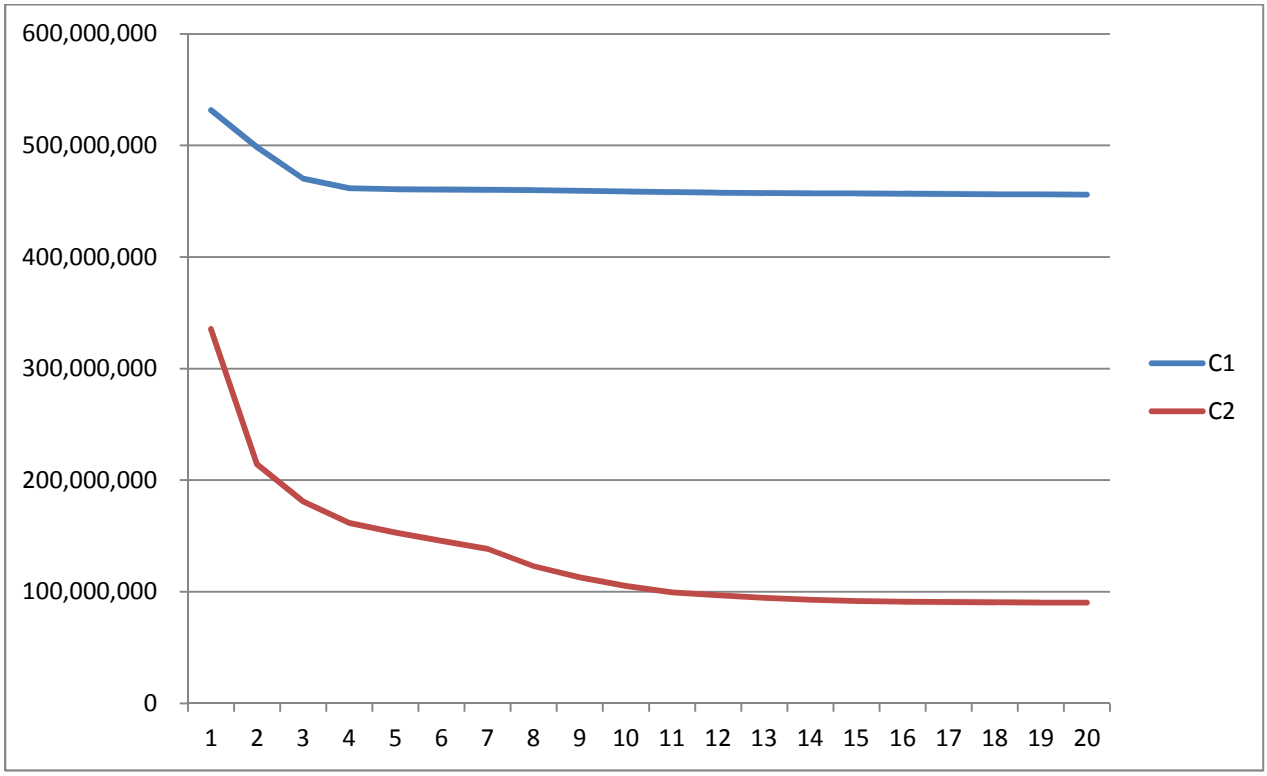
Now consider what happens when centroids are initialized such that two initial centroids are in cluster 3, 0 initial centroids are in cluster 1, and 1 initial centroid is in each of clusters 2, 5, and 5, as shown below.



In the above situation, the k-means algorithm will converge to a local optimum where clusters 1 and 2 will be assigned to the centroid furthest to the left. The points in cluster 3 will share two centroids. In this situation, $\phi = O(r^2 n)$.

As you can see, the cost function can be made arbitrarily larger by increasing the value of r.

4b



4c

No, random initialization of k-means using c1.txt is not better than using c2.txt in terms of cost.

When we start with the random initialization using c1.txt the k-means algorithm levels off in terms of improvement after about 5 iterations. Even after 20 iterations, the "c1" method's error is significantly higher than the "c2" method's error after merely 1 iteration.

The percentage change after 10 iterations of the "c1" method was -16%.

