Clustering in Networks

(Spectral Clustering with the Graph Laplacian . . . a brief introduction)

Tom Carter

Computer Science
CSU Stanislaus

http://csustan.csustan.edu/~ tom/Clustering

April 1, 2012

Our general topics:

What is Clustering?

- Many real-world data sets consist of (or refer to) collections of entities of some general type. Some example data sets might be:
  - Points in space (e.g., a set of (x, y) coordinates)
  - Populations of people (e.g., census data . . .)
  - Financial instruments (e.g., stocks, with data set the daily closing prices)
  - Collections of texts (e.g., a set of emails, or blog posts, or . . .)
  - A social network (e.g., Facebook, or phone contacts)

Our data sets give us views into collections of entities, and we would like to use the data sets to better understand the entities . . .

- Often we can think of our data set as telling us something about the "relatedness" of our collection of entities. In some cases, the "relatedness" might be very simply and directly represented in the data set. For example, on Facebook, two members might be related if they are "friends" of each other. Two emails might be related if they are from the same person. In the census, two people might be "related" if they live in the same town, or if they have the same job, or etc. Two web pages might be related if one contains a link to the other.

  More generally, we may be able to develop a "relatedness" metric associated with the data set. For example, the

relatedness between two stocks might be the degree to which their prices are correlated with each other over time. The relatedness between two texts might be the number of distinct words they have in common.

- One useful starting point to understanding such data sets is to look for subsets of the collection of entities which are more closely related to each other than they are to other members of the collection. We can think of such closely related subsets as "clusters".

  Our goal, then, would be to find (automated or algorithmic) ways to find such clusters in our data sets. If we are lucky, we can also develop techniques for visualizing our data set through the lens of the clusters we have found. For example, we might develop a graphical display of the

data set where things in a particular cluster are near each other, or have the same color.

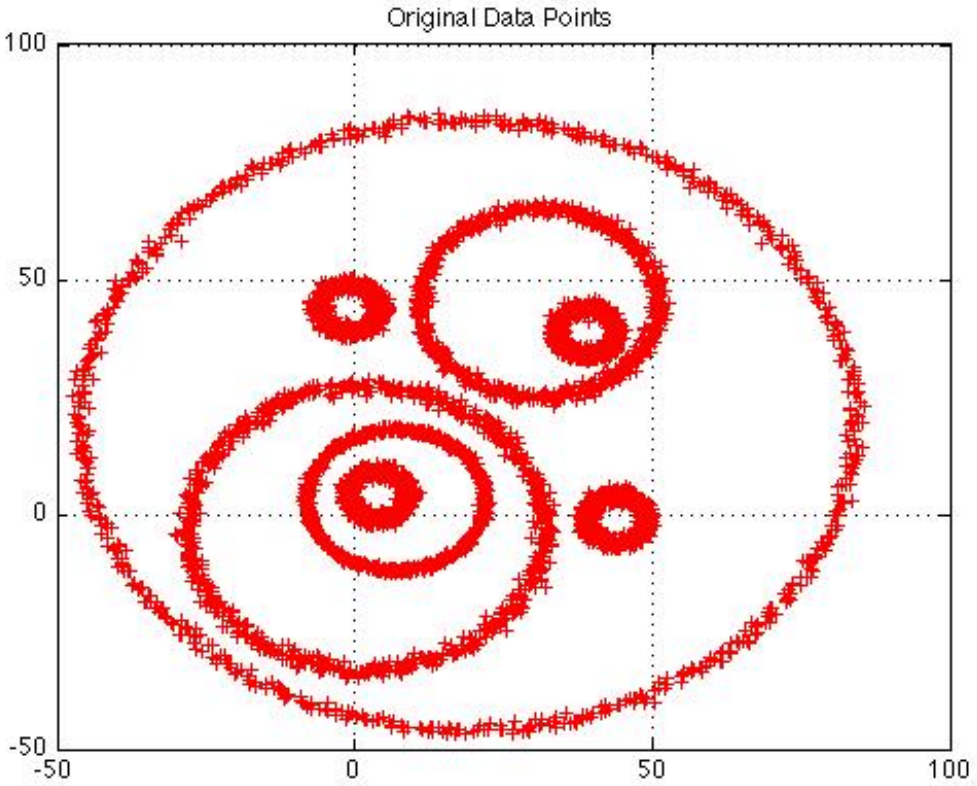
In the case of very large data sets, we can use clustering to reduce the effective size (or dimensionality) of the data set by collapsing clusters into single points, and proceeding from there with our analysis. There are even likely to be times when we want to iterate this process . . .

