CS246: Mining Massive Data Sets **Winter 2016**

Problem Set 2

Due 11:59pm February 4, 2016

Only one late period is allowed for this homework (11:59pm 2/9).

General Instructions

Submission instructions: These questions require thought but do not require long answers. Please be as concise as possible. You should submit your answers as a writeup in PDF format via GradeScope and code via the Snap submission site.

Submitting writeup: Prepare answers to the homework questions into a single PDF file and submit it via http://gradescope.com. Make sure that the answer to each question is on a *separate page*. On top of each page write the number of the question you are answering. Please find the cover sheet and the recommended templates located here:
http://web.stanford.edu/class/cs246/homeworks/hw2/submission_template_hw2.tex
http://web.stanford.edu/class/cs246/homeworks/hw2/submission_template_hw2.pdf
http://web.stanford.edu/class/cs246/homeworks/hw2/submission_template_hw2.docx

Not including the cover sheet in your submission will result in a 2-point penalty. It is also important to tag your answers correctly on Gradescope. We will deduct 5/N points for each incorrectly tagged subproblem (where N is the number of subproblems). This means you can lose up to 5 points for incorrect tagging.

Submitting code: Upload your code at http://snap.stanford.edu/submit. Put all the code for a single question into a single file and upload it.

Questions

1 Recommendation Systems (35 points)

Consider a user-item bipartite graph where each edge in the graph between user U to item I, indicates that user U likes item I. We also represent the ratings matrix for this set of users and items as R, where each row in R corresponds to a user and each column corresponds to an item. If user i likes item j, then $R_{i,j} = 1$, otherwise $R_{i,j} = 0$. Also assume we have m users and n items, so matrix R is $m \times n$.

Let's define a matrix P, $m \times m$, as a diagonal matrix whose i-th diagonal element is the degree of user node i, *i.e.* the number of items that user i likes. Similarly, a matrix Q, $n \times n$,

is a diagonal matrix whose i-th diagonal element is the degree of item node i or the number of users that liked item i. See figure below for an example.

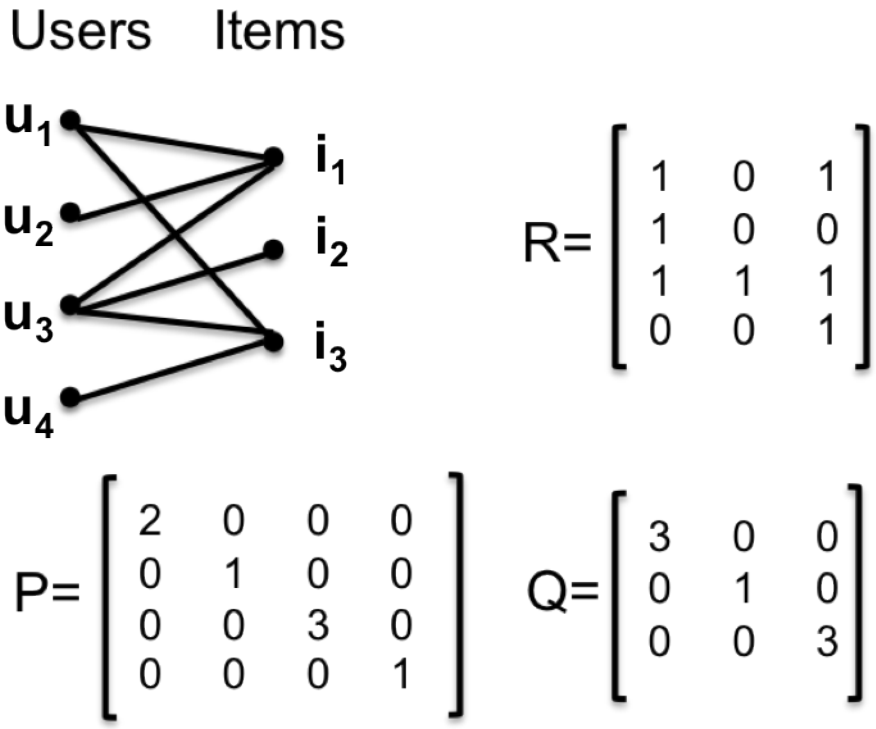


Figure 1: User-Item bipartite graph.

(a) [4 points]

Define the non-normalized user similarity matrix $T = R * R^T$. Explain the meaning of T_{ii} and T_{ij} $(i \neq j)$, in terms of bipartite graph structures (See Figure 1) (e.g. node degrees, path between nodes, etc.).

