CS246: Mining Massive Data Sets

Assignment number: 2

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 and code are due at 5:00 PM on Scoryst and SNAP respectively. Failure to include the coversheet with you assignment will be penalized by 2 points. Each student will have a total of *two* free late periods. *One late period expires at the start of each class.* (Assignments are due on Thursdays, which means the first late period expires on the following Tuesday at 5:00 PM.) Once these late periods are exhausted, any assignments turned in late will be penalized 50% per late period. However, no assignment will be accepted more than one late period after its due date. (If an assignment is due to Thursday then we will not accept it after the following Thursday.)

Your name: Priyank Mathur
Email: priyankm@stanford.edu **SUNet ID:** priyankm

Collaborators: Emrah Budur, Shundan Xiao, Dibyajyoti Ghosh, Arkajyoti Misra

I acknowledge and accept the Honor Code.

(Signed) Priyank Mathur

Answer to Question 1.a

Let the cosine similarity of vectors u and v be represented as

$$cos(u, v) = \frac{\sum_i u_i * v_i}{||u|| * ||v||}$$

Let a row of a matrix be represented as where $R(i, :)$ and a column as where $R(:, j)$.

Now, consider the matrix calculated as $T = R^T R$, which will be an $n \times n$ matrix with each element as -

$$T_{1,1} = R(:, 1)R(:, 1)$$

$$T_{1,2} = R(:, 1)R(:, 2)$$

$$T_{3,2} = R(:, 3)R(:, 2)$$

$$\dots$$

We see that this matrix is symmetrical and each element represents a dot product between 2 columns (movies) of matrix R. These elements are also the numerators in the cosine similarity we defined above.

Let matrix $Q^{\frac{1}{2}}$ represent the element wise square root of a matrix Q. We know that matrices P and Q are diagonal matrices and inverting a diagonal matrix is equivalent to dividing 1 by each of its non-zero elements, i.e.

$$\begin{bmatrix} x_{11} & 0 & 0 & \dots & 0 \\ 0 & x_{22} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & x_{nn} \end{bmatrix}^{-1} = \begin{bmatrix} 1/x_{11} & 0 & 0 & \dots & 0 \\ 0 & 1/x_{22} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1/x_{nn} \end{bmatrix}$$

where x_{ii} are the magnitudes of rows or columns in R.

We know the following about any diagonal matrix X:

$$\begin{bmatrix} x_{11} & 0 & 0 & \dots & 0 \\ 0 & x_{22} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & x_{nn} \end{bmatrix} \times \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} x_{11}a_{11} & x_{11}a_{12} & \dots & x_{11}a_{1n} \\ x_{22}a_{21} & x_{22}a_{22} & \dots & x_{22}a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{nn}a_{n1} & x_{nn}a_{n2} & \dots & x_{nn}a_{nn} \end{bmatrix}$$

and

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \times \begin{bmatrix} x_{11} & 0 & 0 & \dots & 0 \\ 0 & x_{22} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & x_{nn} \end{bmatrix} = \begin{bmatrix} x_{11}a_{11} & x_{22}a_{12} & \dots & x_{nn}a_{1n} \\ x_{11}a_{21} & x_{22}a_{22} & \dots & x_{nn}a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{11}a_{n1} & x_{22}a_{n2} & \dots & x_{nn}a_{nn} \end{bmatrix}$$

Hence, the matrix $[Q^{\frac{1}{2}}]^{-1}T[Q^{\frac{1}{2}}]^{-1}$ represents a matrix where each element has numerator representing dot product of 2 columns and denominator is the product of magnitudes of same 2 columns. Therefore, each each element is cosine similarity between 2 columns.

$$S_I = [Q^{\frac{1}{2}}]^{-1}R^T R[Q^{\frac{1}{2}}]^{-1}$$

Similarly, consider

$$U = RR^T$$

which will be a $m \times m$ matrix with numerators representing the dot products between any 2 users. By using the results from above, we can say that

$$S_U = [P^{\frac{1}{2}}]^{-1}RR^T[P^{\frac{1}{2}}]^{-1}$$

Answer to Question 1.b

$r_{u,s} = \sum_{x \in users} cos(x, u) \times R_{x,s}$ represents the recommendation method.
e.g.

$$r_{1,1} = cos(1,1)R_{11} + cos(1,2)R_{21} + \ldots + cos(1,m)R_{m1}$$

From previous question, we know that

$$S_U = [P^{\frac{1}{2}}]^{-1} R R^T [P^{\frac{1}{2}}]^{-1}$$

is an $m \times m$ symmetric matrix representing the cosine similarity for all users vs. all other users. So,

$$r_{1,1} = S_{11}R_{11} + S_{12}R_{21} + \ldots + S_{1m}R_{m1} = r_{1,1}$$

Then let $A = S_U R$, where

$$A_{1,1} = S_{11}R_{11} + S_{12}R_{21} + \ldots + S_{1m}R_{m1} = r_{1,1}$$

$$A_{1,2} = S_{11}R_{12} + S_{12}R_{22} + \ldots + S_{1m}R_{m2} = r_{1,2}$$

$$A_{1,3} = S_{11}R_{13} + S_{12}R_{23} + \ldots + S_{1m}R_{m3} = r_{1,3}$$

$$\ldots$$

Therefore, each element of matrix A represents the metric r for that user.

$$\Gamma_U = S_U R = [P^{\frac{1}{2}}]^{-1} R R^T [P^{\frac{1}{2}}]^{-1} R$$

which is an $m \times n$ matrix.

Using the same approach as above,

$$\Gamma_I = R S_I = R [Q^{\frac{1}{2}}]^{-1} R^T R [Q^{\frac{1}{2}}]^{-1}$$

which is an $m \times n$ matrix.

Answer to Question 1.c

$$T = RR^T$$

From part 1.a we know that T is a symmetric matrix with each element representing the dot product of 2 rows of the matrix R. This can also be thought of as each element representing the dot product of 2 users.

Since each element in R is either 0 or 1 represents an edge in the graph, the dot product of 2 user rows represents the degree of overlap between the 2 vectors.

The entries along the main diagonal of T, (T_{ii}), represent the overlap between the user and itself. In other words, this represents the out degree of that users node in the bipartite user-to-item graph.

The other entries of T, (T_{ij}), represent the overlap between a user and another user. In other words, this represents the number of items that both these users liked, i.e. the number of that item nodes in the bipartite user-to-item graph that both users are connected to.

