Expectation-Maximization

- EM is an elegant and powerful method for MLE problems with latent variables
- Main idea: model parameters and latent variables are estimated iteratively, where average over the latent variables (expectation)
- A typical example application of EM is the Gaussian Mixture model (GMM)
- However, EM has many other applications
- First, we consider EM for GMMs

Expectation-Maximization for GMM

- First, we define the **responsibilities:**

$$\gamma(z_{nk}) = p(z_{nk} = 1 \mid \mathbf{x}_n) \qquad z_{nk} \in \{0, 1\}$$

$$\sum_k z_{nk} = 1$$

Expectation-Maximization for GMM

- First, we define the **responsibilities:**

$$\gamma(z_{nk}) = p(z_{nk} = 1 \mid \mathbf{x}_n)$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

Expectation-Maximization for GMM

- First, we define the **responsibilities:**

$$\gamma(z_{nk}) = p(z_{nk} = 1 \mid \mathbf{x}_n)$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

- Next, we derive the log-likelihood wrt. to $\boldsymbol{\mu}_k$:

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \boldsymbol{\mu}_k} \overset{!}{=} \mathbf{0}$$

Expectation-Maximization for GMM

- First, we define the **responsibilities:**

$$\gamma(z_{nk}) = p(z_{nk} = 1 \mid \mathbf{x}_n)$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

- Next, we derive the log-likelihood wrt. to $\boldsymbol{\mu}_k$:

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \boldsymbol{\mu}_k} \overset{!}{=} \mathbf{0}$$

and we obtain: $$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

Expectation-Maximization for GMM

- We can do the same for the covariances:

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \Sigma_k} \overset{!}{=} \mathbf{0}$$

and we obtain:

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

- Finally, we derive wrt. the mixing coefficients π_k :

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \pi_k} \overset{!}{=} \mathbf{0} \qquad \text{where:} \qquad \sum_{k=1}^{K} \pi_k = 1$$

Expectation-Maximization for GMM

- We can do the same for the covariances:

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \Sigma_k} \overset{!}{=} \mathbf{0}$$

  and we obtain:

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

- Finally, we derive wrt. the mixing coefficients π_k :

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \pi_k} \overset{!}{=} \mathbf{0} \quad \text{where:} \quad \sum_{k=1}^{K} \pi_k = 1$$

  and the result is: $\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk})$

Algorithm Summary

1. Initialize means $\boldsymbol{\mu}_k$ covariance matrices Σ_k and mixing coefficients $\boldsymbol{\pi}_k$
2. Compute the initial log-likelihood $\log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$
3. **E-Step.** Compute the responsibilities:

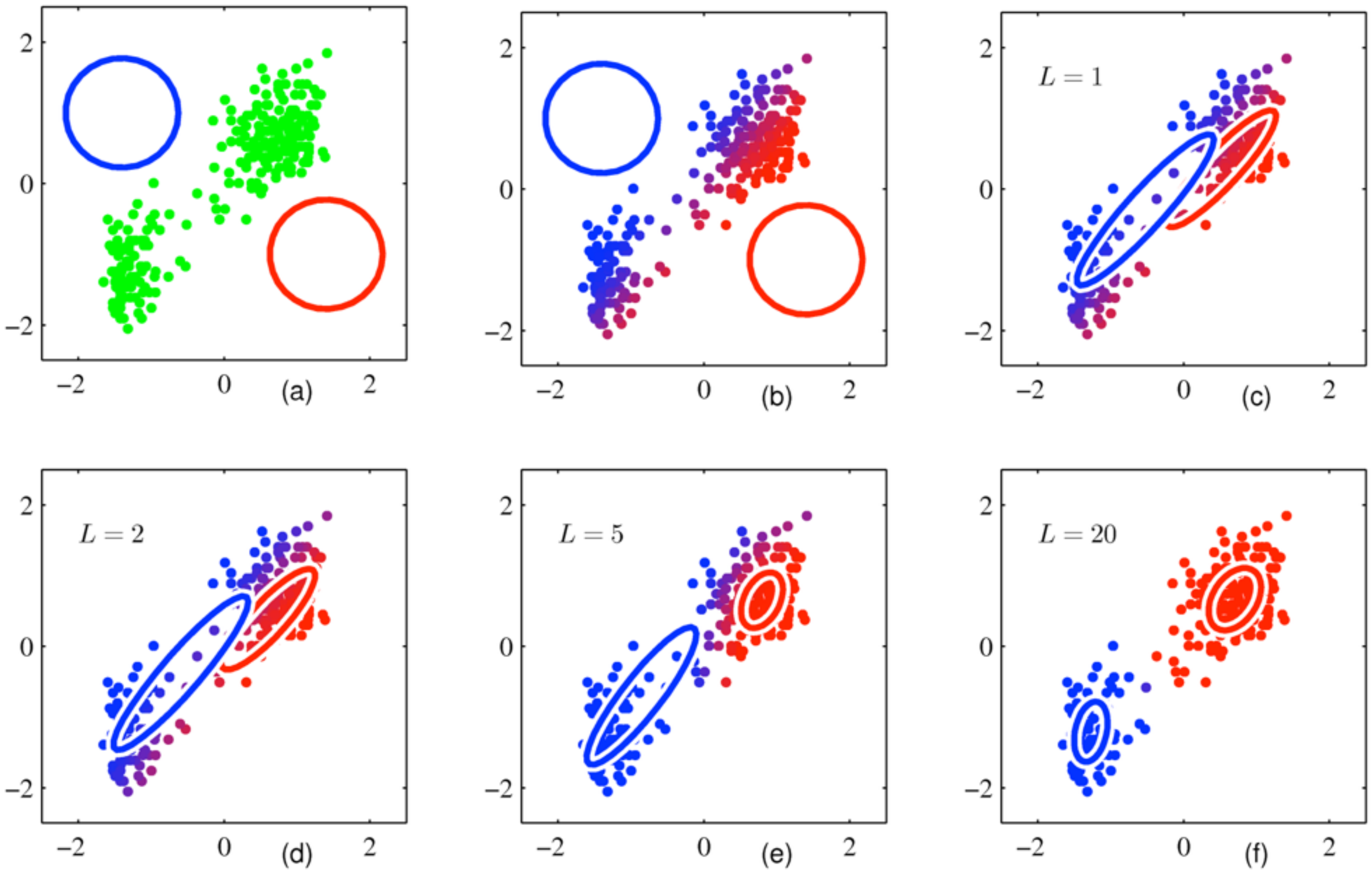
$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

