Lecture 13: Clustering

Foundations of Data Science: Algorithms and Mathematical Foundations

Mihai Cucuringu
mihai.cucuringu@stats.ox.ac.uk

CDT in Mathematics of Random System
University of Oxford

September 26, 2023

Goals and motivation

The k-means algorithm

- k-means++

Spectral Clustering

- Graph Laplacians
- Spectral clustering of graphs & Normalized cuts

Isoperimetric number and conductance

- Cheeger's Inequality

Spectral bi-clustering

Clustering

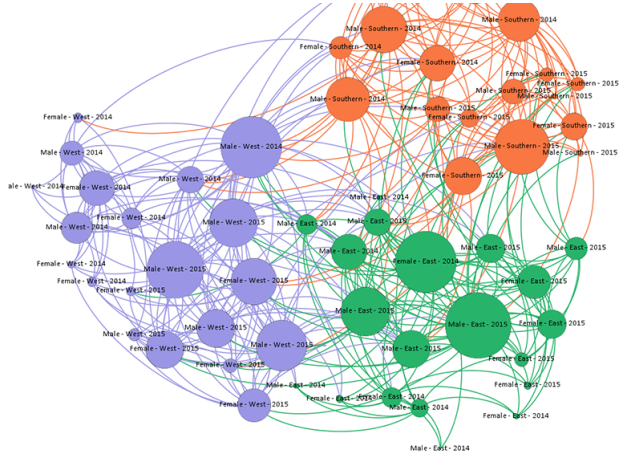
- one of the most widely used techniques in data analysis
- many data sets consist of multiple heterogeneous subsets
- aims to identify groups of nodes that exhibit similar features
- spectral clustering methods have become a fundamental tool with a broad range of applications in many areas (network science, machine learning and data mining)
- on the theoretical side
  - understanding the spectrum of the adjacency matrix (and its Laplacians) is crucial for the development of efficient algorithms with performance guarantees
  - leads to a very mathematically rich set of open problems.

Goal: Given an unlabelled data set, aim to automatically group the data points into coherent subsets/clusters. Applications:

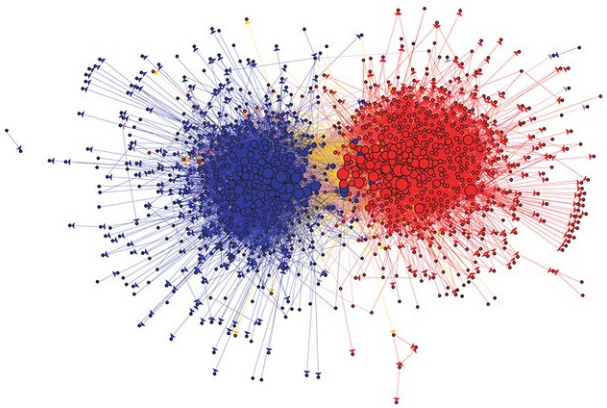
- market segmentation of shoppers based on their browsing and purchase histories
- different types of cancer from gene expression measurements
- discovering communities in social networks
- image segmentation

Clustering

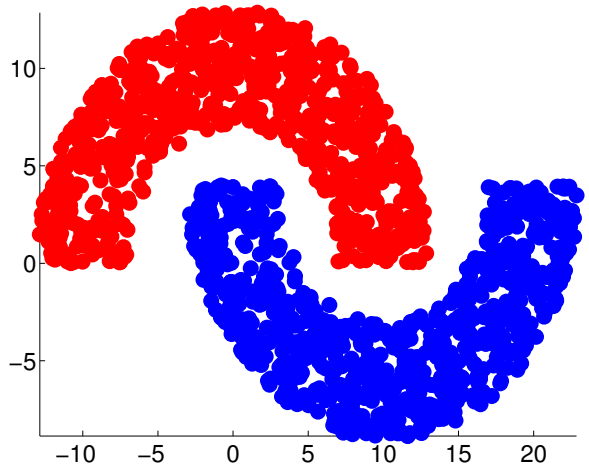
(a)



(b)



(c)



(d)

(a) Zhao, Hengshuang, et al. ”Icnet for real-time semantic segmentation on high-resolution images.” Proceedings of the European Conference on Computer Vision (ECCV) 2018

(b) `https://usaidlearninglab.org/lab-notes/demystifying-social-network-analysis-development-five-key-design-considerations`

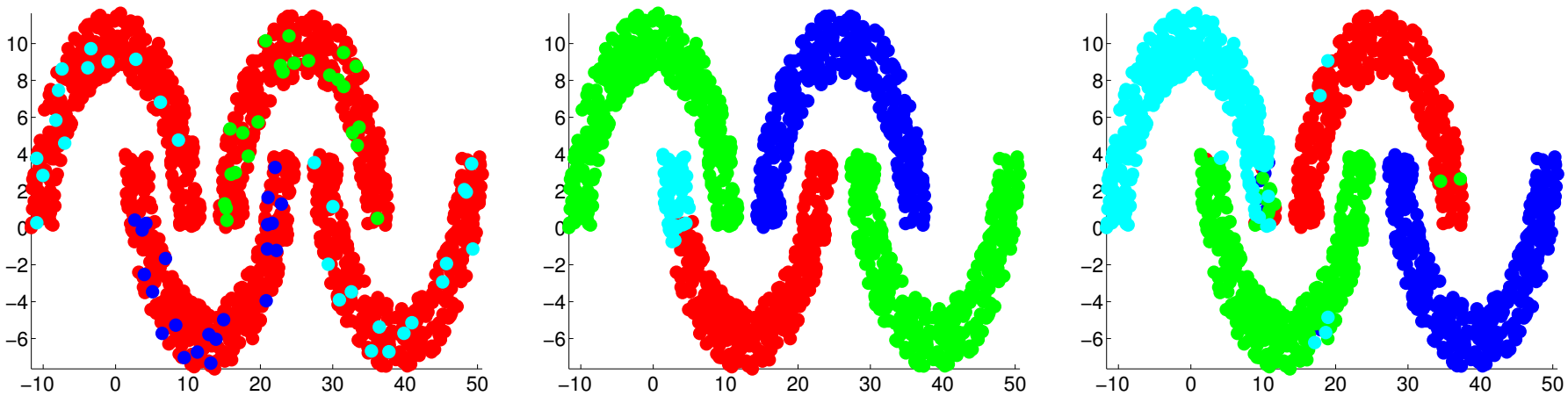
(c) Lada Adamic’s famous visual of Democrat and Republican blogs during the 2004 US election (source: Lada Adamic)

(d) Cucuringu, M., Koutis, I., Chawla, S., Miller, G., and Peng, R. (2016, May). Simple and scalable constrained clustering: a generalized spectral method. AISTATS 2016 (pp. 445-454)

Four-Moons: NoisyKnn($n = 1500, k = 30, l = 15$)

Each node is connected to its $k = 30$ nearest neighbors, and a random set of $l = 15$ nodes from throughout the network.

Why/how can clustering algorithms go wrong?



(e) CSP ($c = 75$) (f) COSf ($c = 75$) (g) FAST-GE ($c = 75$)

Figure: Segmentation for a random instance of the Four-Moons synthetic data set produced by various (constrained clustering) algorithms.

The goal of clustering

Clustering aims to simultaneously

- group similar items together *and*
- place separate dissimilar items into different groups.

Two objectives may contradict each other:

- similarity is not a transitive relation, while
- being in the same cluster is an equivalence relation.

The notion of similarity/dissimilarity between data items is central

- many ways to define; choice depends on the data set analyzed
- may be dictated by domain specific knowledge

Partition-based clustering: divides n data points into K clusters $C_1, \ldots, C_K$ such that for all $k, k' \in \{1, \ldots, K\}$

$$C_k \subset \{1, \ldots, n\}, \quad C_k \cap C_{k'} = \emptyset \ \ \forall k \neq k', \quad \bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}.$$

Clustering points in $\mathbb{R}^k$

- fundamental task in machine learning
- given a set of data points, the goal is to partition the data into a set of clusters where data points assigned to the same cluster correspond to nearby data points (points whose **Euclidean distance** in the ambient space is small)

k-means clustering (Lloyd's algorithm 1982)

- one the most popular methods used for clustering
- given $x_1, \ldots, x_n \in \mathbb{R}^p$, k-means partitions the data points in clusters $C_1, \ldots, C_k$ with centers $\mu_1, \ldots, \mu_k \in \mathbb{R}^p$

$$\min_{\substack{C_1,\ldots,C_k \\ \mu_1,\ldots,\mu_k}} \sum_{\ell=1}^{k} \sum_{i \in C_\ell} ||x_i - \mu_\ell||^2 \tag{1}$$

- given the partition, the optimal centers are given by

$$\mu_\ell = \frac{1}{|C_\ell|} \sum_{i \in C_\ell} x_i, \quad \ell = 1, 2, \ldots, k \tag{2}$$

(take partial derivatives wrt μ_ℓ, and set to zero).

Clustering points in $\mathbb{R}^k$: Lloyd's algorithm (1982)

k-means: iterative alternating optimization approach that alternates:

- (1) Given centers $\mu_1, \ldots, \mu_k \in \mathbb{R}^p$, assign each point to cluster
$$\ell = \text{argmin}_{\ell=1,\ldots,k} ||x_i - \mu_\ell||$$
- (2) Update the centers
$$\mu_\ell = \frac{1}{|C_\ell|} \sum_{i \in C_\ell} x_i$$

Each step decreases the OBJ, guaranteed to converge (as seen later)

Drawbacks:

- not guaranteed to converge to the global optimum
- may get stuck in local optima (suboptimal solutions)
- actually, it might not necessarily converge to a local minimum
- optimizing is NP-hard: no poly. time algo that works in worst-case
- need to know k a-priori (or run the algo for different k's)
- expects points to be defined in a Euclidean space; sometimes we only have **pairwise** affinities btw. points
- no theoretical approximation guarantees exist

Exercise: Show that

- $p = 1$: the algorithm terminates in at most n^{k-1} steps
- updating cluster means μ_ℓ^{t+1} does not increase the obj. fcn.

Within-cluster deviance

- Goal: divide data items into a *pre-assigned number K of clusters* $C_1, \ldots, C_K$ where for all $k, k' \in \{1, \ldots, K\}$,

$$C_k \subset \{1, \ldots, n\}, \quad C_k \cap C_{k'} = \emptyset \ \ \forall k \neq k', \quad \bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}.$$

- Define $W(C_k)$ to be a measure of how different the observations are within cluster k; most common choice is to use squared distances

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \|x_i - x_{i'}\|_2^2 = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \qquad (3)$$

Exercise:

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \|x_i - x_{i'}\|_2^2 = 2 \sum_{i \in C_k} \|x_i - \mu_k\|_2^2, \qquad (4)$$

where $\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$.

k-means

- recall from earlier slide: given $x_1, \ldots, x_n \in \mathbb{R}^p$, k-means partitions the data points in clusters $C_1, \ldots, C_k$ with centers $\mu_1, \ldots, \mu_k \in \mathbb{R}^p$

$$\min_{\substack{C_1,\ldots,C_k \\ \mu_1,\ldots,\mu_k}} \sum_{\ell=1}^{k} \sum_{i \in C_\ell} ||x_i - \mu_\ell||^2 \tag{5}$$

- based on previous exercise, the above is equivalent to solving

$$\min_{C_1,\ldots,C_k} \sum_{\ell=1}^{k} \frac{1}{|C_\ell|} \sum_{i,j \in C_\ell} ||x_i - x_j||^2 \tag{6}$$

- this cost function is a weighted average of the cluster variances, where the weights are proportional to cluster size (number of points $|C_\ell|$).
- finding the solution to the k-means objective is a highly non-convex problem (NP-hard problem)
- assuming the conjecture $P \neq NP$, there is no polynomial-time algorithm for solving the k-means objective.

Within-cluster deviance

Each cluster is represented using a *cluster centroid* (or *prototype*) μ_k

Within-cluster deviance:

$$W(C_k, \mu_k) = \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 = \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2 \tag{7}$$

The overall quality of the clustering is given by the total within-cluster deviance

$$W = \sum_{k=1}^{K} W(C_k, \mu_k) = \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2 \tag{8}$$

k-means

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|_2^2 \tag{9}$$

where $c_i = k$ if and only if $i \in C_k$.