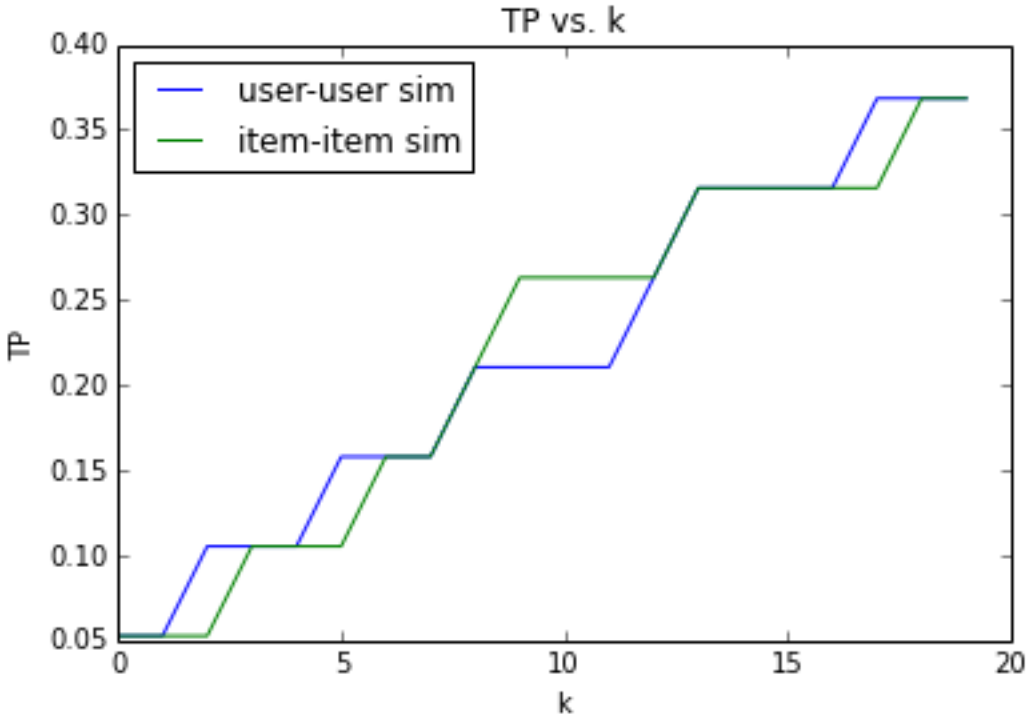
Answer to Question 1.d

Top movies using user-user similarity

"FOX 28 News at 10pm"	908.480053476
"Family Guy"	861.175999287
"2009 NCAA Basketball Tournament"	827.601295474
"NBC 4 at Eleven"	784.781958904
"Two and a Half Men"	757.601118102

Top movies using item-item similarity

"FOX 28 News at 10pm"	31.3647016783
"Family Guy"	30.0011417989
"NBC 4 at Eleven"	29.3967977734
"2009 NCAA Basketball Tournament"	29.2270015615
"Access Hollywood"	28.9712776741



From the above graph, we can not adequately distinguish between the per-

formance of the two methods. The rate of increase of true positive rate for $k \in [1, 19]$ for both the methods is comparable.

Answer to Question 2a

1.

We know that $C = A^T A$. We can prove C is symmetric by the following argument -

$C_{ij} = (A^T A)_{ij}$
$= \sum_k [A^T_{ik} A_{kj}]$
$= \sum_k [A_{ki} A^T_{jk}]$ as $A_{ij} = A^T_{ji}$
$= \sum_k [A^T_{jk} A_{ki}]$
$= (A^T A)_{ji}$
$= C_{ji}$

Similarly, $K = AA^T$ is also symmetric since $K_{ij} = K_{ji}$ by the above argument.

$K_{ij} = (AA^T)_{ij}$
$= \sum_k [A_{ik} A^T_{kj}]$
$= \sum_k [A^T_{ki} A_{jk}]$ as $A_{ij} = A^T_{ji}$
$= \sum_k [A_{jk} A^T_{ki}]$
$= (AA^T)_{ji}$
$= K_{ji}$

We know that -

- $A = USV^T$.
- C is a real, symmetric, and square $n \times n$ matrix which can be decomposed as

$$C = Q \Sigma Q^T \tag{1}$$

  where Q is an orthogonal matrix containing the eigenvectors of C as its columns and $\sum$ contains the eigenvalues of B.

$C = A^T A = (USV^T)^T USV^T$
$= VS^T U^T USV^T$
$= VS^T SV^T$ Since $U^T U = I$

$= V\Sigma V^T$ where $\Sigma = S^T S$

Comparing the result with (1), we know that Σ contains the eigen values of C and V has the eigenvectors of C.

$\Sigma = S^T S$ is an $n \times n$ matrix where S is a diagonal matrix containing σ_{kk} which are the singular values of A. Since $n \gg m$ we will only have m non zero eigenvalues.

Therefore Σ is also a diagonal matrix with eigenvalues (λ) as -

$$\lambda_{kk} = \sigma_{kk}^2$$

for $k \leq m$ and 0 for $k > m$. The matrix V has the eigenvectors for matrix C in its columns.

Using the same logic for $K = AA^T$, we get -
$K = AA^T = USV^T(USV^T)^T$
$= USV^T VS^T U^T$
$= USS^T U^T$ Since $V^T V = I$
$= U\Sigma U^T$ where $\Sigma = SS^T$

Therefore Σ is also a matrix with eigenvalues of K (λ) as -

$$\lambda_{kk} = \sigma_{kk}^2$$

for $k \leq m$. The matrix U has the eigenvectors for matrix K in its columns.

2.

We should use K for computing the singular values of A using matrix K's eigenvalue decomposition. K is a $m \times m$ matrix, and in this case $n \gg m$. Since the operation of calculating eigenvalues is expensive $O(d^3)$, we can use the smaller matrix K to get its eigenvalue decomposition and then calculate the singular values from it by simply using the square root of each eigenvalue.

3.

In part 3.a, we saw that matrix C has eigenvectors as columns of matrix V and eigenvalues as $S^T S$. That is

$$\begin{aligned} C &= VS^T SV^T \\ \implies A^T A &= VS^T SV^T \end{aligned}$$

Since $V^T = V^{-1}$ as V is orthogonal

$$\begin{aligned} \implies A^T A &= VS^T SV^{-1} \\ \implies A^T AV &= VS^T S \end{aligned}$$

Here S is a matrix containing singular values. We can obtain S by taking the square root of each of the eigenvalues.

We can write each column of V as

$$(A^T A)v_k = \lambda_k v_k$$

where λ_k is the eigenvalue which is equal to σ_k^2, the singular value. Hence we can extract the right singular matrix V and the singular values S from the above equations.