The general approach I'll briefly describe here is sometimes called "spectral clustering." It involves representing the "relatedness" as a weighted graph (or network), and applying various mathematical/algorithmic techniques. We can represent the graph as an "adjacency" matrix – i.e., we can construct a symmetric square matrix A from the data set where $A(i, j)$ is the "level of relatedness" between entities i and j. In simple traditional graph theory, two

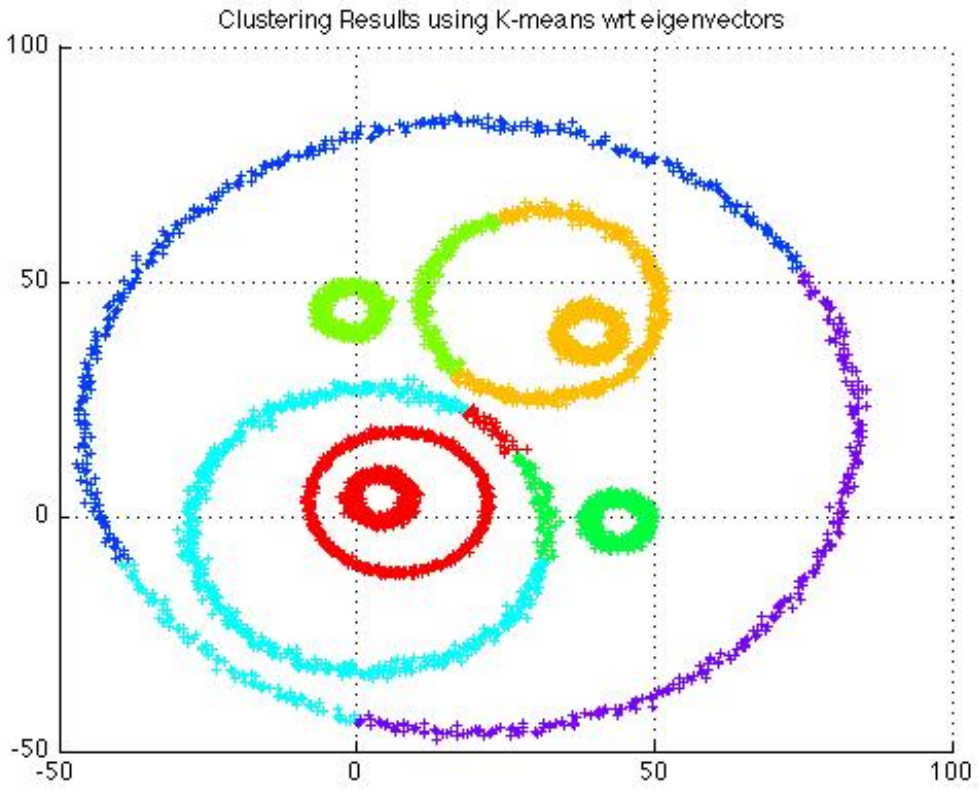
nodes (or vertices) are "related" if there is an edge between them, and the adjacency matrix has $A(i,j) = 1$, and 0 otherwise. Many of our examples will be of the more general "weighted graph" type.

An Example

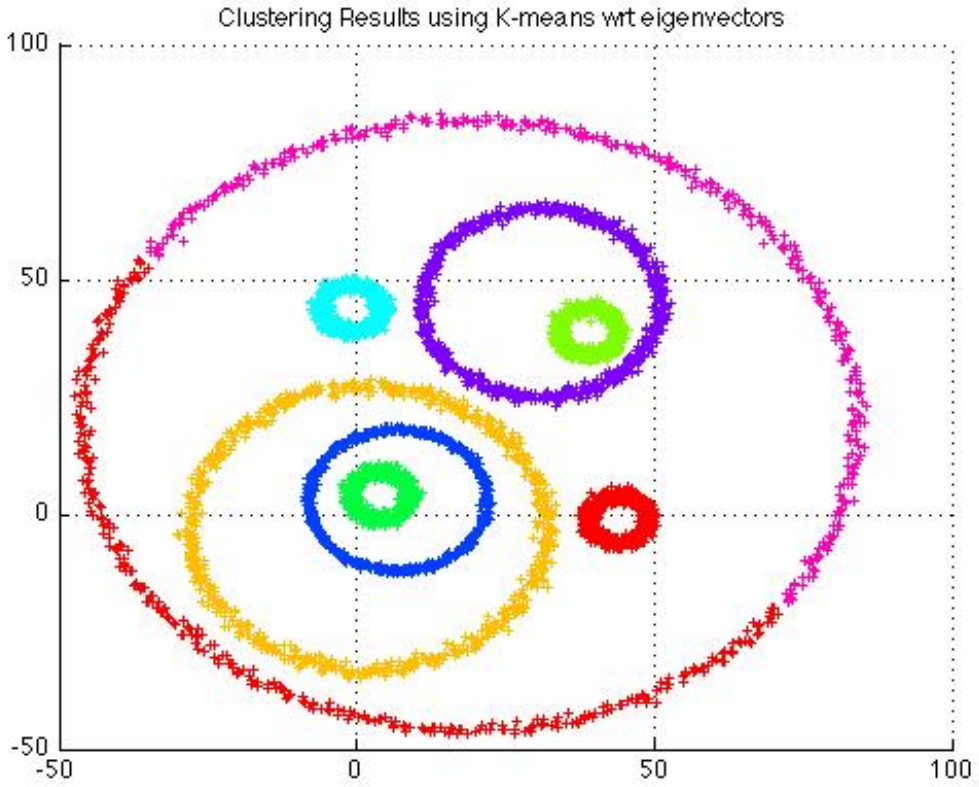
- Let's start with a (relatively) simple example. Consider the following, a collection of points in 2-dimensional space:



- Two points in this data set are related by being near each other. A comparatively naive "clustering" algorithm might find this:



- But we can see that there is more structure to the data set. We would really like something more like this:



This version of clustering was done via an algorithm in a general class often called spectral clustering.

