CMSC 25025 / STAT 37601

Machine Learning and Large Scale Data Analysis

HW 2

Due: Tuesday, April 4, 2023 at 2:00 pm.

Please hand in this Homework as follows:

Upload to Gradescope `HW2`:
A pdf of the theoretical homework.

Upload the ipynb file for problem 3 to the Gradescope `HW2 programming` .

1. *PCA* (25 points)

   The problem of fitting the best k-dimensional subspace to data $x_1, \ldots, x_n \in \mathbb{R}^d$ can be written as the optimization

   $$\min_{\mu, \{\lambda_i\}, V_k} \sum_{i=1}^{n} \|x_i - \mu - V_k \lambda_i\|^2$$

   where V_k is an $d \times k$ orthonormal matrix.

   (a) Show that for fixed V_k an optimum over the variables $\mu \in \mathbb{R}^d$ and $\lambda_i \in \mathbb{R}^k$ is given by

   $$\widehat{\mu} = \overline{x}_n = \frac{1}{n} \sum_{i=1}^{n} x_i$$
   $$\widehat{\lambda}_i = V_k^T (x_i - \overline{x}_n).$$

   Show that $\widehat{\mu}$ is not unique, and characterize the set of possible solutions.

   (b) In class we showed that once we have $\widehat{\mu}$ and $\widehat{\lambda}_i, i = 1, \ldots, k$ for a fixed V_k the PCA problem reduces to maximizing $tr(V_k^t C V_k)$, where C is the sample covariance matrix and the maximum is over all $d \times k$ orthonormal matrices. Let C be a non-negative $d \times d$ symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_d$. Let U be the orthogonal matrix of eigenvectors of C. Let $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d)$ be the diagonal matrix of the eigenvalues so that $C = U \Lambda U^t$. Show that the $d \times k$ matrix U_k of the first k columns of U maximizes:

   $$tr(V_k^t C V_k),$$

   over all orthonormal $d \times k$ matrices.

2. *Ncuts and eigenvalue problems:* (25 points) We have a weighted graph with a set V of n vertices and symmetric weight matrix $W_{ij}, i, j = 1, \ldots, n$. The edge set of the graph is $E = \{(i, j) : W_{ij} > 0\}$. with non-negative entries. Define the degree of a vertex as $d_i = \sum_j W_{ij}$. The degree of a subset A of vertices is the sum of their degrees: $d(A) = \sum_{i \in A} d_i$, and $d = (d_1, \ldots, d_n)$. Let A, B be a partition of the n nodes and define the normalized cut

$$\text{Ncut}(A, B) = \text{cut}(A, B) \left(\frac{1}{d(A)} + \frac{1}{d(B)} \right),$$

where $\text{cut}(A, B) = \sum_{i \in A, j \in B} W_{ij}$.

   (a) Define the Laplacian of the graph as $D - W$, where $D = \text{diag}(d)$. Show that $f^t L f = \frac{1}{2} \sum_{ij} W_{ij}(f_i - f_j)^2$. And explain why the constant vector of 1's, $\mathbf{1}$ is the eigenvector of L with eigenvalue 0.

   (b) Assume the graph is connected, i.e for any two vertices i, j there is a path in the graph between i and j. Show that any eigenvector with eigenvalue 0 has to be of the form $c\mathbf{1}$ for some $c \in R$.

   (c) Let $f \in R^n$ have only two values, i.e. $f_i \in \{a, b\}, a \neq b$. Let $A = \{i : f_i = a\}, B = A^c$. Assuming $f^t d = 0$, what is the unique value of a/b.

   (d) Let f be as in **??**, show that

$$\frac{f^t L f}{f^t D f} = Ncut(A, B).$$

   And conclude that finding the minimum Ncut is equivalent to minimizing

$$\frac{f^t L f}{f^t D f}, \text{ s.t. } f^t D \mathbf{1} = 0, \text{ and } f_i \in \{a, b\}.$$

   (e) Show that f_* minimizes the relaxed problem (i.e. dropping the last constraint) if and only if $f_* = D^{-1/2} u_*$ where u_* minizes $\frac{u^t \widetilde{L} u}{u^t u}$ subject to $u^t D^{1/2} \mathbf{1} = 0$, where $\widetilde{L} = I - D^{-1/2} W D^{-1/2}$.

   (f) Show that $D^{1/2}\mathbf{1}$ is the eigenvector of $\widetilde{L}$ with eigenvalue 0, and u^* is the eigenvector of $\widetilde{L}$ with second smallest eigenvalue.

