



# 12. Clustering

# Motivation

- Supervised learning is good for interaction with humans, but labels from a supervisor are hard to obtain
- Clustering is unsupervised learning, i.e. it tries to learn only from the data
- Main idea: find a similarity measure and group similar data objects together
- Clustering is a very old research field, many approaches have been suggested
- Main problem in most methods: how to find a good number of clusters



# Clustering using Mixture Models

- The full posterior of the Gaussian Mixture Model is

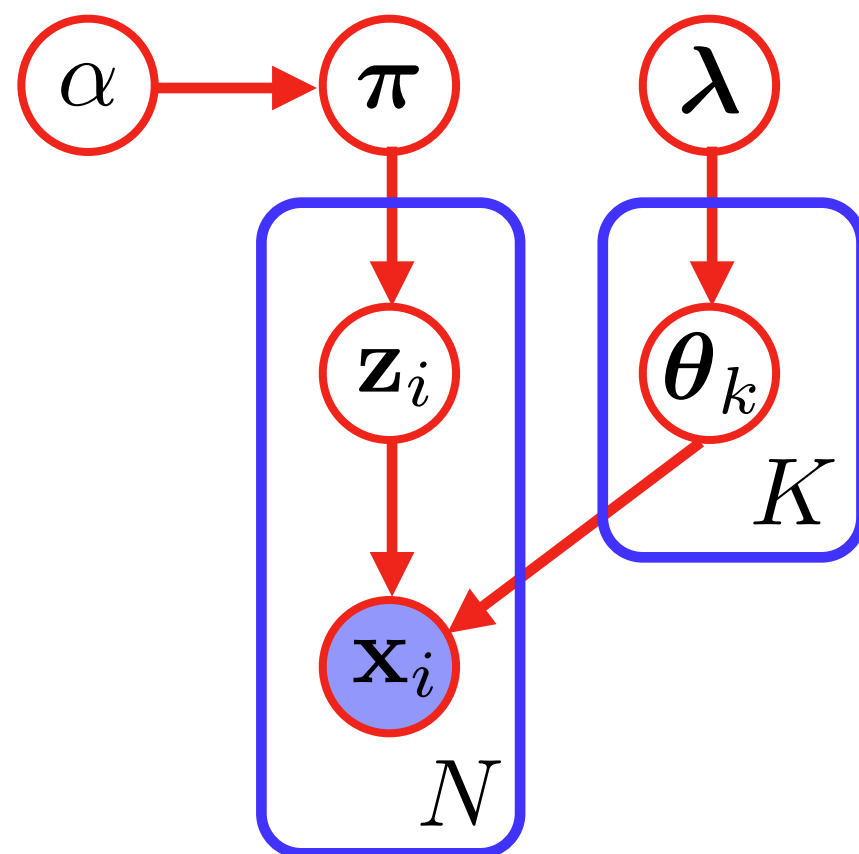
$$p(X, Z, \mu, \Sigma, \pi) = \underbrace{p(X | Z, \mu, \Sigma)}_{\text{data likelihood (Gaussian)}} \underbrace{p(Z | \pi)}_{\text{correspondence prob. (Multinomial)}} \underbrace{p(\pi | \alpha)}_{\text{mixture prior (Dirichlet)}} \underbrace{p(\mu, \Sigma | \lambda)}_{\text{parameter prior (Gauss-IW)}}$$

data likelihood  
(Gaussian)

correspondence  
prob. (Multinomial)

mixture prior  
(Dirichlet)

parameter prior  
(Gauss-IW)



In this model, we use:

- $\mu = (\mu_1, \dots, \mu_K)$
- $\Sigma = (\Sigma_1, \dots, \Sigma_K)$
- $(\mu_k, \Sigma_k) = \theta_k$

Simplification for now:

- Assume  $\Sigma_k$  are known
- Thus:  $\theta_k = \mu_k$



# Clustering using Mixture Models

- The full posterior of the Gaussian Mixture Model is

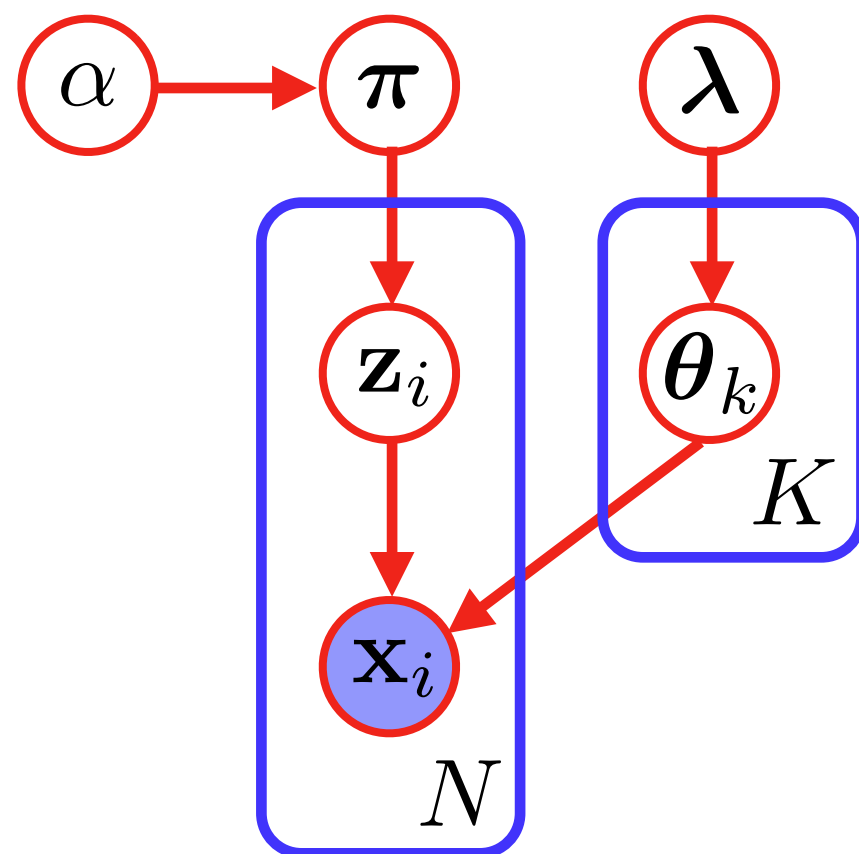
$$p(X, Z, \mu, \Sigma, \pi) = \underbrace{p(X | Z, \mu, \Sigma)}_{\text{data likelihood (Gaussian)}} \underbrace{p(Z | \pi)}_{\text{correspondence prob. (Multinomial)}} \underbrace{p(\pi | \alpha)}_{\text{mixture prior (Dirichlet)}} \underbrace{p(\mu, \Sigma | \lambda)}_{\text{parameter prior (Gauss-IW)}}$$

data likelihood  
(Gaussian)

correspondence  
prob. (Multinomial)

mixture prior  
(Dirichlet)

parameter prior  
(Gauss-IW)