To find U, we can solve the below equation

$$\begin{aligned} A = USV^T = USV^-1 \\ \implies AV = US \end{aligned}$$

4.

In part 3.a, we saw that matrix K has eigenvectors as columns of matrix U and eigenvalues as SS^T. That is

$$\begin{aligned} K &= USS^TU^T \\ \implies AA^T &= USS^TU^T \end{aligned}$$

Since $U^T = U^{-1}$ as U is orthogonal

$$\begin{aligned} \implies AA^T &= USS^TU^{-1} \\ \implies AA^TU &= USS^T \end{aligned}$$

Here S is a matrix containing singular values. We can obtain S by taking the square root of each of the eigenvalues.

We can write for each column of U as

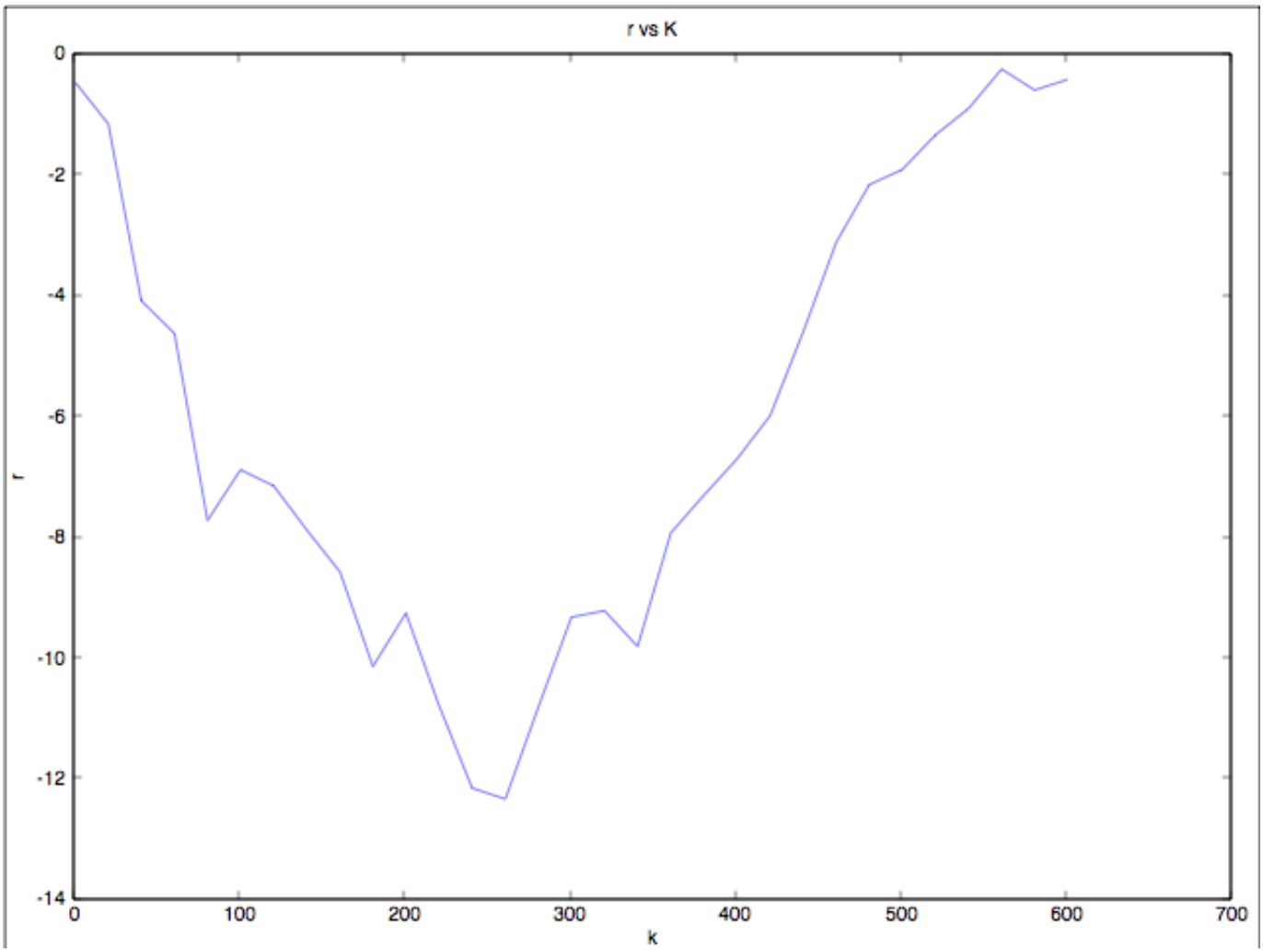
$$(AA^T)u_k = \lambda_k u_k$$

where λ_k is the eigenvalue which is equal to σ^2, the singular value. Hence we can extract the left singular matrix U and the singular values S from the above equations.

To find V, we can solve the below equation

$$\begin{aligned} A = USV^T = USV^-1 \\ \implies AV = US \\ \implies AVS^{-1} = U \end{aligned}$$

Answer to Question 2b



From the above curve we see that we achieve minimum r of -12.347 at k=261.

Answer to Question 2c

1.

To find W that minimizes $||A - XW||_F^2$, we take its derivative w.r.t. W and solve

$$\frac{\partial (A - XW)^T (A - XW)}{\partial W} = 0$$
$$\implies \frac{\partial (A^T A - 2W^T X^T A - W^T X^T X W)}{\partial W} = 0$$
$$\implies \frac{\partial (A^T A - 2W^T X^T A - W^T W)}{\partial W} = 0$$

Solving, we get $W = X^T A$

2.

By setting $X = \bar{U}$, we only consider the first k singular vectors of U. From this, A can be reconstructed as

$$A = \bar{U}\bar{S}\bar{V}^T$$

From part 1,

$$W = X^T A = \bar{U}^T \bar{U}\bar{S}\bar{V}^T$$

Since matrix $\bar{U}$ is orthogonal, we get

$$W = \bar{S}\bar{V}^T$$

3.