- Given partition $\{C_k\}$, we can find the optimal prototypes easily by differentiating W with respect to μ_k:

$$\frac{\partial W}{\partial \mu_k} = 2 \sum_{i \in C_k} (x_i - \mu_k) = 0 \qquad \Rightarrow \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- Given prototypes, we can find the optimal partition by assigning each data point to the closest cluster prototype:

$$c_i = \text{argmin}_k \|x_i - \mu_k\|_2^2$$

However, joint minimization over both partitions and centroids is computationally difficult.

The k-means algorithm (Lloyd's algorithm)

The k-means algorithm returns a *local optimum* of the objective function W, using iterative and alternating minimization.

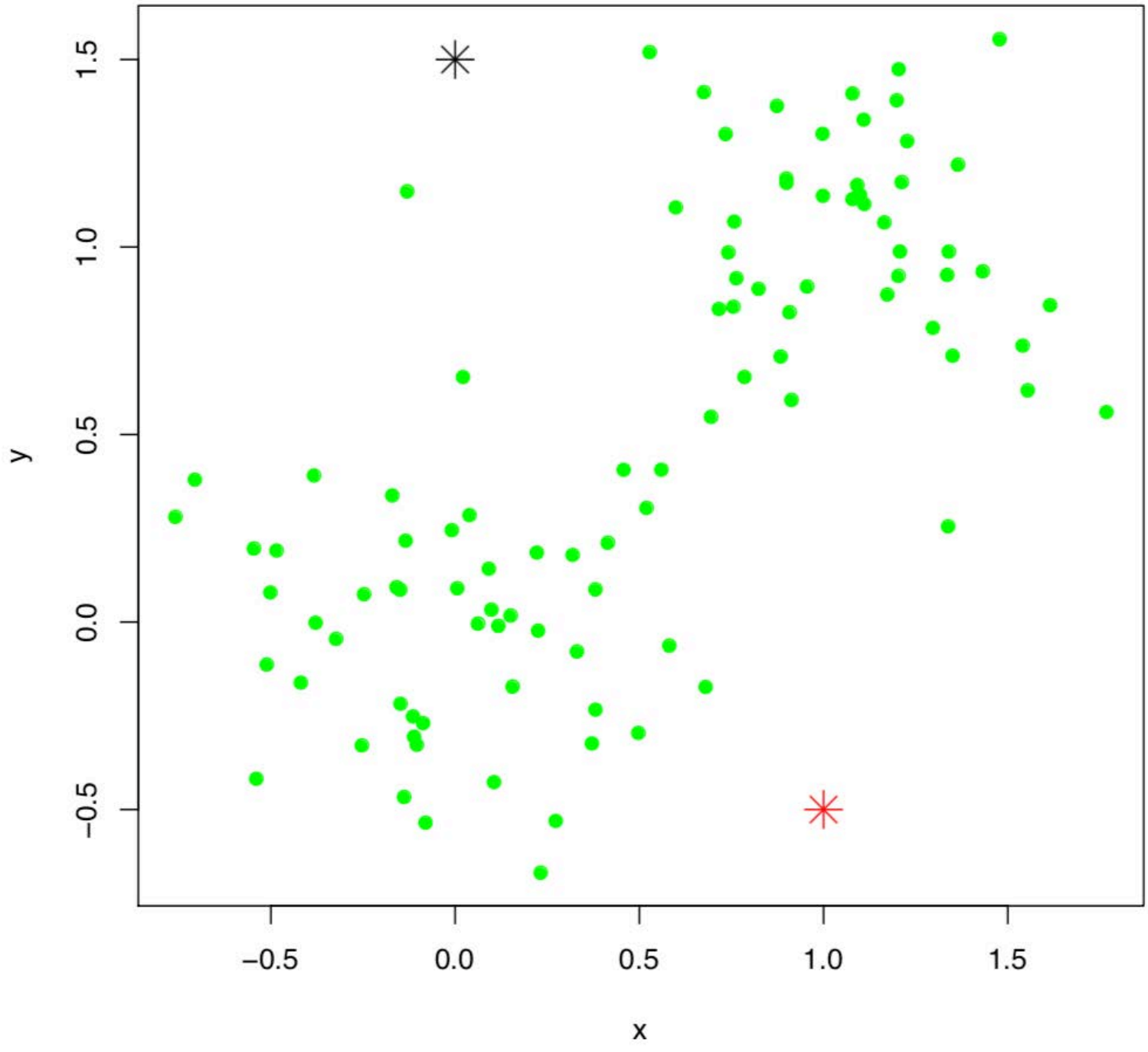
1. Randomly initialize K cluster centroids $\mu_1, \ldots, \mu_K$
2. *Cluster assignment:* For each $i = 1, \ldots, n$, assign each x_i to the cluster with the nearest centroid,

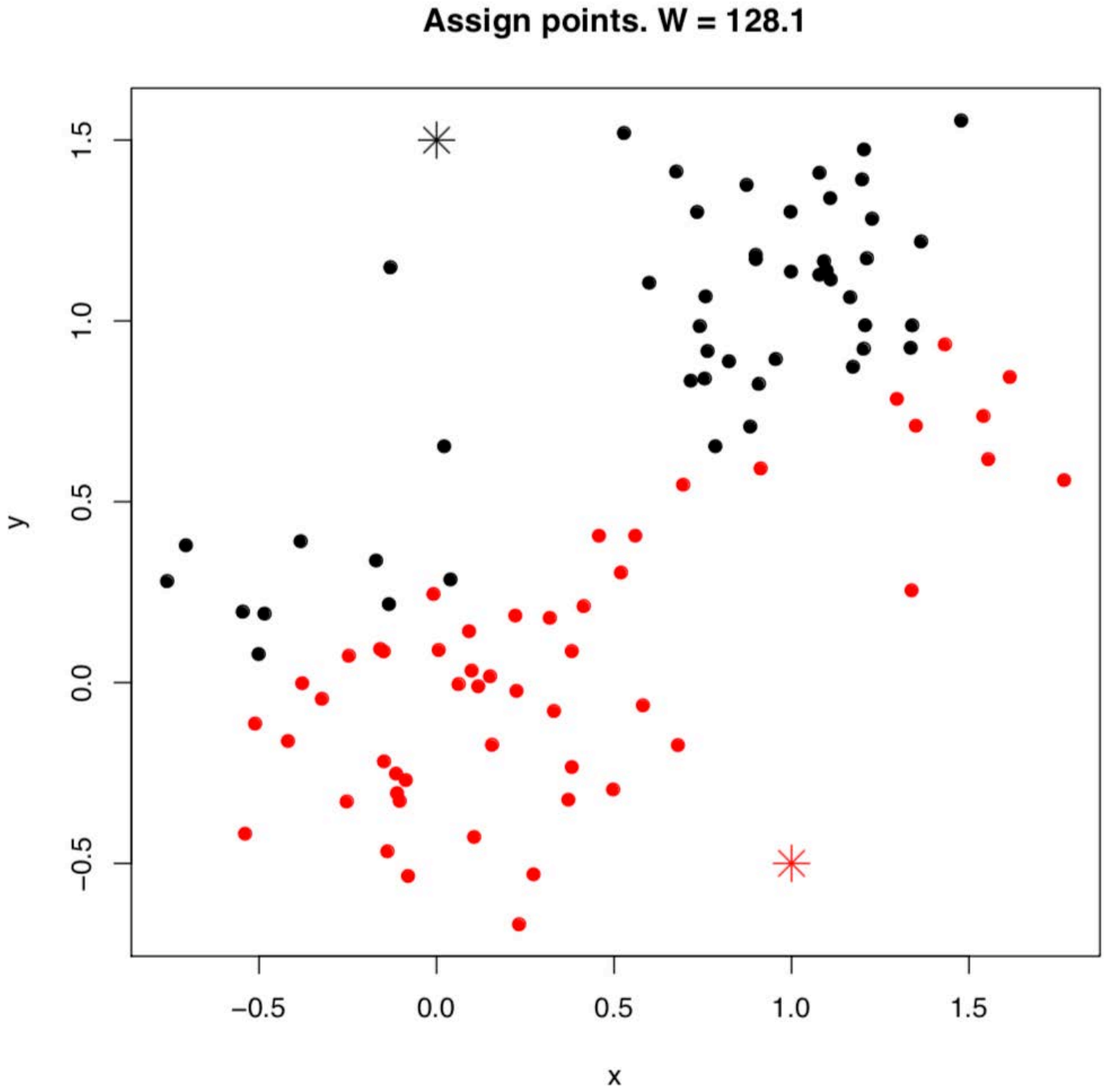
$$c_i := \text{argmin}_k \|x_i - \mu_k\|_2^2$$

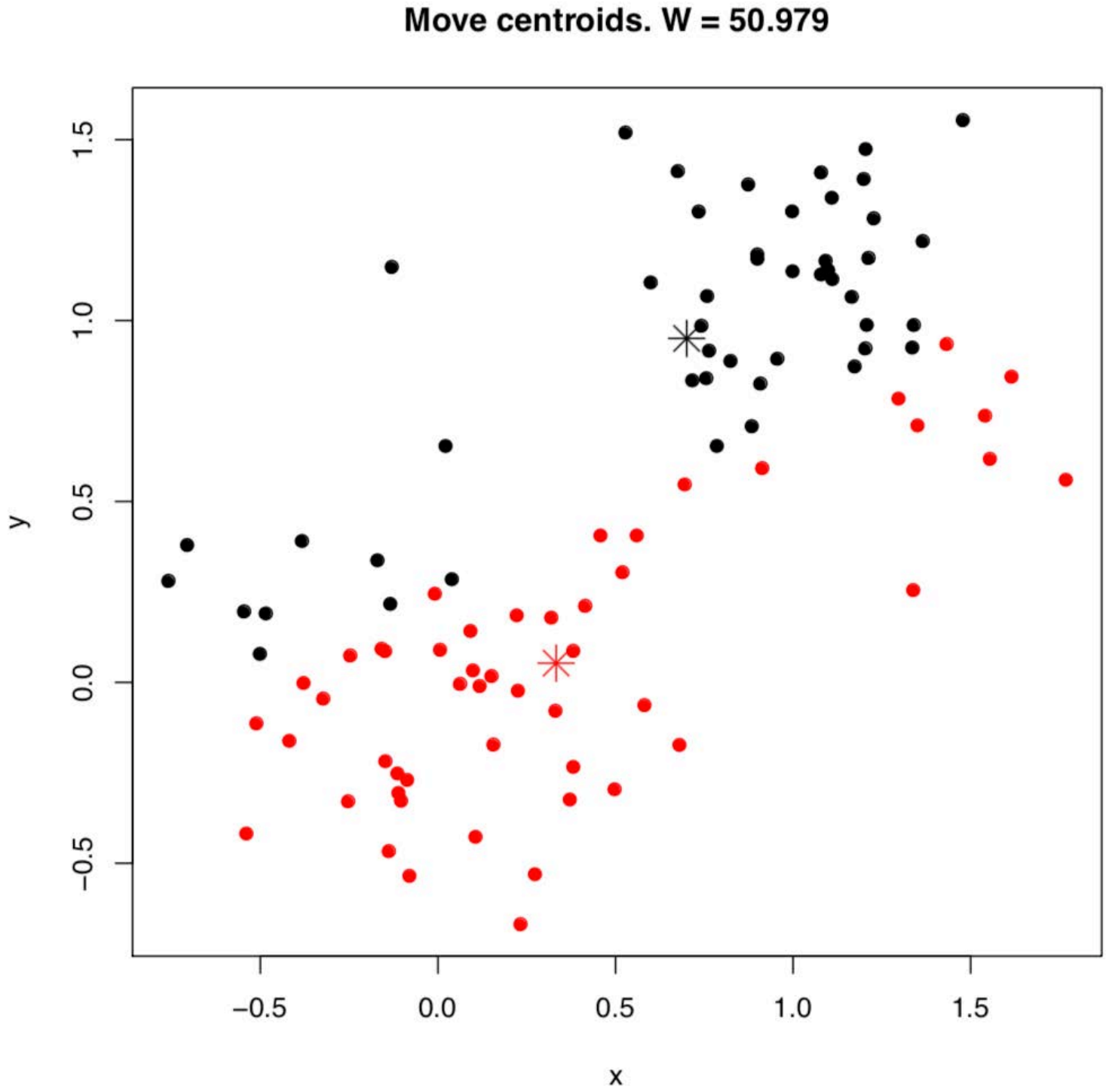
   Set $C_k := \{i : c_i = k\}$ for each k.
3. *Move centroids:* Set $\mu_1, \ldots, \mu_K$ to the averages of the new clusters:

