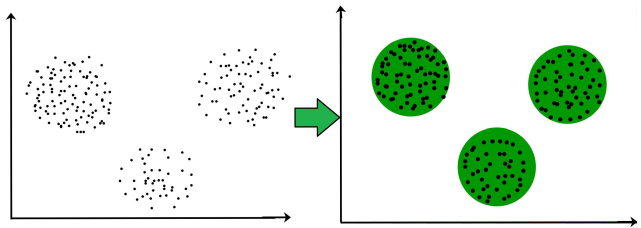
Lecture 11. Graphs, Networks, and Clustering

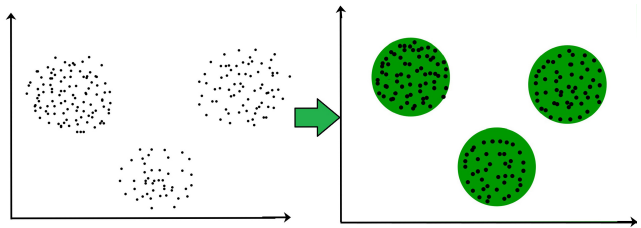
Bao Wang
Department of Mathematics
Scientific Computing and Imaging Institute
University of Utah
Math 5750/6880, Fall 2021

- Hierarchical clustering
- k-means
- PageRank
- Spectral clustering

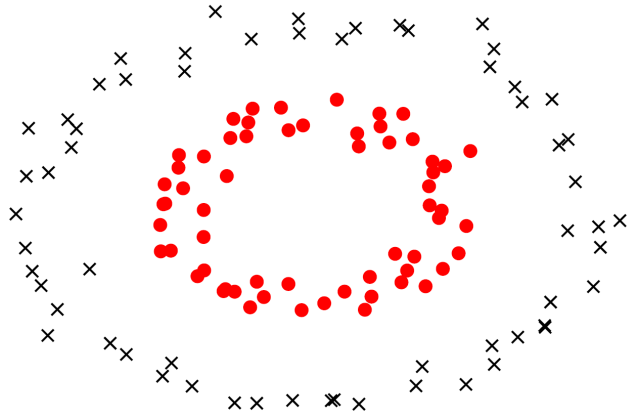
Clustering: grouping a set of objects such that similar objects end up in the same group and dissimilar objects are separated into different groups.



Clustering is an *unsupervised learning* problem: there are no labels that we try to predict. Instead, we wish to organize the data in some meaningful way.



How to cluster data?



1. A set of elements, $\mathcal{X}$.

2. A distance function over $\mathcal{X}$. That is, a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ that is symmetric, satisfies $d(x, x) = 0$ for all $x \in \mathcal{X}$ and often also satisfies the triangle inequality.

2'. Alternatively, the function could be a similarity function $s : \mathcal{X} \times \mathcal{X} \to [0, 1]$ that is symmetric and satisfies $s(x, x) = 1$ for all $x \in \mathcal{X}$.

3. Additionally, some clustering algorithms also require an input parameter k (determining the number of required clusters).

• A partition of the domain set $\mathcal{X}$ into subsets. That is, $C = (C_1, \cdots, X_k)$ where $\bigcup_{i=1}^{k} C = \mathcal{X}$ and for all $i \neq j$, $C_i \bigcap C_j = \emptyset$.

• In some situations the clustering is "soft", namely, the partition of $\mathcal{X}$ into the different clusters is probabilistic where the output is a function assigning to each domain point, $x \in \mathcal{X}$, a vector $(p_1(x), \cdots, p_k(x))$, where $p_i(x) = \mathbb{P}[\boldsymbol{x} \in C_i]$ is the probability that $\boldsymbol{x}$ belongs to cluster C_i.

• Another possible output is a clustering *dendrogram* (from Greek dendron = tree, gramma = drawing), which is a hierarchical tree of domain subsets, having the singleton sets on its leaves, and the full domain as its root.

Linkage-based Clustering Algorithms

Linkage-based clustering algorithms

• Linkage-based clustering algorithms proceed in a sequence of rounds.

• They start from the trivial clustering that has each data point as a single-point cluster. Then, repeatedly, these algorithms merge the "closest" clusters of the previous clustering.

• Consequently, the number of clusters decreases with each such round. If kept going, such algorithms would eventually result in the trivial clustering in which all of the domain points share one large cluster.

• Two parameters, then, need to be determined to define such an algorithm clearly. First, we have to decide how to measure (or define) the distance between clusters, and second, we have to determine when to stop merging. Recall that the input to a clustering algorithm is a between-points distance function, d. There are many ways of extending d to a measure of distance between domain subsets (or clusters).

Linkage-based clustering algorithms: Example

Let the input be the elements $\mathcal{X} = \{a, b, c, d, e\} \subset \mathbb{R}^2$ with the Euclidean distance as depicted on the left, then the resulting dendrogram is the one depicted on the right:

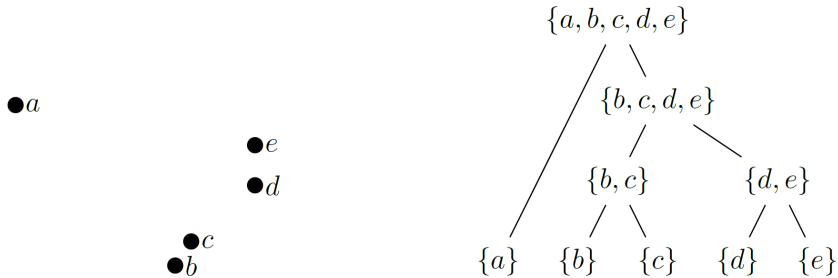


Figure: Illustration of the linkage-based clustering algorithm.

How to measure the distance between clusters?

> Single Linkage clustering, in which the between-clusters distance is defined by the minimum distance between members of the two clusters, namely,

$$D(A, B) := \min\{d(x, y) : x \in A, y \in B\}.$$

> Average Linkage clustering, in which the distance between two clusters is defined to be the average distance between a point in one of the clusters and a point in the order, namely,

$$D(A, B) := \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y).$$

> Max Linkage clustering, in which the distance between two clusters is defined as the maximum distance between their elements, namely,

$$D(A, B) := \max\{d(x, y) : x \in A, y \in B\}.$$

The linkage-based clustering algorithms start from data that is completely fragmented and keep building larger and larger clusters as they proceed. Without employing a stopping rule, the outcome of such an algorithm can be described by a clustering *dendrogram*: that is, a tree of domain subsets, having the singleton sets on its leaves, and the full domain as its root.