4. **M-Step.** Update the parameters:

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})\mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \quad \Sigma_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T}{\sum_{n=1}^{N} \gamma(z_{nk})} \quad \pi_k^{\text{new}} = \frac{1}{N}\sum_{n=1}^{N} \gamma(z_{nk})$$

5. Compute log-likelihood; if not converged go to 3.

The Same Example Again



Why is it Called "EM"?

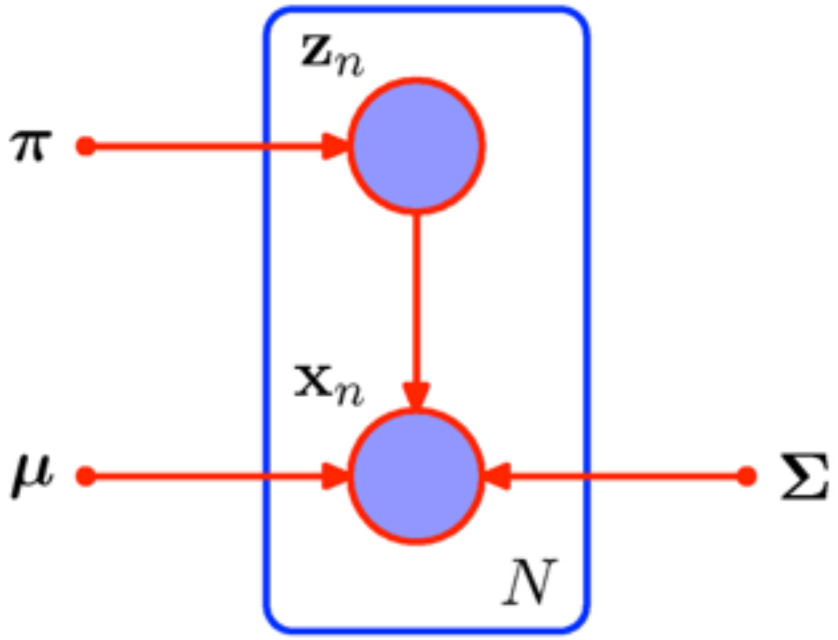
Assume for a moment that we observe X and the binary latent variables Z. The likelihood is then:

$$p(X, Z \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \prod_{n=1}^{N} p(\mathbf{z}_n \mid \boldsymbol{\pi}) p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\mu}, \Sigma)$$

"Complete-data log-likelihood"

where $p(\mathbf{z}_n \mid \boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{z_{nk}}$ and

$$p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\mu}, \Sigma) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)^{z_{nk}}$$



which leads to the log-formulation:

$$\log p(X, Z \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} (\log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k))$$

Why is it Called “EM”?

Instead of maximizing the joint log-likelihood, we maximize its **expectation** under the latent variable distribution:

$$\mathbb{E}_Z[\log p(X, Z \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)] = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_Z[z_{nk}](\log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k))$$

Why is it Called “EM”?

Instead of maximizing the joint log-likelihood, we maximize its **expectation** under the latent variable distribution:

$$\mathbb{E}_Z[\log p(X, Z \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)] = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_Z[z_{nk}](\log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k))$$

where the latent variable distribution per point is:

$$p(\mathbf{z}_n \mid \mathbf{x}_n, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_n \mid \mathbf{z}_n, \boldsymbol{\theta}) p(\mathbf{z}_n \mid \boldsymbol{\theta})}{p(\mathbf{x}_n \mid \boldsymbol{\theta})} \qquad \boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$$

$$= \frac{\prod_{l=1}^{K} (\pi_l \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_l, \Sigma_l))^{z_{nl}}}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

Observations

- Compared to K-means, points can now belong to both clusters (**soft assignment**)
- In addition to the cluster center, a covariance is estimated by EM
- Initialization is the same as used for K-means
- Number of iterations needed for EM is much higher
- Also: each cycle requires much more computation
- Therefore: start with K-means and run EM on the result of K-means (covariances can be initialized to the sample covariances of K-means)
- EM only finds a **local** maximum of the likelihood!

Questions

- How can we determine the number of clusters?
  - different approaches exist, e.g. by “trying out” several values for K and finding the one with highest likelihood
- What if the clusters can not be approximated well by Gaussians?
- Can we formulate an algorithm that only relies on pairwise similarities?

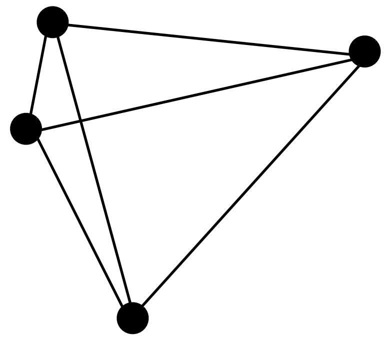
Questions

- How can we determine the number of clusters?
  - different approaches exist, e.g. by “trying out” several values for K and finding the one with highest likelihood
- What if the clusters can not be approximated well by Gaussians?
- Can we formulate an algorithm that only relies on pairwise similarities?