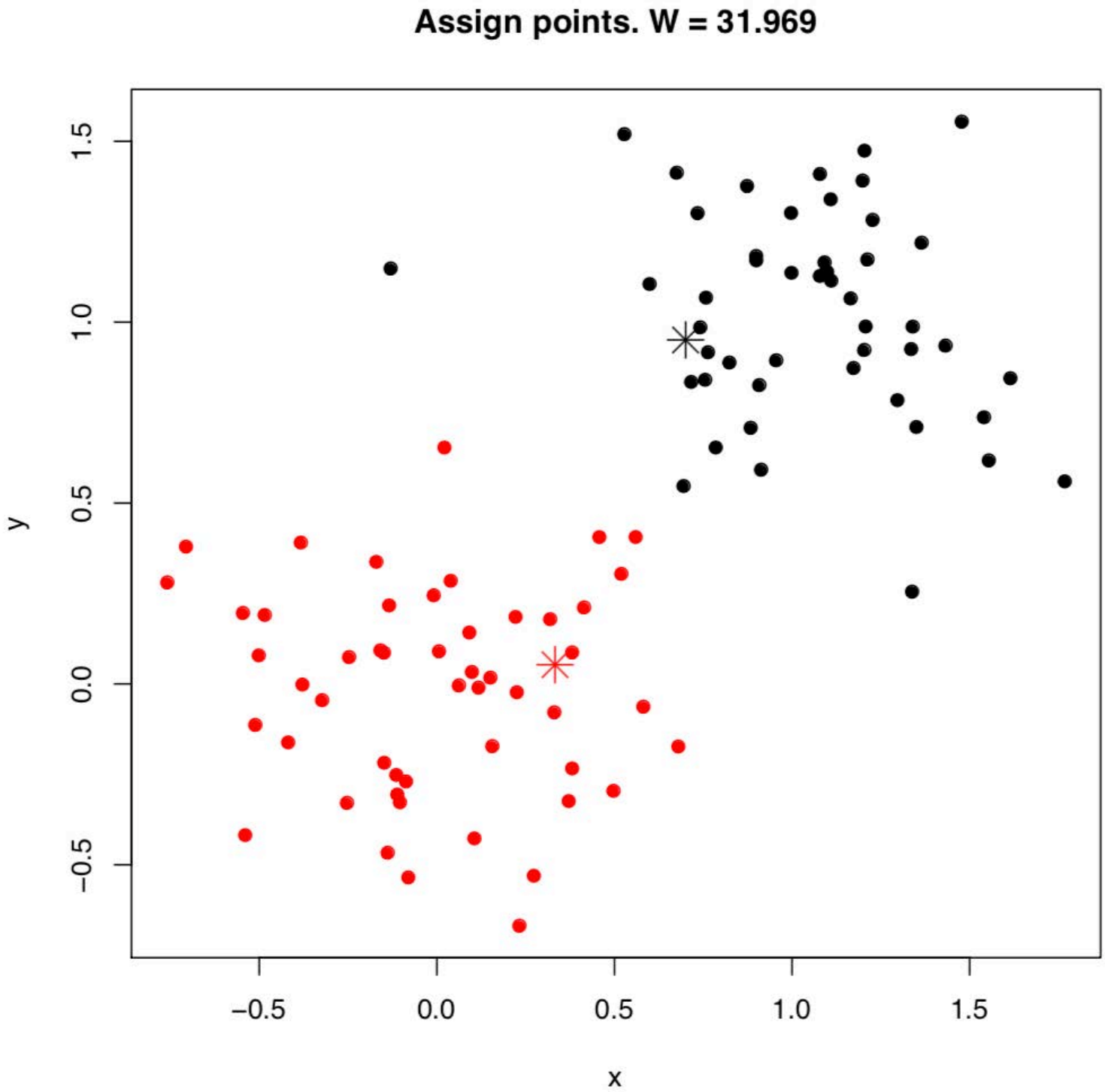
$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

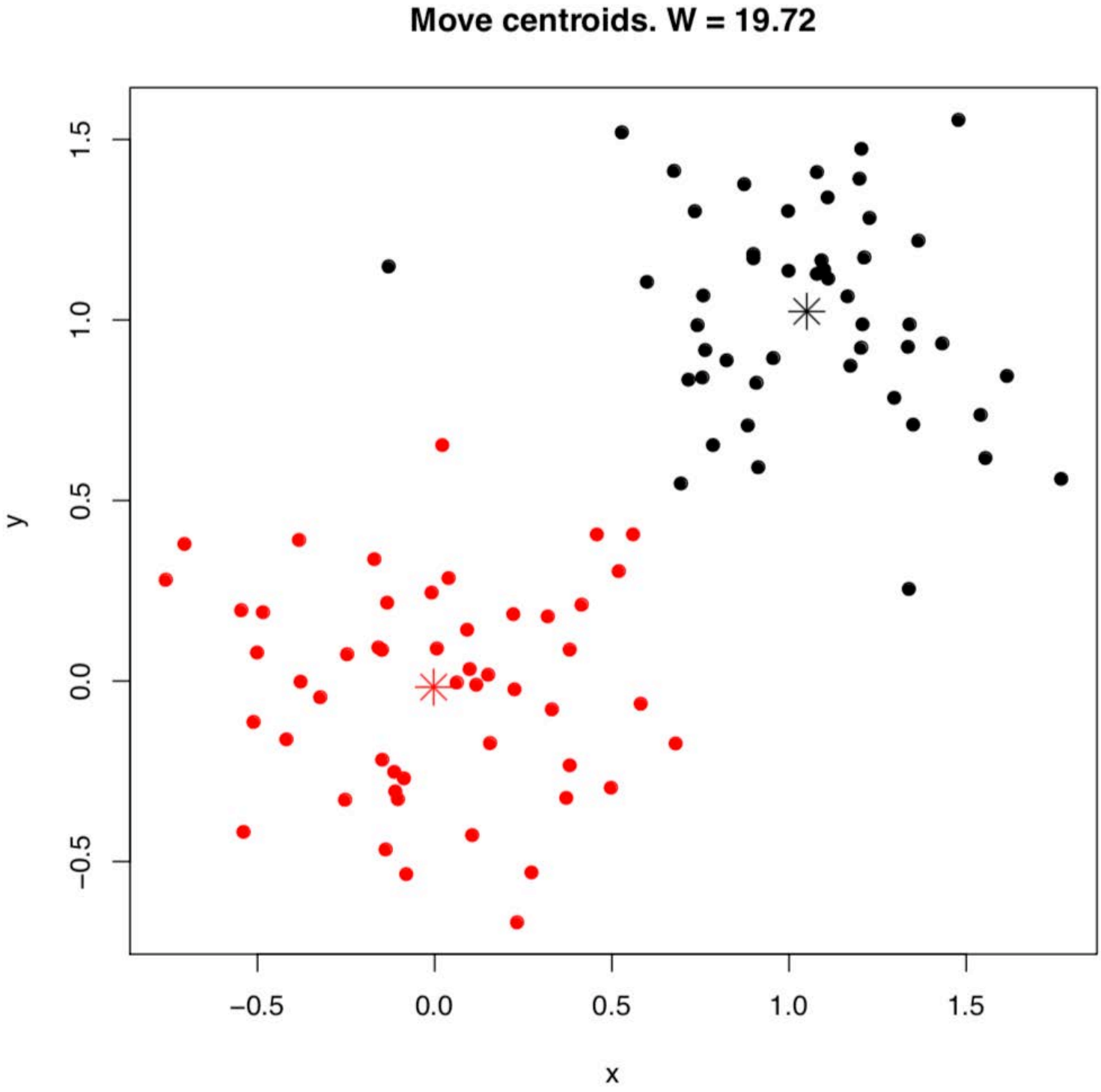
4. Repeat steps 2-3 until convergence.
5. Return the partition $\{C_1, \ldots, C_K\}$ and means $\mu_1, \ldots, \mu_K$.