Height
10
5
0
Alabama
Louisiana
Georgia
Texas
Illinois
New York
Florida
Arizona
New Mexico
Alaska
Colorado
California
Nevada
Idaho
Montana
Nebraska
Minnesota
Wisconsin
Maine
Virginia
Wyoming
Arkansas
Kentucky
Delaware
Missouri
Oregon
Washington
Oklahoma
Indiana
Kansas
Hawaii
Utah

Linkage-based clustering algorithms: Stopping criterion

If one wishes to turn a dendrogram into a partition of the space (a clustering), one needs to employ a *stopping criterion*. Common stopping criteria include

> Fixed number of clusters – fix some parameter, k, and stop merging clusters as soon as the number of clusters is k.

> Distance upper bound – fix some $r \in \mathbb{R}_+$. Stop merging as soon as all the between-clusters distances are larger than r. We can also set r to be $\alpha \max\{d(x, y) : x, y \in \mathcal{X}\}$ for some $\alpha < 1$. In that case the stopping criterion is called "scaled distance upper bound."

k-Means

Formulate clustering as an optimization problem.

• Objective function G: a function from pairs of an input, $(\mathcal{X}, d)$, and a proposed clustering solution $C = (C_1, \cdots, C_k)$, to positive real numbers.

• Goal: Given such an objective function G, the goal of a clustering algorithm is defined as finding, for a given input $(\mathcal{X}, d)$, a clustering C so that $G((\mathcal{X}, d), C)$ is minimized.

Remark. Many common objective functions require the number of clusters, k, as a parameter. In practice, it is often up to the user of the clustering algorithm to choose the parameter k that is most suitable for the given clustering problem.

k-means objective function

- In k-means the data is partitioned into disjoint sets $C_1, \cdots, C_k$ where each C_i is represented by a centroid μ_i.

- It is assumed that the input set $\mathcal{X}$ is embedded in some larger metric space $(\mathcal{X}', d)$ (so that $\mathcal{X} \subset \mathcal{X}'$) and centroids are members of $\mathcal{X}'$.

- The k-means objective function measures the squared distance between each point in $\mathcal{X}$ to the centroid of its cluster.

- The centroid of $\mathcal{C}_i$ is defined to be

$$\mu_i(C_i) = \arg \min_{\mu \in \mathcal{X}'} \sum_{x \in C_i} d(x, \mu)^2.$$

Then, the k-means objective is

$$G_{\mathrm{k-means}}((\mathcal{X}, d), (C_1, \cdots, C_k)) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2.$$

This can also be rewritten as

$$G_{\mathrm{k-means}}((\mathcal{X}, d), (C_1, \cdots, C_k)) = \min_{\mu_1, \cdots, \mu_k \in \mathcal{X}'} \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2. \quad (1)$$

Tasks: 1) Find centers $\mu_1, \cdots, \mu_k$, and 2) Distribute each x to an appropriate cluster.

k-medoids objective function

The k-medoids objective function is similar to the k-means objective, except that it requires the cluster centroids to be members of the input set. The objective function is

$$G_{\mathrm{K-medoid}}((\mathcal{X}, d), (C_1, \cdots, C_k)) = \min_{\mu_1, \cdots, \mu_k \in \mathcal{X}} \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2.$$

k-median objective function

The k-median objective function is quite similar to the k-medoids objective, except that the "distortion" between a data point and the centroid of its cluster is measured by distance, rather than by the square of the distance:

$$G_{\mathrm{K-median}}((\mathcal{X}, d), (C_1, \cdots, C_k)) = \min_{\mu_1, \cdots, \mu_k \in \mathcal{X}} \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i).$$

Pros and cons of median? Robust to outliers but not differentiable.

Center-based objectives

The previous examples can all be viewed as *center-based* objectives. The solution to such a clustering problem is determined by a set of cluster centers, and the clustering assigns each instance to the center closest to it. More generally, center-based objective is determined by choosing some monotonic function $f : \mathbb{R}_+ \to \mathbb{R}_+$ and then defining

$$G_f((\mathcal{X}, d), (C_1, \cdots, C_k)) = \min_{\mu_1, \cdots, \mu_k \in \mathcal{X}'} \sum_{i=1}^{k} \sum_{x \in C_i} f(d(x, \mu_i)),$$

where $\mathcal{X}'$ is either $\mathcal{X}$ or some superset of $\mathcal{X}$.

Later we will discuss graph cut objective, which is not center-based objectives.

The k-means algorithm

• The k-means objective function is quite popular in practical applications of clustering.

• However, it turns out that the optimal k-means solution is often computationally infeasible (the problem is NP-hard, and even NP-hard to approximate to within some constant).–Nonconvex!

• As an alternative, the following simple iterative algorithm is often used, so often that, in many cases, the term k-means clustering refers to the outcome of this algorithm rather than to the clustering that minimizes the k-means objective cost. We describe the algorithm with respect to the Euclidean distance function $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$. (See next slide.)

The k-means algorithm

k-Means algorithm

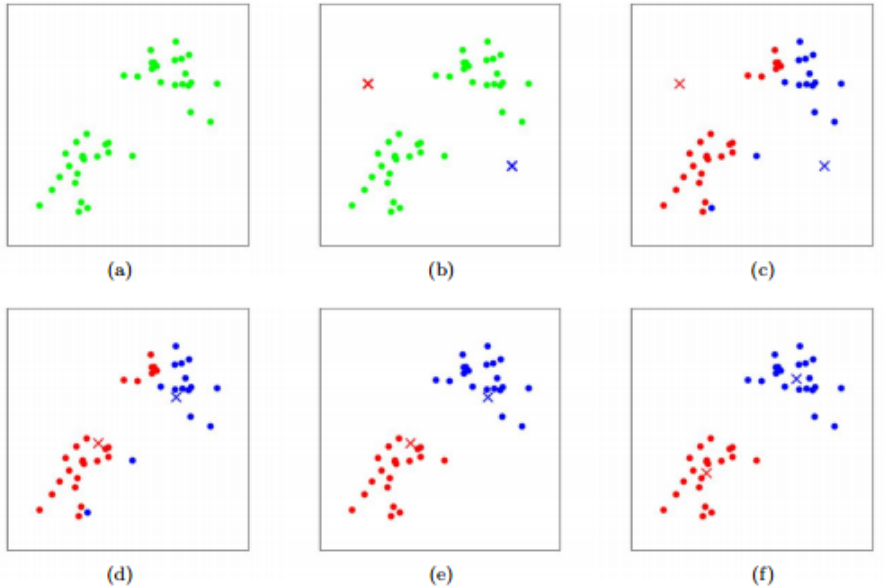
Input: $\mathcal{X} \subset \mathbb{R}^n$; number of clusters k