One example for such an algorithm is Spectral Clustering

Spectral Clustering

- Consider an undirected graph that connects all data points
- The edge weights are the similarities ("closeness")
- We define the weighted degree d_i of a node as the sum of all outgoing edges



$$W = \begin{array}{|c|c|c|c|}\hline & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$

$$d_i = \sum_{j=1}^{N} w_{ij}$$

$$D = \begin{array}{|c|c|c|c|}\hline d_1 & & & \\ \hline & d_2 & & \\ \hline & & d_3 & \\ \hline & & & d_4 \\ \hline \end{array}$$

Spectral Clustering

- The Graph Laplacian is defined as:

$$L = D - W$$

- This matrix has the following properties:
    - the 1 vector is eigenvector with eigenvalue 0

Spectral Clustering

- The Graph Laplacian is defined as:

$$L = D - W$$

- This matrix has the following properties:
  - the 1 vector is eigenvector with eigenvector 0
  - the matrix is symmetric and positive semi-definite

Spectral Clustering

- The Graph Laplacian is defined as:

$$L = D - W$$

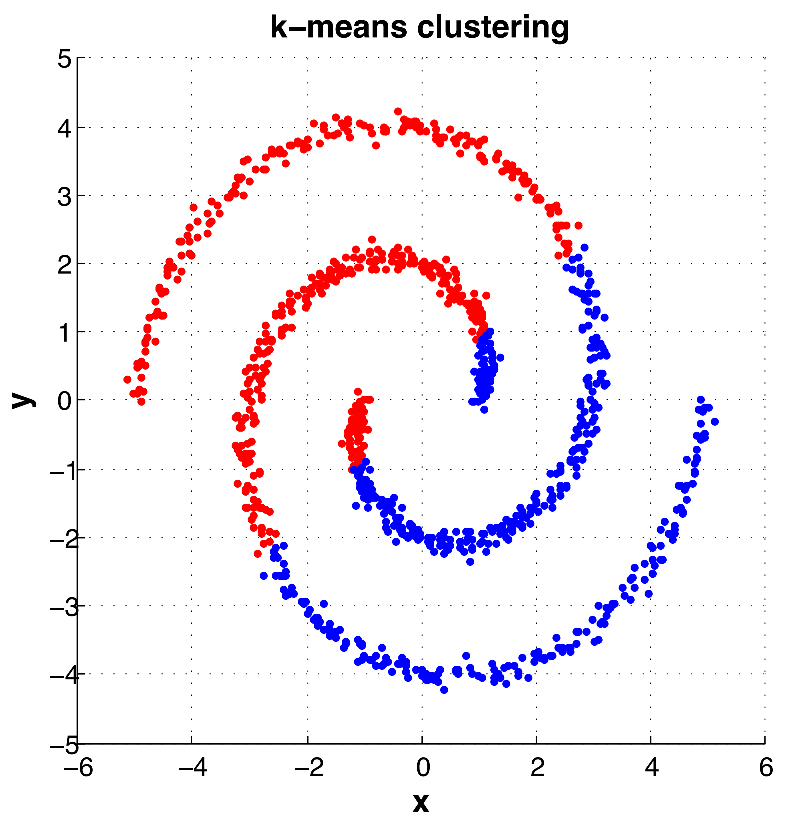
- This matrix has the following properties:
  - the 1 vector is eigenvector with eigenvector 0
  - the matrix is symmetric and positive semi-definite
- With these properties we can show:

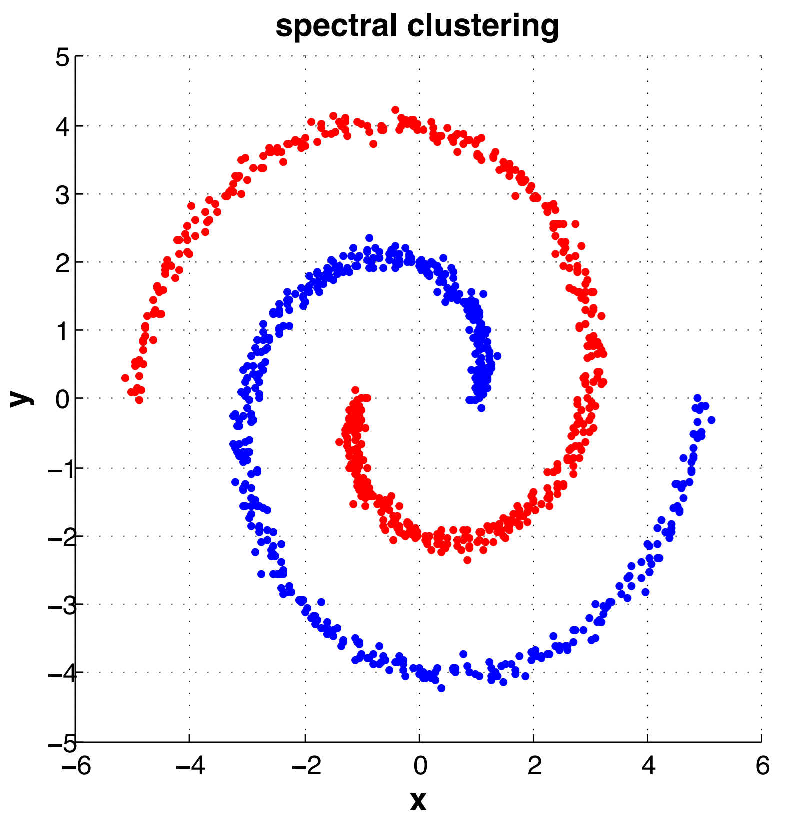
Theorem: The set of eigenvectors of L with eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_K}$, where A_k are the K connected components of the graph.

