CS246: Mining Massive Datasets

Assignment number: 2

Submission time: _____ **and date:** _____

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 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 9:30am.) 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: Blaž Sovdat
Email: blaz.sovdat@gmail.com **SUNet ID:** _____

Collaborators: _____

I acknowledge and accept the Honor Code.

(Signed) _____

(For CS246 staff only)

Late days: 0 1

Section	Score
1	
2	
3	
4	
Total	

Comments:

1 Recommendation systems

Suppose we have a set U of m users and an set I of n items. Define rating matrix $R \in \mathbb{R}^{m \times n}$ as $R_{i,j} := 1$ if user i likes items j and set $R_{i,j} := 0$ otherwise. Now define diagonal $m \times m$ matrix P, with i-th diagonal element defined as the degree of the user node i, i.e., as the number of items user i likes. Similarly define diagonal $n \times n$ matrix Q with i-th diagonal element defined as the degree of the item node i, i.e., as the number of users that like the item i.

(a) Let S_I be the item similarity $n \times n$ matrix, defined as $S_{i,j} := v_i v_j / (\|v_i\| \|v_j\|)$, where v_i and v_j are vectors corresponding to i-th and j-th item, respectively. Our goal is to express S_I in terms of R, P, and Q. Note that $(R^{\mathrm{T}}R)_{i,j} = v_i v_j$ and that $(Q^{-1/2})_{i,i} = 1/\sqrt{v_i v_i} = 1/\|v_i\|$. Note that applying $Q^{-1/2}$ to $R^{\mathrm{T}}R$ from the right gives us $((R^{\mathrm{T}}R)Q^{-1/2})_{i,j} = v_i v_j / \|v_j\|$. Applying $Q^{-1/2}$ from the left gives us $Q^{-1/2}(R^{\mathrm{T}}R) = v_i v_j / \|v_i\|$. This menas that $S_I = Q^{-1/2}((R^{\mathrm{T}}R)Q^{-1/2})$, since $(Q^{-1/2}((R^{\mathrm{T}}R)Q^{-1/2}))_{ij} = v_i v_j / (\|v_i\| \|v_j\|)$. We can thus write

$$S_I = Q^{-1/2}((R^{\mathrm{T}}R)Q^{-1/2}). \tag{1}$$

We now do the same thing for S_U, i.e., the $m \times m$ similarity matrix for users. As before, note that $(RR^{\mathrm{T}})_{i,j} = u_i u_j$, where u_i and u_j represent i-th and j-th user, respectively. Since $(P^{-1/2})_{ii} = 1/\|u_i\|$, we get — by the same reasoning as for S_I above — that $S_U = P^{-1/2}((RR^{\mathrm{T}})P^{-1/2})$, because $(P^{-1/2}((RR^{\mathrm{T}})P^{-1/2}))_{i,j} = u_i u_j / (\|u_i\| \|u_j\|)$. We thus have

$$S_U = P^{-1/2}((RR^{\mathrm{T}})P^{-1/2}). \tag{2}$$

(b) For user-user collaborative recommendation, we let

$$r_{u,s} := \sum_{x \in U} \cos(x, u) R_{x,s}.$$

We know that $(S_U)_{i,j} = \cos(u_i, u_j)$, where u_i denotes i-th user. Note that

$$(S_U R)_{us} = \sum_{x=1}^{m} (S_U)_{ux} R_{xs} = \sum_{x=1}^{m} \cos(x, s) R_{xs} = r_{u,s},$$

where u is u-th user and s is s-th user. This proves that for user-user collaborative filtering we have

$$\Gamma = S_U R. \tag{3}$$

For item-item collaborative recommendation we define

$$r_{u,s} := \sum_{x \in I} R_{u,x} \cos(x, s).$$

As before, we know that $(S_I)_{i,j} = \cos(v_i, v_j)$, where v_i denotes i-th item. Note that

$$(RS_I)_{us} = \sum_{x=1}^{n} R_{ux}(S_I)(xs) = \sum_{x=1}^{n} R_{ux} \cos(x, s) = r_{u,s},$$

where u denotes u-th user and s denotes s-th item. This proves that for item-item collaborative filtering we have

$$\Gamma = RS_I. \tag{4}$$

(c) Define $T := RR^{\mathrm{T}}$ as the non-normalized user similarity matrix. Explain the meaning of $T_{i,i}$ and $T_{i,j}$ for $i \neq j$ in terms of bipartite structures — what these mean in terms of node degrees, path between nodes, etc. Note that $T_{i,i} = (RR^{\mathrm{T}})_{i,i} = \sum_k R_{ik}(R^{\mathrm{T}})_{ki}$ is the number of items that user i likes, i.e., degree of user node i in the bipartite graph.

For $i \neq j$ we have $T_{i,j} = (RR^{\mathrm{T}})_{i,j} = \sum_k R_{ik}(R^{\mathrm{T}})_{kj}$ equal to number of items that both i and j like. In graph-theoretic language, $T_{i,j} = |N_G(u_i) \cap N_G(u_j)|$, i.e., the number of nodes that are at distance[1] 1 from both i and j.[2]

(d) See listing 1 for the code we used. The five TV shows that have the highest similarity scores for Alex for *user-user collaborative filtering*:

(i) `FOX 28 News at 10pm` with score `908.4801`.

(ii) `Family Guy` with score `861.1760`.

(iii) `2009 NCAA Basketball Tournament` with score `827.6013`.

(iv) `NBC 4 at Eleven` with score `784.7820`.

(v) `Two and a Half Men` with score `757.6011`.

The five TV shows that have the highest similairt scores for Alex for *item-item collaborative filtering*:

(i) `FOX 28 News at 10pm` with score `31.3647`.

(ii) `Family Guy` with score `30.0011`.

(iii) `NBC 4 at Eleven` with score `29.3968`.

(iv) `2009 NCAA Basketball Tournament` with score `29.2270`.

(v) `Access Hollywood` with score `28.9713`.

The precision at top-k as a function of $k \in [1, 19]$ is shown in figure 1.

[1] Distance between i and j in such an undirected graph is defined as the number of edges on the shortest path between i and j.