Given this model, we can create new samples:

1. Sample  $\pi, \theta_k$  from priors
2. Sample corresp.  $z_i$
3. Sample data point  $x_i$



# Clustering using Mixture Models

- The full posterior of the Gaussian Mixture Model is

$$p(X, Z, \mu, \Sigma, \pi) = \underbrace{p(X | Z, \mu, \Sigma)}_{\text{data likelihood (Gaussian)}} \underbrace{p(Z | \pi)}_{\text{correspondence prob. (Multinomial)}} \underbrace{p(\pi | \alpha)}_{\text{mixture prior (Dirichlet)}} \underbrace{p(\mu, \Sigma | \lambda)}_{\text{parameter prior (Gauss-IW)}}$$

data likelihood  
(Gaussian)

correspondence  
prob. (Multinomial)

mixture prior  
(Dirichlet)

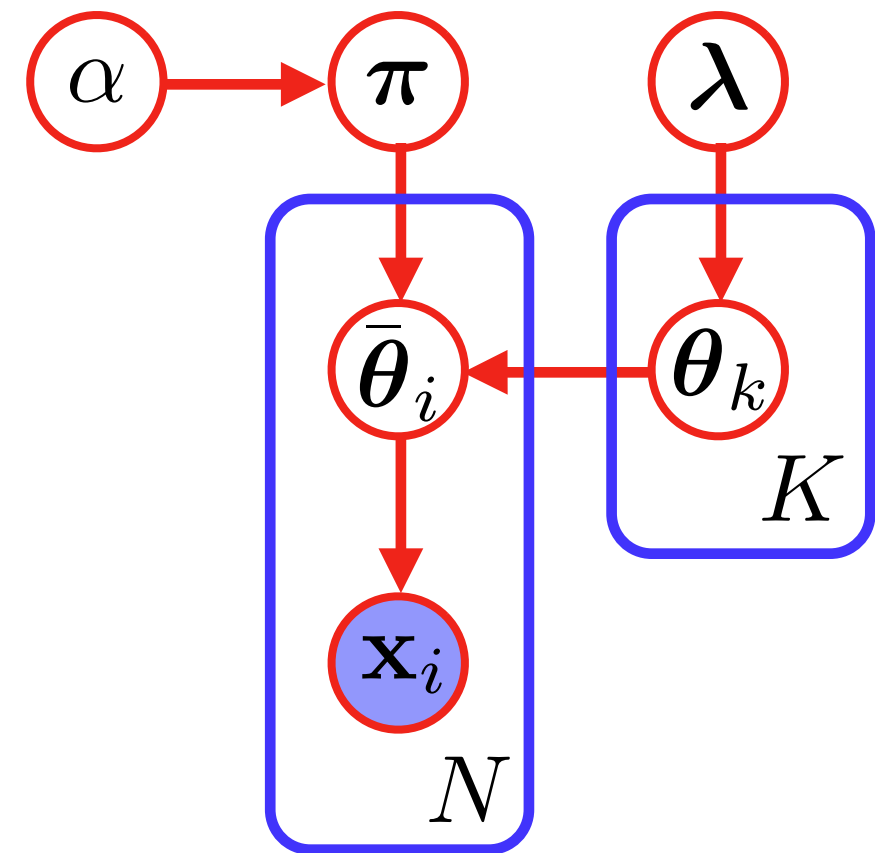
parameter prior  
(Gauss-IW)

An equivalent formulation of this model is this:

1. Sample  $\pi, \theta_k$  from priors
2. Sample params  $\bar{\theta}_i$  from:

$$p(\bar{\theta}_i | \pi, \theta_k) = \sum_{k=1}^K \pi_k \delta(\theta_k, \bar{\theta}_i)$$

3. Sample data point  $\mathbf{x}_i$



# Clustering using Mixture Models

What is the difference in that model?

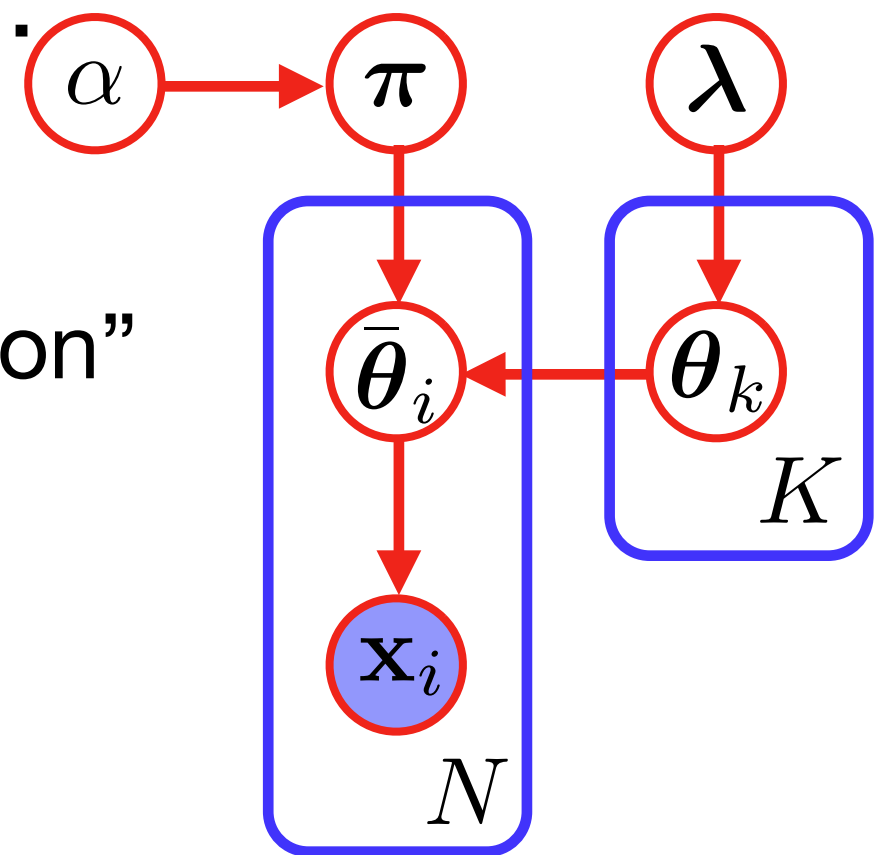
- there is one parameter  $\bar{\theta}_i$  for each observation  $\mathbf{x}_i$
- intuitively: we first sample the location of the cluster and then the data that corresponds to it

In general, we use the notation:

$$\pi \sim \text{Dir}\left(\frac{\alpha}{K} \mathbf{1}\right)$$
$$\theta_k \sim H(\lambda) \quad \text{“Base distribution”}$$

$$\bar{\theta}_i \sim G(\pi, \theta_k) \quad \text{where}$$

$$G(\pi, \theta_k) = \sum_{k=1}^K \pi_k \delta(\theta_k, \bar{\theta}_i)$$



**However: We need to know K**



# The Dirichlet Process

- So far, we assumed that  $K$  is known
- To extend that to infinity, we use a trick:

**Definition:** A Dirichlet process (DP) is a distribution over probability measures  $G$ , i.e.  $G(\theta) \geq 0$  and

$\int G(\theta) d\theta = 1$ . If for any partition  $(T_1, \dots, T_K)$  it holds:

$$(G(T_1), \dots, G(T_K)) \sim \text{Dir}(\alpha H(T_1), \dots, \alpha H(T_K))$$

then  $G$  is sampled from a Dirichlet process.

