Spectral Clustering

By

Jack Melchert
(jmelchert@wisc.edu)

Joseph Fortman
(jfortman@wisc.edu)

Han Xiao
(hxiao46@wisc.edu)

Executive Summary

This activity examines spectral clustering, an unsupervised machine learning algorithm which groups unlabeled data by similar features. It improves upon simple unsupervised algorithms like k-means by grouping the data points using representations of the underlying structures within the data.

By the end of this activity, students will:

- Have a general understanding of the concepts used in spectral clustering, including adjacency matrices, Laplacian matrices, and k-means clustering.

- Understand and complete a spectral clustering algorithm example and observe its benefits over other unsupervised learning algorithms through visualizations.

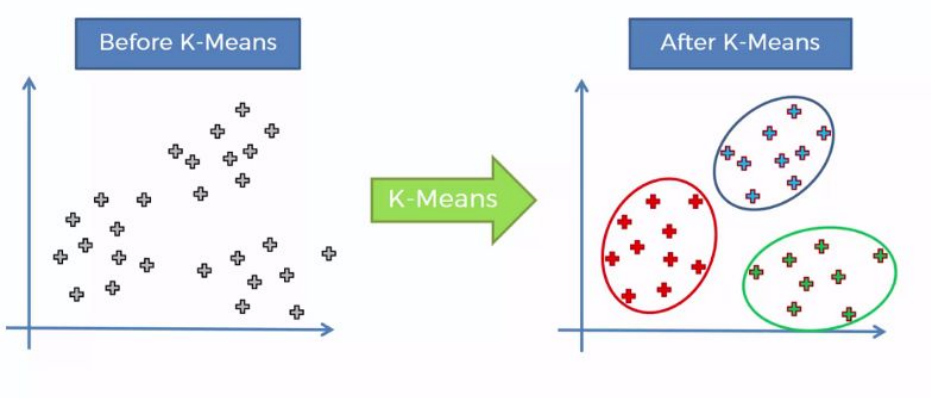
- Be able to identify why spectral clustering can give better results than k-means and in which situations to use either algorithm.

For most of the semester, students have dealt with supervised learning algorithms. Therefore, this project will interest students as it broadens their understanding of the applications of machine learning. Moreover, spectral clustering is conceptually simple enough to understand after completing this activity, while still being an extremely useful machine learning algorithm.

Background

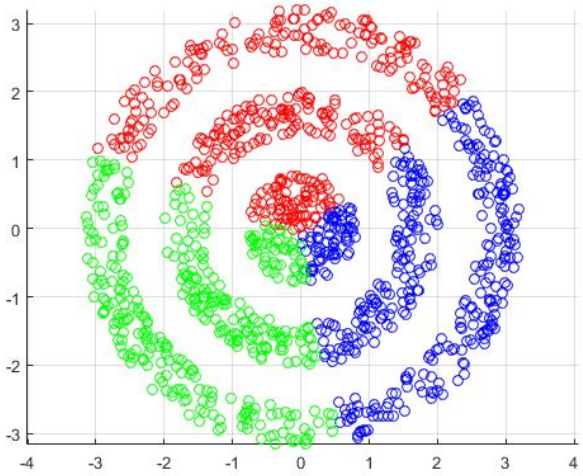
There are two main categories of machine learning algorithms: unsupervised learning and supervised learning. In supervised learning, the input data has information classifying each sample. In contrast, unsupervised learning algorithms will learn patterns in a set of data without being given explicit classification of the data. An example of this would be to construct an algorithm to identify weather patterns. Suppose you organized a training data set that consisted of temperature, wind speed, cloudiness, and precipitation. While the input contains no information about how these features are related, the algorithm would construct groups that would highlight these relationships. It would find that precipitation and cloudiness are highly correlated, and that high temperature and low wind are also correlated. Unsupervised learning is useful for grouping unlabeled data into categories.

The most popular unsupervised learning algorithm is called k-means clustering. First, the algorithm roughly divides the data samples into groups based on similar features (in the examples given in this activity, those features will be the horizontal and vertical coordinates). Then, it iteratively improves the boundaries of those groups based on the mean values of the features contained in each group. At the end of this process, there will be well defined groups that have distinct characteristics.



An example of k-means clustering with 3 clusters (Patil, 2018).

K-means is very effective when the group of points are clearly separated in p dimensional space, where p is the dimension of the original data. However, if clusters are more complex, k-means might give inaccurate results.



K-means clustering of more complex cluster structure.
It is clear that the real clusters have not been identified.