[2] In our case we even have that the graph is bipartite and we know that i and j belong to the same partition, so it is possible that T_{ij} is some well-known property. But the author is not that comfortable with graph theory.

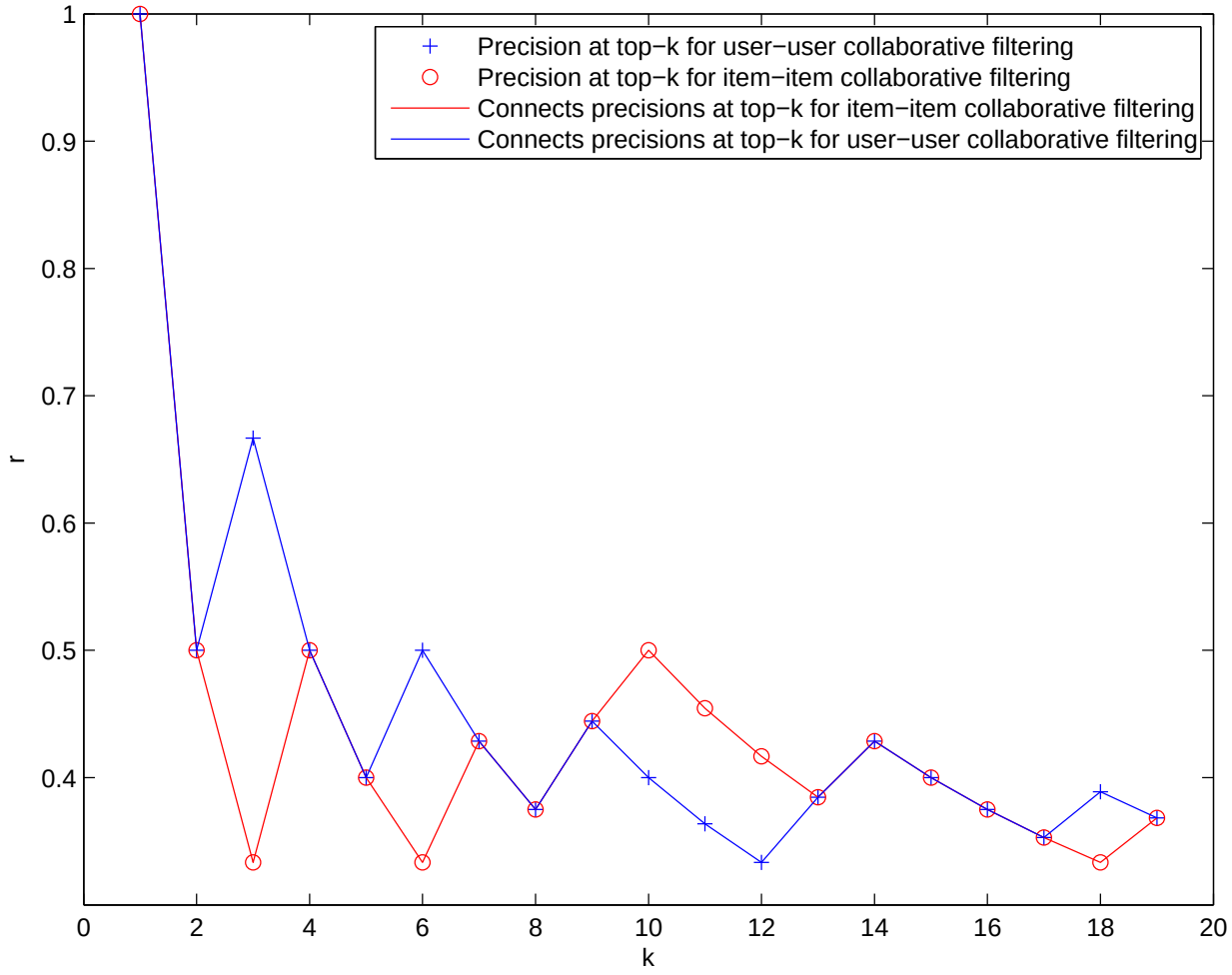

Figure 1: Precision at top-k for user-user (blue) and item-item (red) collaborative filtering.

The precision at top-k is about 0.4 for $k = 19$ for both methods, meaning that Alex watched roughly 2/5 of shows with the highest scores. Both precisions — precision at top-k for user-user and item-item collaborative filtering — converge to the same value. They seem to give roughly the same recommendations, which makes intuitive sense — similar people (that is people with similar tastes) watch similar shows.

Listing 1: Code for HW2Q1.

```
R = importdata('user-shows.txt');

P = diag(diag(R*R'));
Q = diag(diag(R'*R));

Si = Q^(-1/2)*((R'*R)*Q^(-1/2)); % item similarity matrix
Su = P^(-1/2)*((R*R')*P^(-1/2)); % user similairty matrix

% Gi(u, i) is recommendation of item i for user u
Gi = R*Si; % item-item collaborative filtering
% Gu(u, i) is recommendation of item i for user u
Gu = Su*R; % user-user collaborative filtering

% recommendations for Alex based on user-user collaborative filtering:
Su = Gu(:, 1:100); % first hundred columns
% the last five indices in i have the highes recommendation score
% ru contains scores and iu contains indices; sorted in ascending
[ru, iu] = sort(Su(500, :));
iu(90:100) % top 11

% the same thing, except we use item-item collaborative filtering:
Si = Gi(:, 1:100);
[ri, ii] = sort(Si(500, :));
ii(90:100) % top 11

% now plot precision at top-k for various k's
alex = importdata('alex.txt');
% plot for user-user and item-item similarity filtering
N = 19;
xu = 1:N;
xi = 1:N;
yi = []; yu = [];
for k = 1:N
  % fraction of top-k shows watched by alex in reality
  yu = [yu sum(alex(iu(100-k+1:100)))/k];
  yi = [yi sum(alex(ii(100-k+1:100)))/k];
end
plot(xu, yu, 'b+');
hold on;
plot(xu, yu, 'b');
plot(xi, yi, 'ro');
plot(xi, yi, 'r');
print -depsc hw2q2.eps
```

2 Singular Value Decomposition

We consider each point in turn. We use notation from HW2Q2 text, writing I_r for $r \times r$ identity matrix, etc.