4d

instillation

methoxyfenozide

sodium-sensitive

post-infectious

teleological

```
% Philip Scuderi
% Stanford University
% Winter 2013
% CS 246
% Homework 2
% Question 1

function [] = HW2Q1
    AlexRowIndex = 500;
    LastColToCheck = 100;
    NumTopShows = 5;
    R = load('user-shows.txt');
    maxK = 19;

    Alex = load('alex.txt');
    shows = importdata('shows.txt');

    [m, n] = size(R);

    X = R*R';
    P = zeros(m,m);

    for i = 1:m
        P(i,i) = X(i,i);
    end

    SU = (P^-.5)*(X*(P^-.5));
    GammaU = SU*R;

    [sortedUValues, sortedUIndex] = sort(GammaU(AlexRowIndex,1:LastColToCheck),'descend');

    disp('User-User Collaborative Filtering:');
    for i = 1:NumTopShows
        disp(shows(sortedUIndex(i)));
        disp(sortedUValues(i));
    end

    Y = R'*R;
    Q = zeros(n,n);

    for i = 1:n
        Q(i,i) = Y(i,i);
    end

    SI = (Q^-.5)*(Y*(Q^-.5));
    GammaI = R*SI;

    [sortedIValues, sortedIIndex] = sort(GammaI(AlexRowIndex,1:LastColToCheck),'descend');

    disp('Item-Item Collaborative Filtering:');
    for i = 1:NumTopShows
        disp(shows(sortedIIndex(i)));
        disp(sortedIValues(i));
    end

    disp('User-User Collaborative Filtering:');
    for k = 1:maxK
        [precision] = PrecisionAtTopK(Alex, GammaU, k, AlexRowIndex, LastColToCheck);
        disp(['k = ', num2str(k), '; precision = ', num2str(precision)]);
    end

    disp('Item-Item Collaborative Filtering:');
    for k = 1:maxK
        [precision] = PrecisionAtTopK(Alex, GammaI, k, AlexRowIndex, LastColToCheck);
        disp(['k = ', num2str(k), '; precision = ', num2str(precision)]);
    end
end

function [precision] = PrecisionAtTopK(Alex, Gamma, k, AlexRowIndex, LastColToCheck)
    [sortedValues, sortedIndex] = sort(Gamma(AlexRowIndex,1:LastColToCheck),'descend');
```

```
    precision = 0.0;

    for i = 1:k
        if Alex(sortedIndex(i)) == 1;
            precision = precision + 1;
        end
    end

    precision = precision / k;
end
```

```
% Philip Scuderi
% Stanford University
% Winter 2013
% CS 246
% Homework 2
% Question 2

function [] = HW2Q2
    I = load('Matrix.txt');
    [U, S, V] = svd(I, 'econ');

    x = [];
    y = [];

    for k = 197:50:497
        tmpS = S;

        for i = (k+1):497
            tmpS(i,i) = 0;
        end

        x = [x; k]; %#ok<*AGROW>
        y = [y; r(U*tmpS*V')];
    end

    plot(x,y);
    xlabel('k');
    ylabel('r');
end


function [r] = r(I)
    numerator = 0.0;

    for i = 1:99
        for j = (i+1):100
            numerator = numerator + cossim(I(i,:), I(j,:));
        end
    end

    numerator = numerator / nchoosek(100,2);

    denominator = 0.0;

    for i = 1:496
        for j = (i+1):497
            denominator = denominator + cossim(I(i,:), I(j,:));
        end
    end

    denominator = denominator / nchoosek(497,2);

    r = numerator / denominator;
end


function [cossim] = cossim(di, dj)
    cossim = dot(di, dj) / (norm(di) * norm(dj));
end
```

```
// Philip Scuderi
// Stanford University
// Winter 2013
// CS 246
// Homework 2
// Question 4

import java.io.IOException;
import java.util.*;
import java.io.*;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class HW2Q4
{
	public static class Map1 extends Mapper<LongWritable, Text, IntWritable, Text>
	{
		protected static List<List<Double>> centroids = new ArrayList<List<Double>>();
		protected int k;

	  protected void setup(Context context) throws IOException, InterruptedException
	  {
		FileSystem fs = FileSystem.get(context.getConfiguration());

		Path cFile = new Path(context.getConfiguration().get("CFILE"));
		DataInputStream d = new DataInputStream(fs.open(cFile));
		BufferedReader reader = new BufferedReader(new InputStreamReader(d));

		String line;
		while ((line = reader.readLine()) != null)
		{
			StringTokenizer tokenizer = new StringTokenizer(line.toString());

			if (tokenizer.hasMoreTokens())
			{
				List<Double> centroid = new ArrayList<Double>(58);

				while (tokenizer.hasMoreTokens())
				{
					centroid.add(Double.parseDouble(tokenizer.nextToken()));
				}

				centroids.add(centroid);
			}
		}

		k = centroids.size();
	  }

	  private int IndexOfClosestCentroid(List<Double> p)
	  {
		int idxClosestCentoid = -1;
		double distanceToClosestCentroid = Double.MAX_VALUE;

		for (int i = 0; i < k; i++)
		{
			double d = Distance(centroids.get(i), p);
```

```
                        if (d < distanceToClosestCentroid)
