So Far…

- We have discussed
  - Supervised learning
    - Goal: learn a mapping from inputs $\boldsymbol{x}$ to outputs y
    - Training data: a labeled set of input-output pairs
  - Various methods to learn this mapping functions

- It's time for
  - Unsupervised learning
    - We are only given inputs
    - Goal: find "interesting patterns"
    - Discovering clusters: Clustering

Clustering

Deng Cai (蔡登)

College of Computer Science
Zhejiang University

dengcai@gmail.com

What is Clustering (Cluster Analysis)?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters

- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

Measure the Quality of Clustering

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

Distance Measures for Different Kinds of Data

- Numerical (interval)-based:
  - Minkowski Distance:
  - Special cases: Euclidean (L2-norm), Manhattan (L1-norm)
- Vectors: cosine measure
- Binary variables:
  - symmetric vs. asymmetric (Jaccard coeff.)
- Nominal variables: # of mismatches
- Ordinal variables: treated like interval-based
- Ratio-scaled variables: apply log-transformation first
- Mixed variables: weighted combinations

- More important:
  - Distance Metric

Aspects in Clustering Methods

- Partitioning requirement: one level versus hierarchical partitioning
- Separation of clusters: exclusive versus non-exclusive
- Similarity measure: distance versus connectivity based on density or contiguity
- Clustering space: full space versus subspaces

Algorithms

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: **k-means**, **k-medoids**
- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: **GMM**
- Dimensionality reduction approach
  - First dimensionality reduction, then clustering
  - Typical methods: **Spectral clustering**, Ncut

Partitioning Algorithms: Basic Concept

- Partitioning method: partitioning a database D of n objects into a set of k clusters, s.t., min sum of squared distance

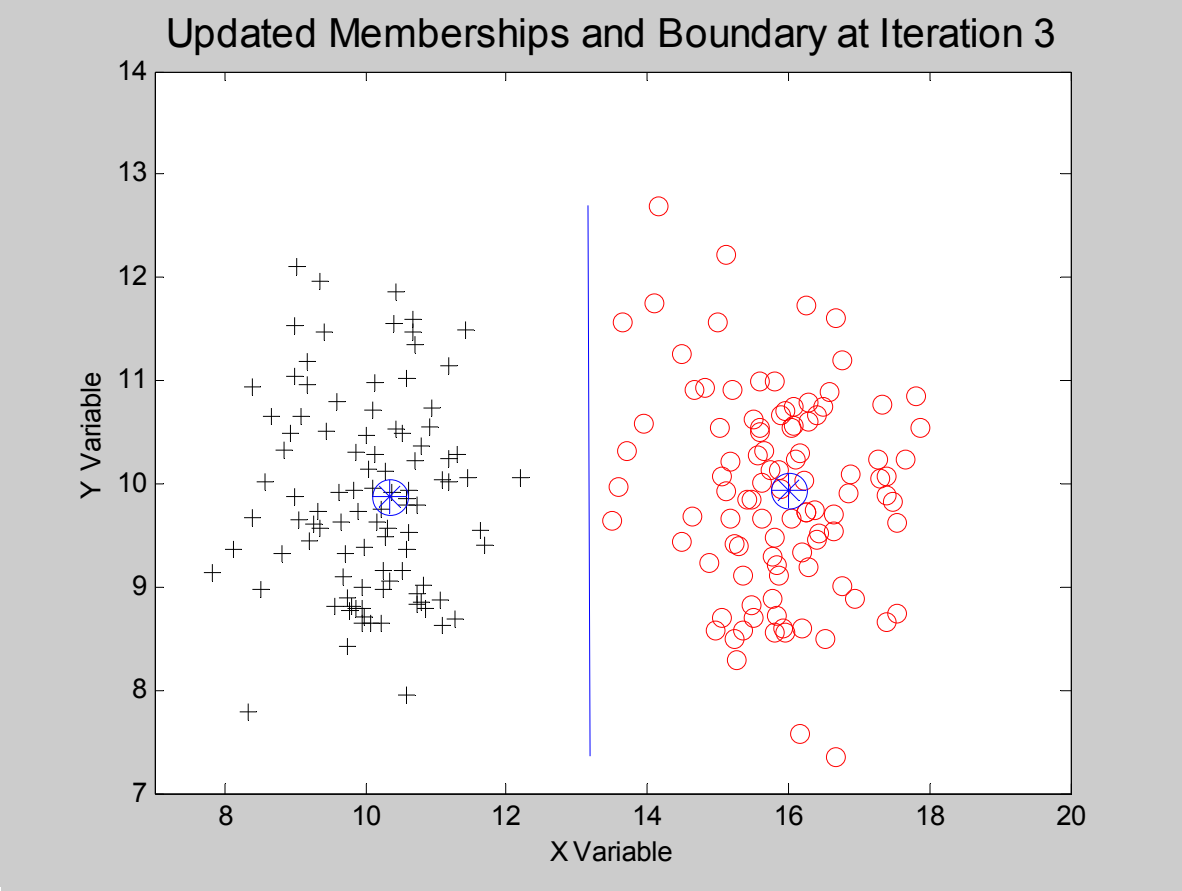
$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in c_i} \|x_j - \mu_i\|^2$$

- Given k, find a partition of k clusters that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: k-means and k-medoids algorithms
  - k-means (MacQueen'67): Each cluster is represented by the center of the cluster
  - k-medoids or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The K-Means Clustering Method

- Given k, the k-means algorithm is implemented in four steps:
  1. Partition objects into k nonempty subsets
  2. Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., mean point, of the cluster)
  3. Assign each object to the cluster with the nearest seed point
  4. Go back to Step 2, stop when the assignment does not change

K-Means Example



K-Means: Time Complexity

- $O(kndt)$ where t is the iteration upper bond
  - Computing distance between two points: $O(d)$
  - Assignment step: $O(kn)$ distance computations, totaled $O(knd)$
  - Update step: each vector gets added once to corresponding centroid, $O(nd)$
  - Multiply the iteration upper bond t: $O(kndt)$

K-Means: Local Optimum

- TSD (Totoal Squared Distance) decreases at each iteration
  - Global minimum of TSE?
  - No, not necessarily.
  - in a sense it is doing “steepest descent” from a random initial starting point, thus, results will be sensitive to the starting point
  - in practice, we can run it from multiple starting points and pick the solution with the lowest TSD

K-Means: Comments

- Implicit assumptions about the "shapes" of clusters
  - Spherical in vector space
  - Sensitive to coordinate changes, weighting
  - Solution: spectral clustering
- Have to manually pick the number of clusters
  - Try and error? Unfortunately not feasible
  - Solution: Hierarchical clustering
- All items forced into a cluster – Hard clustering
  - Small shift of a data point can flip it to a different cluster
  - Solution: soft probabilistic assignments (GMM)
- Doesn't have a notion of "outliers"
  - Other objective functions
  - Solution: K-Medoids

K-Medoids Clustering Method

- Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located object** in a cluster.
  - Medoid: a chosen, centrally located object in the cluster
  - Centroid: the "middle" of a cluster

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} \boldsymbol{x}_i^{(k)}$$

  - not necessarily inside a cluster

How to find the medoid

- Method 1:
  - Compute the pairwise distances matrix D
  - Compute the row (or, column) sum of D, d
  - Finds the smallest entry in d

- Method 2:
  - Compute the centroid μ
  - Finds the point which is closest to μ

How to find the medoid

- For cluster c_i, min sum of squared distance

$$\arg\min_{x_k \in c_i} \sum_{x_j \in c_i} ||x_j - x_k||^2$$

- Let $\mu_i = \frac{1}{n_i} \sum_{x_j \in c_i} x_j$,

$$\sum_{x_j \in c_i} ||x_j - x_k||^2 = \sum_{x_j \in c_i} ||x_j - \mu_i + \mu_i - x_k||^2$$

$$= \sum_{x_j \in c_i} ||x_j - \mu_i||^2 + n_i ||\mu_i - x_k||^2$$

- Choose x_k be the point nearest to μ_i in cluster c_i

K-Medoids Algorithm

- Given k, the k-medoids algorithm is implemented in five steps:
  1. Partition objects into k nonempty subsets
  2. Compute the centroids of the clusters of the current partitioning
  3. Choose the nearest points of the centroids of the clusters as seed points
  4. Assign each object to the cluster with the nearest seed point
  5. Go back to Step 2, stop when the assignment does not change

K-Medoids: Time Complexity

- $O(kndt)$ the same as k-means
  - Computing distance between two points: $O(d)$
  - Assignment step: $O(kn)$ distance computations, totaled $O(knd)$
  - Computing centroid step: each vector gets added once to corresponding centroid, $O(nd)$
  - Update step: each vector gets added once to corresponding seed points, $O(nd)$
  - Multiply the iteration upper bond t: $O(kndt)$

K-Medoids Clustering Method

- One advantage over K-means
  - Sometimes only the distance matrix are provided and the value $||\mu_i - x_k||^2$ couldn't be calculated directly
  - In the case k-means fails, but k-medoids still works after slightly changing its algorithm