Spectral Clustering (a general outline)

- A general outline of a useful approach to a spectral clustering is as follows:
  - Through some mechanism, generate an "adjacency" matrix. I'll sometimes call this an "affinity" matrix. This will be an n x n matrix, where each entry is an "affinity" between a pair of elements in the collection we are studying.
  - Massage the affinity matrix appropriately (art, not science :-)
  - Compute the (a . . .) graph Laplacian from the affinity matrix (see below)

- Calculate eigenvalues and associated eigenvectors from the graph Laplacian

- Choose an appropriate number k of clusters (art, not science :-)

- Select the eigenvectors associated with the largest k eigenvalues

- Make an n x k matrix with those eigenvectors

- Perform k-means clustering on the rows of the n x k matrix

- Display the results, lather, rinse, and repeat . . .

Graph Laplacian(s)

- The general idea of the "graph Laplacian" is to derive a new matrix from the adjacency matrix that can tell us something about the structure of the graph. These have enough similarity to the Laplacian found elsewhere in mathematics that they share the name. There are actually several variations in use. Some versions are outlined in the following. In practice, it is typical for people to work with the eigenvalues and eigenvectors of these derived matrices.

  - Let A (with entries $A(i,j)$) be the adjacency (or affinity) matrix. Note: in the general case, these are frequently labeled W (with entries $W(i,j)$), the weight matrix.

– We now compute the (generalized) degrees D of the nodes in the graph:

$$D(i,i) = \sum_j A(i,j)$$

$$D(i,j) = 0 \text{ if } i \neq j$$

A convenient abbreviation is $D(i) = D(i,i)$.

In the simple case of an unweighted graph, $D(i)$ tells how many other nodes are connected to node i. In the more general case, it tells the total weight of edges emanating from a node.

– The easiest version of a graph laplacian is

$$L = D - A$$

– A second version is often called a "normalized graph laplacian":

$$L_{norm_1} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Or, in coordinate form:

$$L_{norm_1}(i,j) = \begin{cases} 1 & \text{if } i = j \text{ and } D(i) \neq 0 \\ -\frac{A(i,j)}{\sqrt{D(i) * D(j)}} & \text{if } i \neq j \text{ and} A(i,j) \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

– A slightly simpler, but still useful, version of this is:

$$L_{norm_2} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

In other words,

$$L_{norm_2}(i,j) = \frac{A(i,j)}{\sqrt{D(i)D(j)}}$$

– An alternate version of a normalized laplacian is:

$$L_{norm_3} = D^{-1} A$$

This version has the nice property that if we think about a random walk on the graph, with the likelihood of

moving from node i to node j being proportional to $A(i,j)$, then the corresponding matrix of probabilities will be:

$$P = D^{-1}A$$

- more later . . .

←

References

[1] Chung, Fan,
Course notes for UCSD Math 261C (Probabilistic Combinatorics and Algorithms III), Spring 2010
http://math.ucsd.edu/ jthughes/Math261C/Math261C.html

[2] Chung, Fan,
Graph Theory in the Information Age, Notices of the AMS, Vol. 57, No. 6, pp. 726-732
http://www.math.ucsd.edu/ fan/wp/graph.pdf

[3] Horaud, Radu,
A Short Tutorial on Graph Laplacians, Laplacian Embedding, and Spectral Clustering
http://perception.inrialpes.fr/~Boyer/ReadingGroup/GraphLaplacian-tutorial.pdf

[4] von Luxburg, Ulrike,
A Tutorial on Spectral Clustering
http://people.kyb.tuebingen.mpg.de/ule/
publications/publication_downloads/Luxburg07_tutorial.pdf

[5] Spielman, Daniel, and Srivastava, Nikhil,
Graph Sparsification by Effective Resistances
Abstract: http://arxiv.org/abs/0803.0929
Full text: http://arxiv.org/pdf/0803.0929v4

[6] Vejmelka, Martin,
Spectral Graph Clustering
http://ai.ms.mff.cuni.cz/ sui/Seminar2009MartinVejmelka.pdf

[7] Ng, Andrew, Jordan, Michael, and Weiss, Yair,
On Spectral Clustering: Analysis and an Algorithm
http://www.stat.washington.edu/wxs/Stat593-s03/Literature/ng-
jordan-weiss-nips01.pdf

[8] Zhu, Xiaojin,
Course notes, Advanced Natural Language Processing
http://pages.cs.wisc.edu/ jerryzhu/cs769.html

To top ←