**Notation:**  $G \sim \text{DP}(\alpha, H)$

where  $\alpha$  is the **concentration parameter**  
and  $H$  is the **base measure**



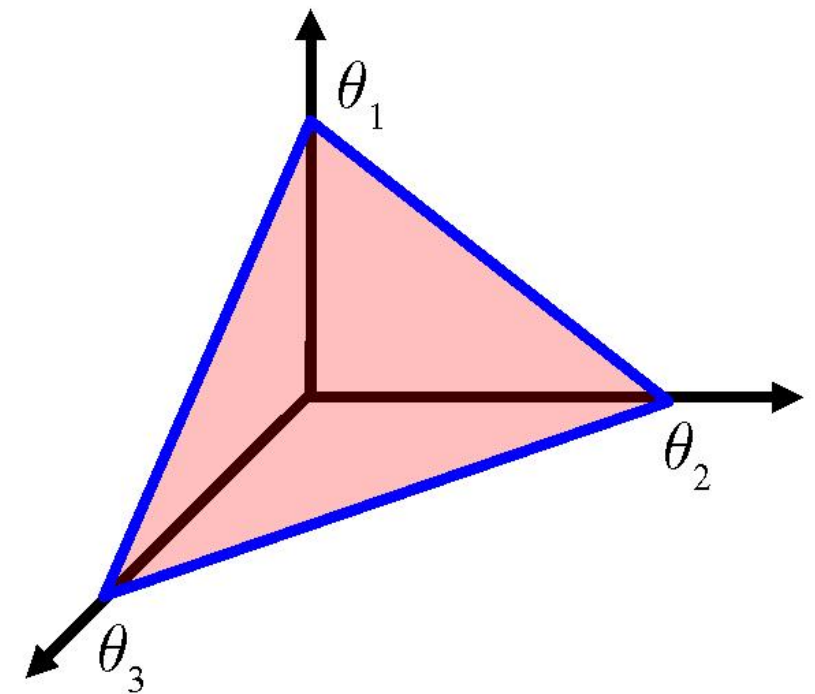
# One Step Backwards...

- The Dirichlet distribution is defined as:

$$\text{Dir}(\boldsymbol{\mu} \mid \boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_K)} \prod_{k=1}^K \mu_k^{\alpha_k - 1} \quad \alpha_0 = \sum_{k=1}^K \alpha_k$$

$$0 \leq \mu_k \leq 1 \quad \sum_{k=1}^K \mu_k = 1$$

- It is the conjugate prior for the multinomial distribution
- There, the parameter  $\alpha$  can be interpreted as the effective number of observations for every state