★ **SOLUTION:** In the user-item bipartite graph, T_{ii} equals the degree of user_i. Since $T_{ii} = \sum_{j=1}^{n} R_{ij} * (R^T)_{ji} = \sum_{j=1}^{n} R_{ij}^2 = \sum_{j=1}^{n} R_{ij}$. Since R_{ij} is 0 or 1, so $T_{ii} = \text{degree}(\text{user}_i)$.

$T_{ij}(i \neq j)$ is the number of paths between user_i and user_j with the length of 2, it also represents the number of items that they both like. $T_{ij} = \sum_{k=1}^{n} R_{ik} * R_{jk}$, $R_{ik} * R_{jk}(\neq 0)$ means there exists a 2-step path starts from user_i to user_j via item_k.

Cosine Similarity: Recall that the cosine similarity of two vectors u and v is defined as:

$$cos\text{-}sim(u,v) = \frac{u \cdot v}{\|u\|\|v\|}$$

(b) [8 points]

Let's define the *item similarity matrix*, S_I, $n \times n$, such that the element in row i and column j is the cosine similarity of *item* i and *item* j which correspond to column i and column j of the matrix R. Express S_I in terms of R, P and Q. Your answer should be an operation on the matrices, in particular you should not define each coefficient of S_I individually.

Repeat the same question for *user similarity matrix*, S_U where the element in row i and column j is the cosine similarity of *user* i and *user* j which correspond to row i and row j of the matrix R.

Your answer should show how you derived the expressions.

(Note: To make the element-wise square root of a matrix, you may write it as matrix to the power of $\frac{1}{2}$.)

★ **SOLUTION:** We denote by $R_{i\cdot}$ the i-th row of R, and by $R_{\cdot j}$ its j-th column.

We know that the vector of an item is defined by one column is R. Furthermore, the norm of, say, item_i is defined by $\sqrt{\sum_{k=1}^{m} R_{ki}^2}$. The sum is in fact the number of users that like this item (because R_{ki} can be either 0 or 1, we have $R_{ki} = R_{ki}^2$). So the norm of item_i is in fact $\sqrt{Q_{ii}}$. We thus have:

$$\begin{aligned}\text{cos-sim}(\text{item}_i, \text{item}_j) &= \frac{R_{\cdot i} \cdot R_{\cdot j}}{\sqrt{Q_{ii}}\sqrt{Q_{jj}}} \\ &= \frac{\sum_{k=1}^{m} R_{ki} R_{kj}}{\sqrt{Q_{ii}}\sqrt{Q_{jj}}}\end{aligned}$$

Now, the matrix Q is diagonal, and its diagonal coefficients are all non-zero (otherwise, some items are liked by nobody, and we might as well remove them, because they are useless and because the angle they form with other items would be ill-defined), so we can denote by $Q^\star$ the diagonal matrix whose diagonal coefficients are defined by $Q^\star_{ii} = Q_{ii}^{-1/2}$. We then have that:

$$\begin{aligned}\left(Q^\star R^T R Q^\star\right)_{ij} &= \sum_{k,l,m} Q^\star_{ik} R_{kl} R_{lm} Q^\star_{mj} \\ &= \sum_{l} Q^\star_{ii} R_{li} R_{lj} Q^\star_{jj} \text{ (because } Q^\star \text{ is diagonal)} \\ &= \sum_{l} \frac{R_{li} R_{lj}}{\sqrt{Q_{ii}}\sqrt{Q_{jj}}} \\ &= \frac{R_{\cdot i} \cdot R_{\cdot j}}{\sqrt{Q_{ii}}\sqrt{Q_{jj}}}\end{aligned}$$

So, the matrix S_I can be expressed in terms of Q and R:

$$S_I = Q^\star R^T R Q^\star$$

To compute a similar expression for S_u, we notice that (R, Q, S_I) and (R^T, P, S_u) play similar roles. Indeed, the relation "user_u likes item_i" can be put backward into "item_i is liked by user_u", which is equivalent to switching users and items, ie to transpose the matrix R. Thus, S_u is given by:

$$S_u = P^\star R R^T P^\star$$

with $P^\star$ being a diagonal matrix whose coefficients are defined by $P^\star_{ii} = P_{ii}^{-1/2}$.

(c) [8 points]

The recommendation method using user-user collaborative filtering for user u, can be described as follows: for all items s, compute $r_{u,s} = \Sigma_{x \in users} \text{cos-sim}(x, u) * R_{xs}$ and recommend the k items for which $r_{u,s}$ is the largest.

Similarly, the recommendation method using item-item collaborative filtering for user u can be described as follows: for all items s, compute $r_{u,s} = \Sigma_{x \in items} R_{ux} * \text{cos-sim}(x, s)$ and recommend the k items for which $r_{u,s}$ is the largest.