K-Medoids Algorithm

- Partitioning Around Medoids (PAM) algorithm

0. Calculate the pair-wise distance matrix W
1. Initialize: randomly select k of the n data points as the medoids
2. Associate each data point to the closest medoid
3. For each cluster, compute its medoid
4. Repeat 2-3 until there is no change in the medoids

K-Medoids: Time Complexity

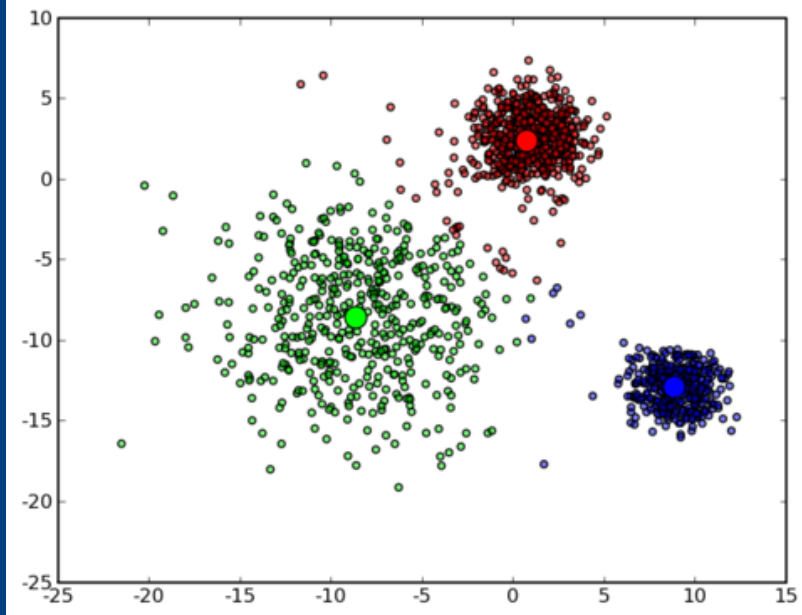
- $O(n^2dt)$
  - Calculate the pair-wise distance:$O\left(n^2d\right)$
  - Assignment step: $O(knd)$ to pick the closest medoid
  - Update medoid step: $O(n)$
  - Multiply the iteration upper bond t: $O(n^2dt)$

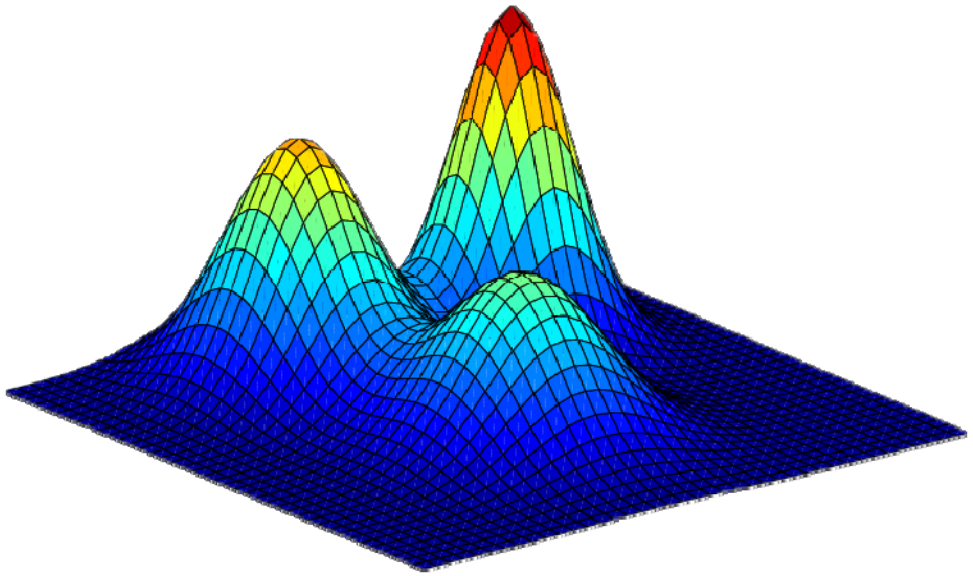
Algorithms

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: **k-means**, **k-medoids**
- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: **GMM**
- Dimensionality reduction approach
  - First dimensionality reduction, then clustering
  - Typical methods: **Spectral clustering**, Ncut

Gaussian Mixture Model

- Gaussian Mixture Model (GMM) is one of the most popular clustering methods which can be viewed as a linear combination of different Gaussian components.





Single Gaussian Model

- Multivariate Gaussian
  - $\boldsymbol{\mu}$: mean of the distribution
  - Σ: covariance of the distribution

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$$

  - Maximum likelihood estimation

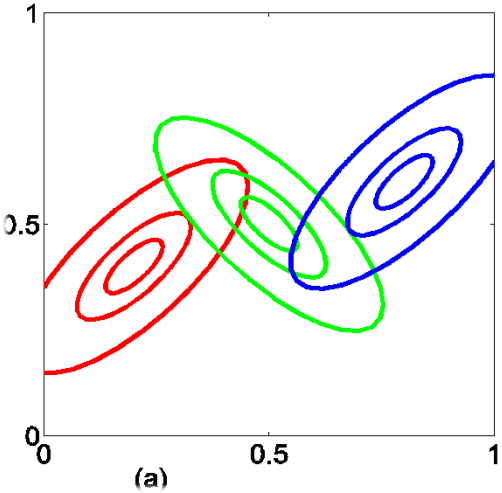
$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x_i}$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x_i}-\hat{\boldsymbol{\mu}})(\boldsymbol{x_i}-\hat{\boldsymbol{\mu}})^{\boldsymbol{T}}$$

Gaussian Mixture Model

- Linear combination of Gaussians
  - Assumption: K Gaussians, each has a contribution of π_k to the data points

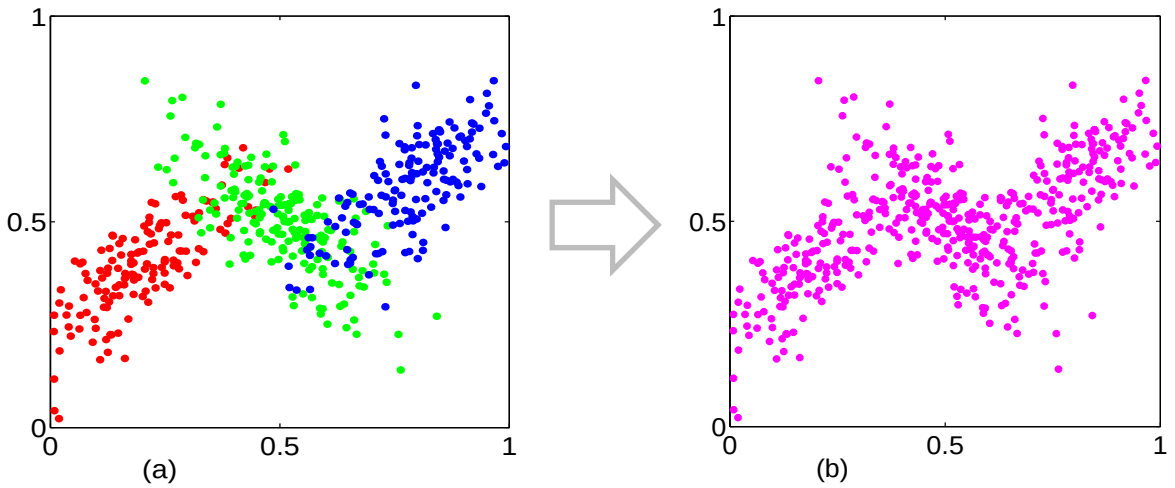
$$\begin{cases} p(\boldsymbol{x};\boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x};\boldsymbol{\theta}_k) \\ \boldsymbol{\Theta} = \{\pi_1, \cdots, \pi_K, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K\}, \sum_{k=1}^{k} \pi_k = 1, \pi_k \in [0,1] \\ p_k(\boldsymbol{x};\boldsymbol{\theta}_k) = \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \end{cases}$$



  - Parameters to be estimated: $\pi_k, \boldsymbol{\mu}_k, \Sigma_k$

Gaussian Mixture Model

- The process of generating a data point
  - first pick one of the components with probability π_k
  - then draw a sample $\boldsymbol{x}_i$ from that component distribution
- Each data point is generated by one of k components



Parameters Estimation for GMM

- The log-likelihood function:

$$\log \prod_{i=1}^{N} p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\Theta}\right) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \right)$$

- How to find the parameters?