The Algorithm

- Input: Similarity matrix W
- Compute L = D - W
- Compute the eigenvectors that correspond to the K smallest eigenvalues
- Stack these vectors as columns in a matrix U
- Treat each row of U as a K-dim data point
- Cluster the N rows with K-means clustering
- The indices of the rows that correspond to the resulting clusters are those of the original data points.

An Example





- Spectral clustering can handle complex problems such as this one
- The complexity of the algorithm is $O(N^3)$, because it has to solve an eigenvector problem
- But there are efficient variants of the algorithm

Further Remarks

- To account for nodes that are highly connected, we can use a normalized version of the graph Laplacian
- Two different methods exist:
    - $L_{rw} = D^{-1}L = I - D^{-1}W$
    - $L_{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$
- These have similar eigenspaces than the original Laplacian L
- Clustering results tend to be better than with the unnormalized Laplacian

Affinity Propagation

- Another algorithm that is based on similarities
- The idea is to determine cluster centers ("exemplars") that explain other data points in an optimal way
- This is similar to k-medoids, but the algorithm is more robust against local minima
- **Idea:** each data point must choose another data point as its exemplar; some points will choose themselves as exemplar
- The number of clusters is then found automatically

Affinity Propagation

- Input: similarity values s(i,j)
- Initialize the responsibilities r(i,j), and the availabilities a(i,j) to 0
- do until convergence:
    - recompute the responsibilities:

      $$r(i,j) = s(i,j) - \max_{j' \neq j}\{a(i,j') + s(i,j')\}$$

    - recompute the availabilities:

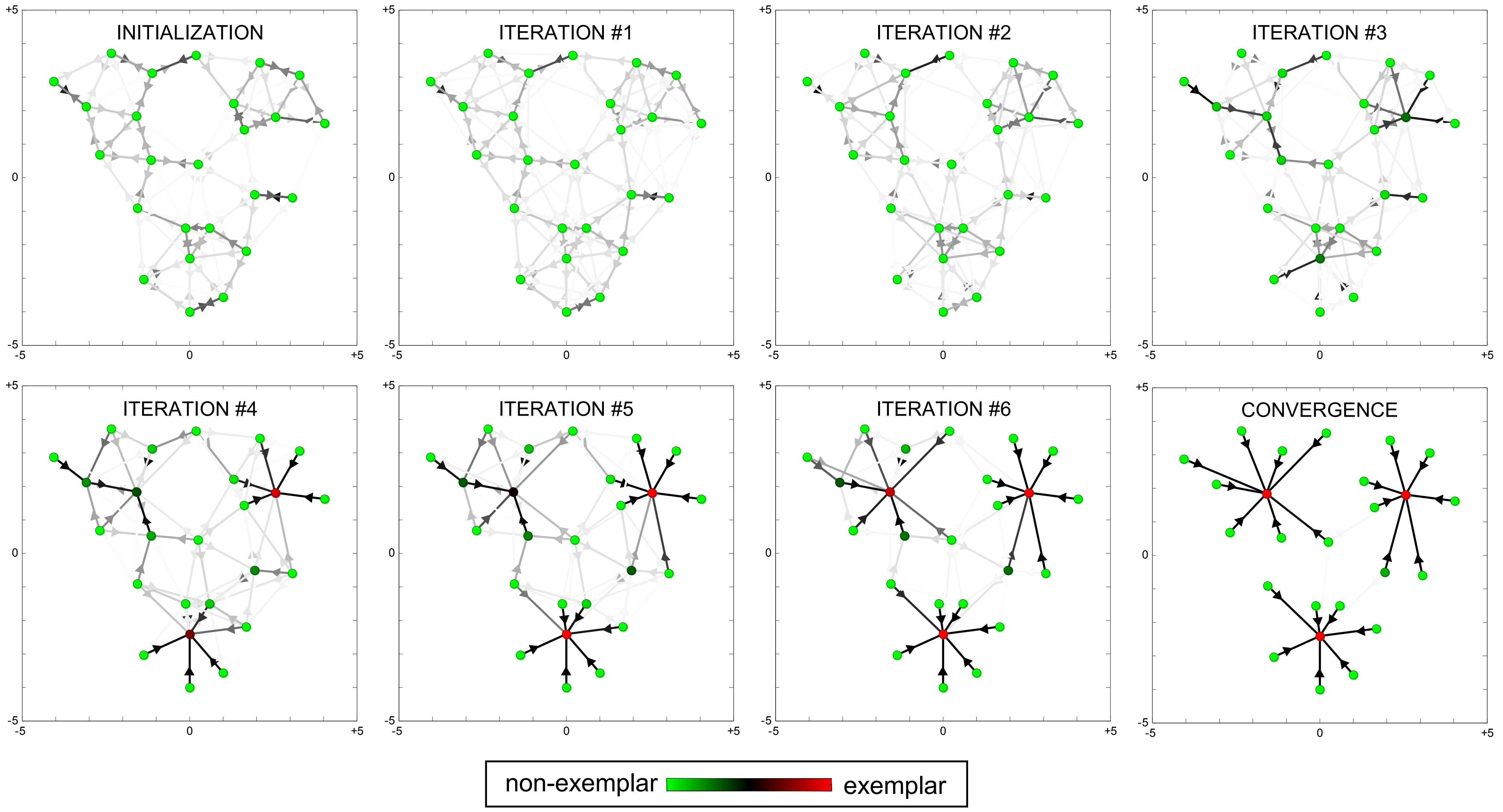
      $$a(i,j) = \min\left\{0, r(j,j) + \sum_{i' \notin \{i,j\}} \max\{0, r(i',j)\}\right\}$$