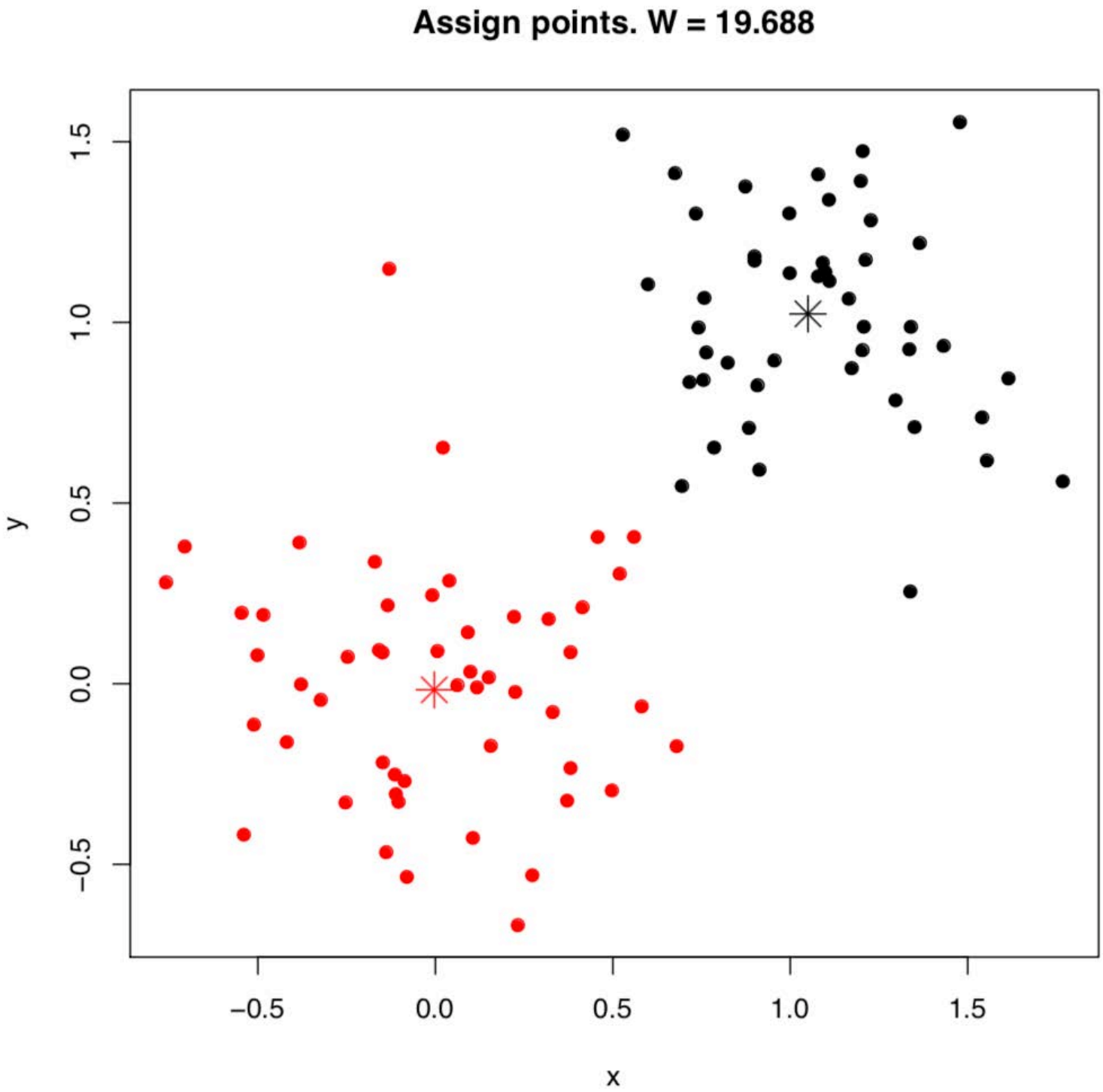

y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5

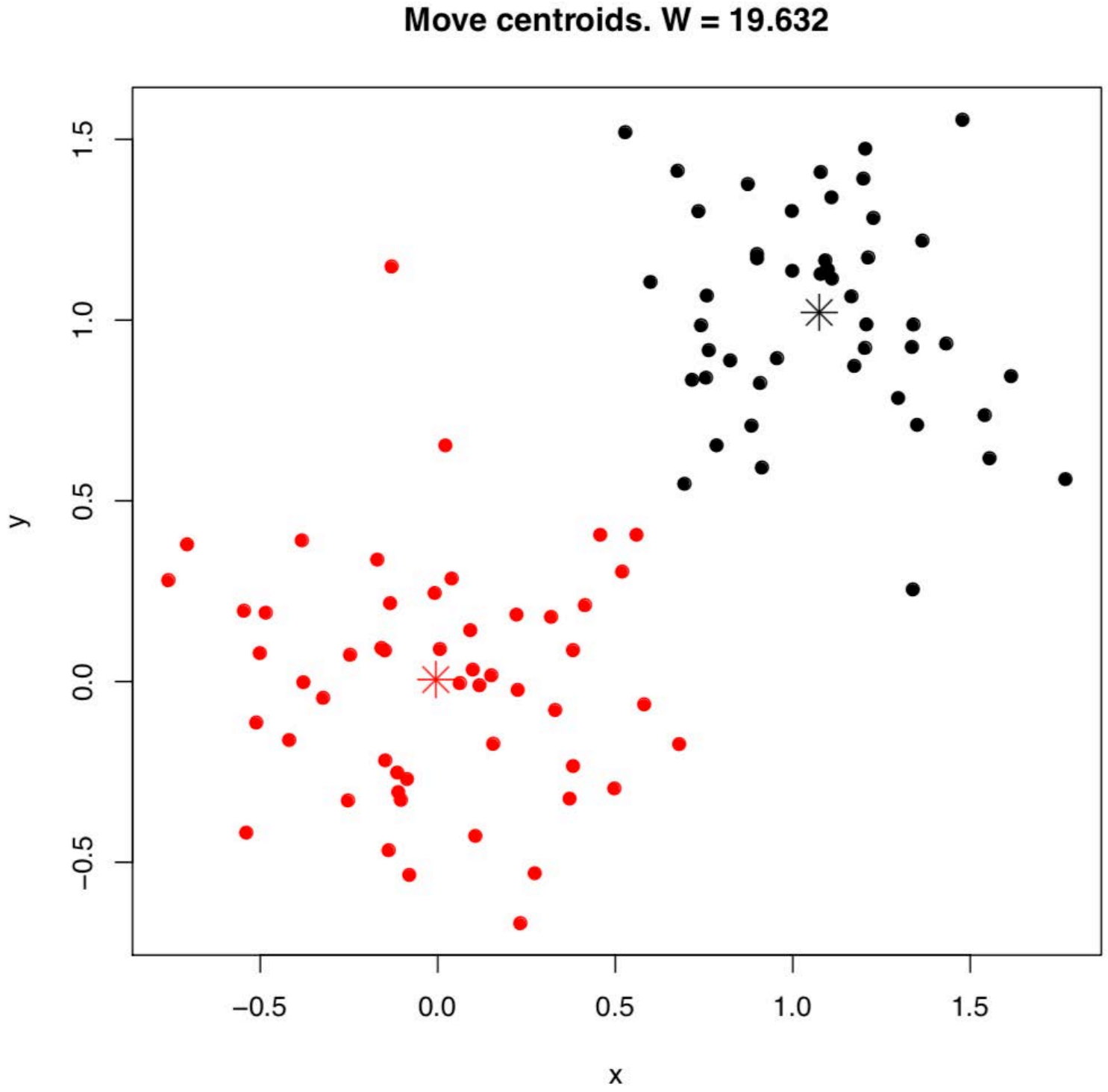

Assign points. W = 128.1
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Move centroids. W = 50.979
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Assign points. W = 31.969
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Move centroids. W = 19.72
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Assign points. W = 19.688
y
x
−0.5
0.0
0.5
1.0
1.5
−0.5
0.0
0.5
1.0
1.5


Move centroids. W = 19.632
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5

k-means: multiple starts

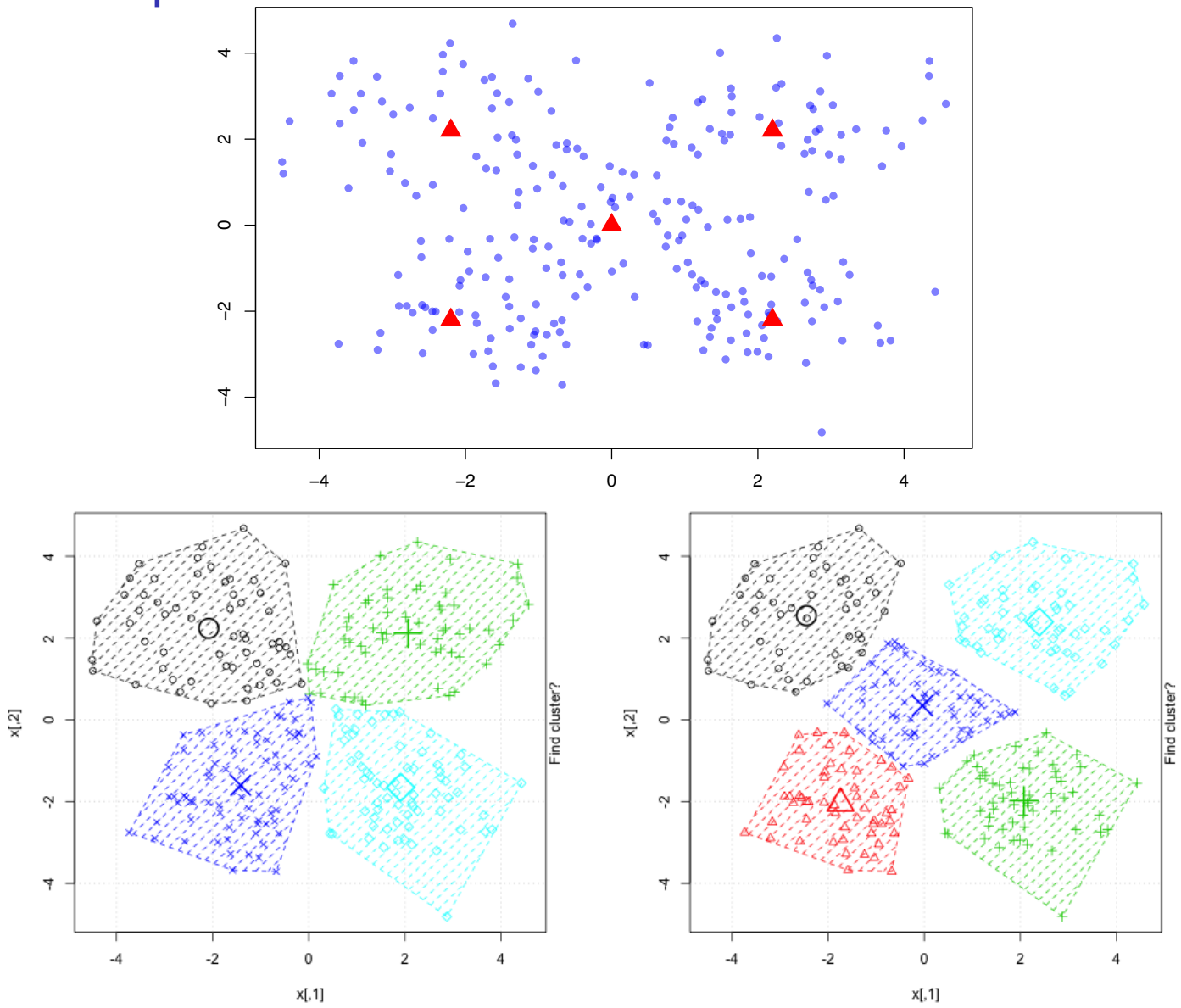


Figure: **Top:** Example data set with 5 close clusters of points; the true cluster means are shown as red triangles. **Bottom:** Two different solutions from k-means with 5 clusters.

k-means (Lloyd's) algorithm

The algorithm stops in a finite number of iterations

- between steps 2 and 3, W either stays constant or it decreases
- this implies that we never revisit the same partition
- as there are only finitely many partitions, the number of iterations cannot exceed this.

The K-means algorithm need not converge to global optimum

- k-means can get stuck at sub-optimal configurations
- the result depends on the starting configuration
- typically perform a number of runs from different initial values, and pick the end result with minimum W.

Complexity of k-means (in practice, linear):

- the running time of Lloyd's algorithm is O($nkpt$), where t is the number of iterations needed until convergence.
- if a strong cluster structure exists, the number of iterations until convergence is often small; little improvement beyond first 10 iters
- worst case running time is super-polynomial in the input size

K-means

- Let $\mathbf{X}$ be an $n \times p$ data matrix, with x_i denoting the ith row of $\mathbf{X}$. Let $C_1, C_2, \ldots, C_K$ be a set of K clusters of observations. Let $\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$ be the mean of the observations in cluster k.

Exercise: show the following identity holds for a given cluster C_k

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2 \tag{10}$$

Solution on the next slide.

- Exercise: show the following identity holds for a given cluster C_k

$$\frac{1}{|C_k|}\sum_{i,i' \in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i \in C_k}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2 \tag{11}$$

Let $n_k = |C_k|$. Then

$$\frac{1}{|C_k|}\sum_{i,i' \in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 \tag{12}$$

$$= \frac{1}{n_k}\sum_{i \in C_k}\sum_{l \in C_k}\sum_{j=1}^{p}(x_{ij}-x_{lj})^2$$

$$= \frac{1}{n_k}\sum_{j=1}^{p}\sum_{i \in C_k}\sum_{l \in C_k}\Big((x_{ij}-\mu_{kj})-(x_{lj}-\mu_{kj})\Big)^2$$

$$= \frac{1}{n_k}\sum_{j=1}^{p}\sum_{i \in C_k}\sum_{l \in C_k}(x_{ij}-\mu_{kj})^2 - 2(x_{ij}-\mu_{kj})(x_{lj}-\mu_{kj}) + (x_{lj}-\mu_{kj})^2$$

$$= \frac{1}{n_k}\sum_{j=1}^{p}\Big(n_k\sum_{i \in C_k}(x_{ij}-\mu_{kj})^2 + n_k\sum_{l \in C_k}(x_{lj}-\mu_{kj})^2\Big)$$

$$= 2\sum_{i \in C_k}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2$$

k-means objective function decomposition

- Exercise: Let v_k be a p-dim vector. Show that

$$\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-v_{kj})^2=\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2+|C_k|\sum_{j=1}^{p}(\mu_{kj}-v_{kj})^2 \quad (13)$$

$$\begin{aligned}\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-v_{kj})^2 &= \sum_{j=1}^{p}\sum_{i\in C_k}\Big((x_{ij}-\mu_{kj})+(\mu_{kj}-v_{kj}\Big)^2\\ &= \sum_{j=1}^{p}\Big(\sum_{i\in C_k}(x_{ij}-\mu_{kj})^2+\sum_{i\in C_k}(\mu_{kj}-v_{kj})^2\Big)\\ &= \sum_{j=1}^{p}\Big(\sum_{i\in C_k}(x_{ij}-\mu_{kj})^2+n_k(\mu_{kj}-v_{kj})^2\Big)\\ &= \sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\mu_{kj})^2+|C_k|\sum_{j=1}^{p}(\mu_{kj}-v_{kj})^2\end{aligned}$$

- Exercise: We run k-means clustering on a 1-D data set ($p=1$, each observation is a single real number; assume these real numbers are distinct). Show that the algorithm terminates in at most n^{K-1} steps.