Initialize: randomly choose initial centroids $\mu_1, \cdots, \mu_k$

repeat until convergence

$\forall i \in [k]$ set $C_i = \{\boldsymbol{x} \in \mathcal{X} : i = \arg\min_j \|\boldsymbol{x} - \mu_j\|\}$ (break ties in some arbitrary manner)

$\forall i \in [k]$ update $\mu_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$


(a)
(b)
(c)
(d)
(e)
(f)

Lemma. Each iteration of the k-means algorithm does not increase the k-means objective function:

$$G_{\mathrm{k-means}}((\mathcal{X}, d), (C_1, \cdots, C_k)) = \min_{\mu_1, \cdots, \mu_k \in \mathcal{X}'} \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu)^2. \tag{2}$$

Proof. We use the short hand $G(C_1, \cdots, C_k)$ for the k-means objective, i.e.,

$$G(C_1, \cdots, C_k) = \min_{\mu_1, \cdots, \mu_k \in \mathbb{R}^n} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \mu_i\|^2. \tag{3}$$

It is convenient to define $\mu(C_i) = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$ and note that

$$\mu(C_i) = \arg\min_{\mu \in \mathbb{R}^n} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \mu\|^2.$$

Therefore, we can rewrite the k-means objective as

$$G(C_1, \cdots, C_k) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \mu(C_i)\|^2. \tag{4}$$

Consider the update at iteration t of the k-means algorithm. Let $C_1^{(t-1)}, \cdots, C_k^{(t-1)}$ be the previous partition, let $\mu_i^{(t-1)} = \mu(C_i^{(t-1)})$, and let $C_1^{(t)}, \cdots, C_k^{(t)}$ be the new partition assigned at iteration t.

Using the definition of the objective as given in (3) we clearly have that (note that the LHS needs to minimize over μ_is, while RHS fixed $\mu_i^{(t-1)}$s.)

$$G(C_1^{(t)}, \cdots, C_k^{(t)}) \leq \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \|\boldsymbol{x} - \mu_i^{(t-1)}\|^2. \tag{5}$$

In addition, the definition of the new partition $(C_1^{(t)}, \cdots, C_k^{(t)})$ implies that it minimizes the expression $\sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \mu_i^{(t-1)}\|^2$ over all possible partitions $(C_1, \cdots, C_k)$. Hence,

$$\sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \|\boldsymbol{x} - \mu_i^{(t-1)}\|^2 \leq \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t-1)}} \|\boldsymbol{x} - \mu_i^{(t-1)}\|^2. \tag{6}$$

Using (4) we have that the right-hand side of (6) equals $G(C_1^{(t-1)}, \ldots, C_k^{(t-1)})$.
Combining this with (5) and (6), we have $G(C_1^{(t)}, \cdots, C_k^{(t)}) \leq G(C_1^{(t-1)}, \cdots, C_k^{(t-1)})$, which concludes our proof.

Remark.

- While the preceding lemma tells us that the k-means objective is monotonically nonincreasing, there is no guarantee on the number of iterations the k-means algorithms needs in order to reach convergence.

- Furthermore, there is no nontrivial lower bound on the gap between the value of the k-means objective of the algorithm's output and the minimum possible value of that objective function. In fact, k-means might converge to a point which is not even a local minimum. Nonconvex. Consider saddle point.

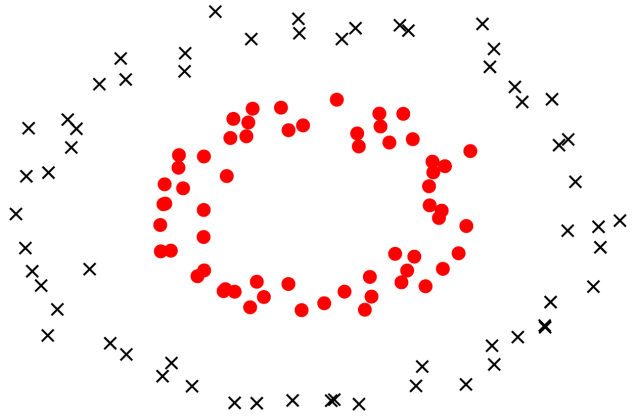
- To improve the results of k-means it is often recommended to repeat the procedure several times with different randomly chosen initial centroids (e.g., we can choose the initial centroids to be random points from the data).

Remarks

• One needs to set the number of clusters a priori (a typical way to overcome this issue is by trying the algorithm for different number of clusters).

• k-means above requires data points to be defined in an Euclidean space, oftentimes we are interested in clustering data for which we only have some measure of affinity between different data points, but not necessarily an embedding in $\mathbb{R}^p$ (this issue can be overcome by reformulating k-means in terms of distances only).

• The solution of k-means are always convex clusters. This means that k-means may have difficulty in finding cluster as in the figure below.



Graphs

• A graph $G = (V, E)$ contains a set of nodes $V = \{v_1, \cdots, v_n\}$ and edges $E \subset \binom{V}{2}$. An edge $(i, j) \in E$ if v_i and v_j are connected.

• A graph is connected if, for all pairs of vertices, there is a path between these vertices on the graph.

A particularly useful way to represent a graph is through its adjacency matrix. Given a graph $G = (V, E)$ on n nodes ($|V| = n$), we define its adjacency matrix $A \in \mathbb{R}^{n \times n}$ as the symmetric matrix with entries

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Another important matrix is the degree matrix D, which is diagonal with $D_{ii} = \sum_{j=1}^{n} A_{ij}$.