- the j that maximizes r(i,j) + a(i,j) is the exemplar of i

Affinity Propagation

- Intuitively:
  - responsibility measures how much i thinks that j would be a good exemplar
  - availability measures how strongly j thinks it should be an exemplar for i
- The algorithm can be shown to be equivalent to max-product loopy belief propagation
- Convergence is not guaranteed, but with “damping” oscillations can be avoided
- The number of clusters can be controlled by the “self-similarity”

Affinity Propagation



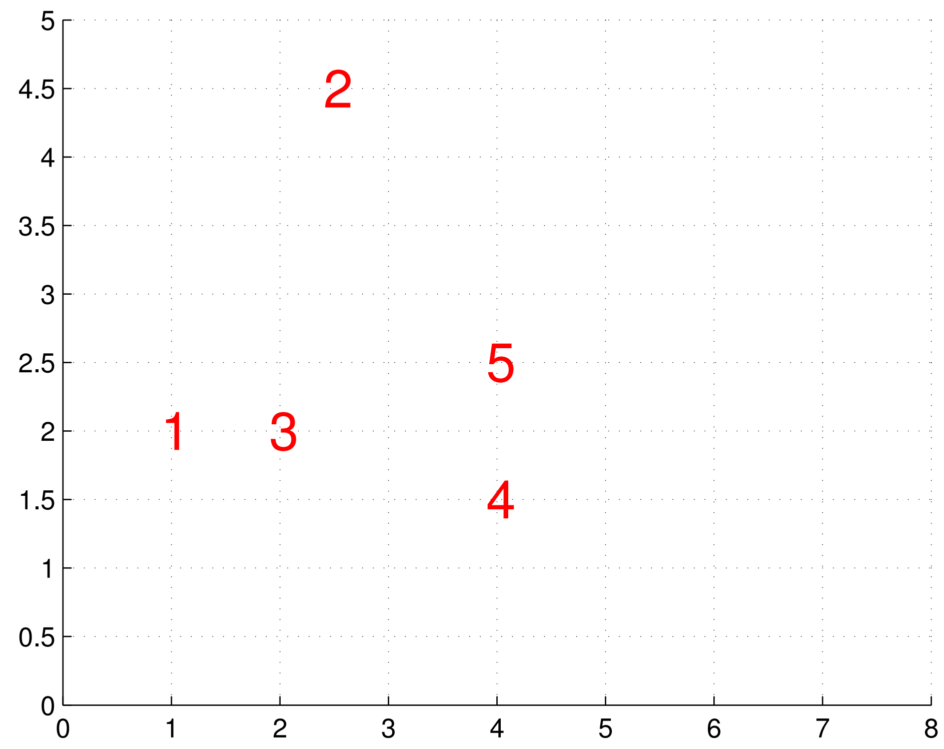
- Colours: how much each point wants to be an exemplar
- Edge strengths: how much a point wants to belong to a cluster