Re-examine GMM

$$p(\boldsymbol{x};\boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x};\boldsymbol{\theta}_k)$$

$$p_k(\boldsymbol{x};\boldsymbol{\theta}_k) = \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$$

$$\sum_{k=1}^{k} \pi_k = 1, \pi_k \in [0,1]$$

- For each point $\boldsymbol{x}^{(i)}$, associate with a hidden variable $z^{(i)}$ denotes which Gaussian $\boldsymbol{x}^{(i)}$ belongs to
- $z^{(i)}$ follows the multinomial distribution $P(z^{(i)} = k) = \pi_k$

$$P(\boldsymbol{x}^{(i)}|z^{(i)} = k) = \mathcal{N}(\boldsymbol{x}^{(i)};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$$

$$P(z^{(i)} = k|\boldsymbol{x}^{(i)}) \triangleq Q_k^{(i)}$$

Parameters Estimation for GMM

$$P\left(z^{(i)}=k|\boldsymbol{x}^{(i)}\right)\triangleq Q_k^{(i)} \qquad \mathcal{N}(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$$

M-Step

$$\boldsymbol{\mu}_k=\frac{\sum_{i=1}^{n}Q_k^{(i)}\,\boldsymbol{x}^{(i)}}{\sum_{i=1}^{n}Q_k^{(i)}} \qquad \hat{\boldsymbol{\mu}}=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x_i}$$

$$\Sigma_k=\frac{\sum_{i=1}^{n}Q_k^{(i)}\left(\boldsymbol{x}^{(i)}-\boldsymbol{\mu}_k\right)\left(\boldsymbol{x}^{(i)}-\boldsymbol{\mu}_k\right)^{\boldsymbol{T}}}{\sum_{i=1}^{n}Q_k^{(i)}} \qquad \hat{\Sigma}=\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x_i}-\hat{\boldsymbol{\mu}})(\boldsymbol{x_i}-\hat{\boldsymbol{\mu}})^{\boldsymbol{T}}$$

$$\pi_k=\frac{\sum_{i=1}^{n}Q_k^{(i)}}{n}$$

$$Q_k^{(i)}\triangleq P\left(z^{(i)}=k|\boldsymbol{x}^{(i)}\right)=\frac{P\left(\boldsymbol{x}^{(i)}|z^{(i)}=k\right)P\left(z^{(i)}=k\right)}{P\left(\boldsymbol{x}^{(i)}\right)}$$

$\mathcal{N}(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$ $\qquad$ π_k

$$=\frac{P\left(\boldsymbol{x}^{(i)}|z^{(i)}=k\right)P\left(z^{(i)}=k\right)}{\sum_{k=1}^{K}P\left(\boldsymbol{x}^{(i)}|z^{(i)}=k\right)P\left(z^{(i)}=k\right)}$$

E-Step

Expectation Maximization

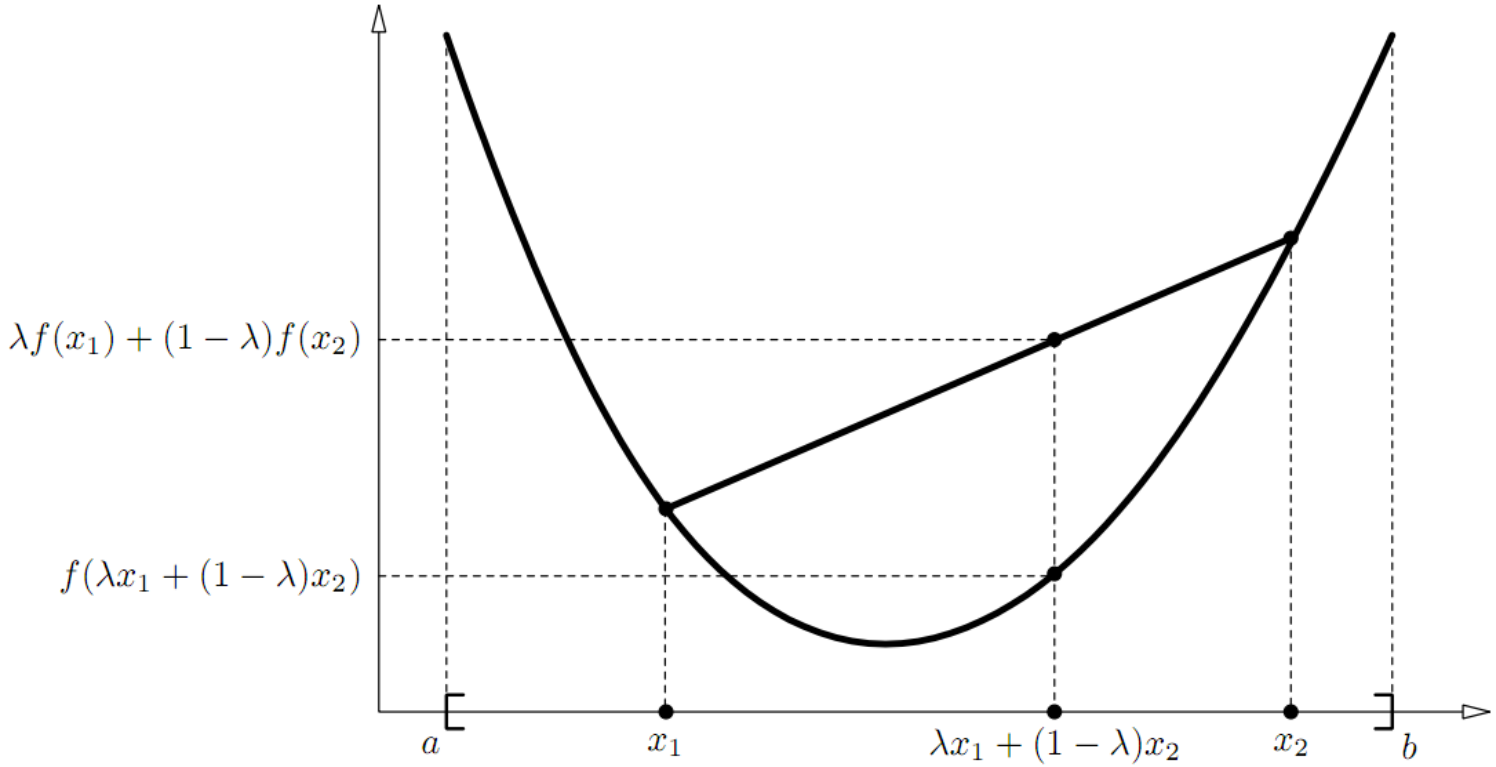
Convex sets

- A set C is convex if the line segment between any two points in C lies in C
- For any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and θ with $0 \leq \theta \leq 1$, we have

$$\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2 \in C$$

Convex functions

- $f: \mathbb{R}^n \to \mathbb{R}$ is convex if $\mathrm{dom} f$ is a convex set and
$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y})$$
for all $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom} f, 0 \leq \theta \leq 1$



- f is concave if $-f$ is convex
- f is strictly convex if $\mathrm{dom} f$ is a convex set and
$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) < \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y})$$
for all $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom} f, \boldsymbol{x} \neq \boldsymbol{y}, 0 < \theta < 1$

- Example:
  - $\log x$, $\ln x$ on $\mathbb{R}^+$ is strictly concave
  - $ax + b$ on $\mathbb{R}$ is both concave and convex
  - e^{ax}, for any $a \in \mathbb{R}$ is convex

Jensen's inequality

- if f is convex, then

$$f\left(\sum_{i=1}^{n}\lambda_i x_i\right) \le \sum_{i=1}^{n}\lambda_i f(x_i) \qquad \lambda_i \ge 0, \sum_i \lambda_i = 1$$

$$f(E[X]) \le E[f(X)]$$

for any random variable X

- if f is strictly convex, then $E[f(X)] = f(E[X])$ holds true if and only if X=$E[X]$ with probability 1 (i.e., if X is a constant)

- $\ln x$: strictly concave

$$\ln\left(\sum_{i=1}^{n}\lambda_i x_i\right) \ge \sum_{i=1}^{n}\lambda_i \ln(x_i)$$

Description

Suppose we have an estimation problem in which we have a training set $\{\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(m)}\}$ consisting of m independent examples. We wish to fit the parameters to the data, where the log-likelihood is given by

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{m} \log p(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

The semicolon means "parameterized". For example, $p(\boldsymbol{x}; \boldsymbol{\theta})$ means the probability of sample $\boldsymbol{x}$ is from the distribution defined by the parameter $\boldsymbol{\theta}$.

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{m} \log p(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

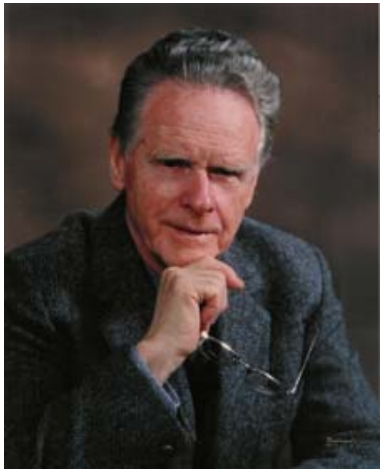
- finding the maximum likelihood estimates of the parameters may be hard

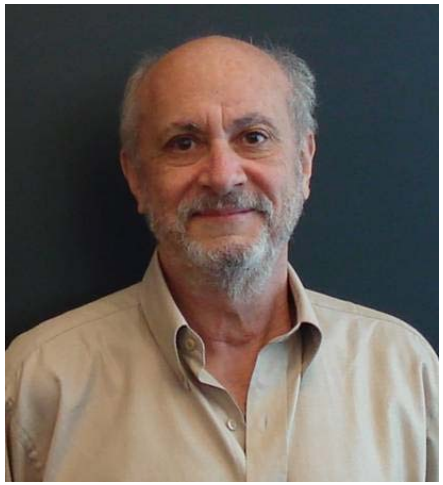
$$l(\boldsymbol{\theta}) = \sum_{i=1}^{m} \log p(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) = \sum_{i=1}^{m} \log \sum_{\boldsymbol{z}^{(i)}} p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta})$$

- Here, the $\boldsymbol{z}^{(i)}$'s are the latent random variables; and it is often the case that if the $\boldsymbol{z}^{(i)}$'s were observed, then maximum likelihood estimation would be easy.