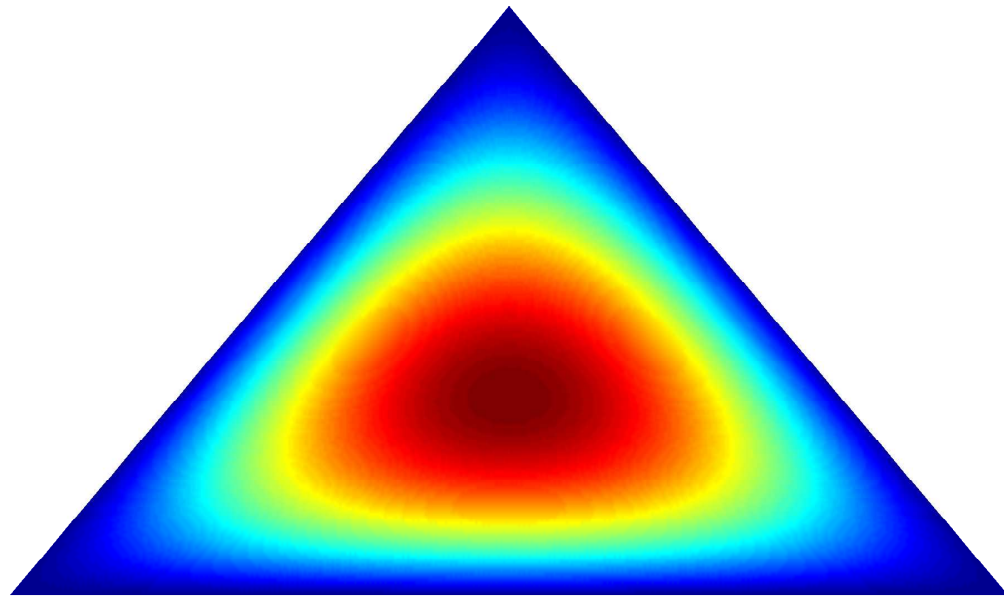


The simplex for K=3



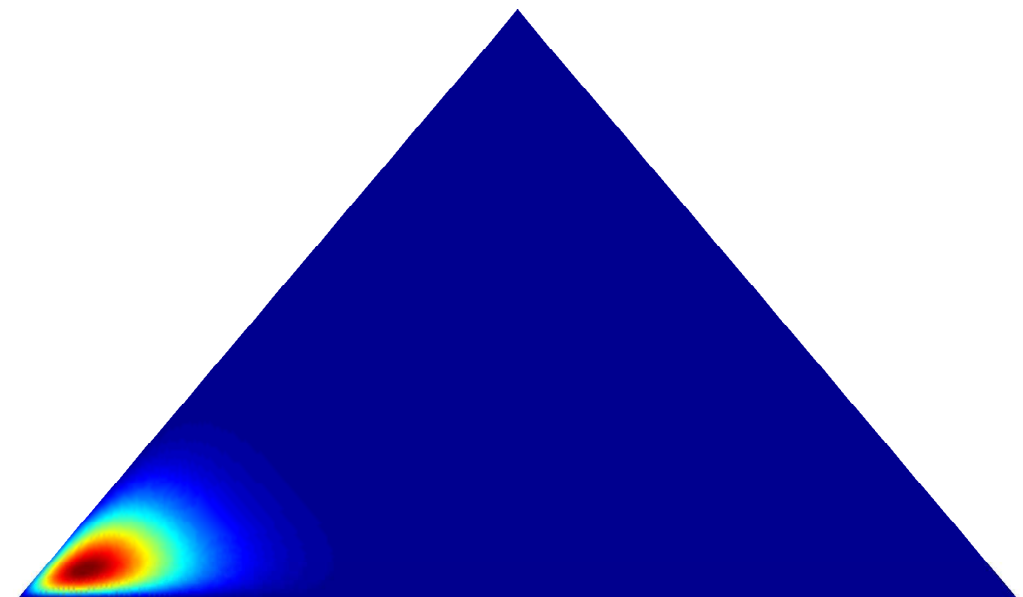


# Some Examples

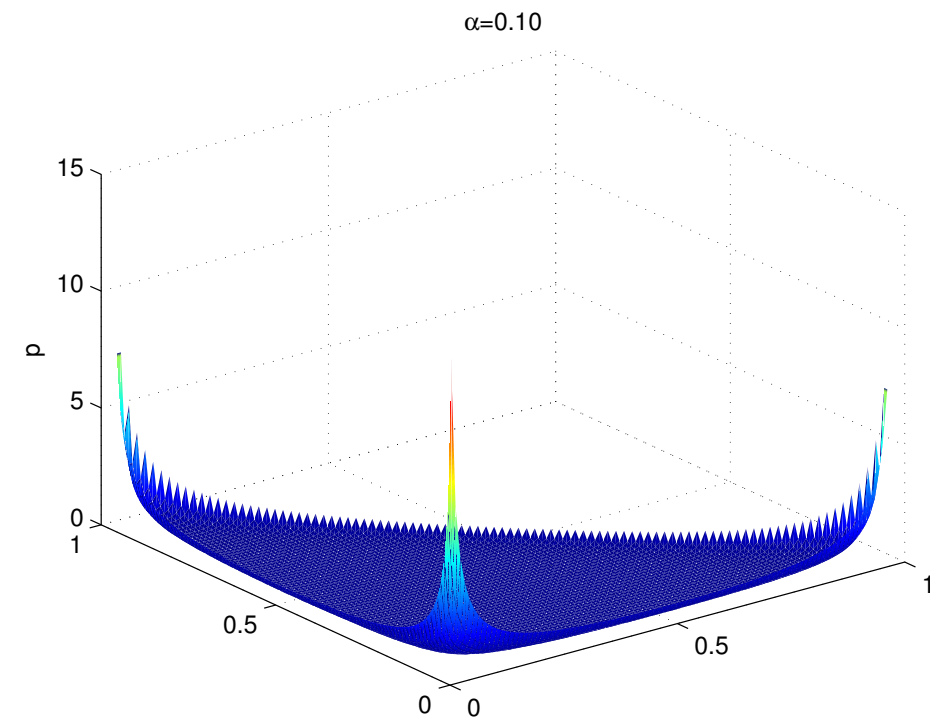


$$\alpha = (2, 2, 2)$$

- $\alpha_0$  controls the strength of the distribution (“peakedness”)
- $\alpha_k$  control the location of the peak



$$\alpha = (20, 2, 2)$$

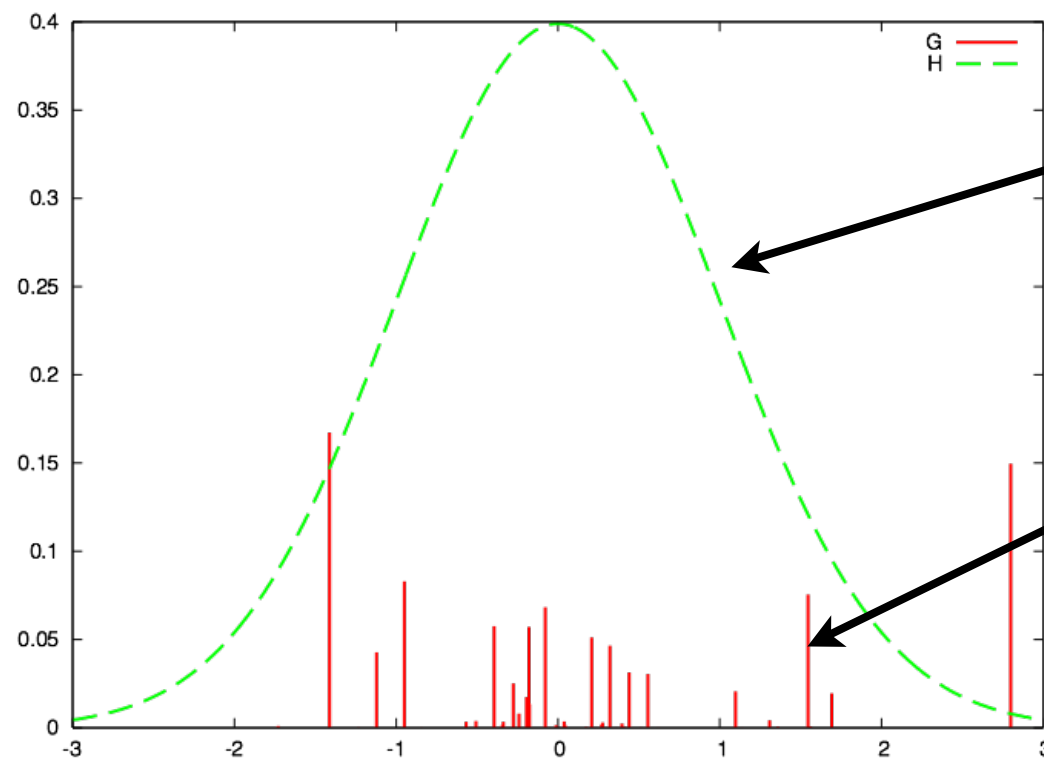


$$\alpha = (0.1, 0.1, 0.1)$$



# Intuitive Interpretation

- Every sample from a Dirichlet distribution is a vector of  $K$  positive values that sum up to 1, i.e. the sample itself is a finite distribution
- Accordingly, a sample from a Dirichlet process is an infinite (but still discrete!) distribution



Base distribution  
(here Gaussian)

Infinitely many  
samples (sum up to 1)

