67750: Advanced Practical Machine Learning	2018/9

Exercise 4 — Clustering

Dr. Tommy Kaplan — *TA: Daniel Gissin*

1 Theoretical Questions (5 Points)

1.1 Convergence of k-means

Prove that k-means converges in a finite number of steps. (Hint: what happens to the cost function $\sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$ at each iteration?)

1.2 Suboptimality of k-means

1. Give an example of data points and an initialization of k-means which converge to a suboptimal solution (a local minima of the cost function).
2. Give an example of data points that have more than one optimal clustering solution (with respect to the cost function).

1.3 Initializations of k-means

Suppose we want to find a globally optimal solution of a k-means clustering problem. We'll assume that if the algorithm is initialized such that every cluster is represented in the random starting points then the algorithm will converge to the optimal solution. That is, a "good" event is one where each example in the initial guess is from a different cluster. Denote the probabilities of sampling a point from each cluster by $\alpha_1, ...\alpha_k$, and:

1. Calculate the expected number of trials we have to perform before we sample a good initialization event (as a function of k, α_i). Do this by calculating the probability of a single "good event", then use the fact that the number of trials is a geometric random variable to get the expectation easily.
2. Give a lower bound on this number as a function of k. Do this by finding the best possible α variables, subject to $\sum_i \alpha_i = 1$.
3. Is this a plausible number of start trials? If not, how can we explain the fact that K-Means often works?

2 Practical Exercise

Please submit a single tar file named "ex4_<YOUR_ID>". This file should contain your code, along with an "Answers.pdf" file in which you should write your answers and provide all figures/data

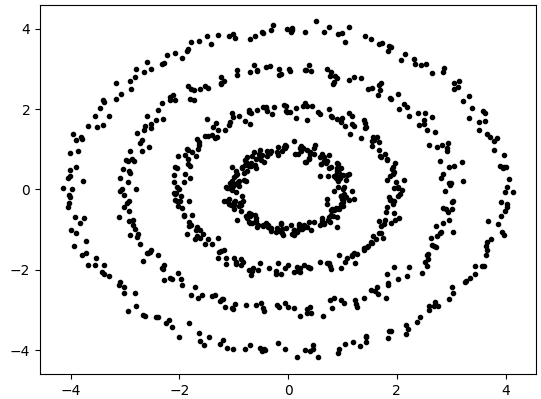

Figure 1: Circles Data

to support your answers (write your ID in there as well, just in case). Your code will be checked manually so please write readable, well documented code.

If you have any constructive remarks regarding the exercise (e.g. question X wasn't clear enough, we lacked the theoretical background to complete question Y, the exercise was too long and question Z felt like a lot of work with little knowledge gained...) we'll be happy to read them in your Answers file.

2.1 Exercise Requirements

1. Implement a K-means++ function and test it on synthetic data (30 points)
2. Write a spectral clustering function and test it on synthetic data (30 points)
3. Demonstrate that you can use the "elbow" method or Silhouette in order to choose k correctly for synthetic data (10 points)
4. Apply your K-means and spectral clustering to the microarray data set. Use K-selection methods to find significant clusters. Visualize the results and the selection process (15 points)
5. Familiarize yourself with the t-SNE algorithm using MNIST and a toy dataset (10 point)

2.2 Data Sets

This exercise contains three data sets for you to use, and you are welcome to create your own synthetic data sets.

2.2.1 Circles Data

An example of this data set is generated in the supplied function "circles_example". It creates four circles and allows for a demonstration of the differences between K-Means and Spectral Clustering.

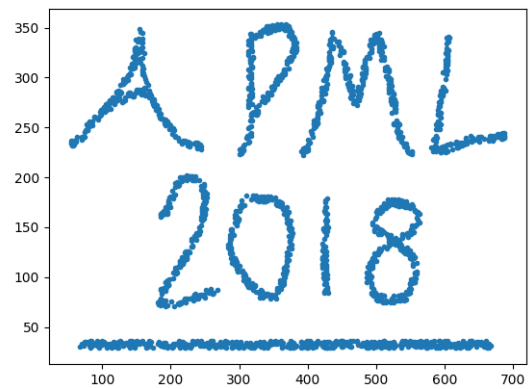

Figure 2: APML Data

2.2.2 APML Picture

Another provided data set can be seen in the supplied function "apml_pic_example", and is given in the supplementary file "APML_pic.pickle". This data set can also be used to show the differences between K-Means and Spectral Clustering.

2.2.3 Microarray

This is a biological data set of high dimensional data that we can test our clustering skills on. An exploration of the data set is provided in a separate PDF file, and in the supplied function "microarray_exploration".

2.3 K-means

In this section we'll **implement the classic K-means algorithm**. Generally, to fully specify the K-means optimization problem, one has to choose a distance metric. We'll implement K-means for euclidean distances (which makes the centroid optimization step simple), but you may experiment with other options if you wish. We'll also **implement the initialization method mentioned in class (K-means++)**. See the function header for more details.

2.4 Spectral Clustering

In this section we'll wrap the k-means function already written with the spectral transformations described in class (see header file for more details). We'll use the symmetric graph Laplacian described in class:

$$L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

Where W is the similarity matrix, and D is the diagonal degree matrix.