While the k-means algorithm found three clusters with a small distance between the points within the clusters, for this dataset an alternative approach might be more effective. Spectral clustering is one such approach.

Spectral clustering originated in the 1990s as scientists attempted to improve computer vision applications. Before the name "spectral clustering" was coined, improved algorithms were generally described as segmentation methods based on eigenvectors of an adjacency matrix (Weiss, 1999). The first successful algorithm of this type may have been the Scott and Longuet-Higgins (SLH) algorithm published in 1991 (Scott et al, 1991).

Overall, the goal is similar to k-means: group similar data points together and distinguish dissimilar data points from each other. While this may be an obvious task when discerning shapes in an image, it is also a common first step when analyzing empirical data in biology, psychology, statistics, data mining, and more (Suryanarayana et al, 2015). Thus, spectral clustering is implemented as a useful, accurate way to simply and efficiently find trends in data, especially when k-means fails.

Modern spectral clustering algorithms often vary in their implementation, but they generally only take a few steps to complete. Although these steps are not always intuitive, a walk-through of a simple spectral clustering algorithm is provided next and the subsequent activities will show how spectral clustering succeeds where other clustering algorithms fail.

Before introducing the spectral clustering algorithm, we will define some terms that may be unfamiliar:

An *adjacency matrix*, A, is a square matrix of size N by N, where N is the number of points in the original graph G. Every entry $A_{i,j}$ in the adjacency matrix represents the edge between points i and j. If an edge exists between these points, then $A_{i,j}$ is 1, otherwise it is 0.

A *degree matrix*, D, is also a square matrix of size N by N. It is a diagonal matrix where every diagonal entry $D_{i,i}$ represents the number of edges connected to point i. An element in the degree matrix can be computed easily from the adjacency matrix by summing every element in the corresponding row in A.

A *Laplacian matrix*, L, is a useful representation of a graph that has been utilized in many areas of mathematical research (Mohar, 1991). It can be computed in unweighted form by subtracting the adjacency matrix from the degree matrix. One especially interesting trait of Laplacian matrices is how their second smallest eigenvector can be related to connectivity, expanding properties, maximum cut, and many other graph properties. While an in depth discussion of what Laplacian matrices are and their many different forms is out of the scope of this activity, it is sufficient to understand that the Laplacian of a graph contains the general information about what the neighborhood of each point looks like. The eigenvectors of this matrix therefore contain information about the underlying structures in the graph.

The Basic Spectral Clustering Algorithm

Given: A graph G with N nodes

(1) **Compute an adjacency matrix *A*.** To create the adjacency matrix, calculate the distance

$$s(i,j) = (G_i - G_j)^2$$

between every pair of points in G. Then, for every distance less than ε, set the corresponding entry $A_{i,j}$ to 1, and every distance greater than or equal to ε, set the corresponding entry $A_{i,j}$ to 0. ε is chosen so that points that are relatively close together will have a connection in the adjacency matrix, and points far away from each other will not.

(2) **Compute a diagonal degree matrix *D*.** This is done summing the entries in each row of A:

$$D_{i,i} = \sum_{j=0}^{N} A_{i,j}$$

(3) **Compute the Laplacian Matrix *L***:

$$L = D - A$$

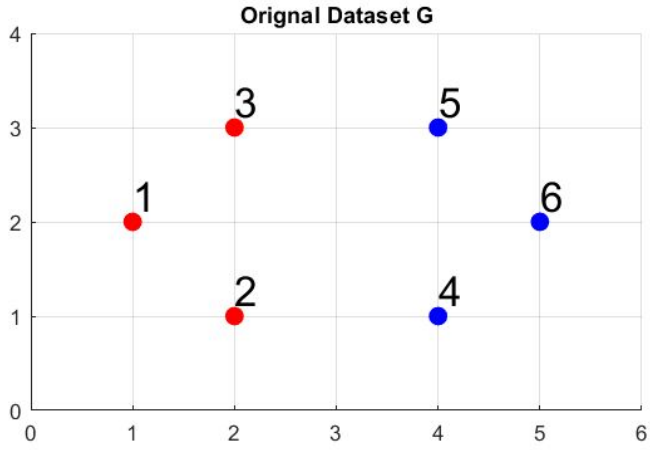
(4) **Compute the eigenvectors and eigenvalues of the Laplacian matrix.** This is done by computing the SVD and using the columns of *U* as eigenvectors and singular values in *S* as eigenvalues.