(a) We find the SVD using $AA^{\mathrm{T}} = A^{\mathrm{T}}A$ for $A \in \mathbb{R}^{m \times n}$.

1) Let $K := AA^{\mathrm{T}} \in \mathbb{R}^{m \times m}$ and $C := A^{\mathrm{T}}A \in \mathbb{R}^{n \times n}$. Since $(C^{\mathrm{T}})_{ij} = C_{ij}$ for arbitrary $1 \leq i, j \leq n$, we have

$$C_{ij} = \sum_{k=1}^{m} A_{ik}(A^{\mathrm{T}})_{kj} = \sum_{k=1}^{m} A_{ik}A_{jk} = \sum_{k=1}^{m} (A^{\mathrm{T}})_{ki}A_{jk} = \sum_{k=1}^{m} A_{jk}(A^{\mathrm{T}})_{ki} = C_{ji} = (C^{\mathrm{T}})_{ij},$$

meaning that $C = C^{\mathrm{T}}$, so C is symmetric. The proof that K is symmetric is the same:

$$K_{ij} = \sum_{k=1}^{n} A_{ik}(A^{\mathrm{T}})_{kj} = \sum_{k=1}^{n} A_{ik}A_{jk} = \sum_{k=1}^{n} A_{jk}(A^{\mathrm{T}})_{ki} = K_{ji}.$$

Simpler way to see this is $(AA^{\mathrm{T}})^{\mathrm{T}} = AA^{\mathrm{T}}$ and $(A^{\mathrm{T}}A)^{\mathrm{T}} = A^{\mathrm{T}}A$, which follow from basic algebraic properties of matrix multiplication.

Next, note that $AA^{\mathrm{T}} = USV^{\mathrm{T}}VS^{\mathrm{T}}U^{\mathrm{T}} = US^2U^{\mathrm{T}}$, since $S = S^{\mathrm{T}}$ and $SI_r = I_rS = S$. By the same reasoning we get $A^{\mathrm{T}}A = VS^2V^{\mathrm{T}}$. This means that S^2 contains eigenvalues of both C and K, that U contains eigenvectors of C as columns, and that V contains eigenvectors of K as columns. Let $k = \arg\min_k\{\sigma_k \mid \sigma_k > 0\}$. Since $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$, the nonzero eigenvalues are exactly $\sigma_1 \geq \ldots \geq \sigma_k > 0$ with the corresponding (nonzero) eigenvectors $u_1, \ldots, u_r$.

2) We should take K if $m \leq n$, and C otherwise. Since we assumed $m = 100$ and $n = 100000$, we should take K in this case.

3) First compute $C := A^{\mathrm{T}}A$ and its eigenvalue decomposition $C = Q_C\Sigma_C Q_C^{\mathrm{T}}$, which equals VS^2V^{T} for unknown V and Σ, equality following because of SVD: $A = U\Sigma V^{\mathrm{T}}$. We trivially have $V = Q_C$. We can thus express $S = \Sigma^{1/2}$ and $U = A(SV^{\mathrm{T}})^{-1} = A(VS^{-1})$.

4) Let $K := AA^{\mathrm{T}}$ compute its eigenvalue decomposition $Q_K\Sigma_K Q_K^{\mathrm{T}}$. Since $A = USV^{\mathrm{T}}$ we have $K = US^2U^{\mathrm{T}}$ for some U, S, V. Similarly as before, we trivially have $U = Q_K$ and $S = \Sigma_K^{1/2}$. Similarly as before we have $V = ((US)^{-1})A)^{\mathrm{T}} = A^{\mathrm{T}}(US^{-1})$.

(b) Document similarity using SVD; see listing 2 for code of this section. See figure 2 for the plot of $r(k)$ for $k \in \{197, 247, 297, 347, 397, 447, 497\}$. We get highest $r = r(k)$ for $k = 347$, i.e., when we reconstruct I using the highest 347 singular values. This could mean that we got rid of the noise by ignoring the "lower" singular values.

Listing 2: Code for HW2Q1.

```
1 I = load('Matrix.txt');
2
```

```
rs = [];
ks = [197, 247, 297, 347, 397, 447, 497];
% S = S.^2;
for k = ks
  [U, S, V] = svd(I, 'econ'); % this is naive; but it works OK for small data;
  % set all except the largest k singular values to 0
  S(k:497, k:497) = 0;
  It = U*S*V';
  Is = It*It';
  N = diag(diag(Is)); % N(i,i) = norm(di)^2
  Ics = N^(-1/2)*(Is*N^(-1/2)); % Ics(i,j) = cossim(di, dj)
  s = 0;
  for i = 1:99
    s = s+sum(Ics(i, i+1:100));
  end
  s = s/(50*99); % equivalent to s := s/binom{100}{2}
  t = 0;
  for i = 1:496
    t = t+sum(Ics(i, i+1:497));
  end
  t = t/(248*497); % equivalent to t := t/binom{497}{2}
  r = s/t;
  rs = [rs r];
end

plot(ks, rs, 'o');
hold on;
plot(ks, rs);
print -desc hw2q2.eps
```

3 Theory of k-means

We have an n-set $\mathcal{X}$ of points from $\mathbb{R}^d$ and a number of clusters k. We want to choose a set of k centers $\mathcal{C}$ that minimize the function

$$\phi := \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} \|x - c\|^2, \tag{5}$$

where $\|x\| := \|x\|_2$ denotes L_2-norm of x. In what follows we often use the fact that $\|x\| = \|-x\|$. (This allows us to be sloppy and write things like $\|x - y\| = \|y - x\|$.)