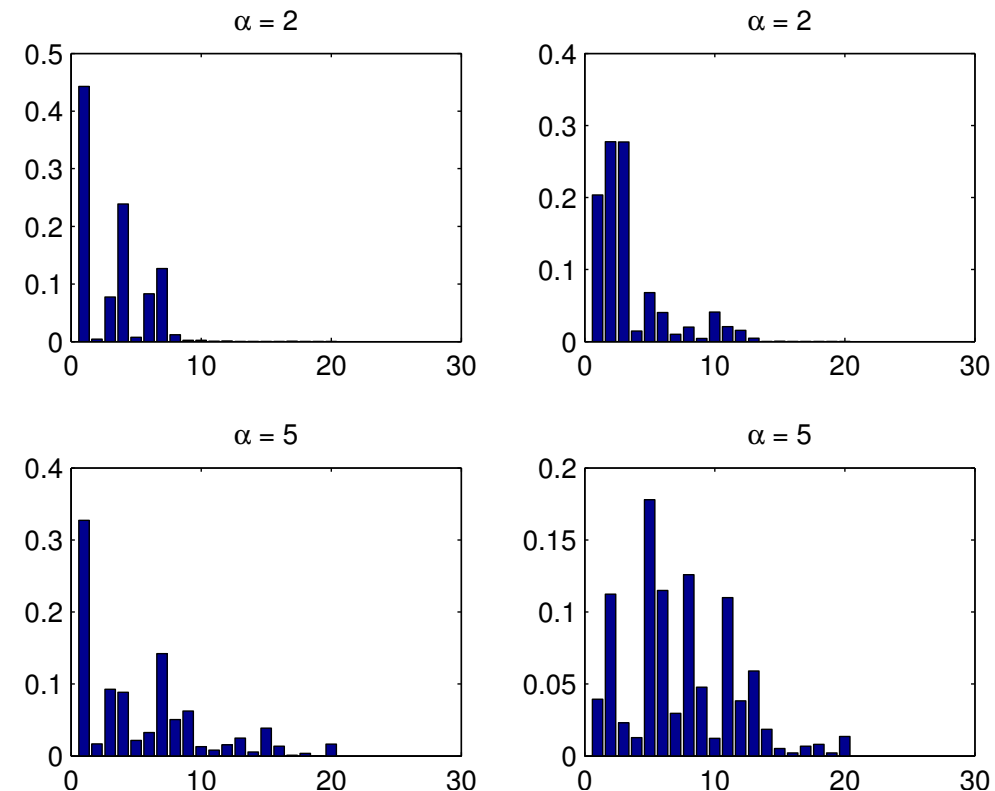
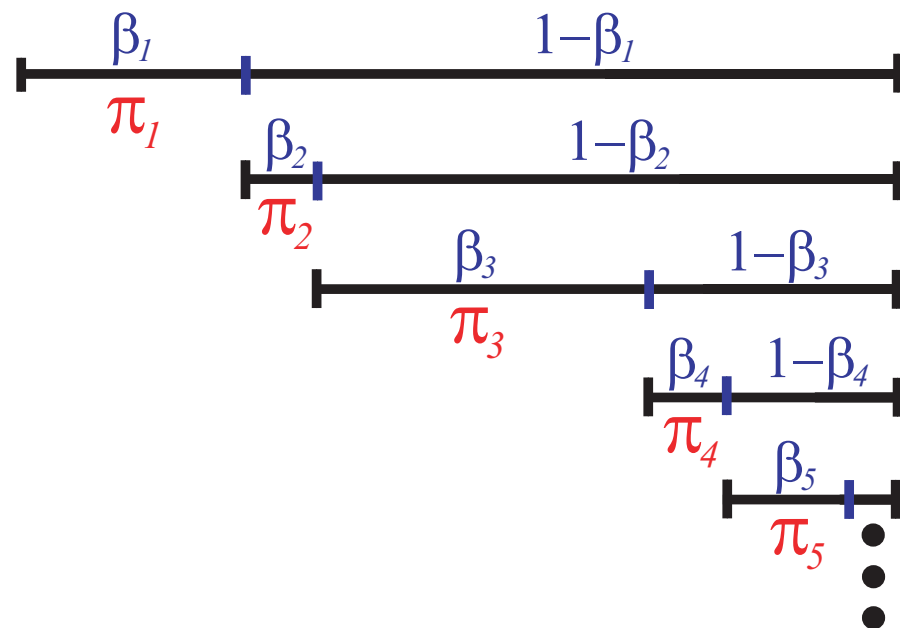


# Construction of a Dirichlet Process

- The Dirichlet process is only defined **implicitly**, i.e. we can test whether a given probability measure is sampled from a DP, but we can not yet construct one.
- A DP can be constructed using the “stick-breaking” analogy:
  - imagine a stick of length 1
  - we select a random number  $\beta$  between 0 and 1 from a Beta-distribution
  - we break the stick at  $\pi = \beta * \text{length-of-stick}$
  - we repeat this infinitely often



# The Stick-Breaking Construction



- formally, we have

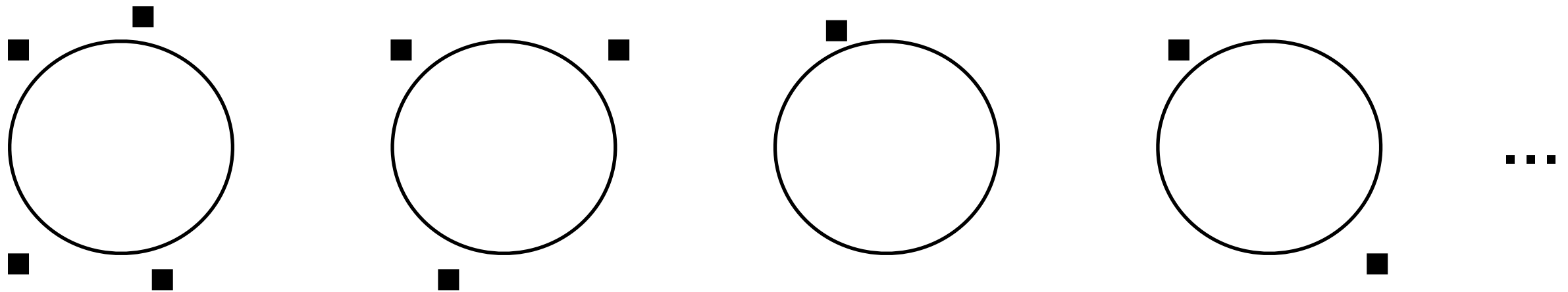
$$\beta_k \sim \text{Beta}(1, \alpha) \quad \pi_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) = \beta_k \left(1 - \sum_{l=1}^{k-1} \pi_l\right)$$

- now we define

$$G(\boldsymbol{\theta}) = \sum_{k=1}^{\infty} \pi_k \delta(\boldsymbol{\theta}_k, \boldsymbol{\theta}) \quad \boldsymbol{\theta}_k \sim H \quad \text{then: } G \sim \text{DP}(\alpha, H)$$



# The Chinese Restaurant Process



- Consider a restaurant with infinitely many tables
- Everytime a new customer comes in, he sits at an **occupied table** with probability **proportional to the number of people** sitting at that table, but he may choose to sit on a **new** table with **decreasing** probability as more customers enter the room.