Let's define the recommendation matrix, Γ, $m \times n$, such that $\Gamma(i, j) = r_{i,j}$. Find Γ for both item-item and user-user collaborative filtering approaches, in terms of R, P and Q.

Your answer should show how you derived the expressions.

★ **SOLUTION:** For the user-user collaborative filtering recommendation, we have that:

$$\begin{aligned}
\Gamma_{ij} &= r_{ij} \\
&= \sum_{x \in \text{users}} \text{cos-sim}(x, i) \times R_{x,j} \\
&= \sum_{x \in \text{users}} \text{cos-sim}(i, x) \times R_{x,j} \\
&= \sum_{x=1}^{m} (S_u)_{i,x} \times R(x, j) \\
&= \big(S_u R\big)_{ij} \\
&= \big(P^\star R R^T P^\star R\big)_{ij}
\end{aligned}$$

Thus,

$$\Gamma = P^\star R R^T P^\star R$$

Similarly, for the item-item collaborative filtering recommendation, we have that:

$$\begin{aligned}\Gamma_{ij} &= r_{ij} \\ &= \sum_{x \in \text{items}} R_{i,x} \times \text{cos-sim}(x, j) \\ &= \sum_{x=1}^{n} R_{i,x} \times (S_I)_{x,j} \\ &= \left(RS_I\right)_{ij} \\ &= \left(RQ^\star R^T RQ^\star\right)_{ij}\end{aligned}$$

Thus,

$$\Gamma = RQ^\star R^T RQ^\star$$

(d) [15 points]

In this question you will apply these methods to a real dataset. The data contains information about TV shows. More precisely, for 9985 users and 563 popular TV shows, we know if a given user watched a given show over a 3 month period.

Download the dataset[1] on http://snap.stanford.edu/class/cs246-data/hw2-q1-dataset.zip.

The ZIP file contains:

- `user-shows.txt` This is the ratings matrix R, where each row corresponds to a user and each column corresponds to a TV show. $R_{ij} = 1$ if user i watched the show j over a period of three months. The columns are separated by a space.
- `shows.txt` This is a file containing the titles of the TV shows, in the same order as the columns of R.

We will compare the user-user and item-item collaborative filtering recommendations for the 500^{th} user of the dataset. Let's call him Alex.

In order to do so, we have erased the first 100 entries of Alex's row in the matrix, and replaced them by 0s. This means that we don't know which of the first 100 shows Alex has watched. Based on Alex's behaviour on the other shows, we will give Alex recommendations on the first 100 shows. We will then see if our recommendations match what Alex had in fact watched.

[1] The original data is from Chris Volinksy's website at http://www2.research.att.com/~volinsky/DataMining/Columbia2011/HW/HW6.html.

- Compute the matrices P and Q.
- Using the formulas found in part (b), compute Γ for the user-user collaborative filtering. Let S denote the set of the first 100 shows (the first 100 columns of the matrix). From all the TV shows in S, which are the five that have the highest similarity scores for Alex? What are their similarity scores? In case of ties between two shows, choose the one with smaller index. Do not write the index of the TV shows, write their names using the file `shows.txt`.
- Compute the matrix Γ for the movie-movie collaborative filtering. From all the TV shows in S, which are the five that have the highest similarity scores for Alex? In case of ties between two shows, choose the one with smaller index. Again, hand in the names of the shows and their similarity score.

Alex's original row is given in the file `alex.txt`. For a given number k, **the true positive rate at top-k** is defined as follows: using the matrix Γ computed previously, compute the top-k TV shows in S that are most similar to Alex (break ties as before). The true positive rate is the number of top-k TV shows that were watched by Alex in reality, divided by the total number of shows he watched in the held-out 100 shows.

- Plot the true positive rate at top-k (defined above) as a function of k, for $k \in [1, 19]$, with predictions obtained by the user-user collaborative filtering.
- On the same figure, plot the true positive rate at top-k as a function of k, for $k \in [1, 19]$, with predictions obtained by the item-item collaborative filtering.

What to submit:

(i) Interpretation of T_{ii} and T_{ij} [for 1(a)]

(ii) Expression of S_I and S_U in terms of R, P and Q and accompanying explanation [for 1(b)]

(iii) Expression of Γ in terms of R, P and Q and accompanying explanation [for 1(c)]

(iv) The answer to this question should include the followings: [for 1(d)]