Sometimes, we will consider weighted graphs $G = (V, E, W)$, where edges may have weights w_{ij}, we think of the weights as non-negative $w_{ij} \geq 0$ and symmetric $w_{ij} = w_{ji}$. In this case, $D_{ii} = \sum_{j=1}^{n} w_{ij}$.

PageRank

PageRank: Backbone of Google's search engine?

The goal of PageRank is to quantitatively rate the importance of each page on the web, allowing the search algorithm to rank the pages and thereby present to the user the more important pages first. How to develop a score of importance for each webpage?

A score will be a nonnegative number. A key idea in assigning a score to any given webpage is that the page's score is derived from the links made to that page from other webpages — "A person is important not if the person knows a lot of people, but if a lot of people know that person".

Suppose the web of interest contains n pages, each page indexed by an integer k, $1 \leq k \leq n$. A typical example is illustrated below, in which an arrow from page k to page j indicates a link from page k to page j. Such a web is an example of a directed graph. The links to a given page are called the backlinks for that page.

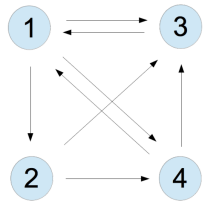


Figure: A toy example of the Internet.

We will use x_k to denote the importance score of page k in the web. x_k is nonnegative and $x_j > x_k$ indicates that page j is more important than page k.

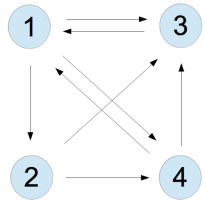


Figure: A toy example of the Internet.

A very simple approach is to take x_k as the number of backlinks for page k. In the example above, we have $x_1 = 2, x_2 = 1, x_3 = 3$, and $x_4 = 2$, so that page 3 is the most important, pages 1 and 4 tie for second, and page 2 is least important. A link to page k becomes a vote for page k's importance.

What is wrong with this?

The above approach ignores an important feature one would explect a ranking algorithm to have, namely, that a link to page k from an important page should boost page k's importance score more than a link from an unimportant page.

In the above example, pages 1 and 4 both have two backlinks: each links to the other, but the second backlink from page 1 is from the seemingly important page 3, while the second backlink for page 4 is from the relatively unimportant page 2. As such, perhaps the algorithm should rate the importance of page 1 higher than that of page 4.

Second attempt

Let us compute the score of page j as the sum of the scores of all pages linking to page j. For the above toy example. The score of page 1 would be determined by the relation $x_1 = x_3 + x_4$. However, since x_3 and x_4 will depend on x_1, this seems like a circular definition, since it is self-referential.

We also seek a scheme in which a webpage does not gain extra influence simply by linking to lots of other pages. We can do this by reducing the impact of each link, as more and more outgoing links are added to a webpage.

If page j contains n_j links, one of which links to page k, then we will boost page k's score by x_j/n_j, rather than by x_j. In this scheme, each webpage gets a total of one vote, weighted by that web page's score, that is evenly divided up among all of its outgoing links. To quantify this for a web of n pages, let $L_k \subset \{1, 2, \cdots, n\}$ denote the set of pages with a link to page k, that is, L_k is the set of page k's backlinks. For each k we require

$$x_k = \sum_{j \in L_k} \frac{x_j}{n_j},$$

where n_j is the number of outgoing links from page j.

Apply the above scheme to the toy example above, then for page 1 we have $x_1 = x_3/1 + x_4/2$, since pages 3 and 4 are backlinks for page 1 and page 3 contains only one link, while page 4 contains contains 2 links (splitting its vote in half). Similarly, $x_2 = x_1/3, x_3 = x_1/3 + x_2/2 + x_4/2$, and $x_4 = x_1/3 + x_2/2$. These conditions can be expressed as linear system of equations $Ax = x$, where $x = [x_1, x_2, x_3, x_4]^\top$ and

$$A = \begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

Thus, we end up with an eigenvalue/eigenvector problem: Find the eigenvector x of the matrix A, associated with the eigenvalue 1.

Why 1 is an eigenvalue of A?

Def. A square matrix is column-stochastic if all its entries are nonnegative and the sum of each column is 1.

Theorem. A column-stochastic matrix A has an eigenvalue equal to 1 and 1 is also its largest eigenvalue.

Proof. Let A be an $n \times n$ column-stochastic matrix. We first note that A and $A^\top$ have the same eigenvalues (there eigenvector will usually be different though). Let $\vec{1} = [1, 1, \cdots, 1]^\top$ be the vector of length n which has all ones as entries. Since A is column-stochastic, we have $A^\top \vec{1} = \vec{1}$ (since all columns of A sum up to 1). Hence $\vec{1}$ is an eigenvector of $A^\top$ (but not for A) with eigenvalue 1. Thus 1 is also an eigenvalue of A.

Lemma. [Gershgorin circle theorem] Let A be a complex $n \times n$ matrix, with entries a_{ij}. For $i \in \{1, \ldots, n\}$ let R_i be the sum of the absolute values of the non-diagonal entries in the i-th row: $R_i = \sum_{j \neq i} |a_{ij}|$. Let $D(a_{ii}, R_i) \subseteq \mathbb{C}$ be a closed disc centered at a_{ii} with radius R_i. Such a disc is called a Gershgorin disc. Then, every eigenvalue of A lies within at least one of the Gershgorin discs $D(a_{ii}, R_i)$.

To show that 1 is the largest eigenvalue of A we apply the Gershgorin circle theorem to $A^\top$. Consider row k of $A^\top$. Let us call the diagonal element $a_{k,k}$ and the radius will be $\sum_{i \neq k} |a_{k,i}| = \sum_{i \neq k} a_{k,i} = 1 - a_{k,k}$. Hence, this circle has 1 on its perimeter. This holds for all Gershgorin circles for this matrix. Thus, since all eigenvalues lie in the union of the Gershgorin circles, all eigenvalues λ_i satisfy $|\lambda_i| \leq 1$.

Back to the toy example

In the above toy example, we obtain as eigenvector x of A associated with eigenvalue 1 the vector $x = [\frac{12}{31}, \frac{4}{31}, \frac{9}{31}, \frac{6}{31}]^\top$. Hence, perhaps somewhat surprisingly, page 3 is no longer the most important one, but page 1. This can be explained by the fact, that the in principle quite important page 3 (which has three webpages linking to it) has only one outgoing link, which gets all its "voting power", and that link points to page 1.

