PR4: Clustering

CS 182/282 Spring 2024

Released: Wednesday, April 24nd, 2024

Due: 11:59pm on Wednesday, May 8th, 2024

Overview

Clustering is an important technique in unsupervised learning that enables inference of complex patterns from large data sets. It has many useful applications in the field of computational biology, from gene expression analysis to population genetic inference to document categorization, and diverse methods have been developed to extract meaning from large-scale bioinformatic data. In this project, you will implement two algorithms to cluster high-dimensional data.

Reading

- A Tutorial on Spectral Clustering (von Luxburg, 2007)

Gradescope

To hand in your project, submit all of your project files to Gradescope.

The Data

For this project, you'll receive three different datasets, all in csv files. In particular, in the stencil from the website, you will find `basic_clusters.csv`, `highdim_clusters.csv`, and `moons_clusters.csv`. `basic_clusters.csv` has 2-dimensional data with 2 clusters on a set of 500 samples. `highdim_clusters.csv` has higher dimensional data for 300 "genes" on a set of 500 "cells" clustered into 5 groups, which we have also generated from a large number of multivariate normal distributions (one distribution per cluster per gene!). `moons_clusters.csv` has two dimensional data that form a pair of interlocking moons (what kind of clustering should this support?).

Implementation

You are free to use existing programming libraries to automate basic mathematical computations or linear algebra manipulations in addition to solving for the eigenvectors of a square matrix; however, **you must implement all algorithmic steps yourself**. In particular, you may **not** make use of any built-in implementations of any existing clustering algorithms. You may, however, use built-in functions for *visualization purposes only* (e.g., to generate a heatmap view of how cells compare to each other based on some distance/similarity metric, or to gauge the spread of your inferred clusters on a PCA or t-SNE plot).

Roadmap

In this project, you will begin by implementing k-means clustering, which you will use to cluster the 2-D and 300-D (but normal) data we have given you. Next, you will implement spectral clustering, which you will use to cluster all the data we have given you. Finally, you will submit a pdf with visualizations of your results.

P1: *k*-Means Clustering (30 points)

Your first task is to cluster the data set using k-means clustering. You should submit a shell script called `kmeans.sh` which can be run as follows:

```
$ sh kmeans.sh <data.csv> <k>
```

`data.csv` is a num samples $\times$ num features matrix in csv format, and k is the number of expected clusters in the data (see above). Your program should print out the cluster labels as integers, each on a separate line. For example, if you cluster your data into 3 clusters, you should print out 0's, 1's, and 2's such that the nth line of your output corresponds to the cluster label for the data point in the nth row of the input data.

You may choose any method of initializing cluster centroids and optimizing k, provided that you are able to successfully cluster the normal data we have given you. We had success using a Euclidean distance similarity function (for comparing points with centroids) and initializing our centroids by choosing one random point, then successively choosing the k additional points that have the maximum [minimum distance to each existing centroid]. We suggest writing this in an extensible way, as you will need to call on it for spectral clustering below.

P2: Spectral Clustering (45 points)

Your second task is to cluster the data using spectral clustering. You should submit a shell script called `spectral.sh` which can be run as follows:

```
$ sh spectral.sh <data.csv> <k> <sigma>
```

Your program should follow the same output specifications as k-means.

You should implement the spectral clustering algorithm using the unnormalized graph Laplacian (see the "tutorial" linked above for insights). We suggest that you follow the outline below, although note that we require you to use Gaussian similarity to compute your Laplacian:

1. Construct a similarity matrix S that holds the Gaussian similarity between samples i and j at S_{ij}. In particular, if $\mathbf{x_i}$ and $\mathbf{x_j}$ are the feature vectors for samples i and j, you should construct your similarity matrix according to the equation

$$S_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$

   where σ is a tunable parameter that your program will take in.

2. Use S as your fully connected graph, and compute the Laplacian by the formula

$$L = D - W = diag[\sum_j S_{ij}] - S$$

   D should be a diagonal matrix with the row sums of S along the diagonal.

3. Find the first k eigenvectors (with smallest eigenvalues) of your newly computed L to produce U.

4. Perform k-means clustering on the rows of U, and assign cells to clusters accordingly.

P3: Report (25 points)

Your final task is to synthesize the results of your two clustering algorithms on the data we have provided you. In particular, you should submit a file called **application.pdf** that contains the following:

1. Plots illustrating the results of both your k-means and spectral clustering algorithms on each of the provided datasets (6 plots total). Note that your k-means algorithm will perform poorly on the `moons_clusters.csv` dataset, although your spectral clustering algorithm should be able to cluster it perfectly. However, you will have to tune your σ parameter to make this work. Think about how the value of σ changes the rate at which the Gaussian similarity between two points decreases as the points get further from each other.
    - For the 2-dimensional datasets, you can plot the data as-is using a package of your choice (examples include matplotlib for python and ggplot2 for R). Be sure to color the points based on their cluster label.
    - For the 300-dimensional dataset, first run PCA on the data and then plot along the first two principal components. You can do this using sklearn for python or the primcomp() function in R.
2. **Bonus:** Answer the following questions in your report. Why do we take the eigenvectors corresponding to the smallest eigenvalues of the laplacian when constructing our U matrix? Please explain this at a high level (no mathematical proof is required). *Hint: Think about the case in which our W matrix represents the binary adjacency matrix of a graph. In this case, if the graph has n separate connected components, the laplacian will have n 0-eigenvectors. What might these eigenvectors look like? Apply this same reasoning to the case where W is a similarity matrix for data points clustered into n clusters. Check out the reading for more info!*

Grading

The autograder will test the ability of your k-means and spectral clustering scripts to cluster the datasets we provide. For full credit, both of your scripts should be able to cluster the data in `basic_clusters.csv` and `highdim_clusters.csv`. Only your spectral clustering script will be able to cluster `moons_clusters.csv` (a suitable value of σ is provided by the autograder). Therefore there are 5 autograder tests worth 15 points each. For full credit on the report you must include and clearly label all 6 plots.