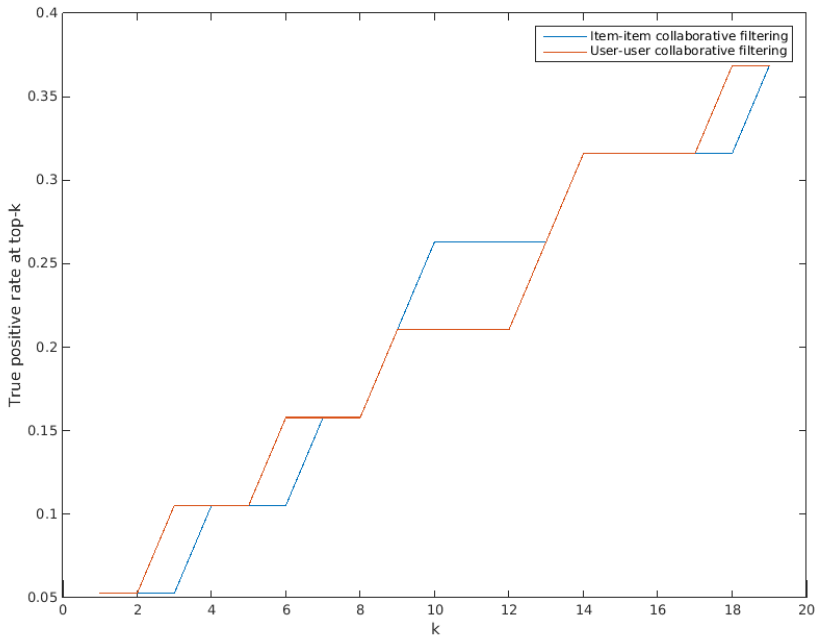
- The five TV shows that have the highest similarity scores for Alex for the user-user collaborative filtering
- The five TV shows that have the highest similarity scores for Alex for the item-item collaborative filtering
- The graph of the true positive rate at top-k for both the user-user and the item-item collaborative filtering
- Upload the source code via the SNAP electronic submission website

★ **SOLUTION:** user_user shows:

- FOX 28 News at 10pm
- Family Guy
- 2009 NCAA Basketball Tournament
- NBC 4 at Eleven
- Two and a Half Men

item_item shows:

- FOX 28 News at 10pm
- Family Guy
- NBC 4 at Eleven
- 2009 NCAA Basketball Tournament
- Access Hollywood



★ **SOLUTION:** Comments: open question. e.g. There is no significant advantage to any of the methods. Or Precision decreases both for user-user and item-item as k increases.

2 Singular Value Decomposition and Principal Component Analysis (25 points)

In this problem we will explore the relationship between two of the most popular dimensionality-reduction techniques, SVD and PCA at a basic conceptual level. Before we proceed with the question itself, let us briefly recap the SVD and PCA techniques and a few important observations:

- First, recall that the eigenvalue decomposition of a *real*, *symmetric*, and *square matrix* B (of size $d \times d$) can be written as the following product:

  $$B = Q\Lambda Q^{\mathrm{T}}$$

  where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ contains the eigenvalues of B (which are always real) along its main diagonal and Q is an orthogonal matrix containing the eigenvectors of B as its columns.

- Principal Component Analysis (PCA): Given a data matrix M (of size $p \times q$), PCA involves the computation of the eigenvectors of MM^T or M^TM. The matrix of these eigenvectors can be thought of as a rigid rotation in a high dimensional space. When you apply this transformation to the original data, the axis corresponding to the principal eigenvector is the one along which the points are most spread out, More precisely, this axis is the one along which the variance of the data is maximized. Put another way, the points can best be viewed as lying along this axis, with small deviations from this axis. Likewise, the axis corresponding to the second eigenvector (the eigenvector corresponding to the second-largest eigenvalue) is the axis along which the variance of distances from the first axis is greatest, and so on.

- Singular Value Decomposition (SVD): SVD involves the decomposition of a data matrix M (of size $p \times q$) into a product: $U\Sigma V^T$ where U (of size $p \times k$) and V (of size $q \times k$) are column-orthonormal matrices[2] and Σ (of size $k \times k$) is a diagonal matrix. The entries along the diagonal of Σ are referred to as singular values of M. The key to understanding what SVD offers is in viewing the r columns of U, Σ, and V as representing concepts that are hidden in the original matrix M.

For answering the questions below, let us define a matrix M (of size $p \times q$) and let us assume this matrix corresponds to a dataset with p data points and q dimensions.

(a) [3 points]

Are the matrices MM^T and M^TM symmetric, square and real? Explain.

[2] A matrix $U \in \mathbb{R}^{p \times q}$ is column-orthonormal if and only if $U^TU = I$ where I denotes the identity matrix

★ **SOLUTION:** Yes, both the matrices are symmetric, square and real.

- Symmetric: $(MM^T)^T = MM^T$ and $(M^TM)^T = M^TM$
- Square: When you multiply a matrix of size $p \times q$ by its transpose of size $q \times p$, you end up with a square matrix of size $p \times p$.
- Real: Given that M is real, the product of M and its transpose will also be real.