EM algorithm

- EM algorithm gives an efficient iterative procedure for maximum likelihood estimation by introducing latent variables $\boldsymbol{z}^{(i)}$'s
- The EM algorithm was explained and given its name in a classic 1977 paper by Arthur Dempster, Nan Laird, and Donald Rubin.





Arthur Dempster, Nan Laird, and Donald Rubin

EM algorithm

- Each iteration of the EM algorithm consists of two processes: The Expectation-step, and the Maximization-step.
- In the expectation, or E-step, the missing data are estimated given the observed data and current estimate of the model parameters.
- In the M-step, the likelihood function is maximized under the assumption that the missing data are known. The estimate of the missing data from the E-step are used in lieu of the actual missing data.
- Convergence is assured since the algorithm is guaranteed to increase the likelihood at each iteration.

Details

Let Q be some distribution over the z's .

Note that $\sum_{\boldsymbol{z}^{(i)}} Q^i(\boldsymbol{z}^{(i)}) = 1$, $Q^i(\boldsymbol{z}^{(i)}) \geq 0$, we get:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{m} \log \sum_{\boldsymbol{z}^{(i)}} p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta})$$

$$= \sum_{i=1}^{m} \log \sum_{\boldsymbol{z}^{(i)}} Q^i(\boldsymbol{z}^{(\mathrm{i})}) \frac{p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta})}{Q^i(\mathbf{z}^{(\mathrm{i})})}$$

$$\geq \sum_{i=1}^{m} \sum_{\boldsymbol{z}^{(i)}} Q^i(\boldsymbol{z}^{(i)}) \log \frac{p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta})}{Q^i(\boldsymbol{z}^{(\mathrm{i})})} = l(\boldsymbol{\theta})$$

The last step of the derivation used Jensen's inequality.

Note that $f(\boldsymbol{x}) = \log(\boldsymbol{x})$ is a concave function and by Jensen's inequality, we have:

$$f\left(E_{\boldsymbol{z}^{(i)}\sim Q}\left[\frac{p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta}\right)}{Q^{i}\left(\boldsymbol{z}^{(i)}\right)}\right]\right) \geq E_{\boldsymbol{z}^{(i)}\sim Q}\left[f\left(\frac{p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta}\right)}{Q^{i}\left(\boldsymbol{z}^{(i)}\right)}\right)\right]$$

where the "$\boldsymbol{z}^{(i)}\sim Q$" subscripts above indicate that the expectations are with respect to $\boldsymbol{z}^{(i)}$ drawn from Q

- We now have the lower-bound $l(\boldsymbol{\theta})$ for the log-likelihood $L(\boldsymbol{\theta})$ that we're trying to maximize
- Determine the lower-bound $l(\boldsymbol{\theta})$ is the E-step
- In the M-step of the algorithm, we then maximize our formula $l(\boldsymbol{\theta})$ with respect to the parameters to obtain a new setting of the $\boldsymbol{\theta}$'s
- The question is: How to choose Q that will guarantee the convergence of the algorithm? Since EM is an iterative algorithm.

- There're many possible choices for the Q's
- It seems natural to try to make the lower-bound tight at that value of $\boldsymbol{\theta}$. I.e., we'll make the inequality above hold with equality at our particular value of $\boldsymbol{\theta}$.
- Actually, such a choice is perfect. And we will see the proof of the convergence proved later.

$$f\left(E_{\boldsymbol{z}^{(i)}\sim Q}\left[\frac{p\left(\boldsymbol{x}^{(i)},\boldsymbol{z}^{(i)};\boldsymbol{\theta}\right)}{Q^i\left(\boldsymbol{z}^{(i)}\right)}\right]\right) \geq E_{\boldsymbol{z}^{(i)}\sim Q}\left[f\left(\frac{p\left(\boldsymbol{x}^{(i)},\boldsymbol{z}^{(i)};\boldsymbol{\theta}\right)}{Q^i\left(\boldsymbol{z}^{(i)}\right)}\right)\right]$$

- Recall the condition for the Jensen's inequality to hold with equality
  - if f is strictly convex, then $E[f(X)] = f(E[X])$ holds true if and only if X=$E[X]$ with probability 1 (i.e., if X is a constant)
- That is, we require that

$$\frac{p\left(\boldsymbol{x}^{(i)},\boldsymbol{z}^{(i)};\boldsymbol{\theta}\right)}{Q^i\left(\boldsymbol{z}^{(i)}\right)} = \text{constant}$$

- Since $\sum_{\boldsymbol{z}^{(i)}} Q^i\left(\boldsymbol{z}^{(i)}\right) = 1$, it is easy to show when

$$Q^i\left(\boldsymbol{z}^{(i)}\right) = \frac{p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta}\right)}{\sum_{\boldsymbol{z}^{(i)}} p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta}\right)} = \frac{p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta}\right)}{p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)} = p\left(\boldsymbol{z}^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

Satisfying the requirement:

$\frac{p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta}\right)}{Q^i\left(\boldsymbol{z}^{(i)}\right)} =$ constant that does not depend on $\boldsymbol{z}^{(i)}$

Proof of convergence

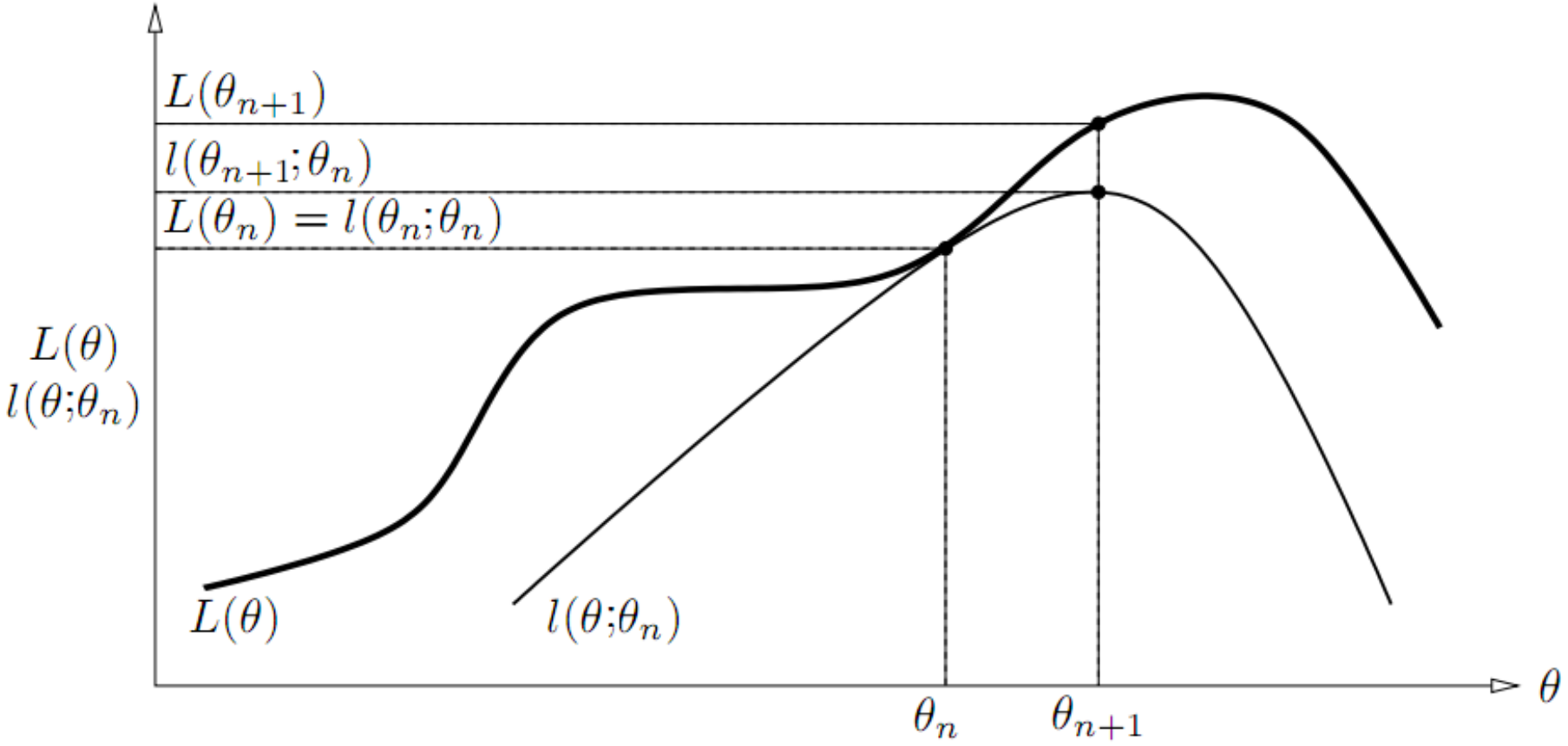
$$L(\boldsymbol{\theta}_{n+1}) \geq l(\boldsymbol{\theta}_{n+1}; \boldsymbol{\theta}_n) = \sum_{i=1}^{m} \sum_{\boldsymbol{z}^{(i)}} Q_n^i(\boldsymbol{z}^{(i)}) \log \frac{p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta}_{n+1})}{Q_n^i(\boldsymbol{z}^{(i)})}$$