Convergence of k-means

- recall: $\min_{\substack{C_1,\ldots,C_k \\ \mu_1,\ldots,\mu_k}} \sum_{\ell=1}^{k} \sum_{i \in C_\ell} ||x_i - \mu_\ell||^2$
- argue that the sum of the squares of the distances, of each point to its cluster center, always improves (decreases)
- for a different set of cluster means, denoted $v_1, \ldots, v_k$, (13) shows

$$\sum_{i \in C_k} ||x_i - v_k||^2 = \sum_{i \in C_k} ||x_i - \mu_k||^2 + |C_k| \cdot \ ||\mu_k - v_k||^2 \qquad (14)$$

- $C^{(t)}$ clustering & $\mu_k^{(t)}$ mean of observations in cluster k at iter. t
- suppose we have just updated the cluster assignments for the next iteration, ie we have determined what $C^{(t+1)}$ is, and so the current value of the objective function (using $C^{(t+1)}$ and $\mu_k^{(t)}$) is

$$\sum_{\ell=1}^{k} \sum_{i \in C_\ell^{(t+1)}} ||x_i - \mu_k^{(t)}||^2 \geq \sum_{\ell=1}^{k} \sum_{i \in C_\ell^{(t+1)}} ||x_i - \mu_k^{(t+1)}||^2 \qquad (15)$$

  where the inequality follows from employing (14). (since $\mu_k^{(t)}$ may not be the same as the cluster means for the assignments $C^{(t+1)}$ (as the cluster assignments can change) when we update the cluster means to $\mu_k^{(t+1)}$ have (15)).
- updating cluster means to $\mu_k^{(t+1)}$ never increases the obj. fcn.

k-means++

- k-means selects the k clusters randomly, which can lead to poor partitions
- k-means++ incrementally chooses a set of k centers by sampling the next center from a distribution where every point has probability proportional to its squared distance to the currently closest center.
- approximation guarantee of $O(\log k)$ (the solution computed by the seeding algorithm has expected cost of $O(\log k)$ times the cost of the optimum solution)
- in practice, it is then often used as a starting solution for Lloyd's algorithm (which never decreases the objective function)

k-means++

Algorithm 1 k-Means++

INPUT: A set of n data points $N \subset \mathbb{R}^d$, the number of clusters k

1. Randomly select an initial center c_1 from N
2. **repeat** for $i \in 1, 2, \ldots, k-1, k$
   Select the next center $c_i = x \in N$ with the probability
   $$P(x) = \frac{D(x)^2}{\sum_{x' \in N} D(x')^2} \tag{16}$$
   Where x' is the closest center that has already been chosen and $D(x')$ is the distance to that center.
3. Continue with the standard k-means algorithm.

Clustering graphs

- Consider an undirected graph $G = (V, E)$ with n vertices
- Each $\{i, j\} \in E$ has an associated **positive** weight $w_{ij} \geq 0$ (similarity between vertices).
- **Goal:** Partition V into **clusters** such that intra-cluster edges have high weight and inter-cluster edges have low weight.
- **Applications:** Statistics, computer science, biology etc.

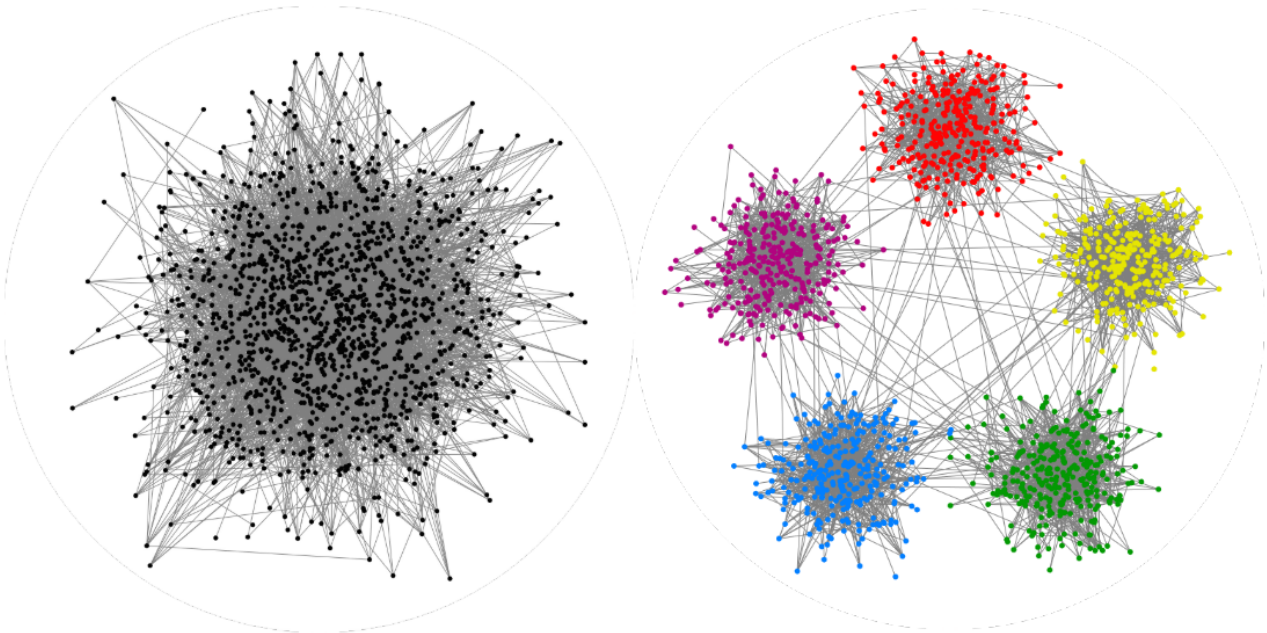


Figure: [Abbe '17] Obtain right graph from left (scrambled) graph

Recall the definition of the graph Laplacian

- Graph Laplacian $L = D - A$ (most popular version)
- A is the adjacency matrix of the graph $A_{ij} \geq 0$
- D is a diagonal matrix, D_{ii} denoting the degree of node i

$$L(i,j) \stackrel{\text{def}}{=} \begin{cases} deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \quad \text{and} \quad (i,j) \in E(G) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

- L is symmetric
- eigenvalues $\lambda_0 \leq \lambda_1 \leq \lambda_{n-1}$, eigenvectors $\phi_0, \phi_1, \ldots, \phi_{n-1}$
- every row sum and column sum of L is zero
- thus, $\lambda_0 = 0$, and $\phi_0 = \mathbf{1} \stackrel{\text{def}}{=} [1, 1, \ldots, 1]^T$ since $L\,\mathbf{1} = 0\,\mathbf{1}$
- the second smallest (smallest non-zero) eigenvalue of L is the algebraic connectivity (Fiedler value, spectral gap) of G

Lemma If $G = (V, E)$ is **connected** and $\lambda_0 \leq \lambda_1 \leq \lambda_{n-1}$ are the eigenvalues of its Laplacian L, then it holds true that $\lambda_1 > 0$ (Stronger result: the multiplicity of the zero eigenvalue is equal to the number of connected components).

Properties of the (random-walk) Laplacian matrix P

Let W denote the adjacency matrix of a weighted graph.

Lemma All the eigenvalues of $P = D^{-1}W$ satisfy $|\lambda_i| \leq 1, \forall i = 1, \ldots, n$

- Let λ be an eigenvalue of P with associated eigenvector x

$$\lambda\, x = P\, x$$

- Let $i_m = \underset{1 \leq i \leq n}{\operatorname{argmax}} |x_i|$,
- Consider the following

$$\lambda x_{i_m} = \sum_{j=1}^{n} P_{i_m j}\, x_j$$

thus

$$|\lambda| = \left| \sum_{j=1}^{n} P_{i_m j}\, \frac{x_j}{x_{i_m}} \right| \leq \sum_{j=1}^{n} P_{i_m j}\, \left| \frac{x_j}{x_{i_m}} \right| \leq \sum_{j=1}^{n} P_{i_m j} = 1 \qquad (18)$$

Combinatorial Laplacian $L = D - A$

We can further redefine this as follows. Let $G_{1,2}$ be the graph on two vertices u and v and one edge $e_{u,v}$
Define

$$L_{G_{1,2}} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

- If $x = [x_1 x_2]^T$, note that

$$x^T L_{G_{1,2}} x = (x_1 - x_2)^2$$

- Let $G_{u,v}$ denote a graph on n vertices with only one edge (between u and v)
- Define the Laplacian of $G_{u,v}$ as

$$L_{G_{u,v}}(i,j) = \begin{cases} 1 & \text{if } i = j \quad \text{and} \quad i \in \{u, v\} \\ -1 & \text{if } i = u \quad \text{and} \quad j = v, \text{ or vice versa} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

Combinatorial Laplacian

For a general graph $G = (V, E)$ define

$$L(G) \stackrel{\text{def}}{=} \sum_{(u,v)\in E} L_{G_{u,v}}$$

- Note that $L_{G_{1,2}}$ has eigenvalues 0 and 2, and so is positive semidefinite
- recall that a symmetric matrix M is positive semidefinite (PSD) if all of its eigenvalues are non-negative
- recall equivalent PSD condition

$$x^T M x \geq 0, \text{ for all } x \in \mathbb{R}^n$$

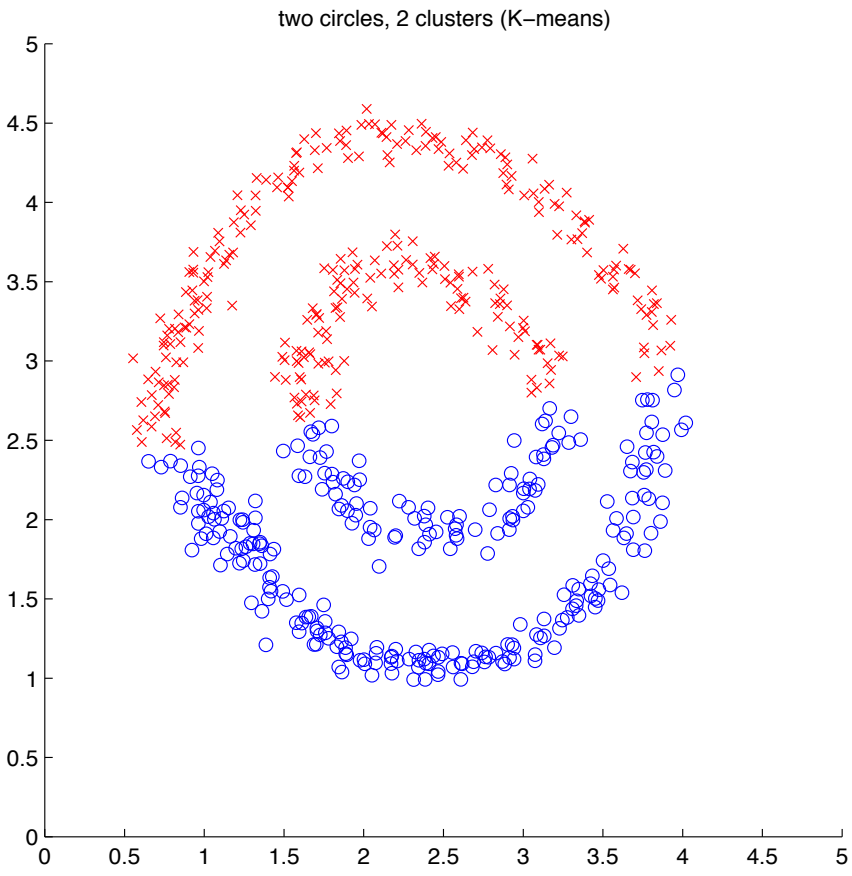
- using the previous observation that

$$x^T L_{G_{1,2}} x = (x_1 - x_2)^2$$

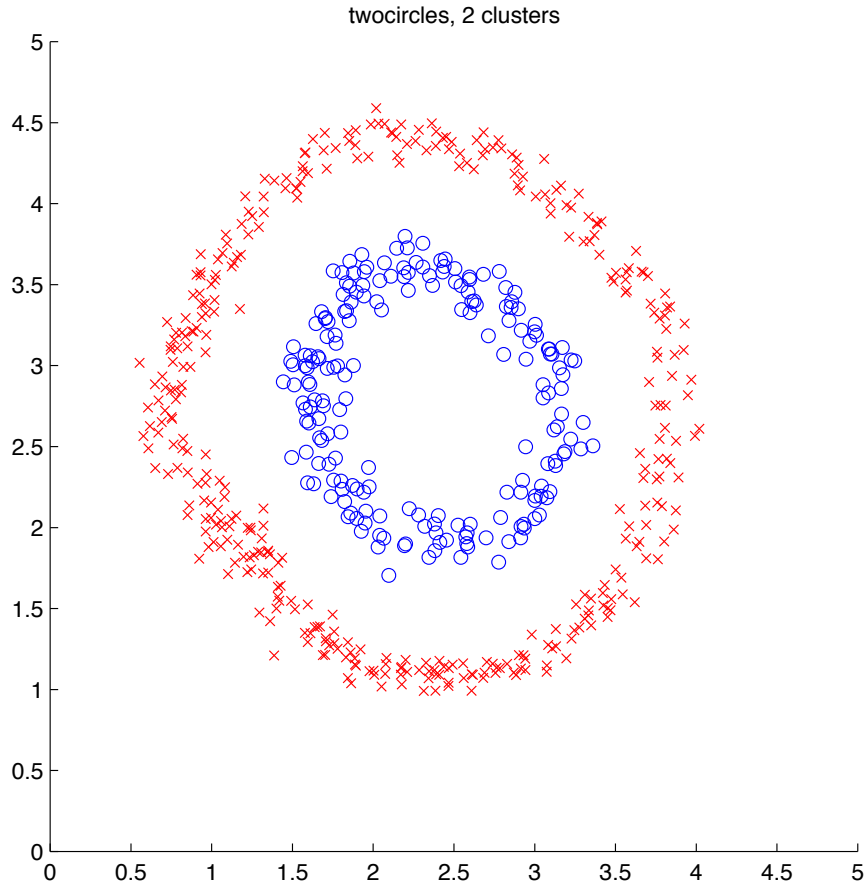
we can show that the Laplacian of every graph is PSD

