CSEP 590A: Machine Learning for Big Data **Spring 2022**

Problem Set 2

Please read the homework submission policies.

Assignment Submission All students should submit their assignments electronically via GradeScope. No handwritten work will be accepted. Math formulas **must** be typeset using LaTeX or other word processing software that supports mathematical symbols (E.g. Google Docs, Microsoft Word). Simply sign up on Gradescope and use the course code X3WYKY. Please use your UW NetID if possible.

For the non-coding component of the homework, you should upload a PDF rather than submitting as images. We will use Gradescope for the submission of code as well. Please make sure to tag each part correctly on Gradescope so it is easier for us to grade. There will be a small point deduction for each mistagged page and for each question that includes code. Put all the code for a single question into a single file and upload it. Only files in text format (e.g. .txt, .py, .java) will be accepted. **There will be no credit for coding questions without submitted code on Gradescope, or for submitting it after the deadline**, so please remember to submit your code.

Coding You may use any programming languages and standard libraries, such as NumPy and PySpark, but you may not use specialized packages and, in particular, machine learning libraries (e.g. sklearn, TensorFlow), unless stated otherwise. Ask on the discussion board whether specific libraries are allowed if you are unsure.

Late Day Policy All students will be given two no-questions-asked late periods, but only one late period can be used per homework and cannot be used for project deliverables. A late-period lasts 48 hours from the original deadline (so if an assignment is due on Thursday at 11:59 pm, the late period goes to the Saturday at 11:59pm Pacific Time).

Academic Integrity We take academic integrity extremely seriously. We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down the solutions and the code independently. In addition, each student should write down the set of people whom they interacted with.

Discussion Group (People with whom you discussed ideas used in your answers):

On-line or hardcopy documents used as part of your answers:

I acknowledge and accept the Academic Integrity clause.

*(Signed)*____

0 HW2 Survey

Please complete HW2 Survey on Gradescope after finishing the homework.

1 Principal Component Analysis and Reconstruction (40 points)

Principal Component Analysis (PCA) is a dimensionality reduction algorithm that represents the data matrix as a set of principal components that are orthogonal to each other. Each of these principal components captures a source of variation in the original data matrix such that we can reduce the dimensions of the data while preserving as much information as possible. Principal components are defined as the eigenvectors of the covariance matrix of the data; thus, we will be implementing PCA using eigendecomposition.

This question will help you understand the basics of eigendecomposition. By examining the dimensionality reduction and reconstruction of images, you will think and learn more about informative representations of data.

Let's do PCA and reconstruct pictures of faces in the PCA basis. As we will be making many visualizations of images, plot these images in a reasonable way (e.g. make the images smaller).

(a) PCA [15 pts]

For this question we will use the dataset `faces.csv` from `pca/data` (adapted from the Extended Yale Face Database B), which is composed of facial pictures of 38 individuals under 64 different lighting conditions such as lit from the top, front, and side (some lighting conditions are missing for some people). In total, there are 2414 images, where each image is 96 pixels high and 84 pixels wide (for a total of 8064 pixels per image). Each row in the dataset is a single image flattened into a vector in a column-major order. The images are in grayscale, so each pixel is represented by a single real number between 0 (black) and 1 (white).

Define $\mathbf{\Sigma}$, a 8064×8064 matrix, as follows:

$$\mathbf{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}}$$

where the $\boldsymbol{x}_i$'s are points in our dataset as **column** vectors and $n = 2414$ is the number of points in the dataset. Now compute the top 50 PCA dimensions; these are the 50 dimensions which best reconstruct the data.

We will be implementing PCA using eigendecomposition. Thus, you may use library functions for obtaining the eigenvalues and eigenvectors of a matrix (e.g. `numpy.linalg.eig` in

Python and `eig` or `eigs` in MATLAB). You should **not** use functions which directly compute the principal components of a matrix. Please ask on Ed regarding other (non-basic) linear algebra functions you may be interested in using.

1. [**3 pts**] What are the eigenvalues λ_1, λ_2, λ_{10}, λ_{30}, and λ_{50}? Also, what is the sum of eigenvalues $\sum_{i=1}^{d} \lambda_i$?

2. [**6 pts**] It can be shown that fractional reconstruction error of using the top k out of d directions is $1 - \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$ (this means that when $k = d$, our reconstruction error is 0, which basically means that we use all the principal components and simply reconstruct the original matrix). Plot this fractional reconstruction error for each of the first 50 values of k (i.e. k from 1 to 50). So the X-axis is k and the Y-axis is the fractional reconstruction error. Make sure your plots are legible at reasonable zoom and the axes are labeled in your write-up.