Practical challenges

In reality, A can be of size billions times billions.

Fortunately, we do not need compute all eigenvectors of A, only the eigenvector associated with the eigenvalue 1, which, as we know, is also the largest eigenvalue of A. This in turn means we can resort to standard power iteration to compute x fairly efficiently (and we can also make use of the fact that A will be a sparse matrix, i.e., many of its entries will be zero).

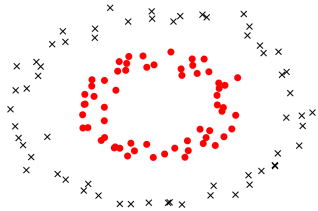
Let the eigenvalue of A be $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ and the associated eigenvectors be $v_1, v_2, \cdots, v_n$. Then any vector x_0 can be written as

$$x_0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n.$$

Then

$$A^k x_0 = c_1 \lambda_1^k v_1 + c_2 \lambda_2^k v_2 + \cdots + c_n \lambda_n^k v_n = c_1 \lambda_1^k (v_1 + c_2 (\frac{\lambda_2}{\lambda_1})^k v_2 + \cdots + c_n (\frac{\lambda_n}{\lambda_1})^k v_n) \to c_1 \lambda_1^k v_1$$

Spectral Clustering



• A natural way to overcome the issues of k-means in the above figure is transforming the data into a graph and cluster the graph.

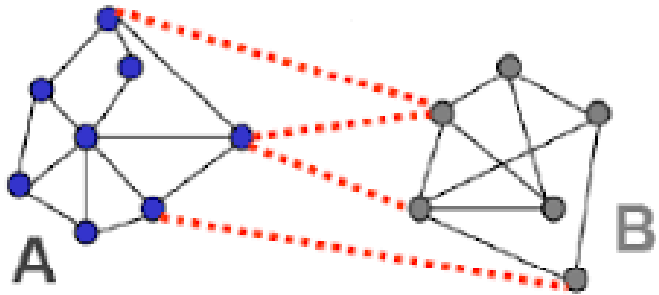
• Given the data points we can construct a weighted graph $G = (V, E, W)$ using a similarity kernel K_ϵ, such as $K_\epsilon(u) = \exp(\frac{1}{2\epsilon}u^2)$, by associating each point to a vertex and, for which pair of nodes, set the edge weight as

$$w_{ij} = K_\epsilon(\|x_i - x_j\|).$$

• Alternatively, we can construct the graph where data points are connected if they correspond to the nearest neighbors. This procedure only needs a measure of distance, or similarity, or data points and not necessarily that they lie in Euclidean space.

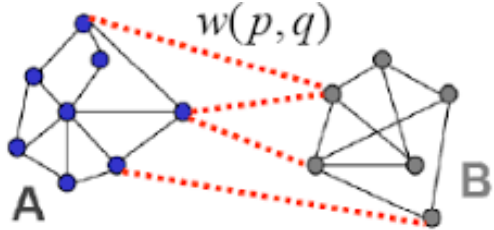
Next, we will address the problem of clustering the nodes of a graph.

Graph representation of data



Given a graph $G = (V, E, W)$, the goal is to partition the graph in clusters in a way that keeps as many of the edges within the clusters and has as few edges as possible across clusters. We will first focus on the case of two clusters.

Graph cut



$$cut\,(A,B) = \sum_{p \in A, q \in B} w(p,q)$$

A natural way to measure a vertex partition (S, S^c) is

$$cut(S) = \sum_{i \in S} \sum_{j \in S^c} w_{ij}.$$

If we represent the partition by a vector $y \in \{\pm 1\}^n$ where $y_i = 1$ if $i \in S$, and $y_i = -1$ otherwise. Then, the cut is a quadratic form on the graph Laplacian.

Def. [Graph Lapplacian and Degree Matrix.] Let $G = (V, E, W)$ be a graph and W the matrix of weights (or adjacency matrix if the graph is unweighted). The degree matrix D is a diagonal matrix with diagonal entries

$$D_{ii} = deg(i) = \sum_{j=1}^{n} w_{ij}.$$

The graph Laplacian of G is given by

$$L_G := D - W.$$

Equivalently,

$$L_G := \sum_{i<j} w_{ij}(e_i - e_j)(e_i - e_j)^\top.$$

Note that the entries of L_G are given by

$$(L_G)_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ deg(i) & \text{if } i = j. \end{cases}$$

If $S \subset V$ and $y \in \{\pm 1\}^n$ such that $y_i = 1$ if $i \in S$, and $y_i = -1$ otherwise, then it is easy to see that

$$cut(S) = \frac{1}{4} \sum_{i<j} w_{ij}(y_i - y_j)^2.$$

Next, we will show that

$$cut(S) = \frac{1}{4} y^\top L_G y, \tag{7}$$

for $y \in \{\pm 1\}^n$ such that $y_i = 1$ if and only if $i \in S$.

Proposition. Let $G = (V, E, W)$ be a graph and L_G be its graph Laplacian, let $y \in \mathbb{R}^n$ then

$$y^\top L_G y = \sum_{i<j} w_{ij}(y_i - y_j)^2.$$

Proof.

$$\begin{aligned}
\sum_{i<j} w_{ij}(y_i - y_j)^2 &= \sum_{i<j} w_{ij}\big[(e_i - e_j)y\big]\big[(e_i - e_j)y\big]^\top \\
&= \sum_{i<j} w_{ij} y^\top (e_i - e_j)(e_i - e_j)^\top y \\
&= y^\top \Big[\sum_{i<j} w_{ij}(e_i - e_j)(e_i - e_j)^\top\Big] y \\
&= y^\top L_G y.
\end{aligned}$$

While $cut(S)$ is a good way of measuring the fit of a partition, it suffers from an issue: the minimum cut is achieved for $S = \emptyset$ (since $cut(\emptyset) = 0$) which is a rather meaningless choice of partition. Next, we will discuss how to promote (almost) balanced partitions.