(b) [5 points]

Prove that the eigenvalues of MM^T are the same as that of M^TM. Are their eigenvectors the same?

★ **SOLUTION:** Let e denote the eigenvector of MM^T and let the corresponding eigen value be λ i.e $MM^T(e) = \lambda(e)$. Multiply both sides of equation by M^T and the result is $M^TMM^Te = M^T\lambda e$ which can be reduced to $M^TM(M^Te) = \lambda(M^Te)$. Therefore, the eigenvalue of $M^TM = \lambda$ too but the eigenvector is M^Te which is different from the eigenvector of MM^T which is e.

(c) [2 points]

Given that we now understand certain properties of M^TM, write an expression for M^TM in terms of Q, Q^T and Λ where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d)$ 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?
Hint: Check the definition of eigenvalue decomposition provided in the beginning of the question to see if it is applicable.

★ **SOLUTION:** $M^TM = Q\Lambda Q^T$

(d) [5 points]

SVD decomposes the matrix M into the product $U\Sigma V^T$ where U and V are column-orthonormal and Σ is a diagonal matrix. Given that $M = U\Sigma V^T$, write a simplified expression for M^TM in terms of V, V^T and Σ?

★ **SOLUTION:** $M^T M = (U\Sigma V^T)^T U\Sigma V^T$
$= V\Sigma^T U^T U\Sigma V^T$
$= V\Sigma^T \Sigma V^T$
$= V\Sigma^2 V^T$

(e) [10 points]

In this question, let us experimentally test if SVD decomposition of M actually provides us the eigenvectors (PCA dimensions) of $M^T M$. We strongly recommend students to use Python and suggested functions for this exercise.[3] Initialize matrix M as follows:

$$M = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix}$$

- 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? *Note: Make sure that the first element of the returned array* Σ *has a greater value than the second element.*

```
>>> U
array([[ 0.27854301,  0.5       ],
       [ 0.27854301, -0.5       ],
       [ 0.64993368,  0.5       ],
       [ 0.64993368, -0.5       ]])
>>> S
array([ 7.61577311,  1.41421356])
>>> Vh
array([[ 0.70710678,  0.70710678],
       [-0.70710678,  0.70710678]])
```

- Compute the eigenvalue decomposition of $M^T M$ (*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 *Evals*. What are the values of *Evals* and *Evecs* (after the sorting and re-arranging process)?

```
>>> P
array([  2.,  58.])
>>> Q
array([[-0.70710678,  0.70710678],
       [ 0.70710678,  0.70710678]])
```

- 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 *Evals* (after the sorting and re-arranging process) produced by eigenvalue decomposition? If so, what is it?
  Note: The function scipy.linalg.svd returns V^T (not V).

  ★ **SOLUTION:** V is equivalent to the matrix of eigenvectors if we reorder the columns as per the ordering of the singular values.

- 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.
  Note: The entries along the diagonal of Σ (part (e)) are referred to as singular values of M. The eigenvalues of M^TM are captured by the diagonal elements in Λ (part (d))

  ★ **SOLUTION:** The singular values of M are square roots of the eigenvalues of M^TM

What to submit:

(i) Written solutions to questions 2(a) to 2(e) with explanations wherever required

(ii) Upload the code via Snap submission site

3 Understanding the Effect of Merging Strategies on Clustering (10 points)

We learned in class that different merging strategies result in different clustering outcomes. In this problem, we explore how various merging strategies impact the clustering outcomes by means of an example. Consider a small 1-dimensional dataset comprising of 9 data points: $\mathcal{P} = \{p_1, p_2, \cdots, p_9\} = \{0.68, 0.72, 0.79, 0.87, 0.92, 0.99, 1.01, 1.22, 1.45\}$. This dataset captures the smoking habits of 9 different individuals. Each of the numbers represent a quantity

[3] Other implementations of SVD and PCA might give slightly different results. Besides, you will just need fewer than five python commands to answer this entire question

called the *Normalized Smoking Rate (NSR)* which is defined as the ratio of the mean number of times an individual i smokes per day to the mean number of times an average individual in the U.S. smokes. This implies that if p_i>1.0 for some individual i, then i smokes more than an average person in the U.S. In our dataset, there are three such individuals who smoke more than an average person the U.S. A survey done by the health department indicated that any individual with an NSR score >1.0 is at a high risk of lung cancer. So, our dataset captures two sub-populations: people with low risk of lung cancer (p_1 to p_6) and people with high risk of lung cancer (p_7 to p_9).