$min_{X^T X = I_K} ||A - XW||_F^2$

Replacing $W = X^T A$ from part 1, we get

$$min_{X^T X = I_K} ||A - XX^T A||_F^2$$
$$\implies min_{X^T X = I_K} ||A(I_k - XX^T)||_F^2$$
$$\implies min_{X^T X = I_K} ||A(X^T X - XX^T)||_F^2$$

Answer to Question 3.a

To prove -

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

We know that

$$||u - v||^2 = (u - v)^T (u - v) = ||u||^2 + ||v||^2 - 2u.v$$

Let S be a set of n points $x_1, x_2 \ldots x_n$. Expanding LHS of the equation, we get -

$$||x_1 - z||^2 + ||x_2 - z||^2 + \ldots + ||x_n - z||^2$$
$$-||x_1 - c(S)||^2 - ||x_2 - c(S)||^2 - \ldots - ||x_n - c(S)||^2$$

$$= ||x_1||^2 + ||z||^2 - 2x_1.z + ||x_2||^2 + ||z||^2 - 2x_2.z + \ldots + ||x_n||^2 + ||z||^2 - 2x_n.z$$
$$-||x_1||^2 - ||c(S)||^2 + 2x_1.c(S) - ||x_2||^2 - ||c(S)||^2 + 2x_2.c(S) + \ldots - ||x_n||^2 - ||c(S)||^2 + 2x_n.c(S)$$

$$= n||z||^2 - n||c(S)||^2 + 2c(S)(x_1 + x_2 + \ldots + x_n) - 2z(x_1 + x_2 + \ldots + x_n)$$

We know that

$$c(S) = \frac{(x_1 + x_2 + \ldots + x_n)}{n}$$

$$\implies (x_1 + x_2 + \ldots + x_n) = n \times c(S) \tag{1}$$

Using (1), we get

$$= n||z||^2 - n||c(S)||^2 + 2nc(S).c(S) - 2nzc(S)$$
$$= n||z||^2 + n||c(S)||^2 - 2nzc(S)$$
$$= n.||c(S) - z||^2 = |S|.||c(S) - z||^2$$

Answer to Question 3.b

From the previous part, we know that

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

Here, if $z = c(S)$,

$$\sum_{x \in S} ||x - z||^2 - \sum_{x \in S} ||x - c(S)||^2 = 0 \tag{1}$$

If $z \neq c(S)$

$$|S|.||c(S) - z||^2 > 0$$

$$\implies \sum_{x \in S} ||x - z||^2 - \sum_{x \in S} ||x - c(S)||^2 > 0 \tag{2}$$

From (1) and (2), we get

$$\sum_{x \in S} ||x - z||^2 \geq \sum_{x \in S} ||x - c(S)||^2 \tag{3}$$

That is, distance of a set of points from a query point is minimum if the query point is the centroid of the set of data.

Let $C_{t1}, C_{t2} \ldots C_{tk}$ be the k clusters at end of t^{th} iteration with $c_{t1}, c_{t2} \ldots c_{tk}$ centroids. Similarly for iteration $(t+1)$, clusters are $C_{(t+1)1}, C_{(t+1)2} \ldots C_{(t+1)k}$ with centroids $c_{(t+1)1}, c_{(t+1)2} \ldots c_{(t+1)k}$.

Let $Cost(C_1, C_2 \ldots C_k, c_1, c_2 \ldots c_k)$ represent the cost while using clusters $C_1, C_2 \ldots C_k$ and centroids as $c_1, c_2 \ldots c_k$.

Now,

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

We know

- $||x - c||^2$ will be minimum when x belongs to the cluster represented by centroid c.

- In each iteration's step (2), a point is placed in the cluster of its nearest centroid. This reduces the error as per the definition of ϕ above, that is

$$Cost(C_{(t+1)1}, C_{(t+1)2} \dots C_{(t+1)k}, c_{t1}, c_{t2} \dots c_{tk}) \leq Cost(C_{t1}, C_{t2} \dots C_{tk}, c_{t1}, c_{t2} \dots c_{tk}) \tag{4}$$

- In each iteration's step (3), the new centorid is calculated as the average of the cluster points, so it is closer to the data points within the cluster, also contributing by reducing error. We can see this by using equation (3) above, where z is the old centroid and c(S) is the new centroid.

$$\begin{aligned} Cost(C_{(t+1)1}, C_{(t+1)2} \dots C_{(t+1)k}, c_{(t+1)1}, c_{(t+1)2} \dots c_{(t+1)k}) \leq \\ Cost(C_{(t+1)1}, C_{(t+1)2} \dots C_{(t+1)k}, c_{t1}, c_{t2} \dots c_{tk}) \end{aligned} \tag{5}$$

Therefore, iteration $(t+1)$ should have better or atleast as good cluster assignments as iteration t.

Hence, from (3), (4) and (5), we can deduce that $\phi^t \geq \phi^{t+1}$.

Answer to Question 3.c

Let $C_{t1}, C_{t2} \dots C_{tk}$ be the k clusters in t^{th} iteration with $c_{t1}, c_{t2} \dots c_{tk}$ centroids. Similarly for iteration $(t+1)$, clusters are $C_{(t+1)1}, C_{(t+1)2} \dots C_{(t+1)k}$ with centroids $c_{(t+1)1}, c_{(t+1)2} \dots c_{(t+1)k}$.

Let $Cost(C_1, C_2 \dots C_k, c_1, c_2 \dots c_k)$ represent the cost while using clusters $C_1, C_2 \dots C_k$ and centroids as $c_1, c_2 \dots c_k$.

We know that -

- From 3.a) that the distance of a set of points from a query point is minimum if the query point is the centroid of the set of data.
- From 3.b) we know that the error function ϕ is a monotonically decreasing function, i.e. $\phi^t \geq \phi^{t+1}$.
- For a fixed k, the global optimal cluster assignment for the data set will give us the lowest possible cost.

Assume we break ties during cluster assignment consistently, for e.g. by assigning to cluster centroid with the lower id (index). Given these facts, for any data set and k (number of clusters), there are only a finite number of cluster assignments.

As we saw in 3.b), step (2) and (3) of the k-means algorithm reduce the overall cost. Hence, cluster assignments will keep changing to reduce ϕ.

$$Cost(C_{(t+1)1}, C_{(t+1)2} \dots C_{(t+1)k}, c_{t1}, c_{t2} \dots c_{tk}) \leq Cost(C_{t1}, C_{t2} \dots C_{tk}, c_{t1}, c_{t2} \dots c_{tk}) \tag{1}$$

and

$$\begin{aligned} Cost(C_{(t+1)1}, C_{(t+1)2} \dots C_{(t+1)k}, c_{(t+1)1}, c_{(t+1)2} \dots c_{(t+1)k}) \leq \\ Cost(C_{(t+1)1}, C_{(t+1)2} \dots C_{(t+1)k}, c_{t1}, c_{t2} \dots c_{tk}) \end{aligned} \tag{2}$$

This is continued until we do not modify the assignments anymore, since we either -

- reached an assignment state where each point is already assigned to its nearest centroid from previous iteration.
- or we cycled through all possible cluster assignments and we reach the optimum cost in the last iteration.