$$x^T L_G \, x = \sum_{(u,v)\in E} (x_u - x_v)^2$$

Motivation: k-means vs. spectral clustering



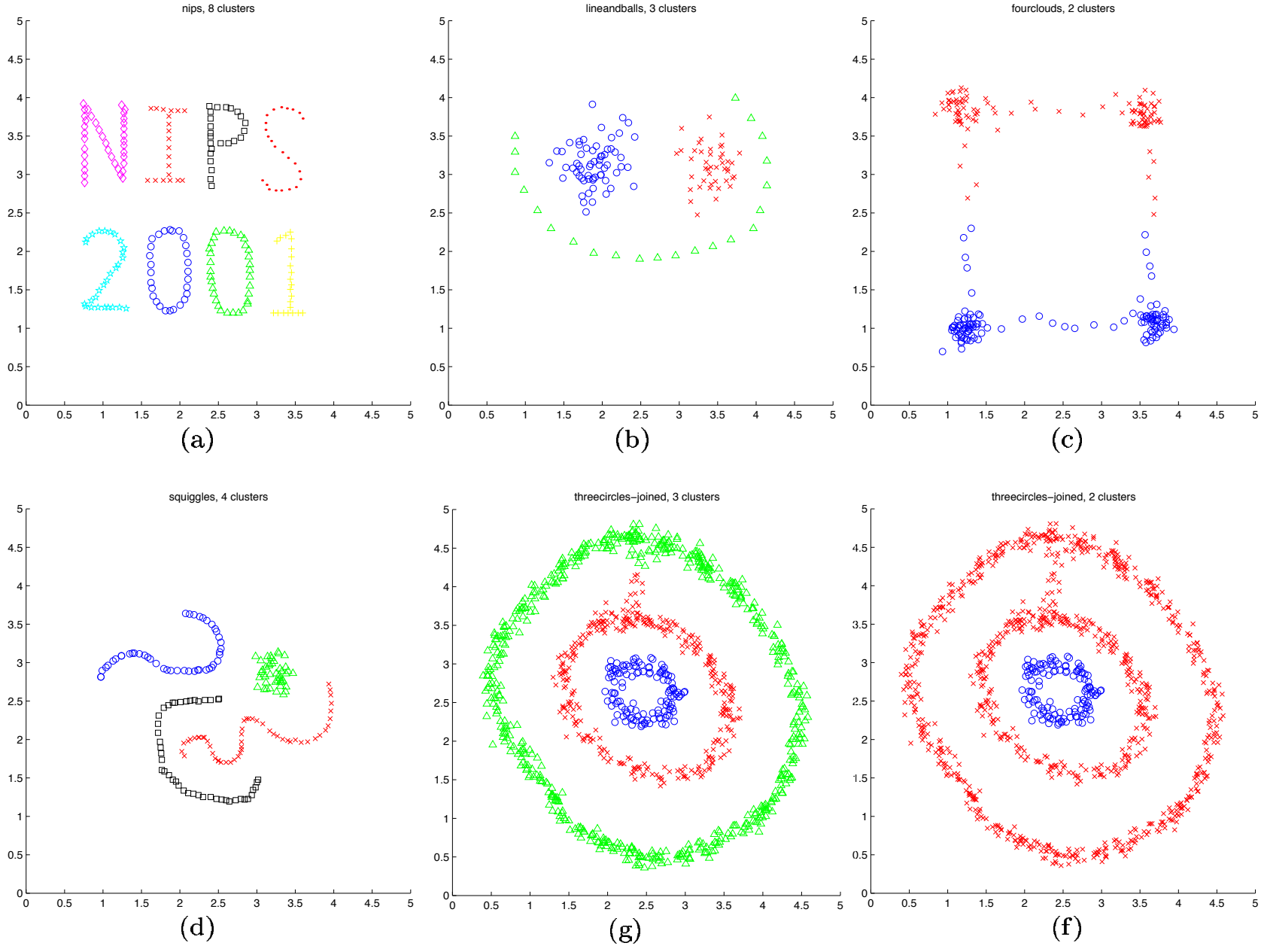
(a) k-means



(b) Spectral clustering

[Shi & Malik 2000; Ng, Jordan, Weiss NIPS 2001]

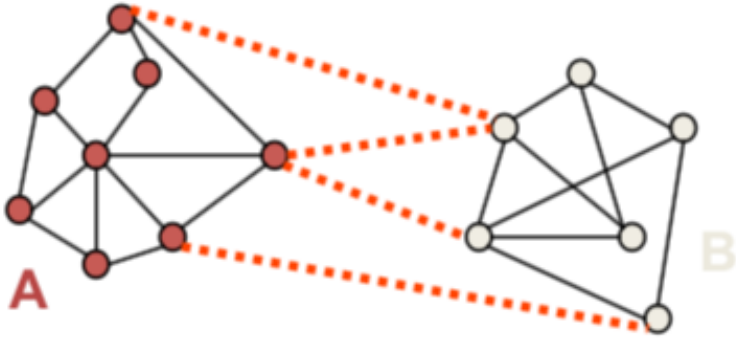
Spectral clustering

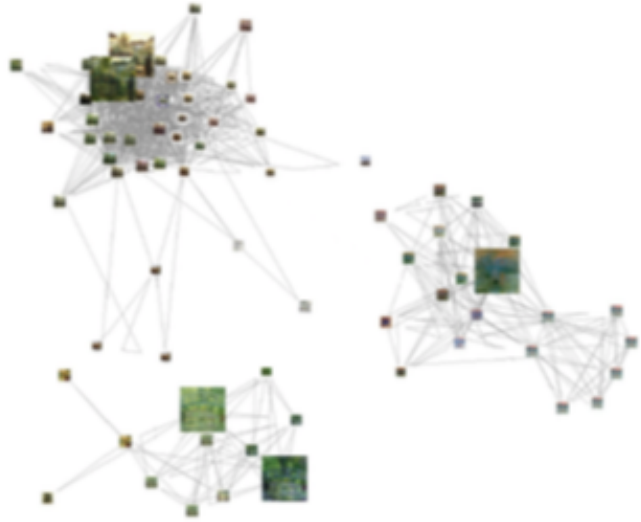


[Shi & Malik 2000; Ng, Jordan, Weiss NIPS 2001]

From points to graphs

- Group points in $\mathbb{R}^D$ together based on links a newly created graph G.
- Each of $x_1, x_2, \ldots, x_n \in \mathbb{R}^D$ becomes a node in the graph G.
- The similarity graph G can be
  - a fully connected graph (will not lead to a scalable approach)
  - k-nearest-neighbor graph (where each node is connected to its k nearest neighbors)
  - disc graph (where each node is connected to everyone within a ball of radius r)
- the latter two approaches will lead to a sparse graph G, which is key in many applications.





[Slide from James Hays]

Clustering graphs

- Consider an undirected graph $G = (V, E)$ with n vertices
- Each $\{i, j\} \in E$ has an associated **positive** weight $w_{ij} \geq 0$ (similarity between vertices).
- Potentially constructed via a kernel K_ϵ such that

$$w_{ij} = K_\epsilon(||x_i - x_j||), \quad (ij) \in E(G) \tag{20}$$

- **Goal:** Partition V into **clusters** s.t. intra-cluster edges have high weight and inter-cluster edges have low weight.
- **Applications:** Statistics, computer science, biology etc.

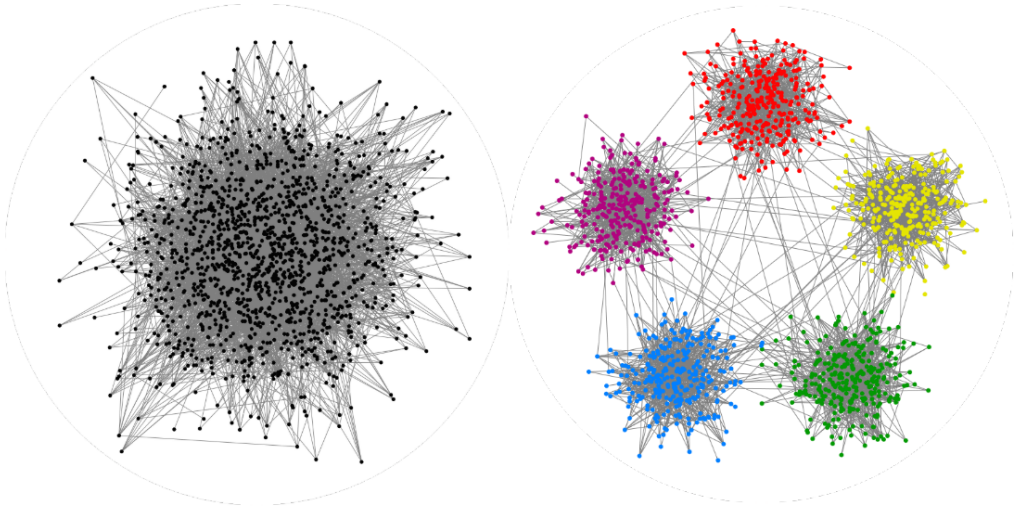


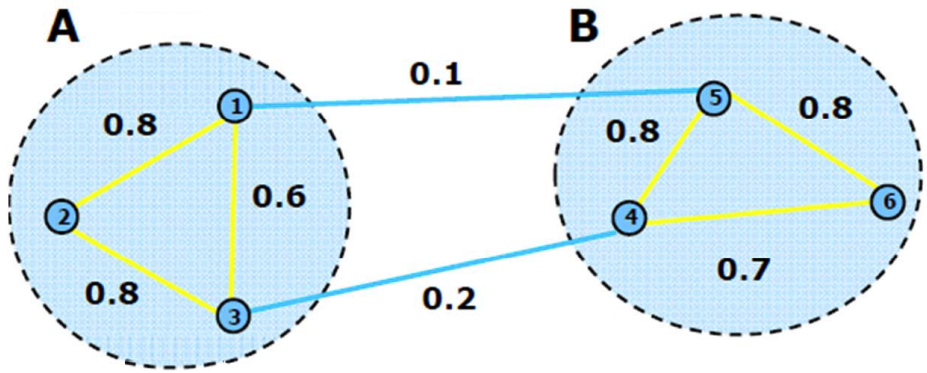
Figure: [Abbe '17] Recover the right graph from the left (scrambled) graph.