We were so enthusiastic about seeing the results of the clustering on this dataset that we already ran an unknown clustering algorithm on this data. The resulting clusters are shown in Figure 2. As can be seen, there are three clusters each of which comprises of three datapoints. Our goal is to group all the datapoints into two clusters. In order to achieve that, you should now decide which pair of the three existing clusters in Figure 2 should be merged so that we are left with two clusters of the data.

Consider the following merging strategies to decide which pair of the existing clusters you would like to merge:

- Centroid based merging: Merge the pair of clusters with the smallest distance between their centroids.
- Closest member based merging: Take the distance between two clusters to be the minimum of the distances between any two points, one chosen from each cluster. Merge the pair of clusters with minimum distance.

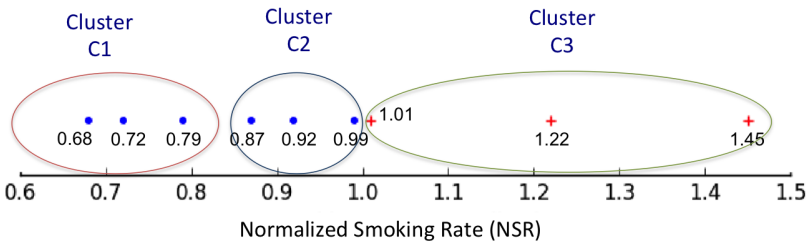


Figure 2: Clusters already obtained from an unknown clustering algorithm

Note: Centroid of a cluster is computed as the average of the all the datapoints in that cluster (rounded to two decimal places). Through out this question, whenever we use the word distance, we are referring to the Euclidean distance metric.

(a) [4 points]

Which pair of clusters shown in Figure 2 will be merged by each of the strategies outlined above (Centroid based merging, Closest member based merging)? Which merge strategy

will leave behind two clusters such that one of the clusters corresponds to low risk of lung cancer and the other represents high risk of lung cancer?

★ **SOLUTION:** Centroid of cluster $C1 = 0.73$ Centroid of cluster $C2 = 0.93$ Centroid of cluster $C3 = 1.23$ Centroid based merging will merge $C1$ and $C2$.

Min distance($C1$,$C2$) = 0.08 Min distance($C2$,$C3$) = 0.02 Min distance($C1$,$C3$) = 0.22

Closest member based merging will merge $C2$ and $C3$.

(b) [4 points]

A merge strategy is called *stable* w.r.t some point p_i and some existing set of clusters S_c if assuming that p_i is absent from the dataset (as well as from its currently assigned cluster) does not change the outcome of the strategy. Of all the merge strategies outlined above, which ones are *stable* w.r.t the point $p_6 = 0.99$ and the existing set of clusters shown in Figure 2 i.e which of the merge strategies are unaffected if we assume that p_6 does not exist in the data and also assume that cluster $C2$ (in Figure 2) just has two data points p_4 and p_5.
Note: When applying centroid based merging, centroid of cluster $C2$ should be computed by assuming that p_6 does not exist.

★ **SOLUTION:** Centroid of cluster $C1 = 0.73$ Centroid of cluster $C2 = 0.90$ Centroid of cluster $C3 = 1.23$ Centroid based merging will merge $C1$ and $C2$. This is stable.

Min distance($C1$,$C2$) = 0.08 Min distance($C2$,$C3$) = 0.09 Min distance($C1$,$C3$) = 0.22

Closest member based merging will merge $C1$ and $C2$. This is unstable.

(c) [2 points]

If we give you a new NSR dataset which is noisy i.e the NSR scores have been miscalculated for about 5% of individuals in this new dataset, would you prefer using a stable merge strategy instead of an unstable one? Explain your reasons.

★ **SOLUTION:** Yes, it is preferable to use stable merge because it is less sensitive to outliers as we saw in the previous example.

What to submit:

(i) Pair of clusters merged for each of the strategies [for 3(a)]

(ii) Indicate the strategy that creates two groups: low risk and high risk of lung cancer [for 3(a)]

(iii) Indicate which strategy (or strategies) are stable [for 3(b)]

(iv) State which strategy would you prefer and why [for 3(c)]

4 k-means on MapReduce (30 points)

Note: This problem requires substantial computing time. Don't start it at the last minute.

This problem will help you understand the nitty gritty details of implementing clustering algorithms on Hadoop. In addition, this problem will also help you understand the impact of using various distance metrics and initialization strategies in practice. Let us say we have a set $\mathcal{X}$ of n data points in the d-dimensional space $\mathbb{R}^d$. Given the number of clusters k and the set of k centroids $\mathcal{C}$, we now proceed to define various distance metrics and the corresponding cost functions that they minimize.