Remark. One way to address the above problem is to simply ask for an exactly balanced partition, $|S| = |S^c|$ (let us assume the number of vertices $n = |V|$ is even). We can then identify a partition with a label vector $y \in \{\pm 1\}^n$ where $y_i = 1$ if $i \in S$, and $y_i = -1$ otherwise. Also, the balanced condition can be written as $\sum_{i=1}^n y_i = \vec{1}y = 0$. This means, we can write the minimum balanced cut as

$$\min_{S \subset V,\ |S| = |S^c|} cut(S) = \frac{1}{4} \min_{y \in \{-1,1\}^n, \vec{1}^\top y = 0} y^\top L_G y.$$

Asking for the partition to be exactly balanced is too restrictive in many cases. There are several ways to evaluate a partition that are variations of $cut(S)$ that take into account the intuition that one wants both S and S^c to not be too small (not necessarily equal to $|V|/2$). A prime example is Cheeger's cut.

Def. [Cheeger's cut] Given a graph and a vertex partition (S, S^c), the Cheeger cut (also known as conductance, or expansion) of S is given by

$$h(S) = \frac{cut(S)}{\min\{vol(S), vol(S^c)\}},$$

where $vol(S) = \sum_{i \in S} deg(i)$. Also, the Cheeger's constant of G is given by

$$h_G = \min_{S \subset V} h(S),$$

where V is the whole vertex set.

Normalized cut: Ncut

A similar object to the Cheeger's cut is the Normalized cut, Ncut, which is given by

$$Ncut(S) = \frac{cut(S)}{vol(S)} + \frac{cut(S^c)}{vol(S^c)}.$$

$Ncut(S)$ and $h(S)$ are tightly related; it is easy to see that:

$$h(S) \leq Ncut(S) \leq 2h(S).$$

Relax balanced cut

Recall: $Cut(S) = \frac{1}{4} y^\top L_G y$. Below we will show Ncut can also be written in terms of a minimization of a quadratic form involving the graph Laplacian L_G.

Recall that the balanced partition can be written as

$$\frac{1}{4} \min_{y \in \{-1,1\}^n, \vec{1}^\top y = 0} y^\top L_G y.$$

An intuitive way to relax the balanced condition is to allow the labels y to take values in two different real values a and b (e.g. $y_i = a$ if $i \in S$ and $y_i = b$ if $i \notin S$) but not necessarily ± 1.

Relax balanced cut

• We can then use the notion of volume of a set to ensure a less restrictive notion of balanced by asking that

$$avol(S) + bvol(S^c) = 0, \quad \text{(Relaxed version of } \vec{1}^\top y = 0) \tag{8}$$

where

$$vol(S) = \sum_{i \in S} deg(i). \tag{9}$$

Thus (8) corresponds to $\vec{1}^\top Dy = 0$, where y is formed by a and b, and D is the degree matrix. (In balanced cut, we have $\vec{1}^\top y = 0$)

• We also need to fix a scale for a and b:

$$a^2 vol(S) + b^2 vol(S^c) = 1,$$

which corresponds to $y^\top Dy = 1$.

This suggests considering

$$\min_{y \in \{a,b\}^n, \vec{1}^\top D y = 0, y^\top D y = 1} y^\top L_G y.$$

We will show this corresponds precisely to Ncut.

Proposition. For a and b to satisfy $avol(S) + bvol(S^c) = 0$ and $a^2 vol(S) + b^2 vol(S^c) = 1$ it must be that

$$a = \left(\frac{vol(S^c)}{vol(S)vol(G)} \right)^{\frac{1}{2}} \quad \text{and} \quad b = -\left(\frac{vol(S)}{vol(S^c)vol(G)} \right)^{\frac{1}{2}},$$

corresponding to

$$y_i = \begin{cases} \left(\frac{vol(S^c)}{vol(S)vol(G)} \right)^{\frac{1}{2}} & \text{if } i \in S, \\ -\left(\frac{vol(S)}{vol(S^c)vol(G)} \right)^{\frac{1}{2}} & \text{if } i \in S^c. \end{cases}$$

Note that *vol* is defined as (9), i.e., the sum of degrees.

Proof. The proof involves only doing simple algebraic manipulations together with noticing that $vol(S) + vol(S^c) = vol(G)$.

We are ready to show relaxed balanced cut is exactly Ncut.

Proposition.

$$Ncut(S) = y^\top L_G y,$$

where y is given by

$$y_i = \begin{cases} \left(\frac{vol(S^c)}{vol(S)vol(G)}\right)^{\frac{1}{2}} & \text{if } i \in S, \\ -\left(\frac{vol(S)}{vol(S^c)vol(G)}\right)^{\frac{1}{2}} & \text{if } i \in S^c. \end{cases}$$

Proof.

$$
\begin{aligned}
y^\top L_G y &= \sum_{i,j} w_{ij}(y_i - y_j)^2 \\
&= \sum_{i\in S}\sum_{j\in S^c} w_{ij}(y_i - y_j)^2 \\
&= \sum_{i\in S}\sum_{j\in S^c} w_{ij}\Big[\Big(\frac{vol(S^c)}{vol(S)vol(G)}\Big)^{\frac{1}{2}} + \Big(\frac{vol(S)}{vol(S^c)vol(G)}\Big)^{\frac{1}{2}}\Big]^2 \\
&= \sum_{i\in S}\sum_{j\in S^c} w_{ij}\frac{1}{vol(G)}\Big[\frac{vol(S^c)}{vol(S)} + \frac{vol(S)}{vol(S^c)} + 2\Big] \\
&= \sum_{i\in S}\sum_{j\in S^c} w_{ij}\frac{1}{vol(G)}\Big[\frac{vol(S^c)}{vol(S)} + \frac{vol(S)}{vol(S^c)} + \frac{vol(S)}{vol(S)} + \frac{vol(S^c)}{vol(S^c)}\Big] \\
&= \sum_{i\in S}\sum_{j\in S^c} w_{ij}\Big[\frac{1}{vol(S)} + \frac{1}{vol(S^c)}\Big] \\
&= cut(S)\Big[\frac{1}{vol(S)} + \frac{1}{vol(S^c)}\Big] = Ncut(S).
\end{aligned}
$$

This means that finding the minimum Ncut corresponds to solving the following relaxed balanced cut

$$\begin{aligned} & \min y^\top L_G y \\ s.t. & \\ & y \in \{a, b\}^n \text{ for some } a \text{ and } b, \\ & y^\top D y = 1, \\ & y^\top D \vec{1} = 0. \end{aligned} \tag{10}$$

Solving (10) is, in general, NP-hard since $y \in \{a, b\}^n$ (combinatorial optimization), we consider a similar problem where the constraint that y can only take two values is removed:

$$\begin{aligned} & \min y^\top L_G y \\ s.t. & \\ & y \in \mathbb{R}^n, \\ & y^\top D y = 1, \\ & y^\top D \vec{1} = 0. \end{aligned} \tag{11}$$

Given a solution of (11) we can round it to a partition by setting a threshold τ and taking $S = \{i \in V : y_i \leq \tau\}$. Next, we will show (11) is an eigenvector problem, and thus we call (11) a spectral relaxation.