Additionally, we need to define the similarity transformation on the data. For this we consider two alternatives:

1. *Symmetric M nearest neighbors*:

$$w_{nn}(x_i, x_j) = \mathbb{1}_{\{x_i \text{ is in } x_j\text{'s m nearest neighbors, } \textbf{or} \text{ vice versa}\}}$$

   You may experiment with the mutual M nearest neighbors if you prefer: $w_{mutual-nn}(x_i, x_j) = \mathbb{1}_{\{x_i \text{ is in } x_j\text{'s m nearest neighbors, } \textbf{and} \text{ vice versa}\}}$

2. *The Heat Kernel*:

$$w_h(x_i, x_j) = e^{-\frac{d(x_i,x_j)^2}{2\sigma^2}}$$

   Where m and σ are parameters of the similarity functions.

Recall that the similarity measure corresponds to the weights on the similarity graph, and we want to somehow approximate the ideal graph with k connected components. As discussed in class, you can plot a histogram of the distances in the data and choose σ as a relevant percentile of the distances.

Implement the spectral clustering wrapper for K-means and execute it on synthetic data using the two similarity transformations. Demonstrate the effect of the parameters (it may be helpful to plot a histogram of the distances in order to choose σ - provide this histogram in your pdf if you indeed use it to choose σ).

Explore the differences between K-Means and Spectral Clustering on the given synthetic datasets. Is the similarity graph a good representation for clustering?

2.4.1 Plotting the Similarity Graph Itself

In spectral clustering, the similarity graph W represents the weights between edges in a graph. One can plot these weights by plotting the matrix (using the matplotlib.pyplot.imshow function for example) to see how every sample from the data is connected to other samples.

To see how spectral clustering uncovers the hidden connected components of the similarity graph, begin by shuffling the data and plotting W. You should see connections without a clear structure. Now, sort the original data according to the clusters given by spectral clustering and calculate W again, over the sorted data. **Compare and discuss the results you got for one of the synthetic datasets**.

2.4.2 Pseudo Code of Spectral Clustering

Given a data matrix:

1. Calculate the distance matrix $S_{i,j} = ||x_i - x_j||$.
2. Move from distances S to similarities W (using the Gaussian kernel or the nearest neighbors).
3. Calculate the diagonal degree D matrix: $D_{i,i} = \sum_j W_{i,j}$.

4. Create the Laplacian matrix $L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.
5. Decompose L into its eigenvectors and select only the ones corresponding to the k lowest eigenvalues.
6. Project the rows of the k eigenvectors onto the unit sphere (normalize the rows, where the columns are the eigenvectors).
7. Run k-means the rows of the representation and return the clustering (row i corresponds to the original x_i).

2.5 Choosing K

We've discussed in class about a few measures we can use to choose the correct number of clusters. You will implement and use these in the exercise. **Provide a demonstration of the effectiveness of one of these measures on synthetic data and discuss the results**.

2.5.1 The "Elbow" Method

We can run k-means with different k's several times, get the best clustering for each k and then plot the best cost function as a function of k. The cost function should decrease as we increase k, but at some k we expect the decreasing to plateau (this is the "elbow").

2.5.2 Silhouette

A second heuristic for choosing k which is a bit more methodical than the "elbow" method is silhouette. In this method, we would like to punish situations where an example is close to a cluster which it isn't a part of. This means we would like to reward in-cluster cohesion and separation between different clusters.

We will have two values. The first, a_i, will give us a measure of how well example i is assigned to its own cluster. The smaller a_i is, the better:

$$a_i = \frac{1}{|C(x_i)| - 1} \sum_{x_j \in C(x_i)} d(x_i, x_j)$$

In words, a_i is the average distance between x_i and the other points within its own cluster. Next, we will define b_i to give us a similar measure of how close it is to the closest other cluster. We would like this value to be as large as possible.

$$b_i = \min_{k \in \{l | x_i \notin C_l\}} \frac{1}{|C_k|} \sum_{j \in C_k} d(x_i, x_j)$$

In words, this is the average distance between x_i and the other points within the closest cluster to x_i which isn't its own cluster.

Now, we will combine these two values and get the silhouette score:

$$S(k) = \frac{1}{N} \sum_{i=1}^{N} \frac{b_i - a_i}{\max(a_i, b_i)}$$

The silhouette score is between -1 and 1, where a high score means that the examples are very close within their clusters (small a_i values) and very distant from other clusters (high b_i values). We can plot S as a function of k and look for the k which gives us the maximum Silhouette.

2.5.3 Eigen Gap

Once you've chosen a good σ or m for your spectral clustering algorithm, you can use the eigenvalues of the Laplacian in order to estimate k. This can be done by finding the first "gap" in the eigenvalues, where they start to be bigger.

2.6 Biological Clustering

Apply your K-means and spectral clustering algorithms, along with the parameter selection methods (k, σ) to the microarray dataset. Visualize and discuss the results. Were you able to detect meaningful clusters? How did you decide how many clusters to look for?

The expected number of clusters for this dataset shouldn't exceed 15, so don't try to cluster the dataset into 200 clusters...

2.7 t-SNE

To get familiar with t-SNE, play around with it on the MNIST data set and on a synthetic high dimensional data set of your choice (produce a high dimensional data set with some structure). If you want to make things go faster, you may use the smaller MNIST dataset of 8×8 images - you can load it from scikit-learn here.

Show results and discuss the effectiveness of t-SNE on the data. You may use the scikit-learn implementation of t-SNE, or any other implementation of your choosing.

Compare the embedding you got from t-SNE to the embedding you get from PCA (like you did in exercise 3 with the autoencoder) - does t-SNE capture the local structure of the data better than PCA?