(5) **Run k-means clustering on the eigenvectors corresponding to the *k* <u>smallest</u> eigenvalues to find the labels for the points in *G*.** The Matlab method *kmeans* may be easily utilized here, where the inputs are (1) the *k* smallest eigenvectors organized as an *N* by *k* matrix, and (2) *k*, the parameter chosen to be the number of clusters that the graph will be partitioned into.

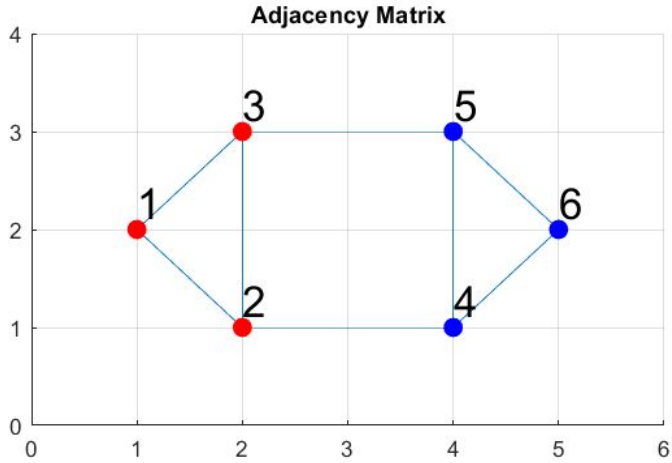
Clustering the *k* smallest eigenvectors of the Laplacian matrix will result in clusters that better represent the underlying structures within the dataset. Because the Laplacian matrix captures information about the neighbors of each point in the graph, the eigenvectors of this Laplacian can represent complex structures, like the structures present in the previous example.

Some variations of the basic spectral clustering algorithms have been introduced to improve the quality of clustering (Luxburg, 2007). There are different choices of similarity functions that are used when computing the adjacency matrix. We described the ε-neighbor graph, where the euclidean distance and a threshold are used to determine which points in the graph are "neighbors". Another popular approach is to use a k-nearest neighbor graph. This approach also uses the euclidean distance between all pairs of points, and point *i*'s neighbors are the *k* points with the smallest distance to *i.* Lastly, it is also common to compute a fully connected graph, where the adjacency matrix has all non-zeros entries calculated from the Gaussian similarity function $s(x_i, x_j) = exp(-\left\|x_i - x_j\right\|^2/(2\sigma^2)$. In all cases, there is a tuning parameter that has to be chosen to ensure that points close together in the original graph will have a connection in the adjacency matrix. We will examine how to compute this tuning parameter in the following activities.

Warm-up



1. Find the adjacency matrix of *G* by calculating the distance between every pair of points. If the distance between two points is less than ε, make a connection in the adjacency matrix. Experiment with different values of ε until you get the adjacency matrix given below. You will need to use Matlab to compute the SVD and k-means later on, so enter the adjacency matrix into Matlab now.



2. Find the degree matrix, *D*, of the adjacency matrix, *A*. Does it matter if you sum the columns or rows in *A*?
3. Compute the Laplacian, *L*, from *D* and *A*.
4. Use the SVD to find the eigenvectors and eigenvalues of *L*.
5. Use Matlab to run k-means clustering on the appropriate eigenvectors. (Hint: Read step (5) of the spectral clustering algorithm carefully) Does your result match the clusters in the original dataset? If not, make sure your adjacency matrix matches the one listed above, and make sure you are using the correct eigenvectors when running k-means.

Main activity

For this activity, please use the spectral_clustering.m script provided. We will ask you to run sections of the code at a time, do this by clicking "Run Section" or using the shortcut Ctrl+Enter.

1. Run the first two sections of the script. This will load the synthetically generated dataset, display the correct clusters, and run the k-means classification on the data. Notice that the clusters produced by k-means don't match the true clusters. Why does k-means struggle when classifying this dataset?

Next, you will examine the adjacency matrix in the third section of the code. The tuning parameter, epsilon, is an important component in the success of spectral clustering.

2. Try determining an appropriate epsilon value. (Hint: Looking at the original data, what distance would you think should be the cutoff for classifying points as neighbors?) Try setting epsilon to 0, and a various other extreme values. How does changing epsilon affect your adjacency matrix?

Finally, after getting an adjacency matrix that appropriately fits the data, look at the last section of the code. This section computes the Laplacian matrix and its SVD, and runs k-means on the resulting eigenvectors.