$$\geq \sum_{i=1}^{m} \sum_{\boldsymbol{z}^{(i)}} Q_n^i(\boldsymbol{z}^{(i)}) \log \frac{p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}; \boldsymbol{\theta}_n)}{Q_n^i(\boldsymbol{z}^{(i)})} = l(\boldsymbol{\theta}_n; \boldsymbol{\theta}_n) = L(\boldsymbol{\theta}_n)$$

Where $l(\boldsymbol{\theta}_{n+1}; \boldsymbol{\theta}_n)$ means the lower-bound function is computed with Q_n parameterized by $\boldsymbol{\theta}_n$

$$Q_n^i(\boldsymbol{z}^{(i)}) = p(\boldsymbol{z}^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}_n)$$

So for each iteration the likelihood $L(\boldsymbol{\theta})$ is nondecreasing.



Graphical interpretation of a single iteration of the EM algorithm. The function is not concave. n represents the index of iterations.

Algorithm Summary

Repeat until convergence {

(E-step) for each i :

$$Q^i\left(\boldsymbol{z}^{(i)}\right) = p\left(\boldsymbol{z}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)$$

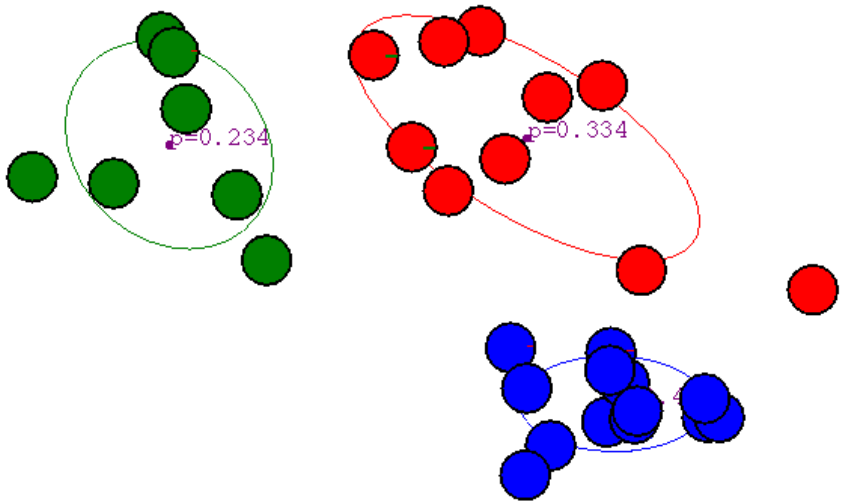
(M-step) Set:

$$\boldsymbol{\theta} := \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \sum_{\boldsymbol{z}^{(i)}} Q^i\left(\boldsymbol{z}^{(i)}\right) \log \frac{p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)};\boldsymbol{\theta}\right)}{Q^i\left(\boldsymbol{z}^{(i)}\right)}$$

}

If the function is not convex, it is possible for the algorithm to converge to local minima or saddle points in unusual cases

Gaussian Mixture Model: An example



K-Means vs. GMM

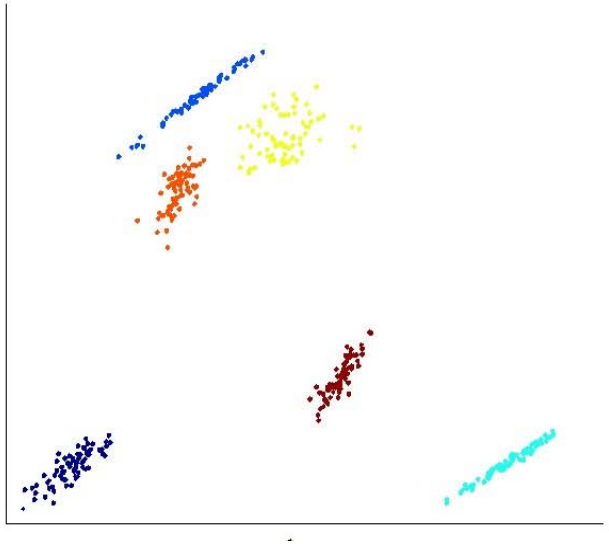
- Objective function:
  - Minimize the TSD
- Can be optimized by an EM algorithm.
  - E-step: assign points to clusters.
  - M-step: optimize clusters.
  - Performs hard assignment during E-step.
- Assumes spherical clusters with equal probability of a cluster.

- Objective function
  - Maximize the log-likelihood.
- EM algorithm
  - E-step: Compute posterior probability of membership.
  - M-step: Optimize parameters.
  - Perform soft assignment during E-step.
- Can be used for non-spherical clusters. Can generate clusters with different probabilities.

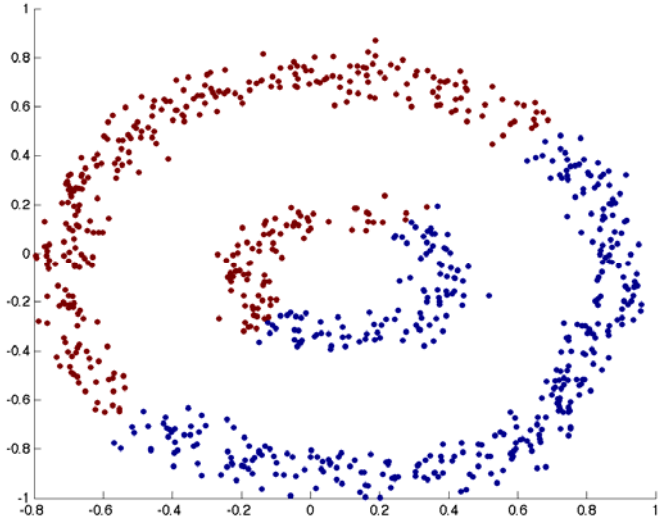
Algorithms

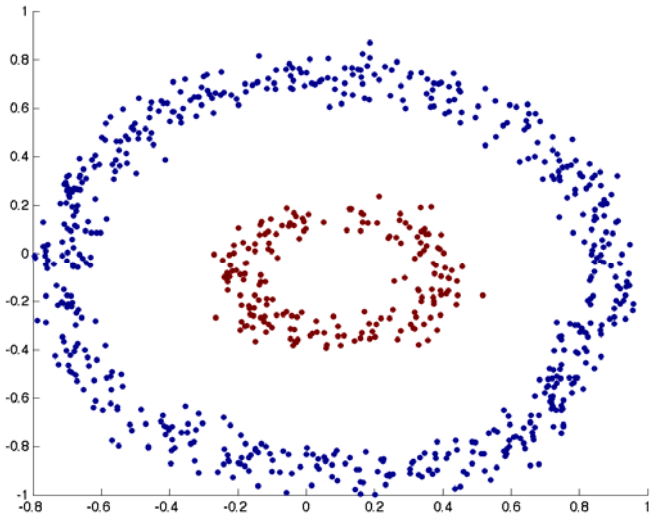
- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: **k-means**, **k-medoids**
- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: **GMM**
- Dimensionality reduction approach
  - First dimensionality reduction, then clustering
  - Typical methods: **Spectral clustering**, Ncut