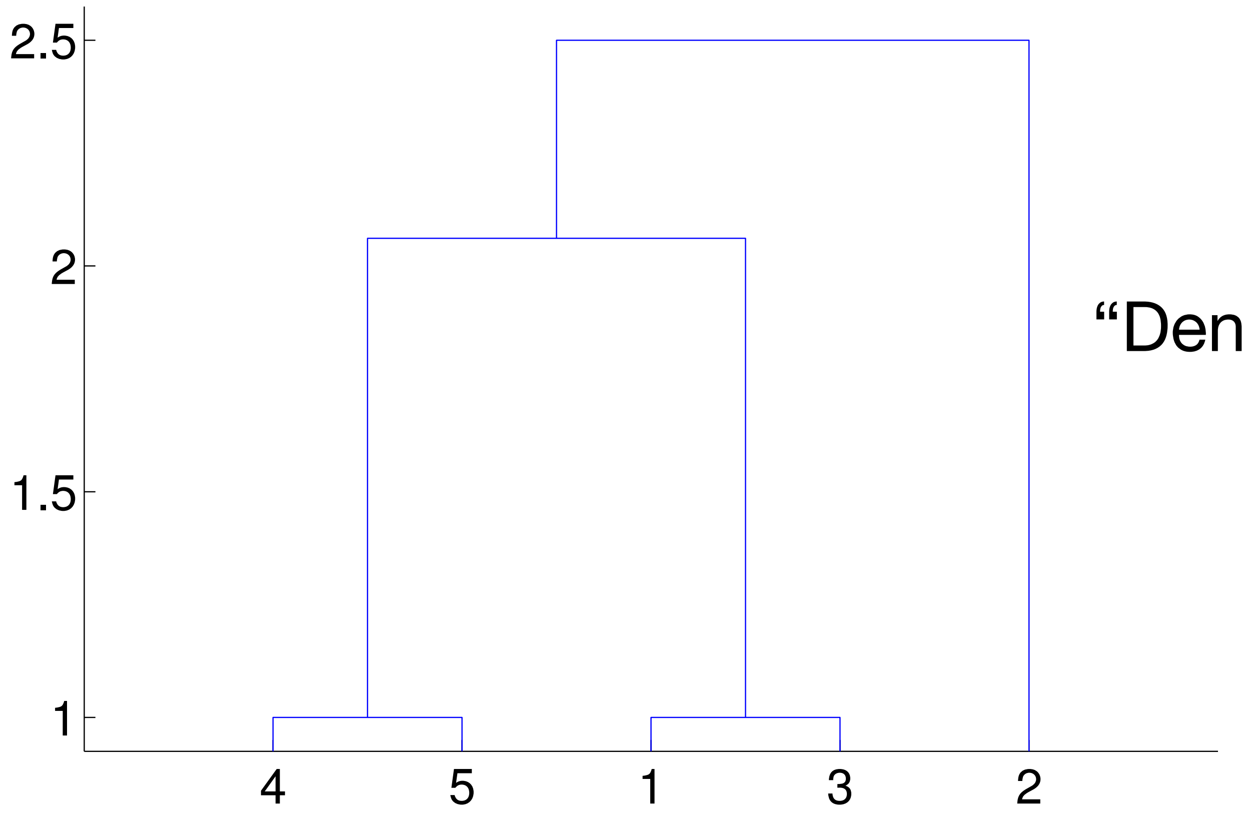
Hierarchical Clustering

- Often, we want to have nested clusters instead of a “flat” clustering
- Two possible methods:
  - “bottom-up” or agglomerative clustering
  - “top-down” or divisive clustering
- Both methods take a dissimilarity matrix as input
- Bottom-up grows merges points to clusters
- Top-down splits clusters into sub-clusters
- Both are heuristics, there is no clear objective function
- They always produce a clustering (also for noise)

Agglomerative Clustering

- Start with N clusters, each contains exactly one data point
- At each step, merge the two most similar groups
- Repeat until there is a single group





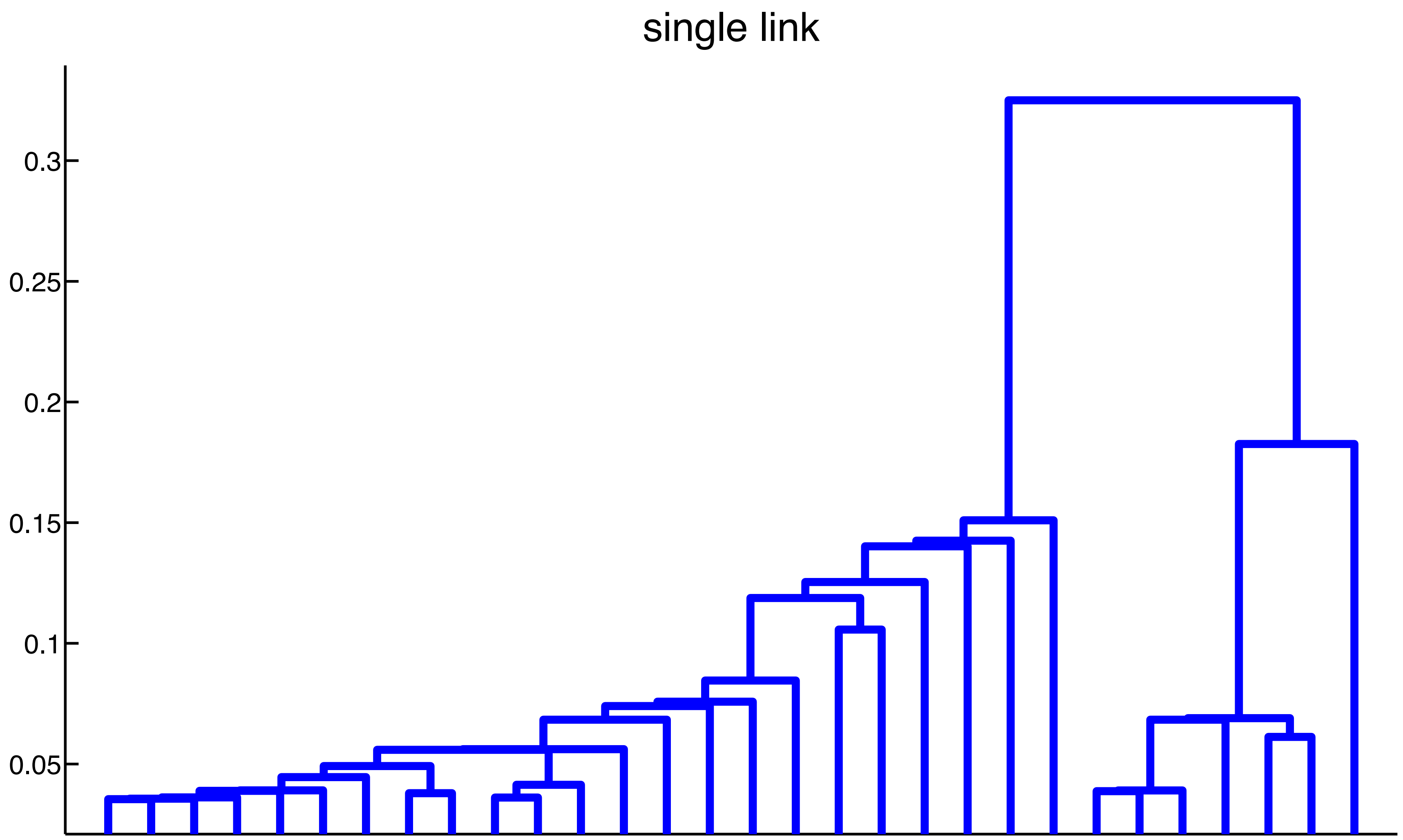
“Dendrogram”

Linkage

- In agglomerative clustering, it is important to define a distance measure between two clusters
- There are three different methods:
  - Single linkage: considers the two closest elements from both clusters and uses their distance
  - Complete linkage: considers the two farthest elements from both clusters
  - Average linkage: uses the average distance between pairs of points from both clusters
- Depending on the application, one linkage should be preferred over the other