Graph Cuts

- consider a partition of the graph G into two subgraphs A and B
- the $Cut(A, B)$ will be given by the sum of the weights of the set of edges that connect the two groups

$$\text{cut(A)} := \sum_{i \in A, j \notin A} w_{ij} \tag{21}$$

- the notion of a **cut** is a fundamental concept in graph clustering
- we aim to find a partition/split of G into A and B in order to minimize the resulting cut.



[Source: David Sontag]

Spectral clustering of graphs

- A popular approach is to perform spectral clustering:
  - **Idea:** Embed V into $\mathbb{R}^k$ and perform k means clustering.
  - Embedding obtained from extremal eigenvectors of suitable graph matrix (eg., Laplacian).
- **Example:** Normalized cut (NC) [Shi and Malik, 2000]

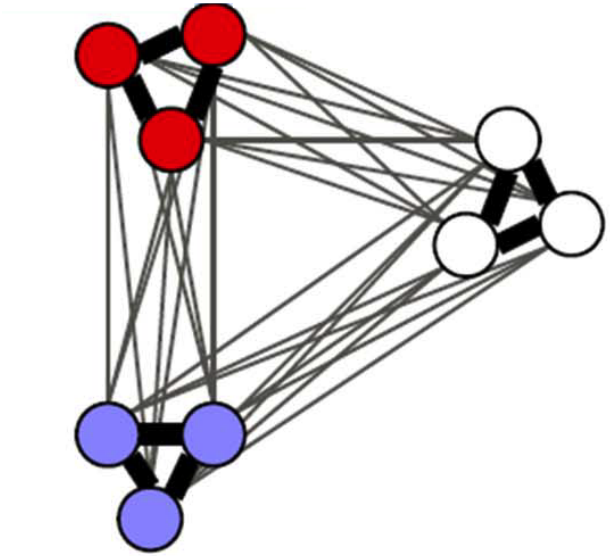
$$\min_{C_1,\ldots,C_k} \sum_{i=1,\ldots,k} \frac{\text{cut}(C_i)}{\text{vol}(C_i)}$$

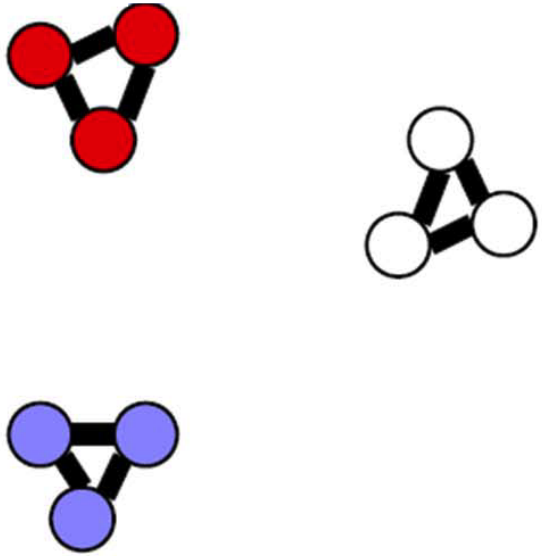
  - $\text{cut}(A) := \sum_{i\in A, j\notin A} w_{ij}$ and $\text{vol}(A) := \sum_{v\in A} \deg(v)$
- NC is a discrete optimization problem, NP-hard in worst case.
- We can "**relax**" the discrete constraints in NC: the solution of new problem is given by the smallest k eigenvectors of

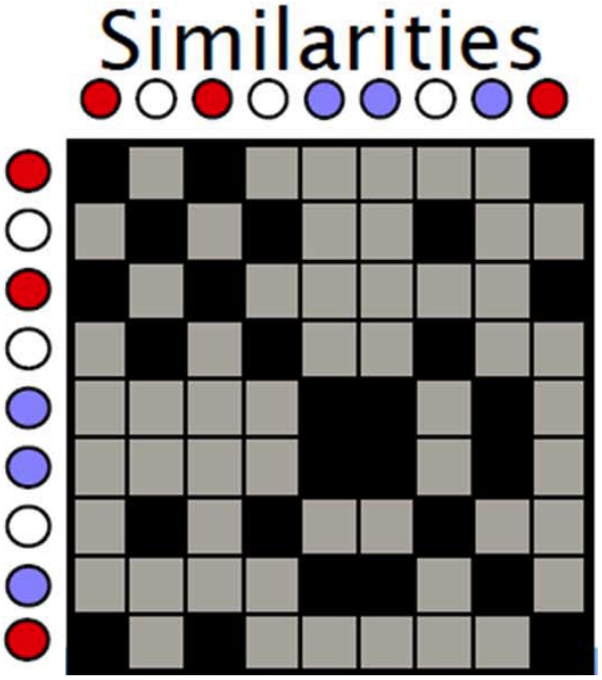
$$D^{-1/2} L D^{-1/2}$$

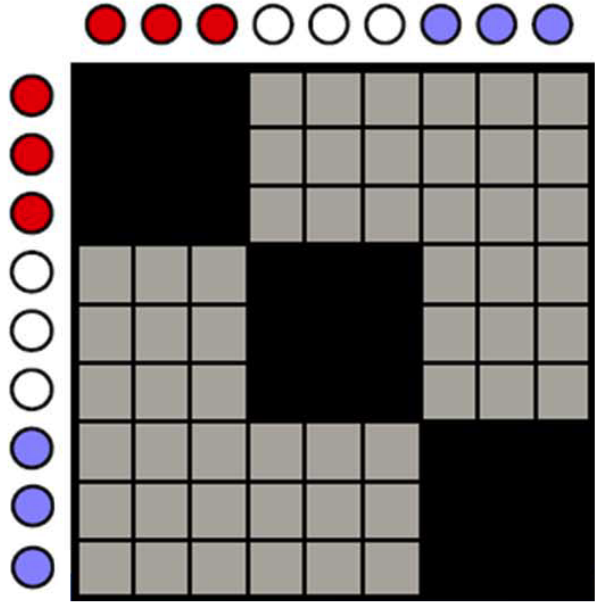
  - D: diagonal matrix with the degrees; L: the Laplacian of G
- $n \times k$ eigenvector matrix is the graph embedding in R^k

Graph partitioning









Source: David Sontag

Isoperimetric number

For a subgraph S, we define $\partial(S)$ as the boundary of S

$$\partial(S) = \{(u, v) | u \in S, v \in \overline{S}\} \tag{22}$$

The **isoperimetric ratio** of S (where $|S|$ is the number of vertices in S) is defined as

$$\Theta(S) \equiv \frac{|\partial(S)|}{|S|} \tag{23}$$

The **isoperimetric number** of the whole graph G is defined as

$$\Theta(G) \equiv \min_{S \subset V; |S| \leq n/2} \Theta(S) \tag{24}$$

• If $\Theta(S)$ is large, it becomes very hard to separate S from the graph, and many edges would have to be cut.

Conductance and Cheeger's Inequality

- another common method is to consider the notion of **conductance** for the graph partitioning problem
- For any $S \subset V$, the conductive of S

$$\Phi(S) = \frac{|\partial S|}{\min\{d(S), d(V - S)\}}$$

- $d(S)$ denotes the sum of degrees (with respect to the graph G) of all vertices in S
- $V - S$ denotes the complement set of S
- $d(V)$ denotes the sum of all vertex degrees in G

The conductance of a graph G, denote $\Phi(G)$ is defined as

$$\Phi(G) = \min_{S \subset V} \Phi(S) \tag{25}$$

where the minimization is taken over all possible subsets of nodes S of $V(G)$.

Cheeger's Inequality

- denote by N the normalized Laplacian

$$N = D^{-1/2} L D^{-1/2}$$

- where $L = D - A$ is the usual Combinatorial Laplacian
- denote the eigenvalues of N by

$$0 = \nu_1 \leq \nu_2 \ldots \leq \nu_n$$

- $\mathbf{d}$: vector of node degrees,
- $\mathbf{d}^{1/2}$: vector holding the square roots of the node degrees

$$\mathbf{d}^{1/2} = [\sqrt{d_1}, \sqrt{d_2}, ..., \sqrt{d_n}]$$

Lemma (Cheeger's inequality)

It holds true that

$$\frac{\nu_2}{2} \leq \Phi(G) \leq \sqrt{2\nu_2}$$

- proof sketch for the lower bound
- the upper bound is much harder to prove

Lemma

For every $S \subset V$, it holds true that $\Phi(S) \geq \frac{\nu_2}{2}$

Proof sketch - main steps:

1. $\mathbf{d}^{1/2}$ is an eigenvector of N of eigenvalue $\nu_1 = 0$
2. starting from
$$\nu_2 = \min_{x \perp \mathbf{d}^{1/2}} \frac{x^T N x}{x^T x}$$
and using the substitution $x = D^{1/2} y$ one can show that
$$\nu_2 = \min_{y \perp \mathbf{d}} \frac{y^T L y}{y^T D y}$$
3. $\sigma = \frac{d(S)}{d(V)}$. Show $y \stackrel{\text{def}}{=} \chi_S - \sigma \mathbf{1}$ is orthogonal to $\mathbf{d}$ ($y^T \mathbf{d} = 0$)
4. show that $y^T L y = |\partial(S)|$
5. if $d(V - S) = d(V) - d(S)$, one can show that
$$y^T D y = \frac{d(S) d(V - S)}{d(V)})$$
6. proof of the Lemma concludes with showing that
$$\nu_2 \leq 2 \frac{|\partial(S)|}{\min\{d(S), d(V - S)\}}$$

Spectral bi-clustering (with known cluster sizes)

Goal: partition the graph G (with adjacency matrix A) into $k = 2$ clusters of sizes n_1, respectively, n_2.

- Let the vector **s** (of size n) be such that

$$s_i = \begin{cases} +1 & \text{if vertex i belongs to group X} \\ -1 & \text{if vertex i belongs to group Y} \end{cases} \tag{26}$$

- Note that

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} +1 & \text{if i and j are in different groups} \\ 0 & \text{if i and j are in the same group.} \end{cases} \tag{27}$$

- Let R denote the size of the cut set of the division

$$\mathrm{R} = \tfrac{1}{4} \sum_{ij} A_{ij}(1 - s_i s_j)$$

which can be shown to equal

$$R = \tfrac{1}{4} \sum_{ij} L_{ij} s_i s_j$$

- In matrix form

$$\mathrm{R} = \tfrac{1}{4} \mathbf{s}^T L \mathbf{s}$$

$L = D - A$ is the combinatorial Laplacian; $\lambda_1 = 0 \leq \lambda_2 \ldots \leq \lambda_n$.

Spectral bi-clustering

We aim to minimize (over the unknown vector **s**) the cut size

$$\mathrm{R} = \tfrac{1}{4}\mathbf{s}^T L\mathbf{s}$$

- the minimization is difficult, since s_i can only equal ± 1
- if s_i could take any **real** value, we could just differentiate to find the optimum
- seek **approximate** solution to the minimization problem
- relax/allow the s_i's to take any values (subject to constraints discussed below)
- and then find the values that minimize R.

The allowed values of the s_i entries are subject to constraints

- initially, $s_i = \pm 1$; **s** lives in the n-dimensional hyper-cube; length of **s** is $\sqrt{n}$
- **relax** this constraint so that the vector **s** can point in any direction, but the length remains $\sqrt{n}$
- **s** now lives in n-dimensional sphere
- enforce $\sum_i s_i = n_1 - n_2$; in vector notation $\mathbf{1}^T\mathbf{s} = n_1 - n_2$

We reduced our initial problem to one of calculus and algebra.