# The Chinese Restaurant Process

- It can be shown that the probability for a new customer is

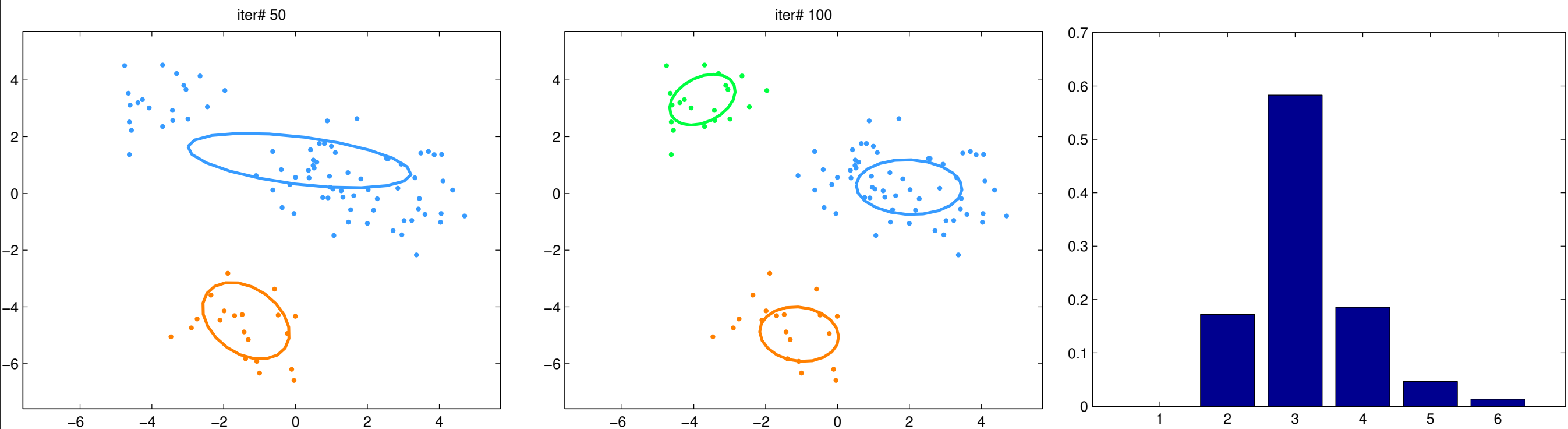
$$p(\bar{\boldsymbol{\theta}}_{N+1} = \boldsymbol{\theta} \mid \bar{\boldsymbol{\theta}}_{1:N}, \alpha, H) = \frac{1}{\alpha + N} \left( \alpha H(\boldsymbol{\theta}) + \sum_{k=1}^K N_k \delta(\bar{\boldsymbol{\theta}}_k, \boldsymbol{\theta}) \right)$$

- This means that currently occupied tables are more likely to get new customers (**rich get richer**)
- The number of occupied tables grows logarithmically with the number of customers



# The DP for Mixture Modeling

- Using the stick-breaking construction, we see that we can extend the mixture model clustering to the situation where  $K$  goes to infinity
- The algorithm can be implemented using Gibbs sampling



# Affinity Propagation

- Often, we are only given a **similarity matrix** for the data points
- The idea of Affinity Propagation 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  $i$

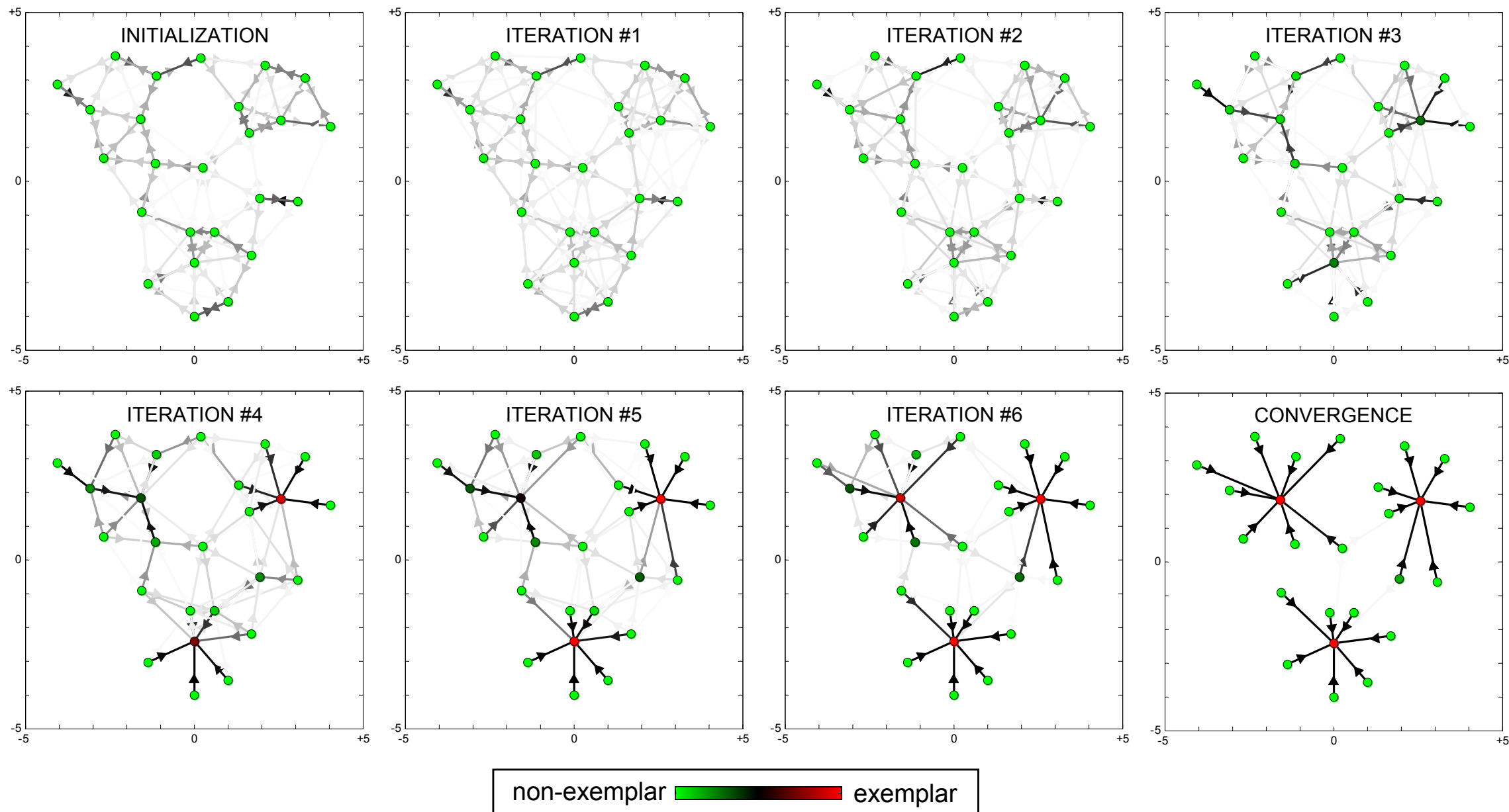


# Affinity Propagation

- 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

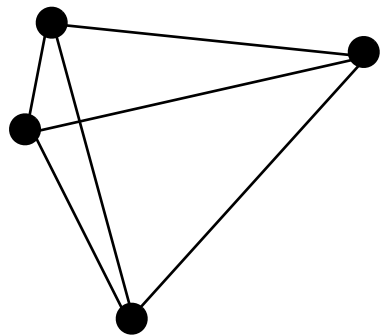


- Colours: how much each point wants to be an exemplar
- Edge strengths: how much a point wants to belong to a cluster



# 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



$W =$


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

$D =$

$d_1$			
	$d_2$		
		$d_3$	
			$d_4$



# 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 eigenvalue 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  $\mathbf{1}$  vector is eigenvector with eigenvalue 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.