At this iteration, using (1) and (2), we can say that ϕ converges to the finite value corresponding to this optimal assignment state. Note that this optimal assignment is not guaranteed to be globally optimum.

Extending the same logic as above, we can argue that k-means algorithm will converge either when -

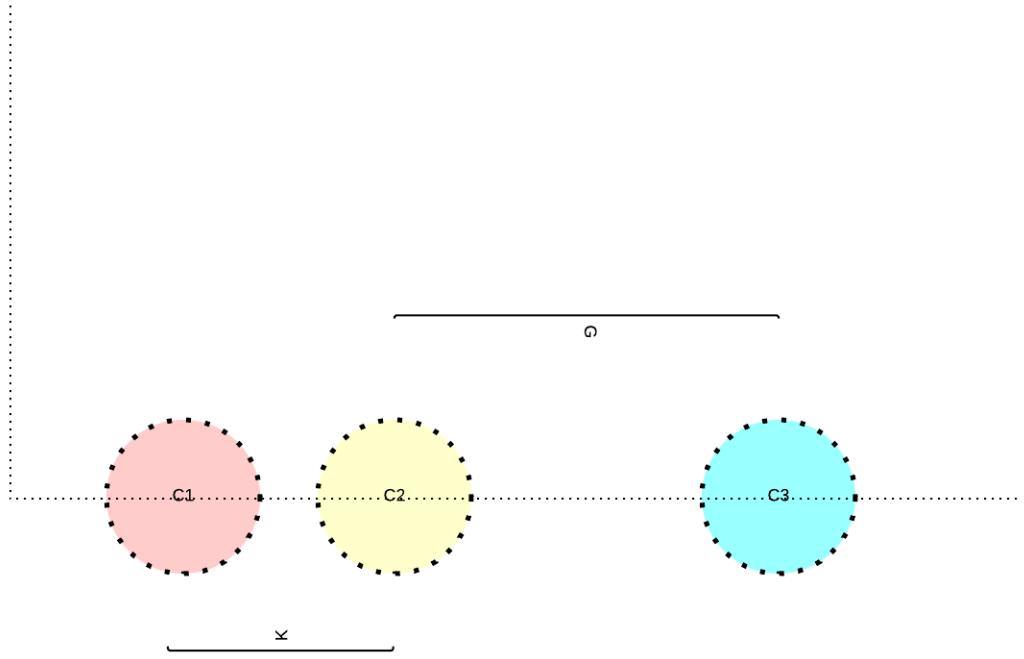
- we reach an assignment state where each point is already assigned to its nearest centroid from previous iteration.
- or we cycled through all possible cluster assignments and we reach the optimum state in the last iteration.

Answer to Question 3.d

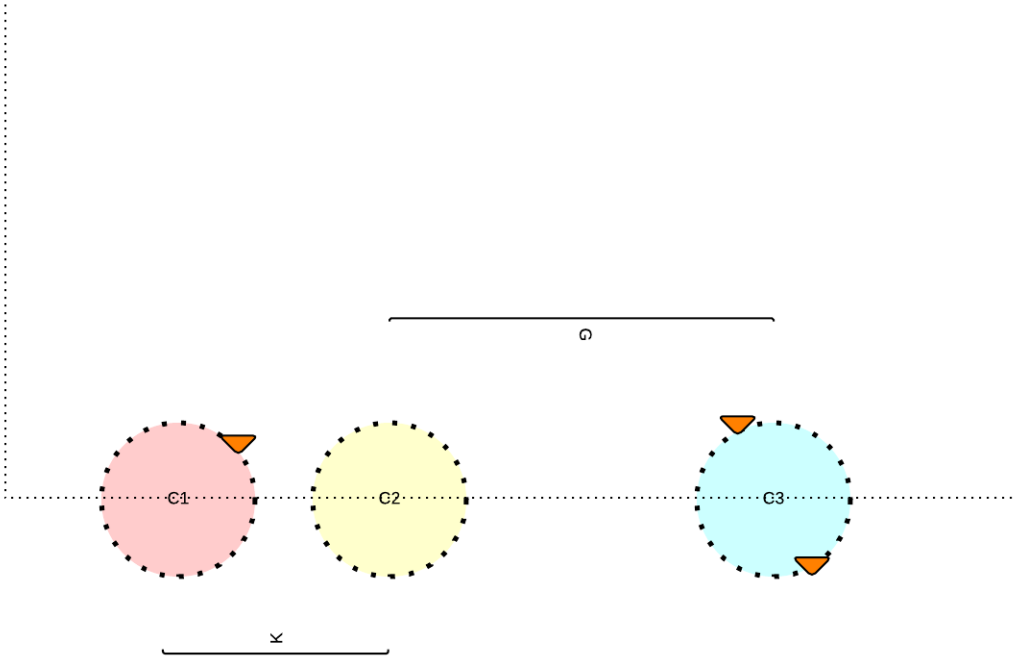
Data distribution Let the actual data be present in the below format. There are 3 true circular clusters with the points lying on the circumference of the circle. In this setting, each circle will ideally have its centroid at the center of the circle. Let each of the 3 clusters have n points. Also for the sake of analysis, we will assume the radius (r) of the 3 circles to be 1.
We will also assume that the distance $G \gg K$