3. [**6 pt**] Using 1-3 sentences, explain what the first eigenvalue captures, and why you think λ_1 is much larger than the other eigenvalues.

(b) Visualization of the Eigen-Directions [11 pts]

Now let us get a sense of the what the top PCA directions are capturing (recall these are the directions which capture the most variance).

1. [**5 pts**] Display the first 10 eigenvectors as images. Label the images to denote which eigenvector each image denotes.

   Hint 1: If the images appear like random lines, try reshaping differently and then transposing.

   Hint 2: The eigenvectors you obtain are normalized to have length 1 in the L_2 norm, thus their entries are extremely small. To avoid getting all-black images, make sure to re-normalize the image. in Python, `matplotlib.pyplot.imshow` does this by default. If you are using `imshow` in MATLAB, you should pass an extra empty vector as an argument, e.g. "`imshow(im, []);`".

2. [**6 pt**] Provide a brief interpretation (4-6 sentences) of what you think each eigenvector captures (*Hint:* Look at each image and think about what features of the image the eigenvector is capturing).

(c) Visualization and Reconstruction [14 pts]

1. [**8 pt**] We will now observe the reconstruction using PCA on a sample of images composed of the following images:

   (a) image 1 (row 0).

(b) image 24 (row 23).

(c) image 65 (row 64).

(d) image 68 (row 67).

(e) image 257 (row 256).

In the previous parts, we used eigendecomposition to extract the top k eigenvectors of Σ. Now, we will examine reconstruction using PCA.

For each of these images, plot the original image and plot the projection of the image onto the top k eigenvectors of Σ for $k = 1, 2, 5, 10, 50$. In particular, if U is the $d \times k$ matrix of the top k eigenvectors, the reconstruction matrix will be UU^{T}.

Specifically, you should obtain a 5×6 table where in each row corresponds to a single image, the first cell in every row is the original image and the following cells are the reconstructions of the image with the 5 different values of k.

Hint: In this problem, we are observing (partial) combinations of images that already have a meaningful scale. Therefore, we want to keep the scale of the results and not re-normalize the images (in contrast to part (c)). If you are using `matplotlib.pyplot.imshow` in Python, you should pass the additional arguments `vmin=0, vmax=1`, e.g. `imshow(im, vmin=0, vmax=1)`. If you are using `imshow` in MATLAB, the default is not re-normalizing, so you should **not** pass additional arguments, e.g. simply use "`imshow(im);`".

2. [**6 pt**] Provide a brief interpretation (4-6 sentences), in terms of your perceptions of the quality of these reconstructions. Explain the results in light of your answers for part (c). Discuss specific examples regarding pairs of images and eigenvector set (i.e. how does including a set of eigenvectors affect specific images?)

What to submit:

(i) The values of λ_1, λ_2, λ_{10}, $\lambda_{30}, \lambda_{50}$ and $\sum_i \lambda_i$, a plot of the reconstruction error vs. k and an explanation for 1(a).

(ii) Plots of the first 10 eigenvectors as images and their interpretation [1(b)].

(iii) Table of original and reconstructed images and their interpretation [1(c)].

(iv) Upload the code to Gradescope.

2 k-means on Spark (30 points)

Note: This problem requires substantial computing time. Don't start it at the last minute. Also, you should **not** use the Spark MLlib clustering library for this problem.

This problem will help you understand the nitty gritty details of implementing clustering algorithms on Spark. 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.

Given two points $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$, such that $\boldsymbol{a} = [a_1, a_2 \cdots a_d]$ and $\boldsymbol{b} = [b_1, b_2 \cdots b_d]$ the *Euclidean distance* or L^2 *distance* between $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as:

$$\|\boldsymbol{a} - \boldsymbol{b}\|_2 = \sqrt{\sum_{i=1}^{d} |a_i - b_i|^2} \tag{1}$$

The *Manhattan distance* or L^1 *distance* between a and b is defined as:

$$\|\boldsymbol{a} - \boldsymbol{b}\|_1 = \sum_{i=1}^{d} |a_i - b_i| \tag{2}$$

The k**-means algorithm** aims to partition $\mathcal{X}$ to k clusters by defining a set $\mathcal{C} = \{\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(k)}\}$ of k centroids and partitioning $\mathcal{X}$ to k clusters $\mathcal{P} = \{P_1, \ldots, P_k\}$, so to minimize the sum of **squared** L^2-distances between every point and the centroid associated with its cluster. Formally, k-means tries to solve the following minimization problem:

$$\min_{\mathcal{C},\mathcal{P}} \Phi(\mathcal{C}, \mathcal{P}) = \min_{\mathcal{C},\mathcal{P}} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in P_i} \|\boldsymbol{x} - \boldsymbol{c}^{(i)}\|_2^2$$

where $\Phi(\mathcal{C}, \mathcal{P}) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in P_i} \|\boldsymbol{x} - \boldsymbol{c}^{(i)}\|_2^2$ is the "cost" associated with the set of centroids $\mathcal{C}$ and the partition $\mathcal{P}$.