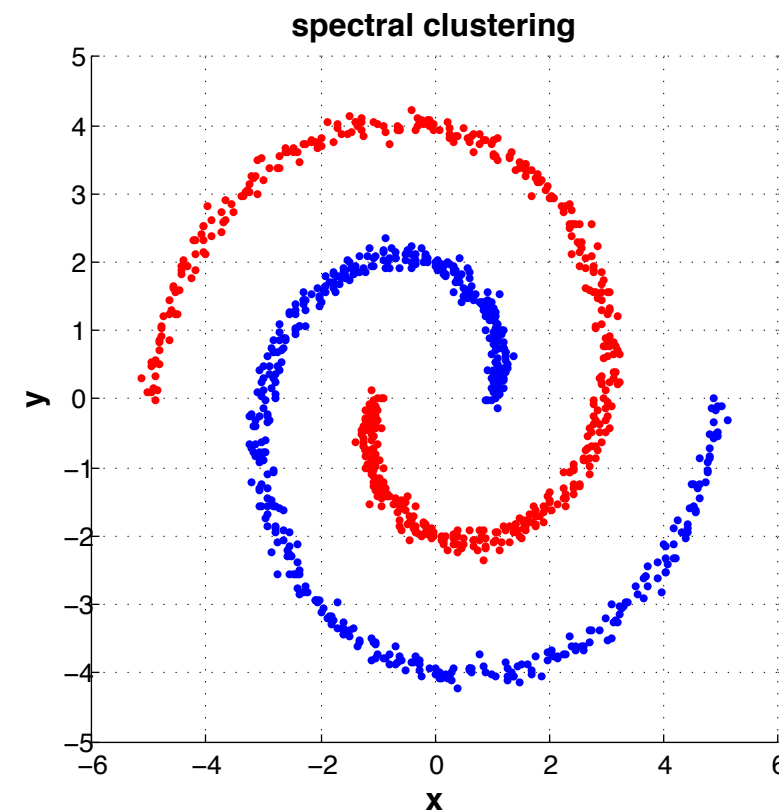
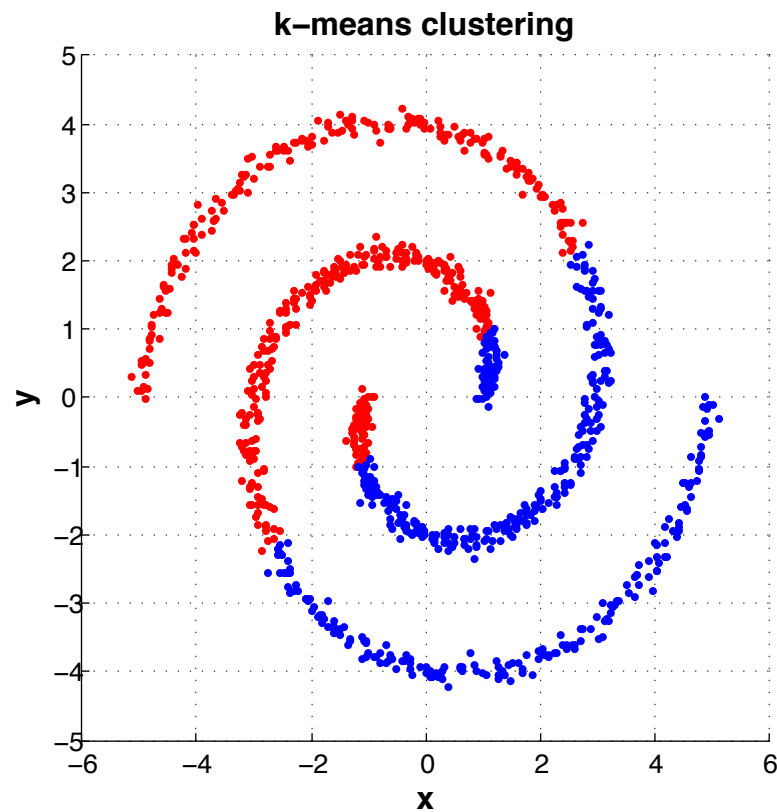
# The Algorithm

- Input: Similarity matrix  $W$
- Compute  $L = D - W$
- Compute the eigenvectors that correspond to the  $K$  smallest eigenvalues
- Stack these vectors as rows 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