1) We now prove the following identity:

$$\sum_{x \in S} \|x - z\|^2 - \sum_{x \in S} \|x - c(S)\|^2 = |S| \cdot \|z - c(S)\|^2, \tag{6}$$

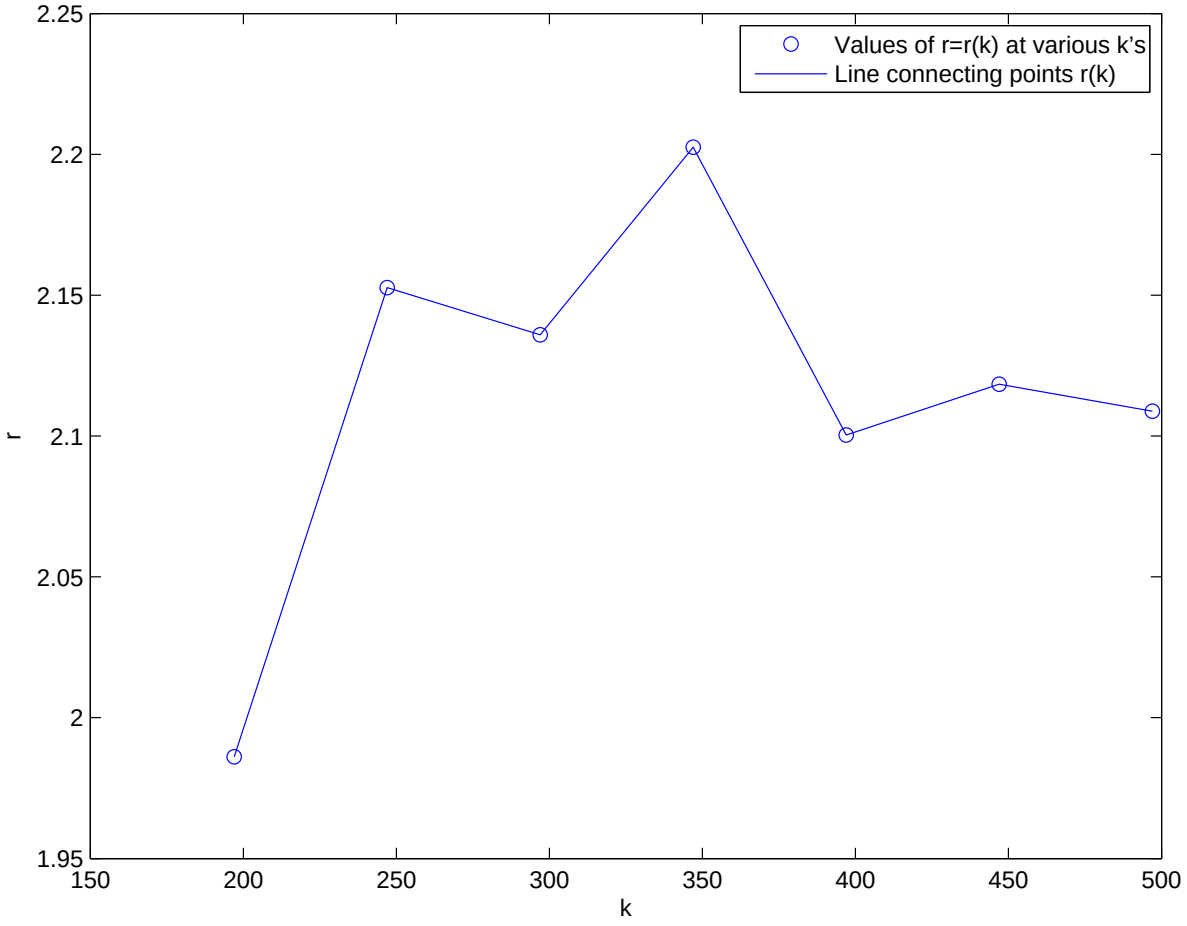

Figure 2: Plot of $r = r(k)$ for $k \in \{197, 247, 297, 347, 397, 447, 497\}$.

where $c(S) := \frac{1}{|S|}\sum_{x\in S} x$ and $S \subseteq \mathbb{R}^d$, and $z \in \mathbb{R}^d$. Note that

$$\begin{aligned}
&|S| \cdot \|z - c(S)\|^2 - \sum_{x\in S} \left(\|x - z\|^2 - \|x - c(S)\|^2\right) \\
&= \sum_{x\in S} \left(\|z - c(S)\|^2 - \|x - z\|^2 + \|x - c(S)\|^2\right) \\
&= \sum_{x\in S} \left((x - c(S))^2 - (x - z)^2 + (x - c(S))^2\right) \\
&= 2\sum_{x\in S} \left(\|c(S)\|^2 - zc(S) - xc(S) + xz\right) \\
&= 2|S| \cdot \left(\|c(S)\|^2 - zc(S)\right) + 2\sum_{x\in S}(xz - xc) \\
&= 2|S| \cdot \left(\|c(S)\|^2 - zc(S)\right) + 2|S| \cdot c(S)z - 2|S| \cdot \|c(S)\|^2 \\
&= 0,
\end{aligned}$$

thus establishing (6). (Note that we wrote $\sum_{x\in S} xz = z|S|\frac{1}{|S|}\sum_{x\in S} x = |S|zc(S)$; similarly $\sum_{x\in S} xc(S) = c(S)|S|\frac{1}{|S|}\sum_{x\in S} x = c(S)c(S) = \|c(S)\|^2$.)

2) Let $\phi_t := \sum_{x\in\mathcal{X}} \min_{c\in\mathcal{C}_t} \|x-c\|^2$ be the objecitve function in the t-th iteration, where $\mathcal{C}_t$ is the set of centroids in this iteration and C_i^t is the set of points assigned to i-th centroid in t-th iteration, i.e., let C_i^t be the cluster in t-th iteration that corresponds to i-th centroid. We now prove that $\phi_t \geq \phi_{t+1}$ for all $t \geq 0$. First note that by (6) we have

$$\begin{aligned}
\phi_t &= \sum_{x\in\mathcal{X}} \min_{x\in\mathcal{C}_t} \|x - c\|^2 \\
&= \sum_{i=1}^{k} \sum_{x\in C_i^t} \|x - c(C_i^{t-1})\|^2 \\
&= \sum_{i=1}^{k} \left(|C_i^t| \cdot \|c(C_i^{t-1}) - c(C_i^t)\|^2 + \sum_{x\in C_i^t} \|x - c(C_i^t)\|^2\right),
\end{aligned}$$