Euclidean distance Given two points A and B in d dimensional space such that $A = [a_1, a_2 \cdots a_d]$ and $B = [b_1, b_2 \cdots b_d]$, the Euclidean distance between A and B is defined as:

$$||a - b|| = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \tag{1}$$

The corresponding cost function ϕ that is minimized when we assign points to clusters using the Euclidean distance metric is given by:

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

Manhattan distance Given two random points A and B in d dimensional space such that $A = [a_1, a_2 \cdots a_d]$ and $B = [b_1, b_2 \cdots b_d]$, the Manhattan distance between A and B is defined as:

$$|a - b| = \sum_{i=1}^{d} |a_i - b_i| \tag{3}$$

The corresponding cost function ψ that is minimized when we assign points to clusters using the Manhattan distance metric is given by:

$$\psi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} |x - c| \tag{4}$$

Iterative k-Means Algorithm: We learned the basic k-Means algorithm in class which is as follows: k centroids are initialized, each point is assigned to the nearest centroid and the centroids are recomputed based on the assignments of points to clusters. In practice, the above steps are run for several iterations. We present the resulting iterative version of k-Means in Algorithm 1.

Algorithm 1 Iterative k-Means Algorithm

```
1: procedure ITERATIVE k-MEANS
2:     Select k points as initial centroids of the k clusters.
3:     for iterations := 1 to MAX_ITER do
4:         for each point p in the dataset do
5:             Assign point p to the cluster with the closest centroid
6:         end for
7:         for each cluster c do
8:             Recompute the centroid of c as the mean of all the data points assigned to c
9:         end for
10:    end for
11: end procedure
```

Iterative k-Means clustering on Hadoop: Implement iterative k-means using MapReduce where a single step of MapReduce completes one iteration of the k-means algorithm. So, to run k-means for i iterations, you will have to run a sequence of i MapReduce jobs.

Please use the dataset at http://snap.stanford.edu/class/cs246-data/hw2-q4-kmeans.zip for this problem.

The zip has 4 files:

1. `data.txt` contains the dataset which has 4601 rows and 58 columns. Each row is a document represented as a 58 dimensional vector of features. Each component in the vector represents the importance of a word in the document.

2. `c1.txt` contains k initial cluster centroids. These centroids were chosen by selecting $k = 10$ random points from the input data.

3. `c2.txt` contains initial cluster centroids which are as far apart as possible. (You can do this by choosing 1st centroid c1 randomly, and then finding the point c2 that is farthest from c1, then selecting c3 which is farthest from c1 and c2, and so on).

Set number of iterations (`MAX_ITER`) to 20 and number of clusters k to 10 for all the experiments carried out in this question.

*Hint about **job chaining**:*

We need to run a sequence of Hadoop jobs where the output of one job will be the input for the next one. There are multiple ways to do this and you are free to use any method you are

comfortable with. One simple way to handle such a multistage job is to configure the output path of the first job to be the input path of the second and so on.

The following pseudo code demonstrates job chaining.

```
var inputDir
var outputDir
var centroidDir

for i in no-of-iterations (
     Configure job here with all params
     Set job input directory = inputDir
     Set job output directory = outputDir + i
     Run job
     centroidDir =  outputDir + i
)
```

You will also need to share the location of the centroid file with the mapper. There are many ways to do this and you can use any method you find suitable. One way is to use the Hadoop Configuration object. You can set it as a property in the Configuration object and retrieve the property value in the Mapper setup function.

For more details see :

1. http://hadoop.apache.org/docs/r1.0.4/api/org/apache/hadoop/conf/Configuration.html#set(java.lang.String,java.lang.String)
2. http://hadoop.apache.org/docs/r1.0.4/api/org/apache/hadoop/conf/Configuration.html#get(java.lang.String)
3. http://hadoop.apache.org/docs/r1.0.4/api/org/apache/hadoop/mapreduce/Mapper.html#setup(org.apache.hadoop.mapreduce.Mapper.Context)

(a) Exploring initialization strategies with Euclidean distance [15 pts]

1. **[10 pts]** Using the Euclidean distance (refer to Equation 1) as the distance measure, compute the cost function $\phi(i)$ (refer to Equation 2) for every iteration i. This means that, for your first MapReduce job iteration, you'll be computing the cost function using the initial centroids located in one of the two text files. Run the k-means on `data.txt` using `c1.txt` and `c2.txt`. Generate a graph where you plot the cost function $\phi(i)$ as a function of the number of iterations i=1..20 for `c1.txt` and also for `c2.txt`.

   (Hint: Note that you do not need to write a separate MapReduce job to compute $\phi(i)$. You can just incorporate the computation of $\phi(i)$ into the Mapper/Reducer.)