To see (11) is an eigenvector problem (and thus computationally tractable), set $z = D^{\frac{1}{2}} y$ and

$$\mathcal{L}_G = D^{-\frac{1}{2}} L_G D^{-\frac{1}{2}}, \tag{12}$$

then (11) is equivalent to

$$\begin{aligned} & \min z^\top \mathcal{L}_G z \\ s.t. & \\ & z \in \mathbb{R}^n, \\ & \|z\|^2 = 1, \\ & \left(D^{\frac{1}{2}} \vec{1}\right)^\top z = 0. \end{aligned} \tag{13}$$

Note that $\mathcal{L}_G = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. We order its eigenvalues in increasing order $0 = \lambda_1(\mathcal{L}_G) \leq \lambda_2(\mathcal{L}_G) \leq \cdots \leq \lambda_n(\mathcal{L}_G)$. ($D^{-1/2} W D^{-1/2}$ is a column stochastic matrix, thus the largest eigenvalue is 1.)

Based on the min-max theorem, we have the following variational interpretation of eigenvalues:

$$\sigma_k = \max_{S:dim(S)=n-k+1} \min_{x \in S, \|x\|=1} \|Ax\|,$$

where σ_k denotes the k-th entry in the increasing sequence of σ's, so that $\sigma_1 \leq \sigma_2 \leq \cdots$, where σ_k is the k-th singular value of A, i.e., the square root of the eigenvalue of $A^\top A$.

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

• Note the constraint $(D^{\frac{1}{2}}\vec{1})z = 0$ constraints the space S to dimensional $n-1$. By the variational interpretation of eigenvalues, the minimum of (13) is $\lambda_2(\mathcal{L}_G)$ and the minimizer is given by the second smallest eigenvector of $\mathcal{L}_G = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, which we call v_2.

• Note that v_2 is also the second largest eigenvector of $D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. This means that the optimal y in (11) is given by $\phi_2 = D^{-\frac{1}{2}} v_2$.

The above arguments motivate the following spectral clustering algorithm.

Algorithm 1. Spectral Clustering

- Given a graph $G = (V, E, W)$, let v_2 be the eigenvector corresponding to the second smallest eigenvalue of the normalized Laplacian $\mathcal{L}_G$, as defined in (12).
- Let $\phi_2 = D^{-\frac{1}{2}} v_2 \in \mathbb{R}^n$.
- Given a threshold τ (one can try all different possibilities, or run k-means for $k = 2$), set

$$S = \{i \in V : \phi_2(i) \leq \tau\}.$$

The relaxation (11) is obtained from (10) by removing a constraint, therefore we have

$$\lambda_2(\mathcal{L}_G) \leq \min_{S \subset V} Ncut(S).$$

This means that

$$\frac{1}{2}\lambda_2(\mathcal{L}_G) \leq h_G.$$

Recall that $h(S) \leq Ncut(S) \leq 2h(S)$ for any S. Collect to Cheeger's cut.

Lemma. There is a threshold τ producing a partition S such that

$$h(S) \leq \sqrt{2\lambda_2(\mathcal{L}_G)}.$$

Remark. The lemma above implies in particular that

$$h(S) \leq \sqrt{4h_G},$$

meaning that Algorithm 1 is suboptimal at most by a square-root factor. Note that the optimal is h_G.

Proof. We will show that given $y \in \mathbb{R}^n$ satisfying

$$R(y) := \frac{y^\top L_G y}{y^\top D y} \leq \delta,$$

and $y^\top D\vec{1} = 0$. There is a "rounding of it", meaning a threshold τ and a corresponding choice of partition

$$S = \{i \in V : y_i \leq \tau\},$$

such that

$$h(S) \leq \sqrt{2\delta},$$

since $y = \phi_2$ satisfies the conditions and gives $\delta = \lambda_2(\mathcal{L}_G)$ this proves the Lemma.

We will pick this threshold at random and use the probabilistic method to show that at least one of the thresholds works.

W.L.O.G., up to relabel the vertices, we assume $y_1 \leq \cdots \leq y_n$. Also, note that scaling of y does not change the value of $R(y)$. Also, note that scaling of y does not change the value of $R(y)$. Also, if $y^\top D\vec{1} = 0$ adding a multiple of $\vec{1}$ to y can only decrease the value of $R(y)$: the numerator does not change and the denominator

$$(y + c\vec{1})^\top D(y + c\vec{1}) = y^\top Dy + c^2\vec{1}^\top D\vec{1} \geq y^\top Dy.$$

This means that we can construct (from y by adding a multiple of $\vec{1}$ and scaling) a vector x such that

$$x_1 \leq \cdots \leq x_n,\ x_m = 0,\ \text{and } x_1^2 + x_n^2 = 1,$$

and

$$\frac{x^\top L_G x}{x^\top Dx} \leq \delta,$$

where m be the index for which $vol(\{1, \cdots, m-1\}) \leq vol(\{m, \cdots, n\})$ but $vol(\{1, \cdots, n\}) > vol(\{m, \cdots, n\})$.