implying

$$\begin{aligned}
\sum_{x\in C_i^t} \|x - c(C_i^{t-1})\|^2 &\geq \sum_{x\in C_i^t} \|x - c(C_i^t)\|^2 \\
&\geq \sum_{x\in C_i^t} \min_{x\in\mathcal{C}_{t+1}} \|x - c\|^2,
\end{aligned}$$

because $c(C_i^t) \in \mathcal{C}_{t+1}$, i.e., $c(C_i^t)$ will be one of the centers in the $(t+1)$-th iteration. We

thus have

$$\begin{aligned}\phi_t &= \sum_{i=1}^{k} \sum_{x \in C_i^t} \|x - c(C_i^{t-1})\|^2 \\ &\geq \sum_{i=1}^{k} \sum_{x \in C_i^t} \min_{c \in \mathcal{C}_{t+1}} \|x - c\|^2 \\ &= \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}_{t+1}} \|x - c\|^2 \\ &= \phi_{t+1},\end{aligned}$$

establishing $\phi_t \geq \phi_{t+1}$ for all t, since t was arbitrary. (Several times we used the obvious fact that the clusters $C_1^t, \ldots, C_k^t$ form partition of $\mathcal{X}$ in each iteration t.)

3) We now show that when running k-means, the objective function converges to a finite value. Note that we trivially have $\phi_t \geq 0$ for all t, because ϕ_t is a sum of squared terms. Since $\phi_t \geq \phi_{t+1}$ for all t, we have that the sequence $\{\phi_t\}_{t \geq 0}$ is monotonically decreasing and bounded from below. The convergence of $\{\phi_t\}_{t \geq 0}$ follows from the well-known bounded monotone convergence theorem, stating that *every bounded monotonic sequence of real numbers is convergent*, because our sequence is both monotonically decreasing and bounded.[3] We conclude that $\{\phi_t\}_{t \geq 0}$ converges to C for some $C \in [0, n \max_{x,y \in \mathcal{X}} \|x - y\|^2]$. (We showed that there *exists* some (finite) value $0 \leq C \leq n \max_{x,y \in \mathcal{X}} \|x - y\|^2$ such that ϕ_t converges to C.)

   We now argue (heuristically) that k-means converges, i.e., that the algorithm reaches a state where the centroids do not change. If all "terms" of the cost function do not change — if C_i^t and C_i^{t+1} all have the same contributions for all i, i.e., for each cluster — then the algorithm will not change clusters anymore. Our reasoning is very hand-wavy:

   - The fact that objective function value remains the same, i.e., $\phi_t = \phi_{t+1}$, does not imply that all the "terms" remain the same.
   - In the previous paragraph we showed that $\{\phi_t\}_{t \geq 0}$ converges to some (finite) value $C \in \mathbb{R}_0^+$. This means that for every $\epsilon > 0$ there exists $N \geq 0$ such that for all $t \geq N$ we have $\phi_t - C < \epsilon$. We *have not* shown $\phi_t = \phi_{t+1}$ for all sufficiently large t.

   However, we believe that our reasoning is sufficient for the purposes of heuristic argument that this part of the question asks for.

4) Finally, we show that for arbitrary $r > 1$ there is a "bad" initialization that will give at leat r-times larger cost than the optimal clustering. The idea is extremely simple.

   Suppose we have k "uniformly filled" disks of n points $D_i \subseteq \mathbb{R}^2$ for fixed $k > 0$, and the first $k - 1$ disks have center of the form $(x_0, 0)$ and all have the same radius, i.e., they lie

[3] Note that it is fine to say bounded because $\{\phi_t\}_{t \geq 0}$ is also bounded from above by $n \max_{x,y \in \mathcal{X}} \|x - y\|^2$, but this is not very relevant to our discussion.

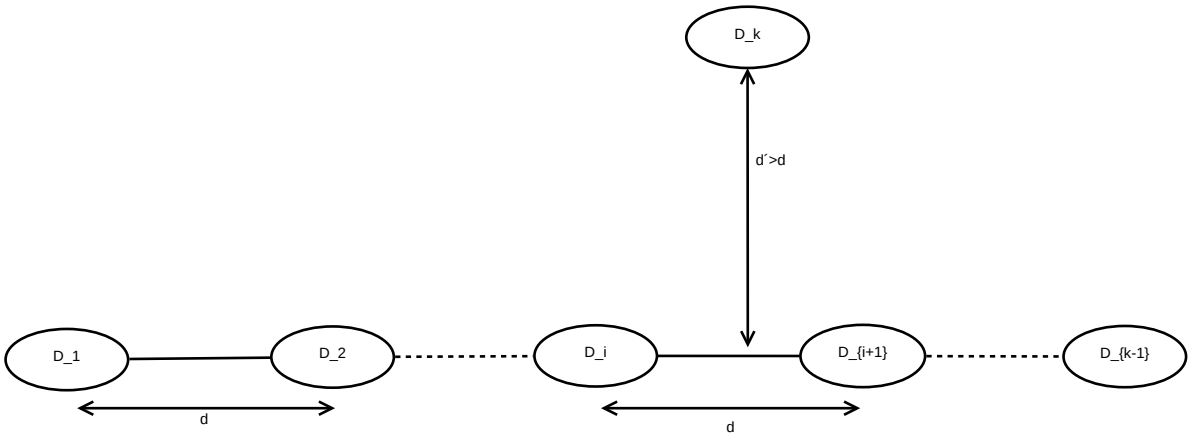

Figure 3: Rough sketch of our setup.

on a horizontal line. Also suppose that the distance between $c(D_i)$ and $c(D_{i+1})$ is $d > 0$ for all $1 \leq i \leq k-2$. Suppose k-th disk is centered at $(0, y_k)$ and is farther than d away from all other disks. (See figure 3 for sketch of our setup.) The rough structure of the optimal clustering is obvious: Place i-th center in the center of the disk D_i. Let ϕ_0 be the cost of the optimal clustering.