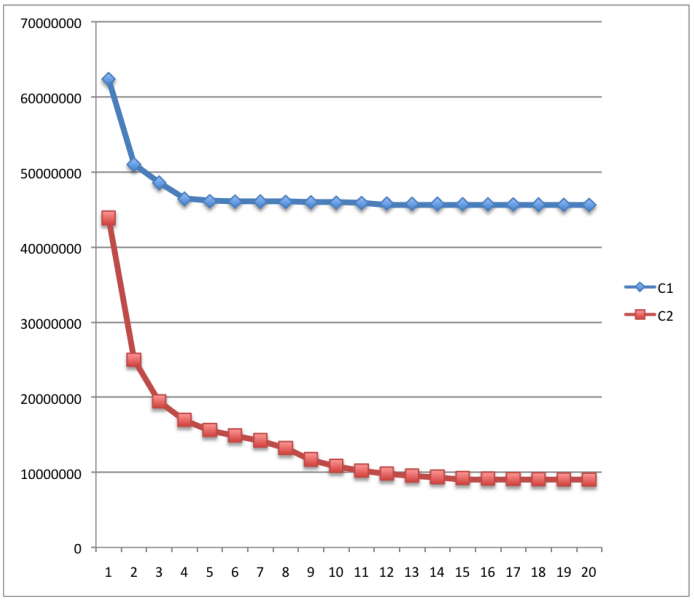


Figure 3: Cost vs Iteration

2. **[5 pts]** What is the percentage change in cost after 10 iterations of the K-Means algorithm when the cluster centroids are initialized using `c1.txt` vs. `c2.txt` and the distance metric being used is Euclidean distance? Is random initialization of k-means using `c1.txt` better than initialization using `c2.txt` in terms of cost $\phi(i)$? Explain your reasoning.

★ **SOLUTION:** c1 improves by 26% after 10 iterations.
c2 improves by 75% after 10 iterations.
Reasoning:
1) Farthest initialization is better than Random initialization because it avoids overlap/splitting of clusters.
2) Farthest initialization minimizes the impact of outliers contributing high costs due to squared cost function.
3) If the data is indeed distributed in clear clusters that are far from each other (not true in general), then using farthest initialization will result in lower costs.

(b) Exploring initialization strategies with Manhattan distance [15 pts]

1. **[10 pts]** Using the Manhattan distance metric (refer to Equation 3) as the distance measure, compute the cost function $\psi(i)$ (refer to Equation 4) for every iteration i. This means that, for your first MapReduce job iteration, you'll be computing the cost function using the initial centroids located in one of the two text files. Run the k-means on `data.txt` using `c1.txt` and `c2.txt`. Generate a graph where you plot the

cost function $\psi(i)$ as a function of the number of iterations i=1..20 for `c1.txt` and also for `c2.txt`.

(Hint: This problem can be solved in a similar manner to that of part (a))

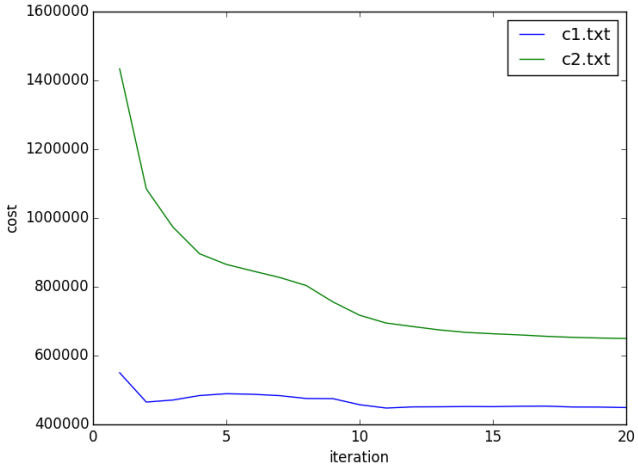


Figure 4: Cost vs Iteration

2. **[5 pts]** What is the percentage change in cost after 10 iterations of the K-Means algorithm when the cluster centroids are initialized using `c1.txt` vs. `c2.txt` and the distance metric being used is Manhattan distance? Is random initialization of k-means using `c1.txt` better than initialization using `c2.txt` in terms of cost $\psi(i)$? Explain your reasoning.

★ **SOLUTION:** c1 improved by 19%, and c2 improved by 52%. c1 is better than c2 in this case. This demonstrates the fact that controlled initialization is not always better than random initialization. One possible explanation is that random initialization can get lucky to be close to the global optima.

What to submit:

(i) Upload the code for 4(a) and 4(b) to Snap submission site

(ii) A plot of cost vs. iteration for two initialization strategies for 4(a)

(iii) Percentage improvement values and your explanation for 4(a)

(iv) A plot of cost vs. iteration for two initialization strategies for 4(b)

(v) Percentage improvement values and your explanation for 4(b)