Thus, the optimal cost is equal to -

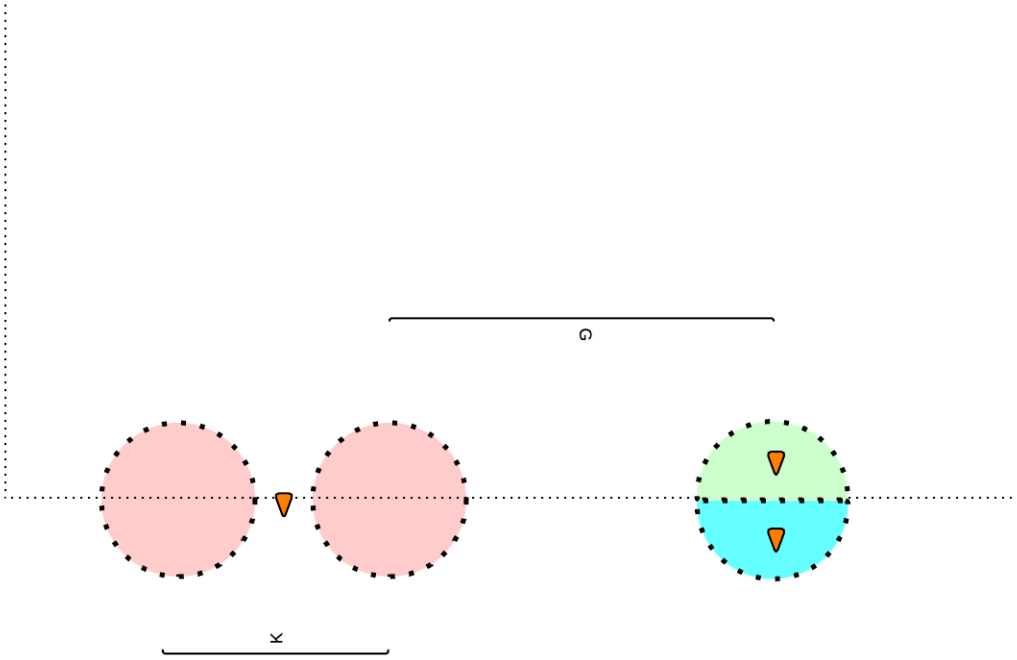
$$\phi_{Opt} = 3 \times n^2 r = 3n^2$$



Now, lets assume a bad initialization as shown in diagram below. The farthest cluster gets 2 centroid allocations.



In this case, the k-means algorithm will converge with the following cluster assignments.



From 3.a, we know that

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

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

where z are the centroids in the bad initialization.

Therefore, for cluster C1 and C2, the new cost will be

$$\phi_{C1} + \phi_{C2} = n^2 + n(\frac{K}{2})^2 + n^2 + n(\frac{K}{2})^2$$

$$\phi_{C1} + \phi_{C2} = 2n^2 + n\frac{K^2}{2}$$

Cluster 3 is actually divided into 2 new clusters and hence its cost will reduce, therefore the new cost for C3 is -

$$\phi_{C3} = n^2 - 2 \times \frac{n}{2}(\frac{4}{3\pi})^2$$

Since the centroid of a semi circle is located at $(0, \frac{4r}{3\pi})$.

$$\phi_{C3} = n^2 - n\frac{16}{9\pi^2}$$

Therefore, total new cost =

$$\phi_{new} = 2n^2 + n\frac{K^2}{2} + n^2 - n\frac{16}{9\pi^2}$$

$$\implies \phi_{new} = 3n^2 + n(\frac{K^2}{2} - \frac{16}{9\pi^2})$$

$$\implies \phi_{new} = 3n^2 + n(\frac{K^2}{2} - 0.18)$$

Hence, the ratio of costs is

$$r = \frac{\phi_{new}}{\phi_{Opt}} = \frac{3n^2 + n(\frac{K^2}{2} - 0.18)}{3n^2}$$

$$r = 1 + \frac{1}{3n}(\frac{K^2}{2} - 0.18)$$

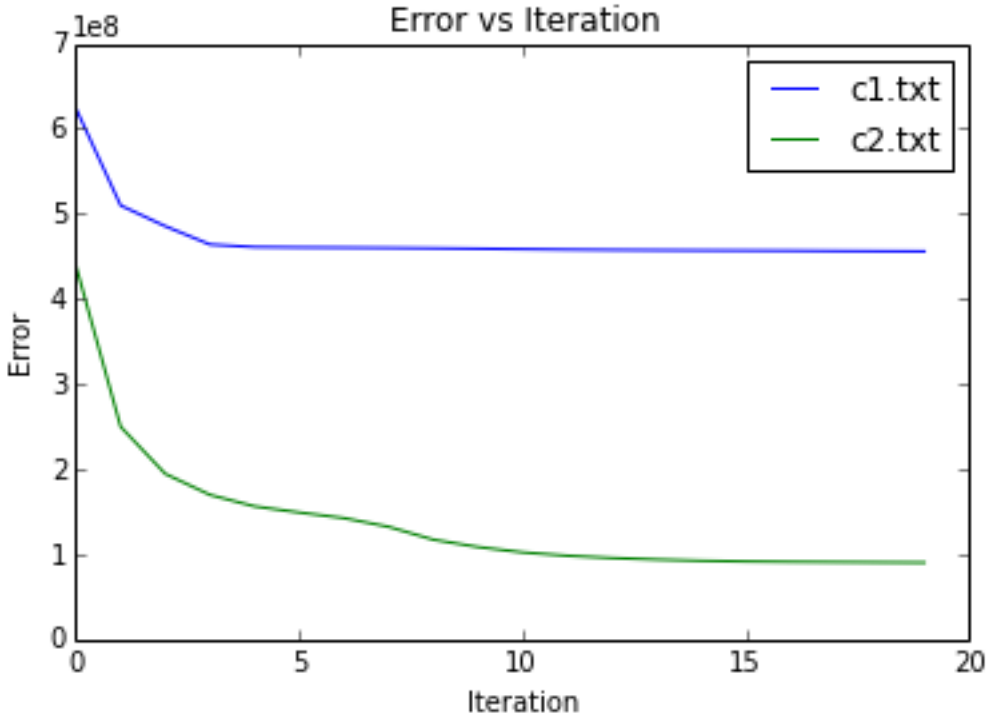
We know that for true clusters C1 and C2 to be different, $K > 2 \times radius > 2$. Hence the ratio r is always greater than 1.

We can make r arbitrarily greater than 1 by increasing distance K and G, as long as $G \gg K$.

Answer to Question 4a

Code at the end of the file.

Answer to Question 4b



As we can see, the cost decreases in each iteration till the algorithm converges. We also see that random initialization works much worse than clusters chosen in c2.txt.

Answer to Question 4c

As we can see from the above graph, the random cluster initialization performs worse than using initial centroids as computed for c2.txt. We saw in question 3 that due to certain bad initialization, k-means can reach convergence at a sub optimal value which seems to be the case with c1.txt.
Using initial centroids that are far away from each other gives us a higher chance of getting clusters that are governed by data distribution, since we get centroids that represent even the farthest points.

Change in cost (ϕ) for c1.txt $= \frac{459018805.0-623658084.0}{623658084.0} \times 100 = -26.39$

That is a decrease of 26.39%.

Change in cost (ϕ) for c2.txt $= \frac{108545091.0-438745437.0}{438745437.0} \times 100 = -75.26$

That is a decrease of 75.26%.

As we see, c2.txt performs much better job of decreasing the error.