Let $r > 1$ be an arbitrary real number. We now show how to arrange the points D_i and pick the centers $c_1, \ldots, c_k$ so that the cost of the resulting clustering will be at least $r\phi_0$. Place the first center, $c_1 = (x_1, 0)$, in the middle of the line. Now set the remaining $k-1$ centers $c_i := (0, y_i)$ inside the outlier disk D_k. It should now be obvious that all the points from $D_1 \cup \ldots \cup D_k$ will belong to a single cluster, i.e., the one that corresponds to the center c_1, while points from D_k will be partitioned into $k-1$ clusters corresponding to $c_2, \ldots, c_k$. With such an initialization the center c_1 will never move. Now stretch the disks of points $D_1, D_2, \ldots, D_{k-1}$ farther apart — sets of points D_i and D_{i+1} are still equally spaced (we move the whole disks D_i, so local structure is presrved), but the distance between them is now larger — so that $\sum_{x \in \cup_{i=1}^{k-1} D_i} \|x - c_1\| \geq r\phi_0$. (It is worth noting that the optimal cost ϕ_0 also increases; the trick is that ϕ_0 is increasing much slower than $\sum_{x \in \cup_{i=1}^{k-1} D_i} \|x - c_1\|$.) This is obviously achievable. The cost of the resulting clustering with such an initialization will thus be at least $r\phi_0$, where ϕ_0 is the cost of the optimal clustering.

We have shown that we can pick centers in such a way that k-means gives arbitrarily bad clustering. Our example was in $\mathbb{R}^2$ with some fixed $k \in \mathbb{K}$.

4 k-means on MapReduce

(a) See the appendix for source code. Our solution consists of the main class `KMeans` (see listing 4) and the helper class `Point` (see listing 3). Note that reducer ouputs pairs (c_i^{t-1}, c_i^t), where c_i^t denotes i-th center in t-th iteration. We used `hadoopjar~/workspace/kmeans/`

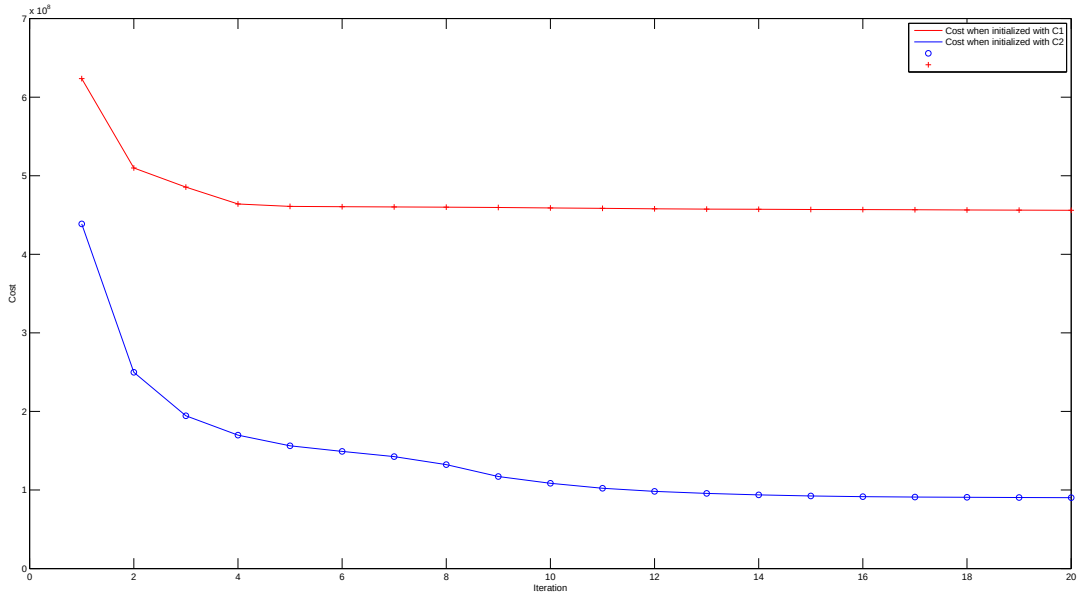

Figure 4: Plot of the cost as a function of the number of iterations of k-means when initialized with `c1.txt` (red) and `c2.txt` (blue).

`target/kmeans-0.0.1-SNAPSHOT.jarhomework\_hw2.kmeans.kmeans.KMeans-Dmapred.job.tracker=local-Dfs.defaultFS=local` to run Hadoop. See source code for details regarding directory structure. We assume all data resides in `/home/cloudera/Documents/hw-2/`.

(b) We compare costs ϕ_t for $t = 1, 2, \ldots, 20$ for initializations `c1.txt` (red) `c2.txt` (blue) on figure 4.

(c) It is clear from figure 4 that random initialization `c1.txt` is not better than `c2.txt`. An intuitive reason for k-means initialized with `c2.txt` performing better than the one initialized with `c1.txt` could be that the centroids `c2.txt` better "cover" the dataset, thus allowing k-means to discover good clusters, while we could have bad luck when picking `c1.txt`. When initialized with `c1.txt` the percentage change in cost is 10.183%, i.e., the cost drops for $\approx 10.2\%$ if instead of the initial ones we use centroids obtained after 10 iterations of k-means. For `c2.txt` the cost drops $\approx 60.7\%$ if instead of the initial ones we use centroids obtained after 10 iterations of k-means.

(d) For the second document we would apply the following 5 tags:

(i) `instillation`,

(ii) `methoxyfenozide`,

(iii) `sodium-sensitive`,

(iv) `masses`,

(v) `post-infectious`.

These were generated using code from listing 6. The code uses clusters computed in 20-th iteration by k-means on Hadoop when initialized with `c1.txt`.

A Source code for question 4

To make grader's life easier, this appendix contains source code associated with the last question.