                        {
                                idxClosestCentoid = i;
                                distanceToClosestCentroid = d;
                        }
                }

                return idxClosestCentoid;
        }

        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException
        {
                // read in a document (point in 58-dimensional space)

                List<Double> p = GetPoint(value.toString());

                int idxClosestCentoid = IndexOfClosestCentroid(p);

                context.write(new IntWritable(idxClosestCentoid), value);
        }
}

public static class Reduce1 extends Reducer<IntWritable, Text, IntWritable, Text>
{
        protected static List<String> newCentroids = new ArrayList<String>(10);

        public void reduce(IntWritable key, Iterable<Text> values, Context context) throws IOException, InterruptedException
        {
                List<Double> newCentroid = null;
                int numPoints = 0;
                List<List<Double>> points = new ArrayList<List<Double>>();

            for (Text value : values)
            {
                ++numPoints;
                List<Double> p = GetPoint(value.toString());
                points.add(p);

                if (newCentroid == null)
                {
                        // initialize the new centroid to the first element
                        newCentroid = new ArrayList<Double>(p);
                }
                else
                {
                        for (int i = 0; i < newCentroid.size(); i++)
                        {
                                newCentroid.set(i, newCentroid.get(i) + p.get(i));
                        }
                }
            }

            // now the newCentroid contains the sum of all the points
            // so to get the average of all the points, we need to
            // divide each entry by the total number of points

                for (int i = 0; i < newCentroid.size(); i++)
                {
                        newCentroid.set(i, newCentroid.get(i) / (double)numPoints);
                }

                // now create a string containing all the new centroid's coordinates

                String s = null;
                for (Double d : newCentroid)
                {
```

```
                        if (s == null)
                        {
                                s = d.toString();
                        }
                        else
                        {
                                s += " " + d.toString();
                        }
                }

                newCentroids.add(s);

                if (newCentroids.size() == 10)
                {
                        WriteNewCentroids(context);
                }

                double cost = 0.0;

            for (List<Double> p : points)
            {
                cost += Math.pow(Distance(newCentroid, p), 2.0);
            }

                // output the centroid ID and the centroid data
            context.write(key, new Text(Double.toString(cost)));
        }

        private static void WriteNewCentroids(Context context) throws IOException
        {
                FileSystem fs = FileSystem.get(context.getConfiguration());
                Path nextCFile = new Path(context.getConfiguration().get("NEXTCFILE"));
                DataOutputStream d = new DataOutputStream(fs.create(nextCFile, false));
                BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(d));

                for (String centroid : newCentroids)
                {
                        writer.write(centroid + "\n");
                }

                writer.close();
        }
    }

public static double Distance(List<Double> p1, List<Double> p2)
{
        double sumOfSquaredDifferences = 0.0;

        for (int i = 0; i < p1.size(); i++)
        {
                sumOfSquaredDifferences += Math.pow(p1.get(i) - p2.get(i), 2.0);
        }

        return Math.pow(sumOfSquaredDifferences, 0.5);
}

public static List<Double> GetPoint(String s)
{
        StringTokenizer tokenizer = new StringTokenizer(s);

        List<Double> p = new ArrayList<Double>(58);

        while (tokenizer.hasMoreTokens())
        {
                p.add(Double.parseDouble(tokenizer.nextToken()));
        }
```

```
        return p;
    }

        public static void main(String[] args) throws Exception