3. *Analysis of SOU similarities* (50 points)

   In this problem you will use the classic *vector space model* from information retrieval to find similar SOU addresses. Your code from Assignment 1 may come in handy here.

In the vector space model, a document of words d is represented by a TF-IDF vector $\mathbf{t}(d) = (t_1(d), t_2(d), \ldots, t_V(d))$ of length V, where V is the total number of words in the vocabulary. The TF-IDF weights are given by

$$t_i(d) = n_i(d) \cdot \log\left(\frac{|\mathcal{D}|}{\sum_{d' \in \mathcal{D}} \mathbb{1}(v_i \in d')}\right)$$

where $n_i(d)$ is the number of times term v_i appears in document d, $\sum_{d' \in \mathcal{D}} \mathbb{1}(v_i \in d')$ is the number of documents that contain term v_i, and $|\mathcal{D}| = \sum_{d' \in \mathcal{D}} 1$ is the total number of documents in the collection $\mathcal{D}$. This weighting scheme favors terms that appear in few documents, but frequently in the current document. The tf-idf can be motivated with concepts in Information Theory that will be introduced later in the course.

(a) Compute the TF-IDF vectors for each SOU address. You should lower case all of the text, and remove punctuation. For example, you could use something like this:

```
import string
import re

def clean_and_split(s):
    # string.punctuation is a string with all the different punctuation symbols.
    # 'tr' is an object that will assign blank space to each punctuation symbol.
    tr=str.maketrans(string.punctuation," "*len(string.punctuation))
    s=s.lower().translate(tr)
    # replace \r\n
    s = re.sub('(\r\n)+',' ',s)
    # replace whitespace substrings with one whitespace and remove leading/trailing whitespac
    s = re.sub(' +',' ',s.strip())
    s=s.split(' ')
    return s
```

You will have to make choices about the size of the term vocabulary to use—for example throwing out the 20 most common words, and words that appear fewer than, say, 50 times, and removing numbers.

(b) A similarity measure between documents is

$$\text{sim}(d, d') = \frac{\mathbf{t}(d) \cdot \mathbf{t}(d')}{\|\mathbf{t}(d)\|\|\mathbf{t}(d')\|},$$

the cosine of the angle between the corresponding TF-IDF vectors. In terms of this measure, find the

- 50 most similar pairs of SOUs given by different Presidents.
- 50 most similar pairs of SOUs given by the same President.
- 25 most similar pairs of *Presidents*, averaging the cosine similarity over all pairs of their SOUs.

When you read the above speeches, do they indeed seem similar to you? (You can read the speeches in a more reader-friendly format here: `http://www.presidency.ucsb.edu/sou.php`) Comment on what you find, and describe what is needed to construct a better similarity measure between documents.

(c) Using this vector representation, cluster the speeches using k-means. To do this you can use the `sklearn` implementation of k-means:

```
from sklearn.cluster import KMeans
```

then we can run the method `KMeans.train`:

```
model = KMeans(n_clusters=c)
sou_clust=model.fit(tf_idf_mat)
```

The options here limit the number of iterations of kmeans to 50, the number of clusters to 10, the clusters are initialized randomly.

Experiment with different number of clusters, and display the clusters obtained (in some manner that you choose). Comment on the clustering results, and whether or not the results are interpretable.

(d) Now try performing spectral clustering using the tf-idf vectors. Compute the matrix A of distances between pairs of vectors and then compute $\widetilde{W}$ as defined in class. A rule of thumb for the bandwidth h is to use the mean of A, but you should experiment with a few values around that as well. Its a good idea to compute A ahead of time, and then get `W=exp(-A/h)` for different values of h.

i. Plot the top 30 eigenvalues of $\widetilde{W}$, (the top eigenvalue should be 1.)

ii. Take top three eigenvectors u_1, u_2, u_3 multiply them on the left by $D^{-1/2}$: $v_i = D^{-1/2}u_i$, and plot the three vectors. The first should be constant corresponding to the eigenvalue 1.

iii. Plot a scatter plot of v_2, v_3, and describe what you observe. Why do you think the scatter plot looks like this?

iv. Another possibility is to do the spectral clustering on the unit length vectors: $\frac{\mathbf{t}(d)}{\|\mathbf{t}(d)\|}$. Perform the same spectral analysis as before, and have the year of each dot printed next to it (you will need a larger figure, so use the command: `plt.figure(figsize=(10,10))`.) What do you observe in this spectral projection? Why is the analysis of the unit length vectors so different from the analysis of the original vectors?