A.1 Implementation of k-means on MapReduce

Listing 3: Source of the helper class `Point.java`.

```
1  package homework_hw2.kmeans.kmeans;
2
3  public class Point {
4    private final int DIM = 58;
5    private double [] p;
6    public Point(String p) {
7      this.p = new double[this.DIM];
8      int i = 0;
9      p = clean(p);
10     for (String x : p.split(" ")) {
11       this.p[i++] = Double.parseDouble(x);
12     }
13   }
14   // ignore previous centers
15   public static String clean(String p) {
16     if (p.indexOf('\t') == -1) { return p; }
17     else {
18       String [] arr = p.split("\t");
19       return arr[1];
20     }
21   }
22   // retrieve i-th component
23   public double get(int i) {
24     return p[i];
25   }
26   // distance between self and p2
27   public double dist(Point p2) {
28     double d = 0.0;
29     for(int i = 0; i < p.length; ++i) {
30       d += (p[i]-p2.get(i))*(p[i]-p2.get(i));
31     }
32     return Math.sqrt(d);
33   }
```

```
34    // add p2 to self
35    public void add(Point p2) {
36      for (int i = 0; i < p.length; ++i) { p[i] += p2.get(i); }
37    }
38    // divide self with scalar c
39    public void divide(double c) {
40      for (int i = 0; i < p.length; ++i) { p[i] /= c; }
41    }
42    // convert point to string
43    public String toString() {
44      String s = "";
45      for (int i = 0; i < p.length; ++i) {
46        s += p[i]+" ";
47      }
48      return s;
49    }
50  }
```

Listing 4: Source of the main class `KMeans.java`.

```
1   package homework_hw2.kmeans.kmeans;
2
3   import org.apache.hadoop.conf.Configuration;
4
5   import org.apache.hadoop.conf.Configured;
6   import org.apache.hadoop.util.Tool;
7   import org.apache.hadoop.util.ToolRunner;
8
9   import org.apache.hadoop.mapreduce.Job;
10  import org.apache.hadoop.mapreduce.Mapper;
11  import org.apache.hadoop.mapreduce.Reducer;
12  import org.apache.hadoop.mapreduce.Mapper.Context;
13  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14  import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
15  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
16  import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
17
18  import org.apache.hadoop.fs.Path;
19  import org.apache.hadoop.io.IntWritable;
20  import org.apache.hadoop.io.LongWritable;
21  import org.apache.hadoop.io.Text;
22
23  import java.io.IOException;
24  import java.util.*;
25  import java.io.BufferedReader;
26  import java.io.FileReader;
27
28  public class KMeans extends Configured implements Tool {
29    /**
30     * @param args
31     */
32    public static void main(String[] args) throws Exception {
```

```
33      Configuration conf = new Configuration();
34      int res = ToolRunner.run(conf, new KMeans(), args);
35      System.exit(res);
36    }
37    public static void loadCenters(ArrayList<Point> centers, String centersPath)
          {
38      BufferedReader br = null;
39      try {
40        br = new BufferedReader(new FileReader(centersPath));
41        String line = null;
42        while ((line = br.readLine()) != null) {
43          centers.add(new Point(line));
44        }
45        br.close();
46      } catch (IOException e) {
47        System.err.println("[ERROR] Exception reading centers.");
48      }
49    }
50    @Override
51    public int run(String[] arg0) throws Exception {
52      getConf().set("centersPath", "/home/cloudera/Documents/hw-2/c1.txt");
53      for (int i = 1; i < 20; ++i) {
54        Job job = new Job(getConf(), "KMeans");
55        // Configure job with all parameters
56        job.setMapperClass(Map.class);
57        job.setReducerClass(Reduce.class);
58
59        job.setOutputKeyClass(Text.class);
60        job.setOutputValueClass(Text.class);
61
62        job.setInputFormatClass(TextInputFormat.class);
63        job.setOutputFormatClass(TextOutputFormat.class);
64        // Set job input/output dirs
65        FileInputFormat.addInputPath(job, new Path("/home/cloudera/Documents/hw
              -2/hw2-q4/data.txt"));
66        FileOutputFormat.setOutputPath(job, new Path("/home/cloudera/Documents/
              hw-2/c-"+i+"/"));
67        // Run job
68        job.waitForCompletion(true);
69        // centroid Dir := outputDir + i
70        getConf().set("centersPath", "/home/cloudera/Documents/hw-2/c-"+i+"/part
              -r-00000");
71      }
72      return 0;
73    }
74
75    public static class Map extends Mapper<LongWritable, Text, Text, Text> {
76      @Override
77      public void map(LongWritable key, Text value, Context context)
78          throws IOException, InterruptedException {
79        ArrayList<Point> centers = new ArrayList<Point>();
```

```
80        Text center = new Text(), point = new Text();
81
82        loadCenters(centers, context.getConfiguration().get("centersPath"));
83        Point crrDat = new Point(value.toString());
84        Point crrCnt = centers.get(0);
85        for(Point p : centers) {
86          if (crrDat.dist(crrCnt) > crrDat.dist(p)) { crrCnt = p; }
87        }
88
89        center.set(crrCnt.toString());
90        point.set(crrDat.toString());
91
92        context.write(center, point);
93      }
94    }
95
96    public static class Reduce extends Reducer<Text, Text, Text, Text> {
97      @Override
98      public void reduce(Text key, Iterable<Text> values, Context context)
99          throws IOException, InterruptedException {
100       Text value = new Text();
101       ArrayList<Point> cluster = new ArrayList<Point>();
102
103       for (Text x : values) { cluster.add(new Point(x.toString())); }
104
105       Point c = cluster.get(0);
106       for (int i = 1; i < cluster.size(); ++i) { c.add(cluster.get(i)); }
107       c.divide(cluster.size()); // our new centroid
108
109       value.set(c.toString());
110       context.write(key, value);
111     }
112   }
113 }
```

A.2 Computing $\phi(i)$ at each iteration

Since the `Point.java` remains the same as in listing 3, we only include the slightly changed `KMeans.java`. See listing 5.