Spectral bi-clustering

- differentiate with respect to s_i, enforcing the constraints using two Lagrange multipliers, denoted by λ and 2μ

$$\frac{\partial}{\partial s_i}[\sum_{jk} L_{jk} s_j s_k + \lambda(n - \sum_j s_j^2) + 2\mu((n_1 - n_2) - \sum_j s_j)] = 0$$

- taking derivatives, we arrive at

$$\sum_j L_{ij} s_j = \lambda s_i + \mu$$

or, in matrix notation

$$L\mathbf{s} = \lambda\mathbf{s} + \mu\mathbf{1}$$

- (since $L\mathbf{1} = 0$) left multiply by $\mathbf{1}^T$ and solve for μ

$$\mu = -\frac{n_1 - n_2}{n}\lambda$$

- define the new vector

$$\mathbf{x} = \mathbf{s} + \tfrac{\mu}{\lambda}\mathbf{1} = \mathbf{s} - \tfrac{n_1 - n_2}{n}\mathbf{1}$$

Spectral bi-clustering

- Consider
$$L\mathbf{x} = L(\mathbf{s} + \tfrac{\mu}{\lambda}\mathbf{1}) = L\mathbf{s} = \lambda\mathbf{s} + \mu\mathbf{1} = \lambda\mathbf{x}$$
- $\mathbf{x}$ is an eigenvector of the Laplacian L with eigenvalue λ
- choose an eigenvector that gives the smallest cut size R
- note that
$$\mathbf{1}^T\mathbf{x} = \mathbf{1}^T\mathbf{s} - \tfrac{\mu}{\lambda}\mathbf{1}^T\mathbf{1} = (n_1 - n_2) - \tfrac{n_1 - n_2}{n}n = 0$$
- thus $\mathbf{x}$ is orthogonal to $\mathbf{1}$; it is an eigenvector of L - cannot be the eigenvector $\mathbf{1} = [1, 1, \ldots, 1]$ of eigenvalue $\lambda_1 = 0$
- which eigenvector should we choose? Note that
$$R = \tfrac{1}{4}\mathbf{s^T L s} = \tfrac{1}{4}\mathbf{x^T L x} = \tfrac{1}{4}\lambda\mathbf{x^T x}$$
$$\mathbf{x^T x} = \mathbf{s^T s} + \tfrac{\mu}{\lambda}(\mathbf{s^T 1} + \mathbf{1^T s}) + \tfrac{\mu^2}{\lambda^2}\mathbf{1^T 1} = 4\tfrac{n_1 n_2}{n}$$
which yields $R = \frac{n_1 n_2}{n}\lambda$.
- thus, the cut size R is proportional to the eigenvalue λ
- since our goal is to minimize R, we choose the eigenvector v_2 corresponding to the second smallest eigenvalue λ_2
- recover $\mathbf{s}$ by $\mathbf{s} = \mathbf{x} + \frac{n_1 - n_2}{n}\mathbf{1}$

Algorithm 2 Spectral graph bi-clustering.

Require: Output partition $V = X \cup Y$ that minimizes the cut

$$R = \frac{1}{2} \sum_{i \in X, j \in Y} A_{ij}$$

1: Calculate the eigenvector $\mathbf{v}_2$ corresponding to the second smallest eigenvalue λ_2 of the comb. Laplacian $L = D - A$

2: Sort the elements of the eigenvector in order from largest to smallest.

3: (a) Place the vertices corresponding to the n_1 largest elements in group X, the rest in group Y, and calculate the resulting cut size

3: (b) Place the vertices corresponding to the n_1 smallest elements in group X, the rest in group Y, and recalculate the resulting cut size.

4: Consider the partitions (a) and (b) of the network, and choose the one that gives the smaller cut size.

Q: What about the case when n_1 and n_2 are not specified?

Eigenvector sweep profiles

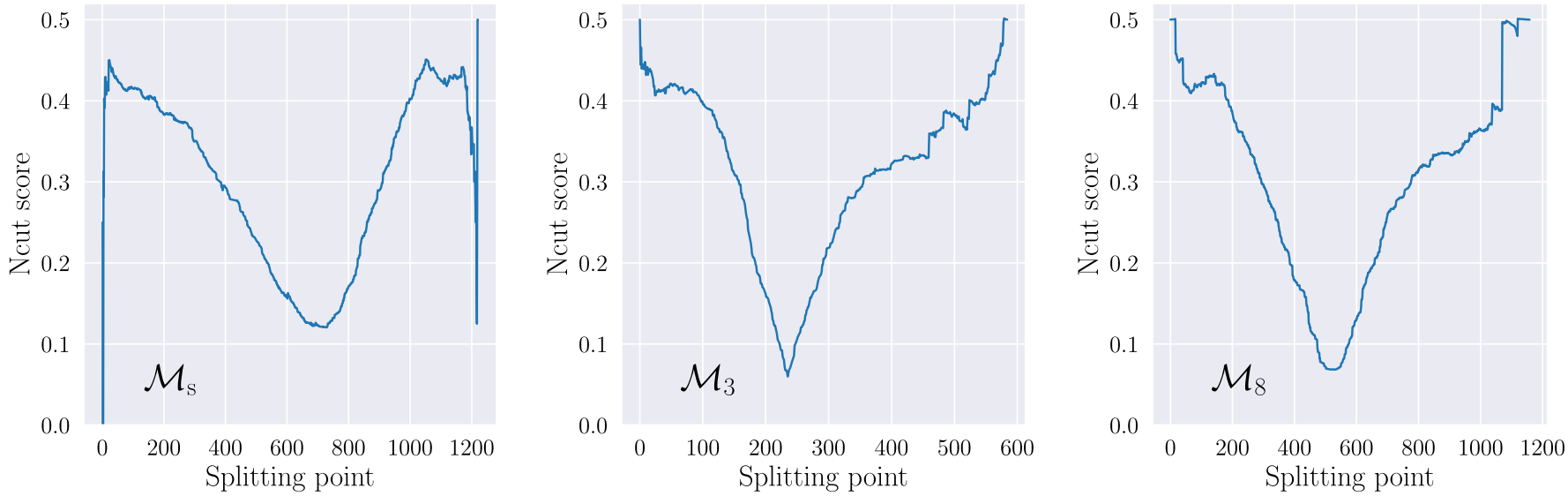


Fig. 14 Sweep profiles for different motifs on the US-POLITICAL-BLOGS network. M_s corresponds to using the symmetrized adjacency matrix

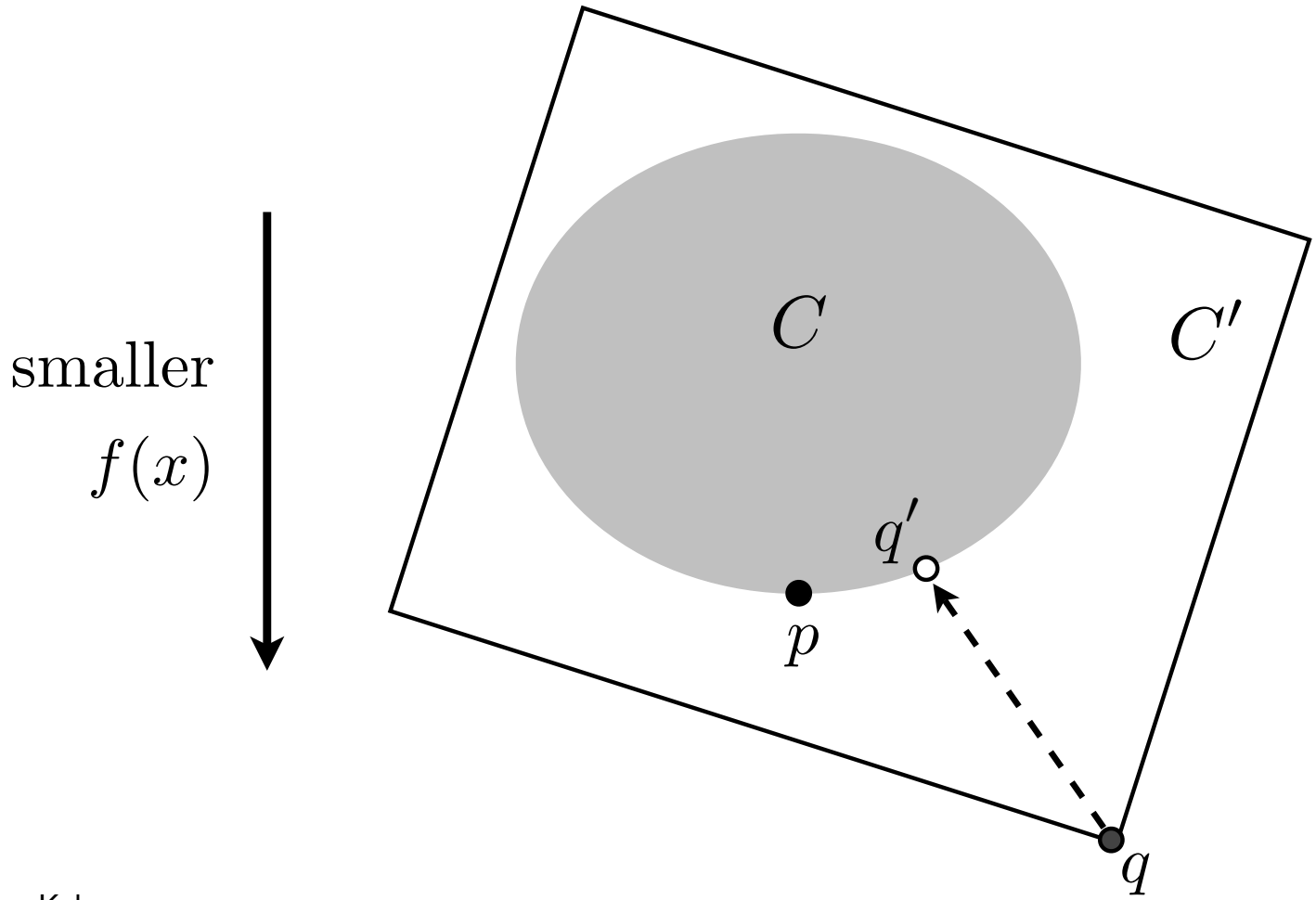
- x-axis: the index of the sorted entries in the Fiedler eigenvector
- y-axis: the normalized cut score (NCut)
- find a splitting point $n_1 + n_2 = n$ such that the NCut is minimized

[Underwood, W.G., Elliott, A. Cucuringu, M. Motif-based spectral clustering of weighted directed networks. Appl Netw Sci 5, 62 (2020). https://doi.org/10.1007/s41109-020-00293-z]

Facing an NP-hard minimization problem?

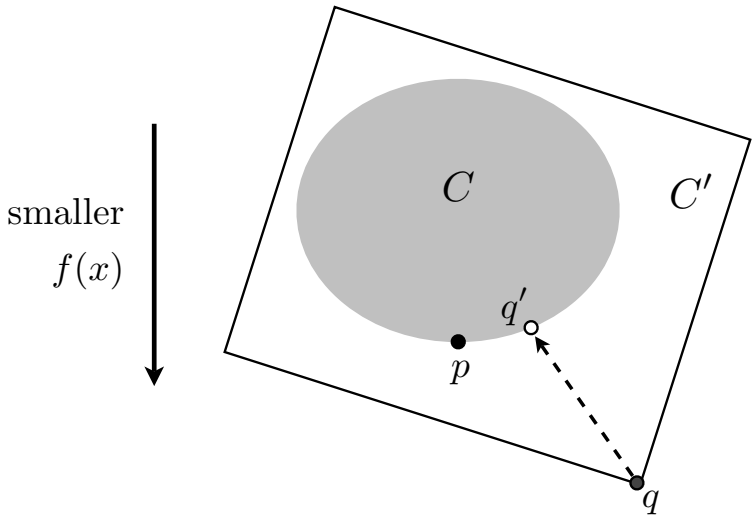
Just relax..

- dropping the constraint $x \in \{-1, 1\}^n$ or $x \in \{0, 1\}^n$ as we have seen so far, is a recurring technique in algorithms



Credit: Jonathan Kelner

General approach to relaxations algorithms



- dropping the constraint $x \in \{-1, 1\}^n$ or $x \in \{0, 1\}^n$ is a recurring technique in algorithms
- can frame this as a more general relaxation technique

- goal is to solve an NP-hard problem which takes the form of

  minimize $f(x)$ subject to the constraint $x \in C$

  where C is a domain specific constrained set
- instead, we relax and choose to minimize $f(x)$ subject to a weaker constraint $x \in C' \supseteq C$
- let p and q be the points that minimize f in C and C', respectively
- since $C \subseteq C'$, we know that $f(q) \leq f(p)$
- to benefit from the relaxation, need to show how to "round" q to a feasible point $q' \in C$, + prove $f(q') \leq \gamma f(q)$ for some const. $\gamma \geq 1$
- implies $f(q') \leq \gamma f(q) \leq \gamma f(p)$; process yields a γ-approximation