Good clustering – we know it when we see it

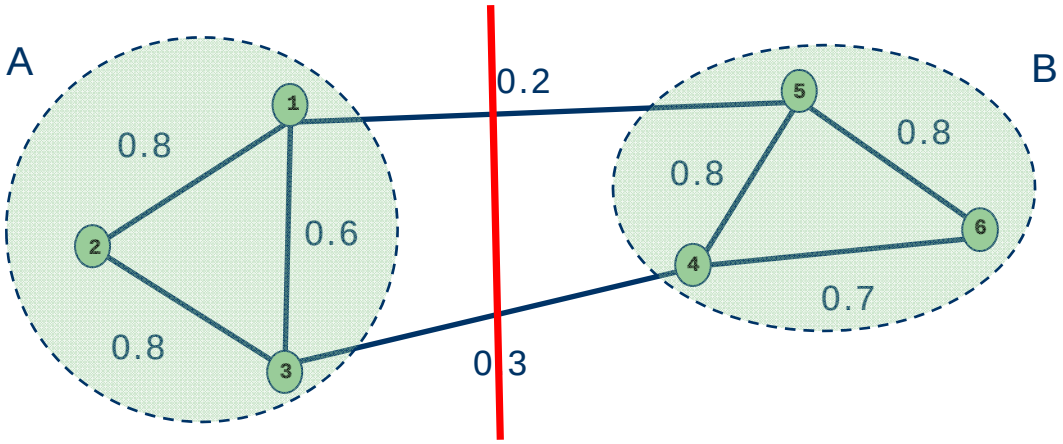


An Example



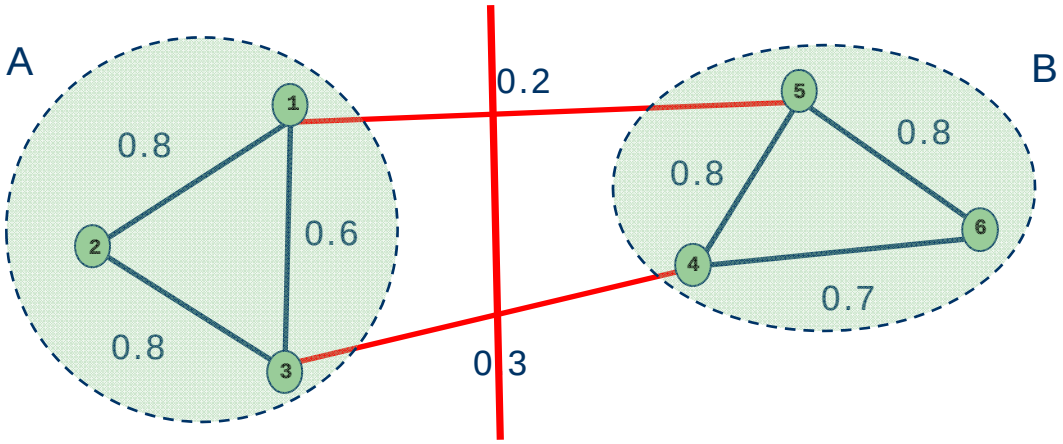


Spectral Clustering



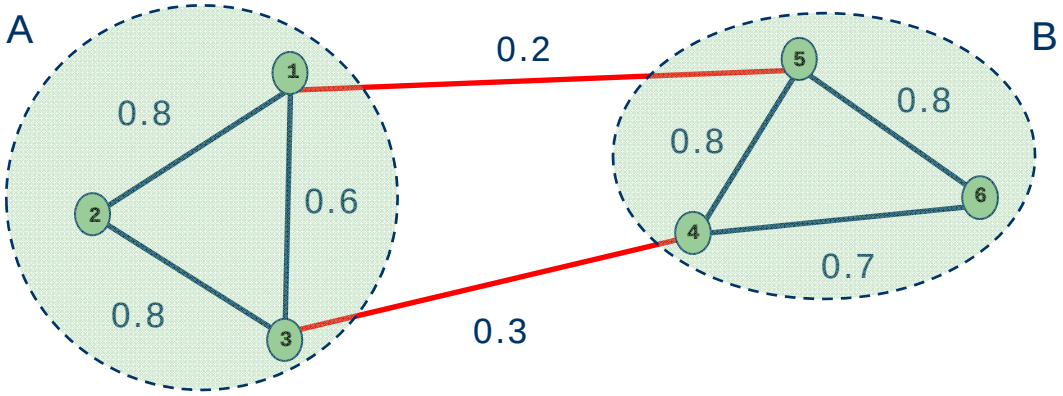
- Represent data points as the vertices V of a graph G.
  - All pairs of vertices are connected by an edge E.
  - Edges have weights W. Large weights mean that the adjacent vertices are very similar; small weights imply dissimilarity.
- Clustering can be viewed as partitioning a similarity graph
  - Divide vertices into two disjoint groups (A,B)

Clustering Objectives



- Traditional definition of a "good" clustering:
  - Points assigned to same cluster should be highly similar.
  - Points assigned to different clusters should be highly dissimilar.
- Apply these objectives to our graph representation
  - Minimize weight of between-group connections

Graph Cuts



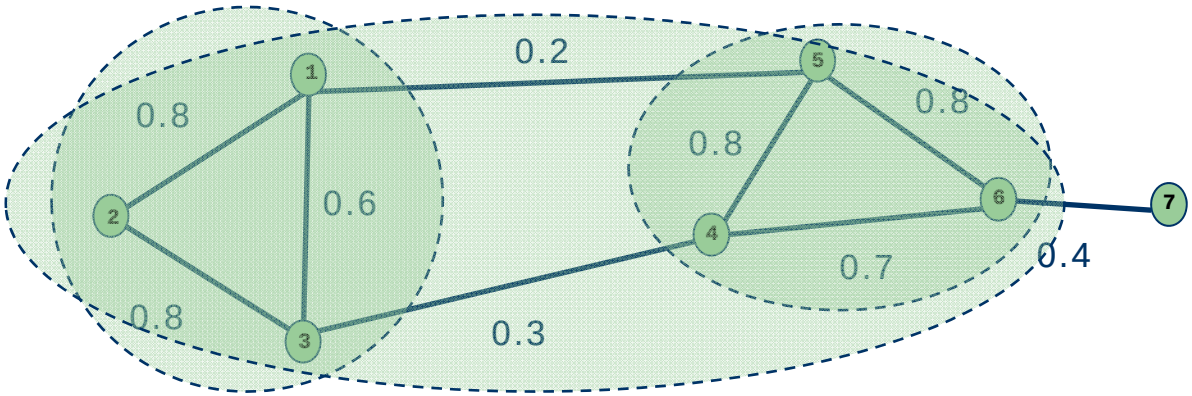
- Express partitioning objectives as a function of the "edge cut" of the partition.
  - ***Cut***: Set of edges with only one vertex in a group.we wants to find the minimal cut beetween groups. The groups that has the minimal cut would be the partition

$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

Graph Cut Criteria

- Criterion: Minimum-cut
  - Minimize the weights of connections between groups

$$\min cut(A, B)$$



- Problem:
  - Only considers the inter-cluster connections
  - Does not consider the intra-cluster density
- Maximize the weights of connections within groups

$$\max\big(assoc(A, A) + assoc(B, B)\big)$$

- $assoc(A, A) = \sum_{i \in A, j \in A} w_{ij}$

Graph Cut Criteria

- Criterion: Normalized-cut (Shi & Malik,'97)
  - Consider the connectivity between groups relative to the density of each group.

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

    - Normalize the association between groups.

$$assoc(A,V) = \sum_{i \in A, j \in V} w_{ij}$$

- Produces more balanced partitions

$$\min Ncut(A,B)$$

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$

Graph Cut Criteria

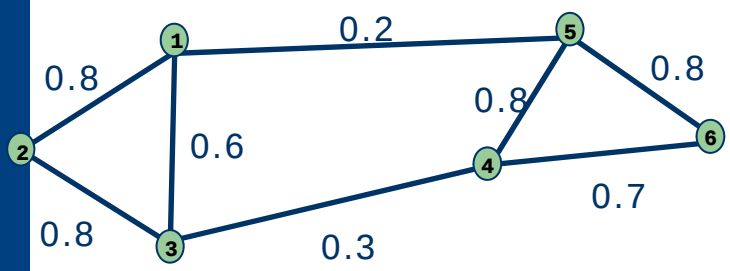
$$cut(A,B) = assoc(A,V) - assoc(A,A)$$

$$cut(A,B) = assoc(B,V) - assoc(B,B)$$

- $Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$
- $= \frac{assoc(A,V)-assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,V)-assoc(B,B)}{assoc(B,V)}$
- $= 2 - \left(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}\right) = 2 - Nassoc(A,B)$

Matrix Representation

- Adjacency matrix (W)
  - $n \times n$ matrix
  - w_{ij}: edge weight between vertex x_i and x_j
  - Symmetric matrix



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.2	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.3	0	0
x_4	0	0	0.3	0	0.8	0.7
x_5	0.2	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

Objective Function of Ncut

$$\boldsymbol{x} \in [1,-1]^n, x_i = \begin{cases} 1 & i \in A \\ -1 & i \in B \end{cases} \qquad d_i = \sum_j w_{ij}$$