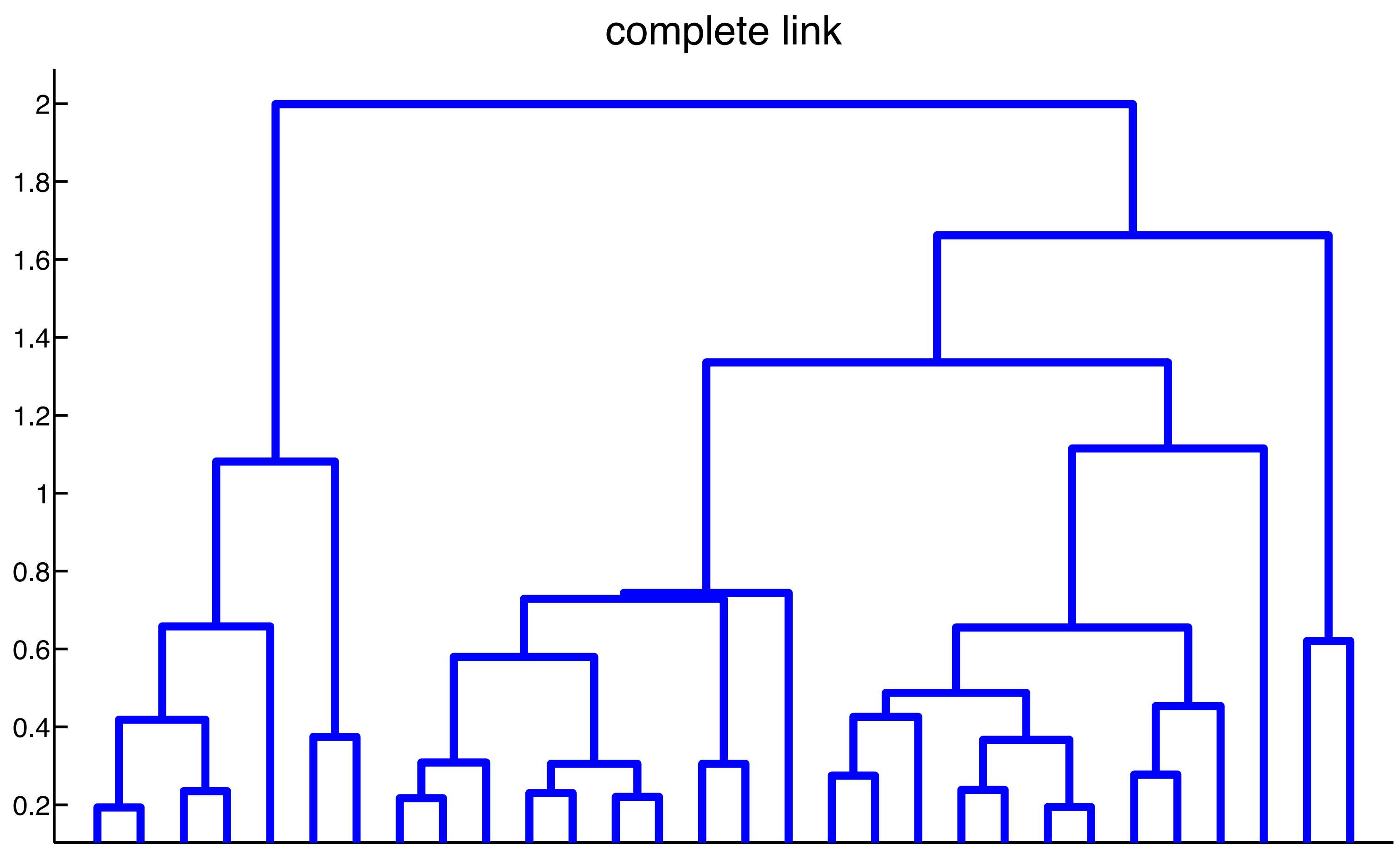
Single Linkage

- The distance is based on $d_{SL}(G,H) = \min_{i \in G, i' \in H} d_{i,i'}$
- The resulting dendrogram is a minimum spanning tree, i.e. it minimizes the sum of the edge weights
- Thus: we can compute the clustering in $O(N^2)$ time