For a fixed set of centroids $\mathcal{C}$, the cost function Φ is minimized by assigning every point to the centroid closest to it (in L^2). Thus, the cost $\phi(\mathcal{C})$ associated with $\mathcal{C}$ is:

$$\phi(\mathcal{C}) = \min_{\mathcal{P}} \Phi(\mathcal{C}, \mathcal{P}) = \sum_{x \in \mathcal{X}} \min_{\boldsymbol{c} \in \mathcal{C}} \|\boldsymbol{x} - \boldsymbol{c}\|_2^2 \tag{3}$$

For a fixed partition $\mathcal{P}$, the cost function Φ is minimized by setting the centroid of every cluster to be the mean of all the points in that cluster. Thus, the algorithm operates as follows: Initially, k centroids are initialized. Thereafter, alternately, given the set of centroids, each point is assigned to the cluster associated with the nearest centroid; and the centroids are recomputed based on the assignments of points to clusters. The above process is executed for several iterations, as is detailed in Algorithm 1.

Algorithm 1 k-means Algorithm

```
1: procedure k-MEANS
2:     Select $k$ points as initial centroids of the $k$ clusters.
3:     for iteration := 1 to MAX_ITER do
4:         for each point $x$ in the dataset do
5:             Cluster of $x \leftarrow$ the cluster with the closest centroid to $x$
6:         end for
7:         for each cluster $P$ do
8:             Centroid of $P \leftarrow$ the mean of all the data points assigned to $P$
9:         end for
10:        Calculate the cost for this iteration.
11:    end for
12: end procedure
```

Instead of minimizing the cost function Φ, which is defined using the squared L^2 distance, we can minimize a cost function defined using the L^1 distance, namely:

$$\Psi(\mathcal{C}, \mathcal{P}) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in P_i} \|\boldsymbol{x} - \boldsymbol{c}^{(i)}\|_1$$

Note that in this case the distance is **not** squared.

Still, for a fixed set of centroids $\mathcal{C}$, the cost function Ψ is minimized by assigning every point to the centroid closest to it (but this time in L^1). Thus the cost $\psi(\mathcal{C})$ associated with $\mathcal{C}$ is:

$$\psi\left(\mathcal{C}\right) = \min_{\mathcal{P}} \Psi(\mathcal{C}, \mathcal{P}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \min_{\boldsymbol{c} \in \mathcal{C}} \|\boldsymbol{x} - \boldsymbol{c}\|_1 \tag{4}$$

In contrast to Φ, for a fixed partition $\mathcal{P}$, the cost function Ψ is minimized setting the centroid of every cluster to be the **median** in every dimension of the points in that cluster. Thus, we obtain a variation on k-means, which is known as **k-medians**.

Please use the dataset from `kmeans/data` within the bundle for this problem.

The folder has 3 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 the first centroid $\boldsymbol{c}^{(1)}$ randomly, and then finding the point $\boldsymbol{c}^{(2)}$ that is farthest from $\boldsymbol{c}^{(1)}$, then selecting $\boldsymbol{c}^{(3)}$ which is farthest from $\boldsymbol{c}^{(1)}$ and $\boldsymbol{c}^{(2)}$, and so on).

Set the number of iterations (`MAX_ITER`) to 20 and the number of clusters k to 10 for all the experiments carried out in this question. Your driver program should ensure that the correct amount of iterations are run.

Note: Please ensure that your solution works correctly with duplicate data points. If not, your computation might be slightly off.

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

Implement k-means using Spark and compute the cost function $\phi(i)$ (Equation 3) after every iteration $i = 0, 1, \ldots$[1]. This means that for your 0th 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`.

Hint: Note that you do not need to write a separate Spark job to compute $\phi(i)$. You should be able to calculate costs while partitioning points into clusters.

Hint 2: The cost after 5 iterations starting from the centroids in `c1.txt` is approximately $460,538,000$.