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$= \frac{\sum_{x_i>0, x_j<0} -w_{ij} x_i x_j}{\sum_{x_i>0} d_i} + \frac{\sum_{x_i<0, x_j>0} -w_{ij} x_i x_j}{\sum_{x_i<0} d_i}$$

$$W \in R^{n \times n} \qquad D \in R^{n \times n} \qquad \boldsymbol{x} \in [1,-1]^n \qquad \mathbf{1} \in [1]^n \qquad k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$$

$$4\, Ncut(A,B) = \frac{(\mathbf{1}+\boldsymbol{x})^T (D-W)(\mathbf{1}+\boldsymbol{x})}{k\mathbf{1}^T D\mathbf{1}} + \frac{(\mathbf{1}-\boldsymbol{x})^T (D-W)(\mathbf{1}-\boldsymbol{x})}{(1-k)\mathbf{1}^T D\mathbf{1}}$$

$$b = \frac{k}{1-k}$$

$$= \frac{[(\mathbf{1}+\boldsymbol{x}) - b(1-\boldsymbol{x})]^T (D-W)[(\mathbf{1}+\boldsymbol{x}) - b(1-\boldsymbol{x})]}{b\mathbf{1}^T D\mathbf{1}}$$

Objective Function of Ncut

$$\boldsymbol{y} = (\mathbf{1} + \boldsymbol{x}) - b(1 - \boldsymbol{x}) \qquad k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i} \qquad b = \frac{k}{1-k} = \frac{\sum_{x_i>0} d_i}{\sum_{x_i<0} d_i}$$

$$\boldsymbol{y}^T D\mathbf{1} = 2\sum_{x_i>0} d_i - 2b\sum_{x_i<0} d_i = 0$$

$$\boldsymbol{y}^T D\boldsymbol{y} = 4\sum_{x_i>0} d_i + 4b^2\sum_{x_i<0} d_i = 4\left(b\sum_{x_i<0} d_i + b^2\sum_{x_i<0} d_i\right)$$

$$= 4b\left(\sum_{x_i<0} d_i + b\sum_{x_i<0} d_i\right) = 4b\mathbf{1}^T D\mathbf{1}$$

$$\min_{\boldsymbol{x}} Ncut(\boldsymbol{x}) = \min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(D-W)\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}$$

$$s.t. \quad \boldsymbol{y} \in [2, -2b]^n, \qquad \boldsymbol{y}^T D\mathbf{1} = 0$$

- NP-hard!

Rayleigh quotient

- Relaxation:

$$\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T (D - W)\boldsymbol{y}}{\boldsymbol{y}^T D \boldsymbol{y}}, \boldsymbol{y} \in \mathcal{R}^n, \boldsymbol{y}^T D \mathbf{1} = 0$$

- $L \equiv D - W$

$$\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T L \boldsymbol{y}}{\boldsymbol{y}^T D \boldsymbol{y}}, \boldsymbol{y} \in \mathcal{R}^n, \boldsymbol{y}^T D \mathbf{1} = 0$$

Rayleigh quotient

$$\max_x \frac{x^T A x}{x^T B x} \quad \Rightarrow \quad \max_x \; x^T A x \;\; s.t. \; x^T B x = 1$$

Lagrangian Function

$$L(x) = \; x^T A x + \lambda(x^T B x - 1)$$

Taking the derivative with respect to x

$$\frac{\partial L(x)}{\partial x} = 0 \quad \Rightarrow \quad (A + A^T)x + \lambda(B + B\hat{}T)x = 0$$

If A and B are symmetric

$$Ax = \kappa Bx, \kappa = -\lambda \quad \Leftarrow \quad \text{General Eigen Decompositio}$$

Generalized Eigen-problem

$$\min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(D-W)\boldsymbol{y}}{\boldsymbol{y}^T D\boldsymbol{y}}, \boldsymbol{y} \in \mathcal{R}^n, \boldsymbol{y}^T D\mathbf{1} = 0$$

$$(D-W)\boldsymbol{y} = \lambda D\boldsymbol{y}$$

- Eigenvector corresponding to the smallest eigenvalue.
- Vector $\mathbf{1}$ is the eigenvector corresponding to the eigenvalue 0.

$$(D-W)\boldsymbol{y} = \lambda D^{\frac{1}{2}} D^{\frac{1}{2}}\boldsymbol{y}$$

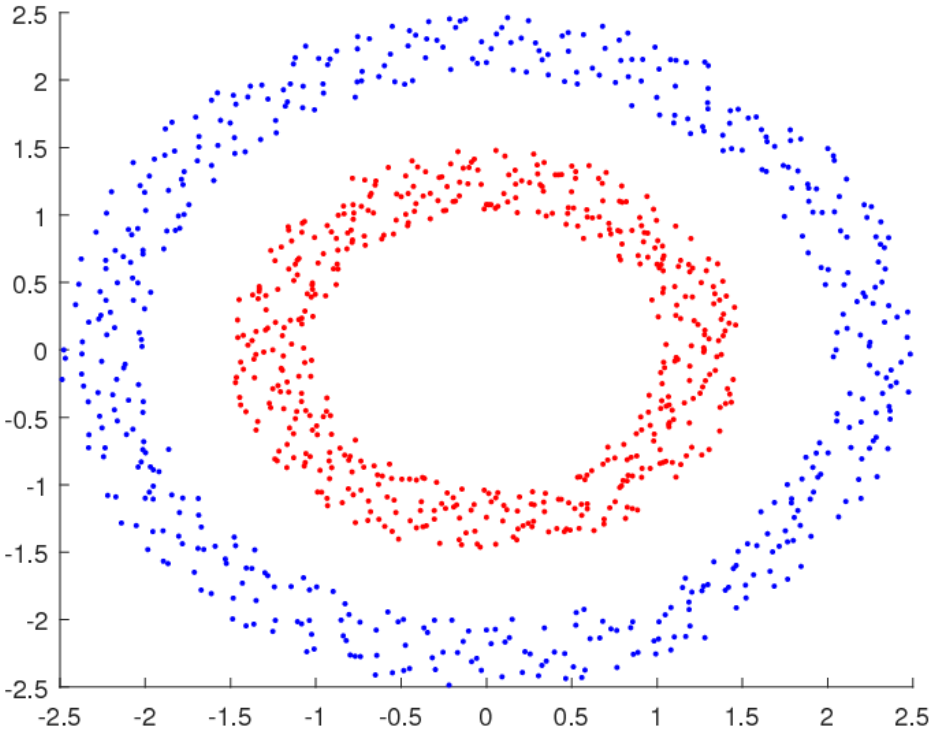
$$D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}D^{\frac{1}{2}}\boldsymbol{y} = \lambda D^{\frac{1}{2}}\boldsymbol{y}$$

$$D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}\boldsymbol{z} = \lambda \boldsymbol{z}$$

- $\boldsymbol{z}_1^T\boldsymbol{z}_2 = 0 \rightarrow \left(D^{\frac{1}{2}}\boldsymbol{y}_1\right)^T\left(D^{\frac{1}{2}}\boldsymbol{y}_2\right) = 0 \rightarrow \boldsymbol{y}_1^T D\boldsymbol{y}_2 = 0$

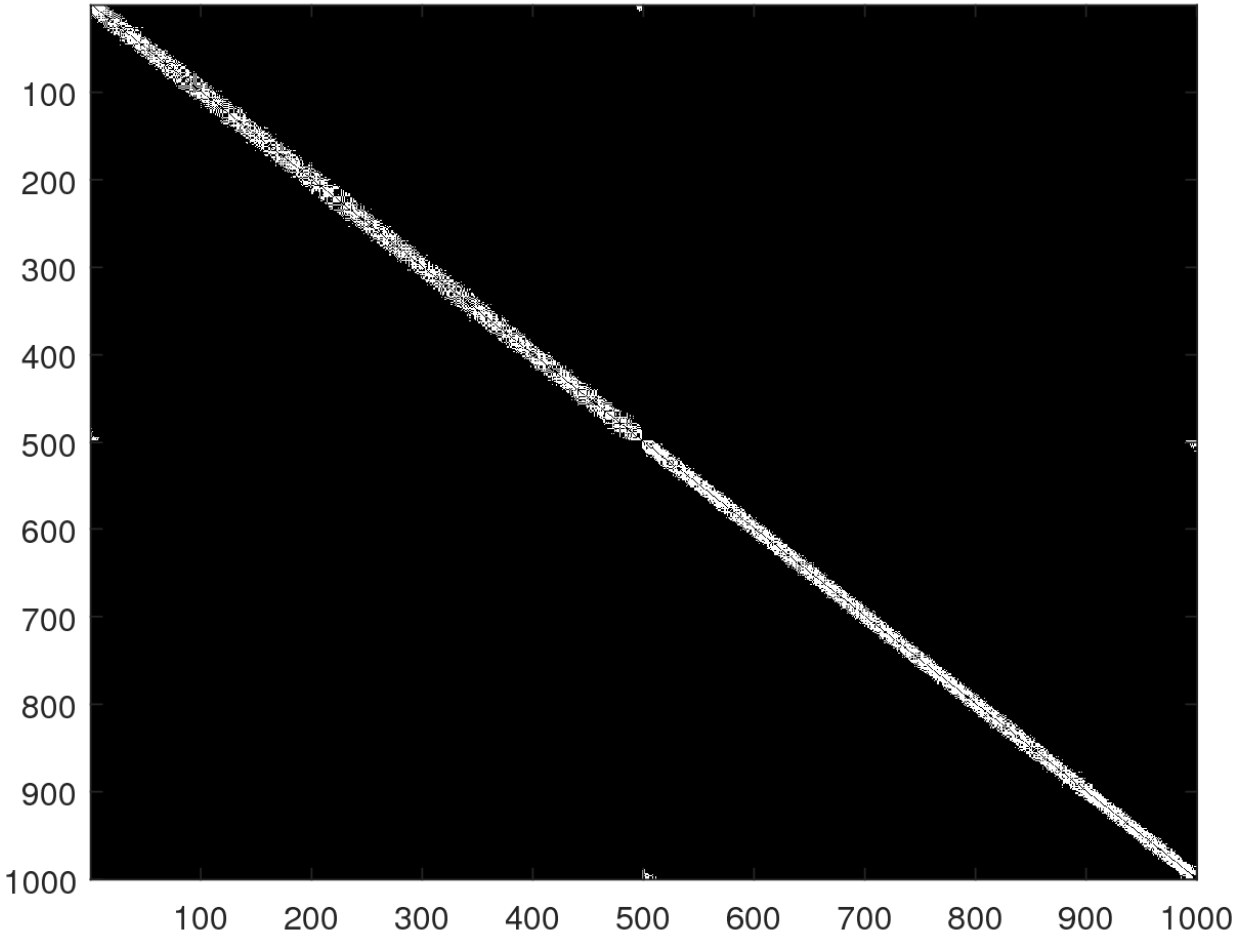
- The eigenvector corresponding to the 2nd small eigenvalue.