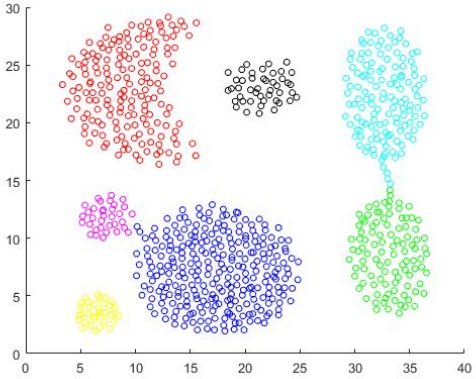
3. There is a line in the code that you must fill in before observing the results of spectral clustering. The variable "eigenvectors" is the input to the k-means algorithm. Use the result of the SVD to construct this matrix.

Notice how Spectral Clustering uses the k-means algorithm to make the cluster decisions.

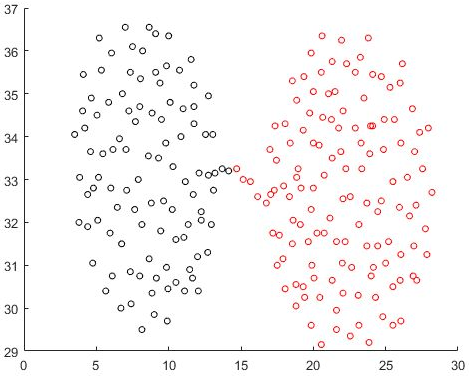
4. Why does running k-means on the k eigenvectors of the Laplacian produce better results than running k-means on the original data?

5. Here are some pictures of classification problems. Circle which ones spectral clustering would be more successful than k-means. Why?

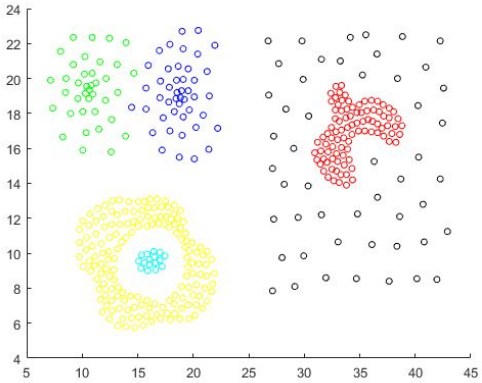
(a)



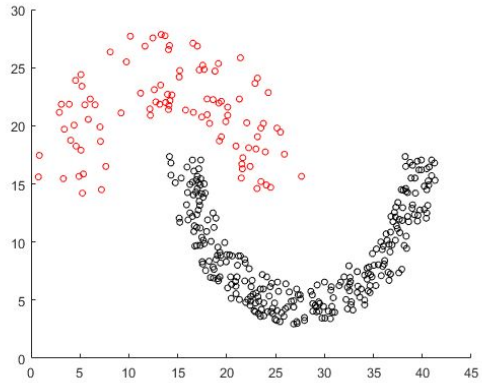
(b)



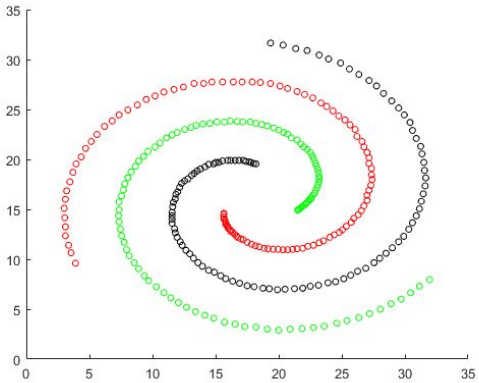
(c)



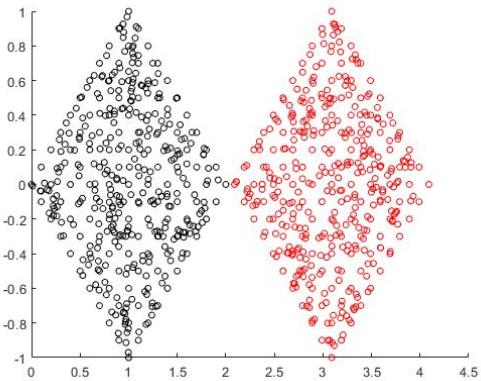
(d)



(e)



(f)