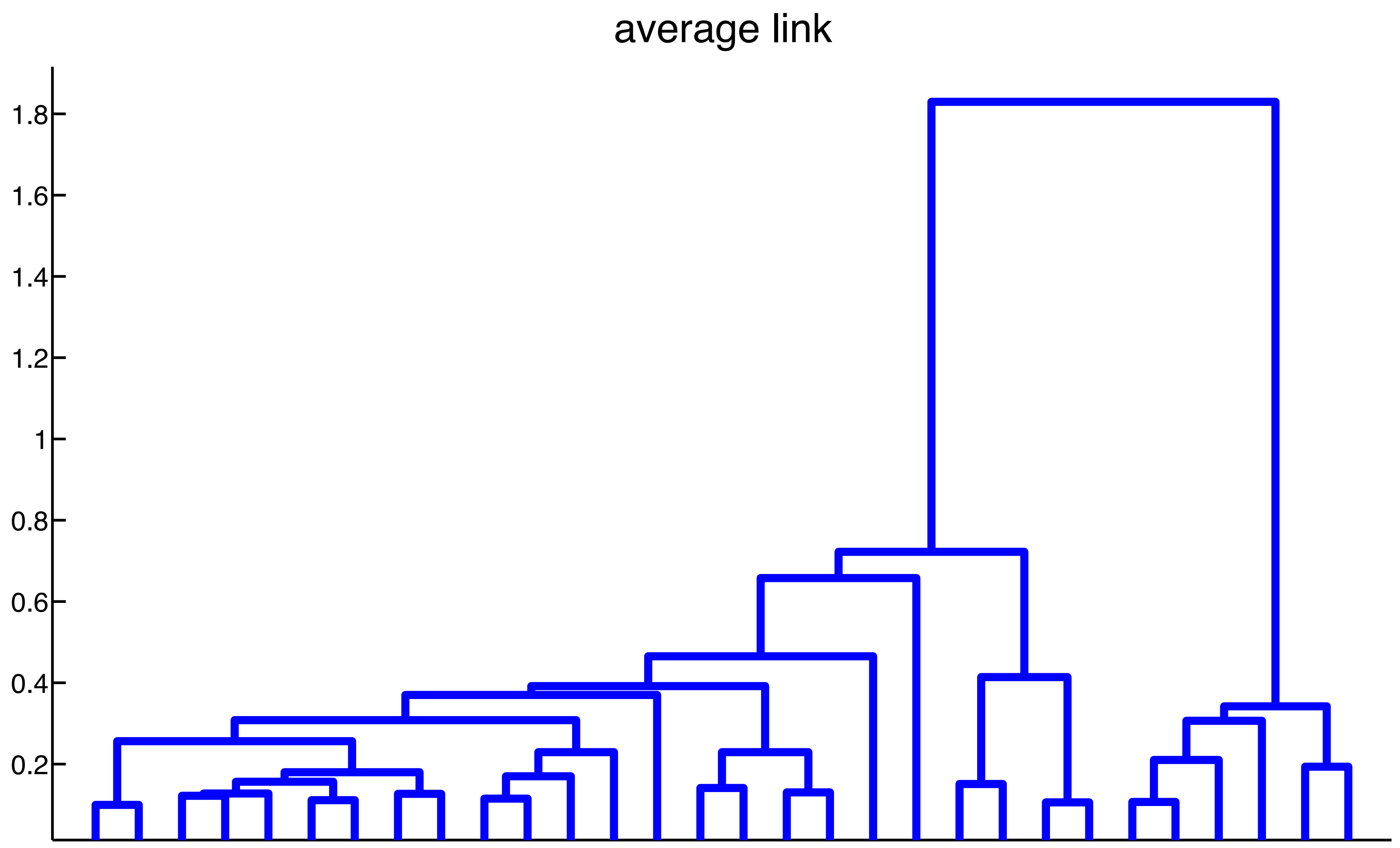
Complete Linkage

- The distance is based on $d_{CL}(G,H) = \max_{i \in G, i' \in H} d_{i,i'}$
- Complete linkage fulfills the **compactness property**, i.e. all points in a group should be similar to each other
- Tends to produce clusters with smaller diameter



Average Linkage

- The distance is based on $d_{avg}(G,H) = \frac{1}{n_G n_H} \sum_{i \in G} \sum_{i' \in H} d_{i,i'}$
- Is a good compromise between single and complete linkage
- However: sensitive to changes on the meas. scale



Divisive Clustering

- Start with all data in a single cluster
- Recursively divide each cluster into two child clusters
- Problem: optimal split is hard to find
- Idea: use the cluster with the largest diameter and use K-means with K = 2
- Or: use minimum-spanning tree and cut links with the largest dissimilarity
- In general two advantages:
  - Can be faster
  - More globally informed (not myopic as bottom-up)

Choosing the Number of Clusters

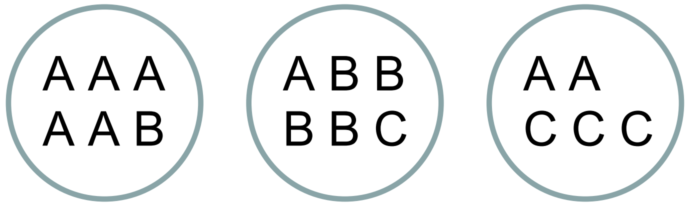
- As in general, choosing the number of clusters is hard
- When a dendrogram is available, a gap can be detected in the lengths of the links
- This represents the dissimilarity between merged groups
- However: in real data this can be hard to detect
- There are Bayesian techniques to address this problem (Bayesian hierarchical clustering)

Evaluation of Clustering Algorithms

- Clustering is unsupervised: evaluation of the output is hard, because no ground truth is given
- Intuitively, points in a cluster should be similar and points in different clusters dissimilar
- However, better methods use external information, such as labels or a reference clustering
- Then we can compare clusterings with the labels using different metrics, e.g.
  - purity
  - mutual information

Purity

- Define N_{ij} the number of objects in cluster i that are in class j
- Define $N_i = \sum_{j=1}^{C} N_{ij}$ number of objects in cluster i
- $p_{ij} = \frac{N_{ij}}{N_i} \qquad p_i = \max_j p_{ij}$ "Purity"
- overall purity
$$\sum_i \frac{N_i}{N} p_i$$



Purity = 0.71

- Purity ranges from 0 (bad) to 1 (good)
- But: a clustering with each object in its own cluster has a purity of 1

Mutual Information

- Let U and V be two clusterings
- Define the probability that a randomly chosen point belongs to cluster u_i in U and to v_j in V

$$p_{UV}(i,j) = \frac{|u_i \cap v_j|}{N}$$

- Also: The prob. that a point is in u_i $\quad p_U(i) = \frac{|u_i|}{N}$

$$\mathbb{I}(U,V) = \sum_{i=1}^{R} \sum_{j=1}^{C} p_{UV}(i,j) \log \frac{p_{UV}(i,j)}{p_U(i)p_V(j)}$$

- This can be normalized to account for many small clusters with low entropy

Summary

- Several Clustering methods exist:
  - K-means clustering and Expectation-Maximization, both based on Gaussian Mixture Models
  - K-means uses hard assignments, whereas EM uses soft assignments and estimates also the covariances
  - Spectral clustering uses the graph Laplacian and performs an eigenvector analysis
- Major Problem:
  - most clustering algorithms require the number of clusters to be given