Answer to Question 4d

Top 5 words for document 2 are -

- instillation
- methoxyfenozide
- sodium-sensitive
- post-infectious
- teleological

Code for Q1

```
# coding: utf-8

# In[1]:

import numpy as np
from numpy import linalg as LA
import matplotlib.pyplot as plt


# In[2]:

path = 'q1-dataset/user-shows.txt'
path_alex = 'q1-dataset/alex.txt'
path_show = 'q1-dataset/shows.txt'


# In[3]:

shows = open(path_show).read().split('\n')


# In[4]:

alex = np.loadtxt(path_alex, delimiter=' ')


# In[5]:

R = np.loadtxt(path, delimiter=' ')


# In[6]:

P = np.diag(R.sum(1))


# In[7]:

Q = np.diag(R.sum(0))


# In[8]:

T = R.dot(R.T)


# In[9]:
```

```
P_sqrt = np.power(P, 0.5)

# In[10]:

Q_sqrt = np.power(Q, 0.5)

# In[11]:

P_sq_inv = LA.inv(P_sqrt)

# In[12]:

Q_sq_inv = LA.inv(Q_sqrt)

# In[12]:



# In[ ]:

SU = P_sq_inv.dot(T).dot(P_sq_inv)

# In[36]:

GU = SU.dot(R)

# In[40]:

GU_sub = GU[:, :100]

# In[94]:

unn_idx = np.argsort(GU_sub[499,:])[::-1][:5]

# In[95]:

unn_scr = GU_sub[499, unn_idx]

# In[65]:
```

```
# In[96]:

for ix in unn_idx:
    print shows[ix], GU_sub[499, ix]


# In[85]:




# In[88]:

SI = Q_sq_inv.dot(R.T.dot(R)).dot(Q_sq_inv)


# In[91]:

GI = R.dot(SI)


# In[93]:

GI_sub = GI[:, :100]


# In[97]:

inn_idx = np.argsort(GI_sub[499,:])[::-1][:5]


# In[98]:

inn_scr = GI_sub[499, inn_idx]


# In[100]:

for ix in inn_idx:
    print shows[ix], GI_sub[499, ix]


# In[103]:




# In[203]:
```

```
ii_rec = np.argsort(GI_sub[499,:])[::-1]
uu_rec = np.argsort(GU_sub[499,:])[::-1]
ttl = alex[:100].sum()
uu_rates = []
ii_rates = []

for k in range(1,21):
    uu_idxs = uu_rec[:k]
    ii_idxs = ii_rec[:k]

    uu_liked = alex[uu_idxs].sum()
    ii_liked = alex[ii_idxs].sum()

    #print uu_liked, ii_liked

    uu_rates.append(uu_liked/ttl)
    ii_rates.append(ii_liked/ttl)


# In[204]:

plt.plot(np.arange(20), uu_rates, label='user-user sim')
plt.plot(np.arange(20), ii_rates, label='item-item sim')
plt.xlabel("k")
plt.ylabel("TP")
plt.title("TP vs. k")
plt.legend(loc=2)


# In[ ]:
```

Code for Q2

```
%% predictor: get agreement for given X
function [outputs] = predictor(X, y)

   modelX = X(41:640, :);
   modelY = y(41:640, :);
   model = sum(bsxfun(@times, modelX, modelY));

   testX = X(1:40, :);
   testY = y(1:40, :);

   mult = repmat(model, size(testY), 1);
   coeff = testX * mult';

   coeff = coeff(:,1);
   prodCoeff = coeff .* testY;

   outputs = sum(prodCoeff) * -1;


%% driver: main driver code
function [op] = driver(X, y)
   k = [1:20:601];

   [U,S,V] = svd(X, 'econ');
   n = size(S,1);

   perf = [];

   for idx = k,
      newX = U(:, [1:idx]);

      res = predictor(newX, y);
      perf = [perf res];
   end

   op = perf;

end


% plot the data
op = driver(X, y);
plot([1:20:601],op);
xlabel('k');
ylabel('Agreement');
title('Agreement vs K');
```