References

- Patil, Prasad. “K Means Clustering : Identifying F.R.I.E.N.D.S in the World of Strangers.” *Towards Data Science*, Towards Data Science, 20 May 2018, towardsdatascience.com/k-means-clustering-identifying-f-r-i-e-n-d-s-in-the-world-of-strangers-695537505d.
- Y. Weiss. Segmentation using eigenvectors: A unifying view. In International Conference on Computer Vision, 1999
- Scott, Guy L., and H. Christopher Longuet-Higgins. “An Algorithm for Associating the Features of Two Images.” *Proceedings: Biological Sciences*, vol. 244, no. 1309, 1991, pp. 21–26
- Luxburg, Ulrike Von. “A Tutorial on Spectral Clustering.” *Statistics and Computing*, vol. 17, no. 4, 2007, pp. 395–416., doi:10.1007/s11222-007-9033-z.
- Mohar, B & Alavi, Y & Chartrand, G & Oellermann, Ortrud & Schwenk, Allen. (1991). The Laplacian spectrum of graphs. Graph Theory, Combinatorics and Applications. 2. 5364.

Appendix

Warm-up Answers:

1. Epsilon = 4
2. The rows are the same as the columns, A is symmetric.

3-5. Use Matlab. Resulting labels = [2; 2; 2; 1; 1; 1]

```
%% (1) Create adjacency (adjacency) matrix
A = [1    1    1    0    0    0;
    1    1    1    1    0    0;
    1    1    1    0    1    0;
    0    1    0    1    1    1;
    0    0    1    1    1    1;
    0    0    0    1    1    1];

%% (2) Create degree matrix from A
D = [3    0    0    0    0    0;
    0    4    0    0    0    0;
    0    0    4    0    0    0;
    0    0    0    4    0    0;
    0    0    0    0    4    0;
    0    0    0    0    0    3];

%% (3) Calculate the Laplacian Matrix
```

L = D - A

%% (4) Get eigenvectors of L
[U, S, V] = svd(L);

%% (5) cluster using last 2 eigenvectors using kmeans function
labels = kmeans(U(:,5:6), 2)
% resulting labels = [2; 2; 2; 1; 1; 1];

Main Activity Answers:

1. K-means struggles to accurately cluster the spiraling data because the the clusters are not clearly separated by a distance in 2 dimensions. The k-means algorithm iteratively improves using the Euclidean distance to find each cluster center. It does not consider the underlying structure of the data or if points are close to each other.
2. The correct range of epsilon is about [2, 13]. Changing the epsilon value affects how our adjacency matrix is connected and which points are considered to be neighbors.
3. By computing the eigenvectors and eigenvalues of the Laplacian (which represents the relationships in our data), the best approximation to the data is found. Therefore, our data has been mapped from p-dimensions to k-dimensions, and running k-means on these vectors considers the underlying structure of the data.
4. eigenvectors = U(:, 310:312);
5. K-means is appropriate for (a), (b), and (f). Spectral Clustering is would perform better for (c), (d), and (e)

```
clear;
close all;

%% Load and plot dataset

load data_Spiral.mat

k = 3; % Number of classes (also number of clusters)
X = D(:, 1); % X values of dataset
Y = D(:, 2); % Y values of dataset
labels = L; % Labels of different classes

% Colors for plotting the dataset
colors = [0,0,0 ; 1,0,0 ; 0,1,0];

figure
scatter(X, Y, 20, colors(labels, :));
title("Original Dataset")
```

```
%% K-means clustering

% km are the labels produced by the k-means clustering
km = kmeans([X, Y],k);

figure
scatter(X, Y, 20, colors(km, :));
title("k-means")

%% Create the adjacency matrix

% FILL IN THIS VALUE
epsilon = 5; % Configuration parameter for adjacency matrix

A = zeros(size(X,1), size(Y,1));
for i = 1:size(X)
    for j = 1:size(Y)
        A(i, j) = (X(i)-X(j))^2 + (Y(i)-Y(j))^2; % Calculate distance between
every pair of points
    end
end

% Threshold the distances for creating the adjacency matrix
A(A <= epsilon) = 1;
A(A > epsilon) = 0;

% Create degree matrix from adjacency matrix
D = zeros(size(A,1));
for i=1:size(A,1)
    D(i,i) = sum(A(i,:));
end

% Graph the connections made in the adjacency matrix
figure
gplot(A,[X,Y],'-*');
title("Adjacency Matrix");


%% Spectral Clustering

% Calculate the Laplacian Matrix
Lap = D - A;

% Calculate SVD
[U, S, V] = svd(Lap);

% FILL IN THIS LINE
eigenvectors = U(:, 310:312);

% Cluster these eigenvectors using k-means
spec_labels = kmeans(eigenvectors,k);

% Plot the resulting clusters
```

```
figure
scatter(X, Y, 20, colors(spec_labels, :));
title("spectral clustering using SVD")
```