        {
                String inputDirectory = "/home/cs246/Desktop/HW2/input";
                String outputDirectory = "/home/cs246/Desktop/HW2/output";
                String centroidDirectory = "/home/cs246/Desktop/HW2/config";

                int iterations = 20;

          for (int i = 1; i <= iterations; i++)
          {
                  Configuration conf = new Configuration();

                  String cFile = centroidDirectory + "/c" + i + ".txt";
                  String nextCFile = centroidDirectory + "/c" + (i+1) + ".txt";
                  conf.set("CFILE", cFile);
                  conf.set("NEXTCFILE", nextCFile);

                  Job job = new Job(conf, "HW2Q4." + iterNum);
                  job.setJarByClass(HW2Q4.class);
                  job.setOutputKeyClass(IntWritable.class);
                  job.setOutputValueClass(Text.class);
                  job.setMapperClass(Map1.class);
                  job.setReducerClass(Reduce1.class);
                  job.setInputFormatClass(TextInputFormat.class);
                  job.setOutputFormatClass(TextOutputFormat.class);

                  FileInputFormat.addInputPath(job, new Path(inputDirectory));
                  FileOutputFormat.setOutputPath(job, new Path(outputDirectory + "/output" + iterNum));

                  job.waitForCompletion(true);
          }
        }
}
```

```
// Philip Scuderi
// Stanford University
// Winter 2013
// CS 246
// Homework 2
// Question 4

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace HW2Q4
{
    class HW2Q4
    {
        private static double Distance(List<double> p1, List<double> p2)
        {
            double sumOfSquaredDifferences = 0.0;

            for (int i = 0; i < p1.Count; i++)
                sumOfSquaredDifferences += Math.Pow(p1[i] - p2[i], 2.0);

            return Math.Pow(sumOfSquaredDifferences, 0.5);
        }

        private static int IndexOfClosestCentroid(List<List<double>> centroids, List<double> p)
        {
            int idxClosestCentoid = -1;
            double distanceToClosestCentroid = Double.MaxValue;

            for (int i = 0; i < centroids.Count; i++)
            {
                double d = Distance(centroids[i], p);

                if (d < distanceToClosestCentroid)
                {
                    idxClosestCentoid = i;
                    distanceToClosestCentroid = d;
                }
            }

            return idxClosestCentoid;
        }

        static void Main(string[] args)
        {
            List<List<double>> data = new List<List<double>>();
            List<string> vocab = new List<string>();
            List<List<double>> centroids = new List<List<double>>();

            using (StreamReader reader = new StreamReader("../../data.txt"))
            {
                String line;
                while ((line = reader.ReadLine()) != null)
                {
                    List<double> document = new List<double>();

                    foreach (string feature in line.Split(' '))
                        document.Add(double.Parse(feature));

                    data.Add(document);
                }
            }

            using (StreamReader reader = new StreamReader("../../vocab.txt"))
            {
                String line;
```

```
            while ((line = reader.ReadLine()) != null)
                vocab.Add(line);
        }

        using (StreamReader reader = new StreamReader("../../c1-21.txt"))
        {
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                List<double> centroid = new List<double>();

                foreach (string feature in line.Split(' '))
                    centroid.Add(double.Parse(feature));

                centroids.Add(centroid);
            }
        }

        int idx = IndexOfClosestCentroid(centroids, data[1]);
        List<double> closestCentroid = centroids[idx];

        Dictionary<int, double> dict = new Dictionary<int, double>();
        int i = 0;

        foreach (double feature in closestCentroid)
            dict.Add(++i, feature);

        i = 0;
        foreach (var feature in dict.OrderByDescending(key => key.Value))
        {
            Console.WriteLine(vocab[feature.Key - 1].PadRight(16) + " " + (i + 1) + " " + feature.Key
+ " " + feature.Value);

            if (++i == 6)
                break;
        }

        Console.ReadKey();
    }
  }
}
```