Code for Q4

```
package edu.stanford.cs246.kmeans;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
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;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class KMeans extends Configured implements Tool {
   public static void main(String[] args) throws Exception {
      System.out.println(Arrays.toString(args));
      int res = ToolRunner.run(new KMeans(), args);

      System.exit(res);
   }

   @Override
   public int run(String[] args) throws Exception {
      System.out.println(Arrays.toString(args));

      String opBase = args[1];
      String jobBase = "KMeansPass";
      ArrayList<Long> errors = new ArrayList<Long>();

      Configuration conf = new Configuration();
      conf.set("centroids", args[2]);
      conf.setInt("pass", 1);
```

```
    Job job = new Job(conf, "KMeansPass1");
    job.setJarByClass(KMeans.class);

    job.setMapOutputKeyClass(IntWritable.class);
    job.setMapOutputValueClass(Text.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(NullWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(opBase+"1"));

    job.waitForCompletion(true);
    errors.add(job.getCounters().findCounter("Errors", "pass1").getValue());

    for (int pass=2; pass<=20; pass++) {
      conf = new Configuration();
      conf.set("centroids", opBase+(pass-1));
        conf.setInt("pass", pass);

        job = new Job(conf, jobBase+pass);
        job.setJarByClass(KMeans.class);

        job.setMapOutputKeyClass(IntWritable.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);

        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(opBase+pass));

        job.waitForCompletion(true);
        errors.add(job.getCounters().findCounter("Errors", "pass"+pass).getValue());
    }

    System.out.println(errors);

    return 0;
}
```

```
public static class Map extends Mapper<LongWritable, Text, IntWritable, Text> {
  ArrayList<String> centroids = new ArrayList<String>();
  ArrayList<Vector> centroidVectors = new ArrayList<Vector>();
  static IntWritable pass = new IntWritable();
  static IntWritable cluster = new IntWritable();
  static Text op = new Text();

  public void setup(Context context) throws IOException, InterruptedException {
     String strLine;

     Configuration conf = context.getConfiguration();
     pass.set(conf.getInt("pass", 0));

     FileSystem fs = FileSystem.get(conf);
     Path centroidFile = new Path(conf.get("centroids"));

     if (fs.isFile(centroidFile)) {
        FSDataInputStream in = fs.open(centroidFile);
        BufferedReader bufread = new BufferedReader (new InputStreamReader (in));

        while ((strLine = bufread.readLine ()) != null) {
             centroids.add(strLine);
          }
          in.close ();
     }
     else {
        FileStatus[] fileStats = fs.listStatus(centroidFile);

        for (int k=0; k < fileStats.length; k++) {
           Path f = fileStats[k].getPath();

           FSDataInputStream in = fs.open(f);
           BufferedReader bufread = new BufferedReader (new InputStreamReader (in));

           while ((strLine = bufread.readLine ()) != null) {
                centroids.add(strLine);
             }
             in.close ();
        }
     }


      for (String str : centroids) {
       centroidVectors.add(new Vector(str));
     }
  }

   @Override
   public void map(LongWritable key, Text value, Context context)
          throws IOException, InterruptedException {
```

```
    String point = value.toString();
    Vector v = new Vector(point);

    double minDist = Double.MAX_VALUE;
    int minIdx = -1;

    for (int j=0; j<centroidVectors.size(); j++) {
       double dist = v.dist(centroidVectors.get(j));

       if (dist < minDist) {
          minDist = dist;
          minIdx = j;
       }
    }

    cluster.set(minIdx);
    context.getCounter("Errors", "pass"+pass).increment((long) Math.pow(minDist, 2));
    op.set(point);

    context.write(cluster, op);

  }
}

public static class Reduce extends Reducer<IntWritable, Text, Text, NullWritable> {
  static final Text op = new Text();
  static IntWritable pass = new IntWritable();

  public void setup(Context context) throws IOException, InterruptedException {
     Configuration conf = context.getConfiguration();
     pass.set(conf.getInt("pass", 0));
  }

   @Override
   public void reduce(IntWritable key, Iterable<Text> values, Context context)
          throws IOException, InterruptedException {

     double cnt = 0;
     Vector vSum = null;

     for(Text val: values) {
        String pt = val.toString();
        Vector ptVec = new Vector(pt);

        if (cnt == 0) {
           vSum  = Vector.getZeroVector(ptVec.coords.size());
        }

        cnt++;
        vSum.add(ptVec);
     }
```

```
        vSum.divideByN(cnt);

        op.set(vSum.toString());
        context.write(op, NullWritable.get());
      }
   }
}
```

```
package edu.stanford.cs246.kmeans;

import java.util.ArrayList;

public class Vector {

   ArrayList<Double> coords = new ArrayList<Double>();

   static Vector getZeroVector(int size) {
      Vector v = new Vector();

      for (int i=0; i<size; i++) {
         v.coords.add(0.0);
      }

      return v;
   }

   public Vector(){}

   public Vector(String pts) {
      String[] coordinates = pts.split(" ");

      for (String str : coordinates) {
         double d = Double.parseDouble(str);
         coords.add(d);
      }
   }

   public void add(Vector v2) {

      for (int i=0; i<v2.coords.size(); i++) {
         this.coords.set(i, this.coords.get(i) + v2.coords.get(i));
      }
   }

   public void divideByN(double n) {

      for (int i=0; i<this.coords.size(); i++) {
         this.coords.set(i, this.coords.get(i)/n);
      }
   }
```

```
    public double dist(Vector v2) {
        double dist = 0;

        for (int i=0; i<v2.coords.size(); i++) {
            dist += Math.pow(this.coords.get(i) - v2.coords.get(i), 2);
        }

        return Math.pow(dist, 0.5);
    }

    public String toString() {
        StringBuffer str = new StringBuffer();

        for (int k=0; k<this.coords.size()-1; k++)
            str.append(this.coords.get(k) + " ");

        str.append(this.coords.get(this.coords.size()-1));

        return str.toString();
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Vector v1 = new Vector("0 1 2 3 4");
        Vector v2 = new Vector("0 1 1 3 3");

        System.out.println(v1.coords);
        System.out.println(v2.coords);
        System.out.println(v1.dist(v2));
        System.out.println(v2.dist(v1));

        v1.add(v2);
        v2.divideByN(3);

        System.out.println(v1.coords);
        System.out.println(v2.coords);
        System.out.println(v1.dist(v2));

        System.out.println(v1.toString());
    }

}

// Python code to generate graph and get best labels
test1 = [6.23658084E8, 5.0986063E8, 4.85478413E8, 4.63994716E8, 4.60967022E8,
    4.6053561E8, 4.60310797E8, 4.60001258E8, 4.59568241E8, 4.59018805E8, 4.5848837E8,
```

```
    4.57941919E8, 4.5755567E8, 4.57287833E8, 4.57048292E8, 4.56889922E8, 4.56701323E8,
    4.56401899E8, 4.56175503E8, 4.55984591E8]
test2 = [4.38745437E8, 2.49801643E8, 1.94492505E8, 1.69802592E8, 1.56293459E8,
    1.4909191E8, 1.42506236E8, 1.32301622E8, 1.17168711E8, 1.08545091E8, 1.02234922E8,
    9.8275677E7, 9.5627978E7, 9.3791002E7, 9.2374815E7, 9.1539349E7, 9.1043261E7,
    9.0749936E7, 9.0467844E7, 9.0214141E7]
plt.plot(test1, label="c1.txt")
plt.plot(test2, label="c2.txt")
plt.legend()
plt.xlabel("Iteration")
plt.ylabel("Error")
plt.title("Error vs Iteration")

// Get best labels from centroid coordinates
centStr = "0.17496894409937883 0.26409937888198765 0.48478260869565226
    0.4175776397515528 0.3142236024844721 0.15795031055900627 0.07546583850931678
    0.21559006211180123 0.33819875776397507 0.6326086956521739 0.12447204968944099
    0.7521118012422355 0.15298136645962745 0.12142857142857141 0.19863354037267086
    0.249689440993789 0.16366459627329188 0.34677018633540374 1.7215527950310567
    0.19913043478260872 1.37223602484472 0.3104968944099379 0.21465838509316768
    0.193416149068323 0.41155279503105574 0.13223602484472052 0.02267080745341615
    0.021304347826086957 0.007018633540372672 0.024472049689440997 0.014906832298136646
    0.0011801242236024845 0.09024844720496894 0.002360248447204969 0.07192546583850934
    0.02708074534161491 0.10708074534161494 0.007018633540372672 0.02372670807453416
    0.025838509316770186 0.005962732919254659 0.04366459627329193 0.01173913043478261
    0.0063975155279503105 0.04291925465838507 0.031242236024844716 0.0027950310559006213
    0.01925465838509317 0.06939130434782607 0.13108074534161487 0.028931677018633542
    0.6866832298136648 0.21185093167701857 0.06648447204968946 10.619267080745331
    187.82608695652175 868.944099378882 0.7453416149068323"
cent = centStr.split()
cent = map(double, cent)
idx = (np.argsort(cent))[::-1]
vocab = open('vocab.txt').read()
words = vocab.split("\n")
cnt = 1
for i in idx:
    print words[i]
    cnt += 1
```