Listing 5: Source of the main class `KMeans.java`.

```
1  package homework_hw2.kmeans.kmeans;
2
3  import org.apache.hadoop.conf.Configuration;
4
5  import org.apache.hadoop.conf.Configured;
6  import org.apache.hadoop.util.Tool;
7  import org.apache.hadoop.util.ToolRunner;
8
9  import org.apache.hadoop.mapreduce.Job;
```

```
10 import org.apache.hadoop.mapreduce.Mapper;
11 import org.apache.hadoop.mapreduce.Reducer;
12 import org.apache.hadoop.mapreduce.Mapper.Context;
13 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
15 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
16 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
17
18 import org.apache.hadoop.fs.Path;
19 import org.apache.hadoop.io.IntWritable;
20 import org.apache.hadoop.io.LongWritable;
21 import org.apache.hadoop.io.Text;
22
23 import java.io.IOException;
24 import java.util.*;
25 import java.io.BufferedReader;
26 import java.io.FileReader;
27
28 public class KMeans extends Configured implements Tool {
29   /**
30    * @param args
31    */
32   public static void main(String[] args) throws Exception {
33     Configuration conf = new Configuration();
34     int res = ToolRunner.run(conf, new KMeans(), args);
35     System.exit(res);
36   }
37   public static void loadCenters(ArrayList<Point> centers, String centersPath)
         {
38     BufferedReader br = null;
39     try {
40       br = new BufferedReader(new FileReader(centersPath));
41       String line = null;
42       while ((line = br.readLine()) != null) {
43         if (!line.startsWith("cost")) { centers.add(new Point(line)); }
44       }
45       br.close();
46     } catch (IOException e) {
47       System.err.println("[ERROR] Exception reading centers.");
48     }
49   }
50   @Override
51   public int run(String[] arg0) throws Exception {
52     getConf().set("centersPath", "/home/cloudera/Documents/hw-2/c2.txt");
53     for (int i = 1; i <= 20; ++i) {
54       Job job = new Job(getConf(), "KMeans");
55       // Configure job with all parameters
56       job.setMapperClass(Map.class);
57       job.setReducerClass(Reduce.class);
58
59       job.setOutputKeyClass(Text.class);
```

```
60        job.setOutputValueClass(Text.class);
61
62        job.setInputFormatClass(TextInputFormat.class);
63        job.setOutputFormatClass(TextOutputFormat.class);
64        // Set job input/output dirs
65        FileInputFormat.addInputPath(job, new Path("/home/cloudera/Documents/hw
              -2/hw2-q4/data.txt"));
66        FileOutputFormat.setOutputPath(job, new Path("/home/cloudera/Documents/
              hw-2/c-"+i+"/"));
67        // Run job
68        job.waitForCompletion(true);
69        getConf().set("centersPath", "/home/cloudera/Documents/hw-2/c-"+i+"/part
              -r-00000");
70      }
71      return 0;
72    }
73
74    public static class Map extends Mapper<LongWritable, Text, Text, Text> {
75      @Override
76      public void map(LongWritable key, Text value, Context context)
77          throws IOException, InterruptedException {
78        ArrayList<Point> centers = new ArrayList<Point>();
79        Text center = new Text(), point = new Text();
80
81        loadCenters(centers, context.getConfiguration().get("centersPath"));
82        Point crrDat = new Point(value.toString());
83        Point crrCnt = centers.get(0);
84        for(Point p : centers) {
85          if (crrDat.dist(crrCnt) > crrDat.dist(p)) { crrCnt = p; }
86        }
87
88        center.set(crrCnt.toString());
89        point.set(crrDat.toString());
90
91        context.write(center, point);
92      }
93    }
94
95    public static class Reduce extends Reducer<Text, Text, Text, Text> {
96      @Override
97      public void reduce(Text key, Iterable<Text> values, Context context)
98          throws IOException, InterruptedException {
99        Text value = new Text();
100       ArrayList<Point> cluster = new ArrayList<Point>();
101
102       Point currCenter = new Point(key.toString());
103       double currCost = 0.0;
104
105       for (Text x : values) { cluster.add(new Point(x.toString())); }
106
107       Point nextCenter = cluster.get(0);
```

```
108         currCost = currCenter.dist(nextCenter);
109         for (int i = 1; i < cluster.size(); ++i) {
110           nextCenter.add(cluster.get(i));
111           currCost += currCenter.dist(cluster.get(i));
112         }
113         nextCenter.divide(cluster.size()); // our new centroid
114
115         // output the cost
116         Text costKey = new Text(); costKey.set("cost");
117         Text costVal = new Text(); costVal.set(String.valueOf(currCost));
118         context.write(costKey, costVal);
119         // output the new center
120         value.set(nextCenter.toString());
121         context.write(key, value);
122       }
123     }
124 }
```

B Automatically tagging documents

We wrote a simple python script that uses centroids computed on hadoop to find the tags. See listing 6.

Listing 6: Source of the main class `KMeans.java`.

```
1  #!/usr/bin/python
2
3  import operator
4
5  dist = lambda x, c: sum([(x[i]-c[i])**2 for i in range(len(x))])
6
7  # d is a document; c is a set of centroids
8  def tags(d, c):
9    v = open('hw2-q4/vocab.txt').read().split('\n')
10   dc = 0
11   # find cluster of the document d
12   for k in c: dc = k if dist(d, c[k]) < dist(d, c[dc]) else dc
13   # python 2.6 doesn't support dict comprehension
14   t = {}
15   for i in range(len(d)): t[i] = d[i]
16   t = sorted(t.iteritems(), key=operator.itemgetter(1))
17   t.reverse()
18   print [v[t[i][0]] for i in range(11)]
19
20 # entry point
21 if __name__ == '__main__':
22   # load documents
23   T = open('hw2-q4/data.txt').read().split('\n')
24   X = []
25   for t in T: X.append([float(el) for el in t.split()])
```

```
26  # load the clusters that we got after 20 iterations
27  # when initialized with c1.txt
28  T = open('c-19/part-r-00000').read().split('\n')
29  C = {}
30  i = 0
31  for t in T[:-1]:
32    if t[:4] != 'cost':
33      C[i] = ([float(el) for el in t.split('\t')[1].split()])
34      i = i+1
35  # compute the tags
36  tags(X[2], C)
```

References

[RLU13] Anand Rajaraman, Jure Leskovec, and Jeffrey D. Ullman. *Mining of Massive Datasets*. Second edition, 2013. Unpublished.