Spectral Clustering Example – 2 Circles



Spectral Clustering Example – 2 Circles

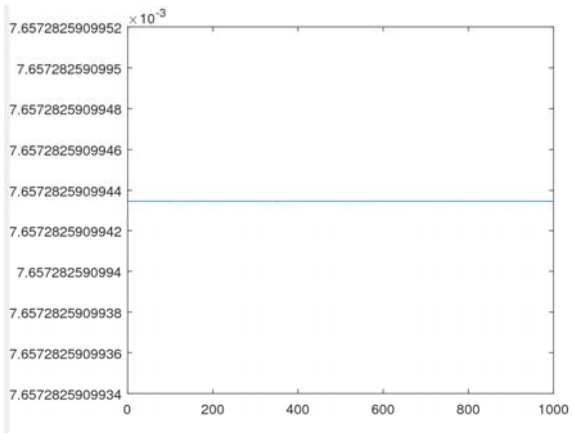
- 15nn nearest neighbor graph

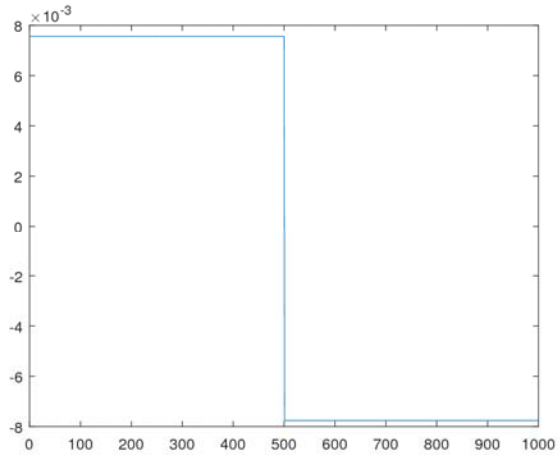


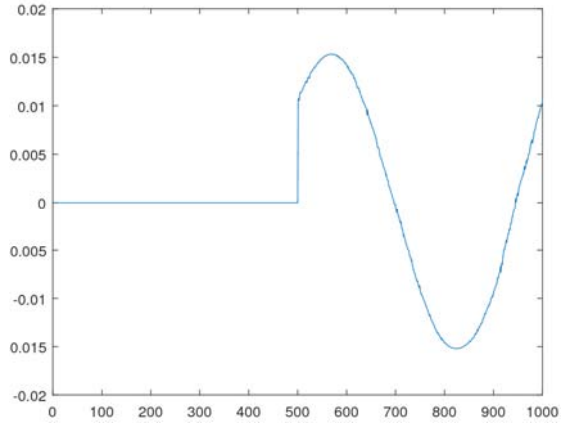
Spectral Clustering Example – 2 Circles

- The top 4 eigenvalues

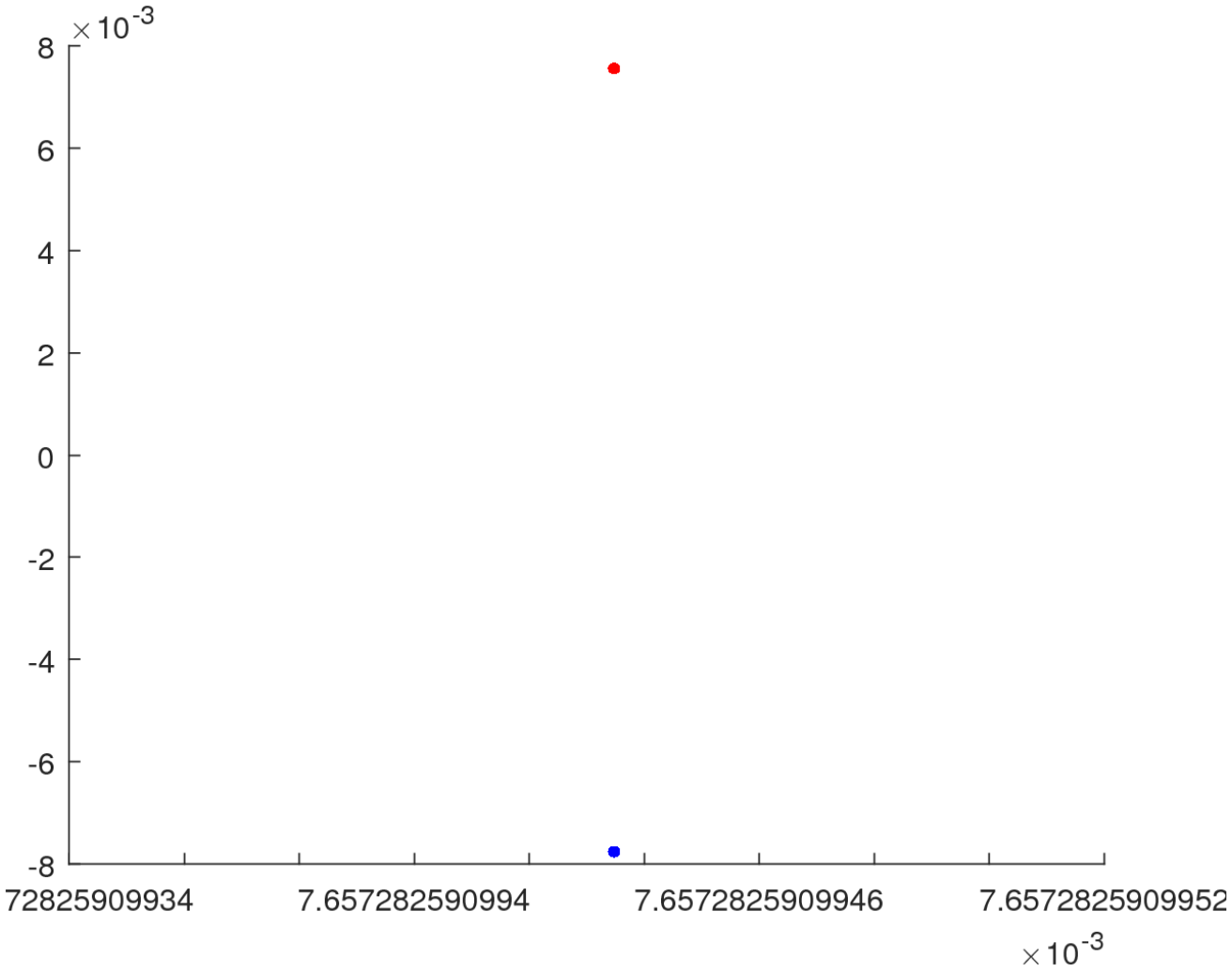
```
eigvalue =

    1.0000
    1.0000
    0.9972
    0.9970
```







Spectral Clustering Example – 2 Circles



K > 2

- Perform Ncut recursively.

- Use more than one eigenvectors.
  - Suppose $\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_k$ are the first k eigenvectors corresponding to the smallest eigenvalues, let
    $$Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k] \in R^{n \times k}$$
  - Each row vector of Y is a k dimensional representation of the original data point.
  - Performing kmeans.

Spectral Clustering Algorithm

1. Graph construction
   - Heat kernel $W_{ij} = \exp\left\{-\frac{\|x_i - x_j\|}{2\sigma^2}\right\}$
   - k-nearest neighbor graph
2. Eigen-problem
   - Compute eigenvalues and eigenvectors of the matrix L
   - Map each point to a lower-dimensional representation based on one or more eigenvectors.
3. Conventional clustering schemes, e.g. K-Means
   - Assign points to two or more clusters, based on the new representation.