1. **[8 pts]** Generate a graph where you plot the cost function $\phi(i)$ as a function of the number of the iteration i=0,. . . ,20 for `c1.txt` and also for `c2.txt`. Label your axes and add a legend for the subgraphs for `c1.txt` and `c2.txt`.

2. **[7 pts]** By how many percent does the cost change over the first 10 iterations of k-means when initializing using `c1.txt` and when using `c2.txt`? (e.g. if the cost before the first iteration was 10 and after the 10th iteration it is 7, then it decreased by 30%).

   Explain which initialization method is better in terms of the cost $\phi(i)$, and your intuition as to why.

   Hint: The cost for c1 decreases by ∼26% after 10 iterations.

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

Implement k-medians using Spark and compute the cost function $\psi(i)$ (Equation 4) after every iteration $i = 0, 1, \ldots$ [2]. This means that, for your 0th iteration, you'll be computing the cost function using the initial centroids located in one of the two text files. Run the k-medians on `data.txt` using `c1.txt` and `c2.txt`.

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

[1] We are overloading the notation here. The more precise way to express $\phi(i)$ would be $\phi(\mathcal{C}_i)$, where $\mathcal{C}_i$ is the set of centroids after iteration i.

[2] Same as footnote 1

Hint 2: The cost after 5 iterations starting from the centroids in **`c1.txt`** *is approximately* $412,012$.

1. [**8 pts**] Generate a graph where you plot the cost function $\psi(i)$ as a function of the number of the iteration i=0,. . . ,20 for `c1.txt` and also for `c2.txt`. Label your axes and add a legend for the subgraphs for `c1.txt` and `c2.txt`.

2. [**7 pts**] By how many percent does the cost change over the first 10 iterations of k-medians when initializing using `c1.txt` and when using `c2.txt`?

   Explain which initialization method is better in terms of cost $\psi(i)$, and your intuition as to why.

   Hint: The cost for c1 decreases by ∼26% after 10 iterations.

What to submit:

(i) Upload the code for 2(a) and 2(b) to Gradescope (Your solution may be in either one or two files).

(ii) A plot of cost vs. iteration for two initialization strategies for 2(a).

(iii) Percentage of change and your explanation for 2(a).

(iv) A plot of cost vs. iteration for two initialization strategies for 2(b).

(v) Percentage of change values and your explanation for 2(b).

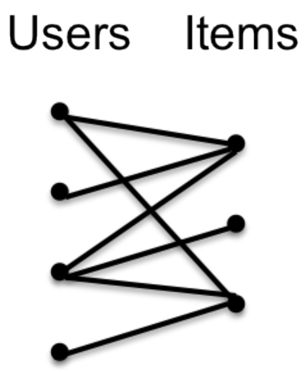
3 Recommendation Systems (30 points)

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.

(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.).



$$R=\begin{bmatrix}1&0&1\\1&0&0\\1&1&1\\0&0&1\end{bmatrix}$$

$$P=\begin{bmatrix}2&0&0&0\\0&1&0&0\\0&0&3&0\\0&0&0&1\end{bmatrix}\quad Q=\begin{bmatrix}3&0&0\\0&1&0\\0&0&3\end{bmatrix}$$

Figure 1: User-Item bipartite graph.

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

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

(b) [**6 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. Show that $S_I = Q^{-1/2} R^T R Q^{-1/2}$, where $Q^{-1/2}$ is defined by $Q_{rc}^{-1/2} = 1/\sqrt{Q_{rc}}$ for all nonzero entries of the matrix, and 0 at all other positions.

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. That is, your expression for S_U should also be in terms of some combination of R, P, and Q. Your answer should be an operation on the matrices, in particular you should not define each coefficient of S_U individually.

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 $\frac{1}{2}$.)

(c) [5 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.

Hint: For the item-item case, $\Gamma = RQ^{-1/2}R^T RQ^{-1/2}$.

Your answer should show how you derived the expressions (even for the item-item case, where we give you the final expression).

(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.

Use the dataset from `recommendation/` within the bundle for this problem.

The folder 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 500th 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.

- Compute the matrices P and Q.
- Using the formulas found in part (c), 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.

What to submit:

(i) Interpretation of T_{ii} and T_{ij} [for 4(a)]

(ii) Expression of S_I and S_U in terms of R, P and Q and accompanying explanation [for 4(b)]

(iii) Expression of Γ in terms of R, P and Q and accompanying explanation [for 4(c)]

(iv) The answer to this question should include the followings: [for 4(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 item-item collaborative filtering
- Upload the source code via the SNAP electronic submission website