We consider a random construction of S with the following distribution.
$S = \{i \in V : x_i \leq \tau\}$ where $\tau \in [x_1, x_n]$ is drawn at random with the distribution

$$\mathbb{P}\{\tau \in [a, b]\} = \int_a^b 2|\tau| d\tau,$$

where $x_1 \leq a \leq b \leq x_n$.
It is not difficult to check that

$$\mathbb{P}\{\tau \in [a, b]\} = \begin{cases} |b^2 - a^2| & \text{if } a \text{ and } b \text{ have the same sign} \\ a^2 + b^2 & \text{if } a \text{ and } b \text{ have different signs} \end{cases}$$

Let us start by estimating $\mathbb{E}[cut(S)]$:

$$\begin{aligned} \mathbb{E}[cut(S)] &= \mathbb{E}\Big[\frac{1}{2} \sum_{i \in V} \sum_{j \in V} w_{ij} 1_{(S, S^c) \text{ cuts the edge } (i,j)}\Big] \\ &= \frac{1}{2} \sum_{i \in V} \sum_{j \in V} w_{ij} \mathbb{P}\{(S, S^c) \text{ cuts the edge } (i, j)\} \end{aligned}$$

Note that $\mathbb{P}\{(S, S^c) \text{ cuts the edge } (i,j)\}$ is $|x_i^2 - x_j^2|$ is x_i and x_j have the same sign and $x_i^2 + x_j^2$ otherwise. Both cases can be conveniently upper bounded by $|x_i - x_j|(|x_i| + |x_j|)$. This means that

$$\mathbb{E}[cut(S)] \leq \frac{1}{2}\sum_{i,j} w_{ij}|x_i - x_j|(|x_i| + |x_j|) \leq \frac{1}{2}\sqrt{\sum_{ij} w_{ij}(x_i - x_j)^2}\sqrt{\sum_{ij} w_{ij}(|x_i| + |x_j|)^2}$$

where the second inequality follows from the Cauchy-Schwarz inequality.
From the construction of x we know that

$$\sum_{ij} w_{ij}(x_i - x_j)^2 = 2x^\top L_G x \leq 2\delta x^\top D x.$$

Also,

$$\sum_{ij} w_{ij}(|x_i|+|x_j|)^2 \leq \sum_{ij} w_{ij}(2x_i^2+2x_j^2) = 2\Big(\sum_i deg(i)x_i^2\Big)+2\Big(\sum_j deg(j)x_j^2\Big) = 4x^\top D x.$$

This means that

$$\mathbb{E}[cut(S)] \leq \frac{1}{2}\sqrt{2\delta x^\top D x}\sqrt{4x^\top D x} = \sqrt{2\delta}x^\top D x.$$

On the other hand,

$$\mathbb{E}[\min\{vol S, vol S^c\}] = \sum_{i=1}^{n} deg(i)\mathbb{P}\big\{x_i \text{ is in the smallest set (in terms of volume)}\big\},$$

to break ties, if $vol(S) = vol(S^c)$ we take the "smallest" set to be the one with the first indices.
Note that m is always in the largest set. Any vertex $j < m$ is in the smallest set if $x_j \leq \tau \leq x_m = 0$ and any $j > m$ is in the smallest set if $0 = x_m \leq \tau \leq x_j$. This means,

$$\mathbb{P}\{x_i \text{ is in the smallest set (in terms of volume)}\} = x_j^2,$$

which means that

$$\mathbb{E}[\min\{vol S, vol S^c\}] = \sum_{i=1}^{n} deg(i)x_i^2 = x^\top Dx.$$

Hence,

$$\frac{\mathbb{E}[cut(S)]}{\mathbb{E}[\min\{vol S, vol S^c\}]} \leq \sqrt{2\delta}.$$

Note however that because $\mathbb{E}[cut(S)]/\mathbb{E}[\min\{volS, volS^c\}]$ is not necessarily the same as $\mathbb{E}[cut(S)]/\mathbb{E}[\min\{volS, volS^c\}]$ and so, we do not necessarily have

$$\frac{\mathbb{E}[cut(S)]}{\mathbb{E}[\min\{volS, volS^c\}]} \leq \sqrt{2\delta}.$$

However, since both random variables are positive,

$$\mathbb{E}[cut(S)] \leq \mathbb{E}[\min\{volS, volS^c\}\sqrt{2\delta}] \quad i.e., \quad \mathbb{E}[cut(S) - \min\{volS, volS^c\}\sqrt{2\delta}] \leq 0,$$

which guarantees, by the probabilistic method, the existence of S such that

$$cut(S) \leq \min\{volS, volS^c\}\sqrt{2\delta},$$

which is equivalent to

$$h(S) = \frac{cut(S)}{\min\{volS, volS^c\}} \leq \sqrt{2\delta},$$

which concludes the proof of the Lemma.

Theorem. [Cheeger's inequality] The following inequality holds

$$\frac{1}{2}\lambda_2(\mathcal{L}_G) \leq h_G \leq \sqrt{2\lambda_2(\mathcal{L}_G)}.$$

Algorithm 2. Spectral Clustering for multi-clusters

- Given a graph $G = (V, E, W)$, let $v_2, \cdots, v_k$ be the eigenvector corresponding to the second through $(k-1)$th eigenvalues of the normalized Laplacian $\mathcal{L}_G = D^{-\frac{1}{2}} L_G D^{-\frac{1}{2}}$.
- Let $\phi_m = D^{-\frac{1}{2}} v_m \in \mathbb{R}^n$.
- Consider the map $\phi : V \to \mathbb{R}^{k-1}$ defined as

$$\phi(v_i) = \begin{bmatrix} \phi_2(i) \\ \vdots \\ \phi_k(i) \end{bmatrix}.$$

Cluster the n points in $k-1$ dimensions into k clusters using k-means.

Multiple clusters

There is an analogue of Cheeger's inequality.

- A natural way of evaluating k-way clustering is via the k-way expansion constant

$$\rho_G(k) = \min_{S_1,\cdots,S_k} \max_{l=1,\cdots,k} \left\{ \frac{cut(S)}{vol(S)} \right\},$$

where the maximum is over all choice of k disjoin subsets of V (but not necessarily forming a partition).

- Another natural definition is

$$\phi_G(k) = \min_{S: volS \leq \frac{1}{k} vol(G)} \frac{cut(S)}{vol(S)}.$$

Clearly,

$$\phi_G(k) \leq \rho_G(k).$$

The following are analogues of Cheeger's inequality for multiple clusters.

Theorem. Let $G = (V, E, W)$ be a graph and k a positive integer

$$\rho_G(k) \leq \mathcal{O}(k^2)\sqrt{\lambda_k}.$$

Also,

$$\rho_G(k) \leq \mathcal{O}\left(\sqrt{\lambda_{2k} \log k}\right).$$