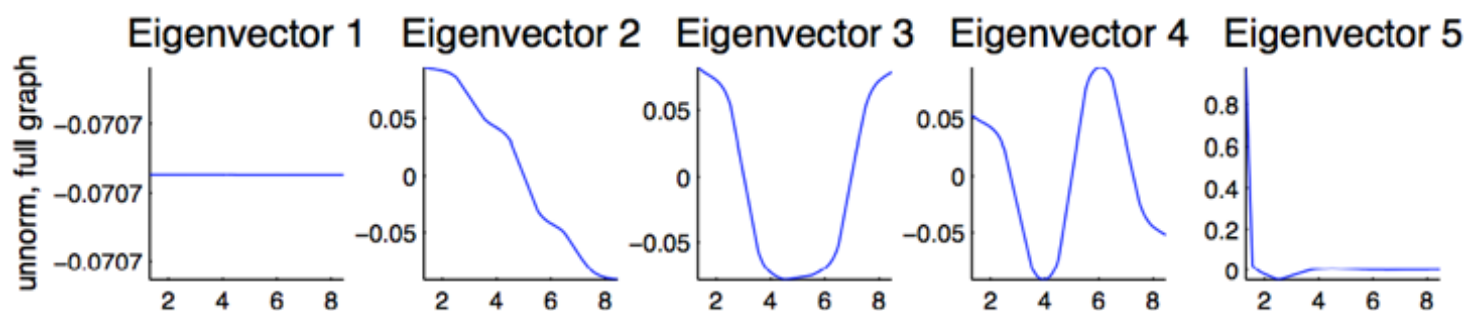
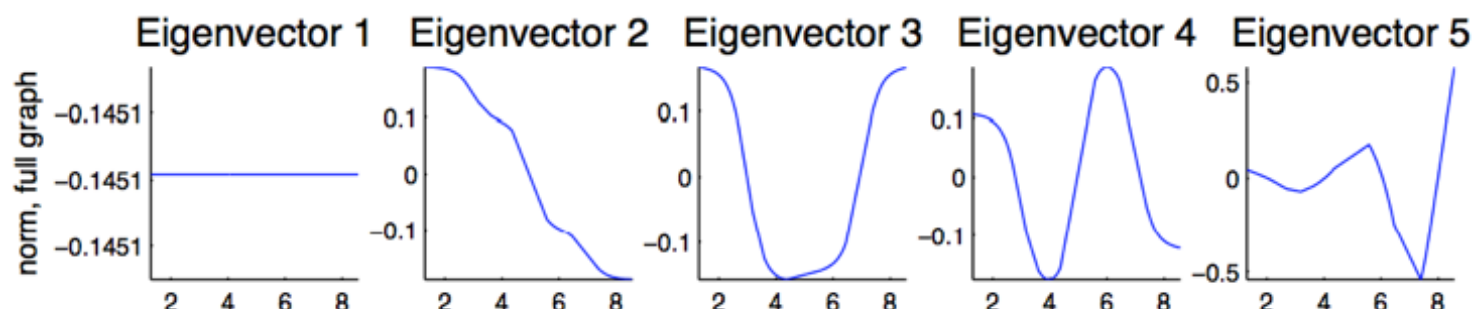
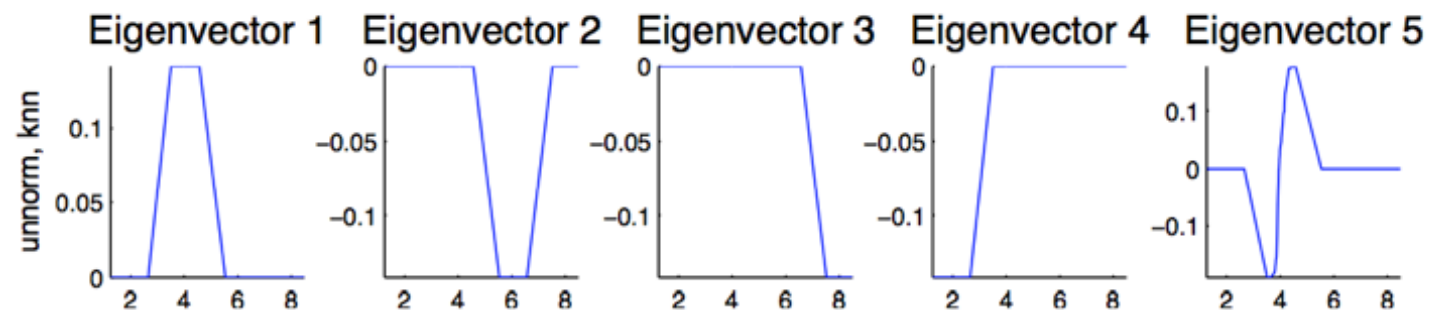
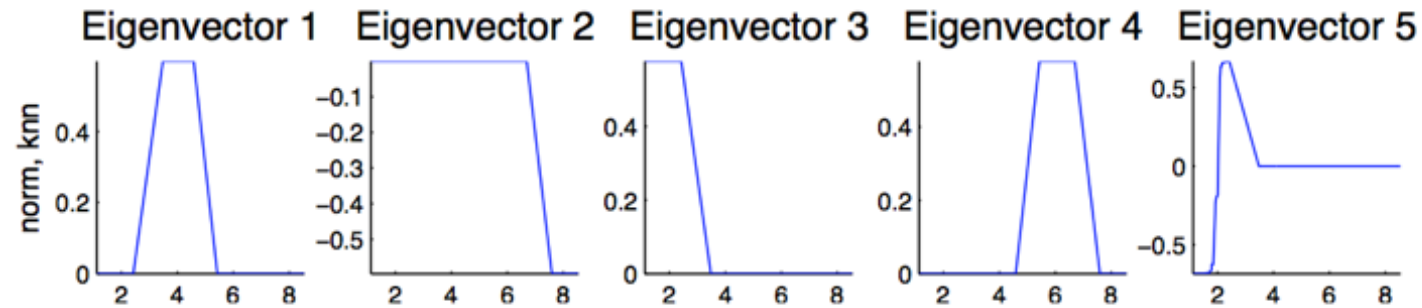
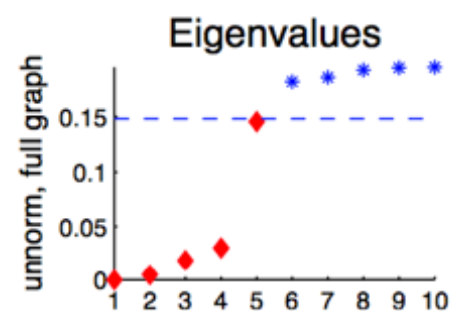
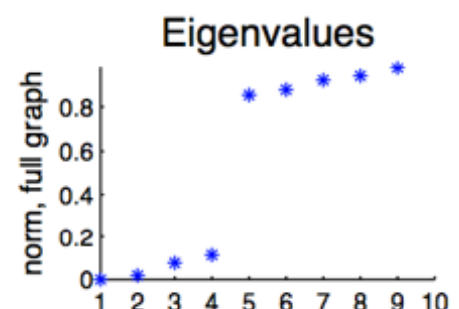
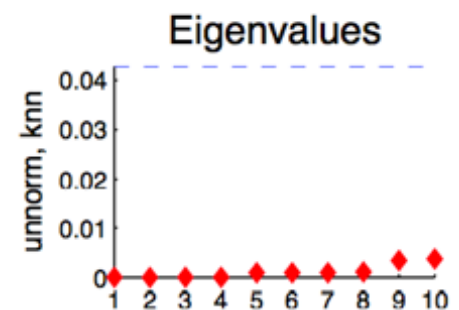
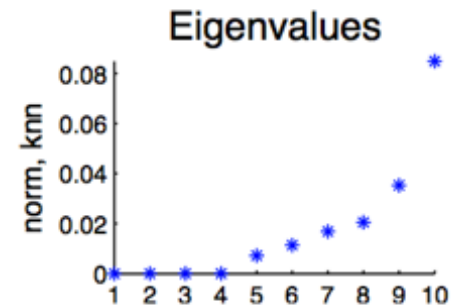
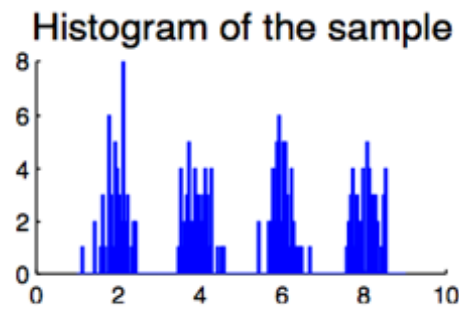


# Further Remarks

- 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



# Another Small Example



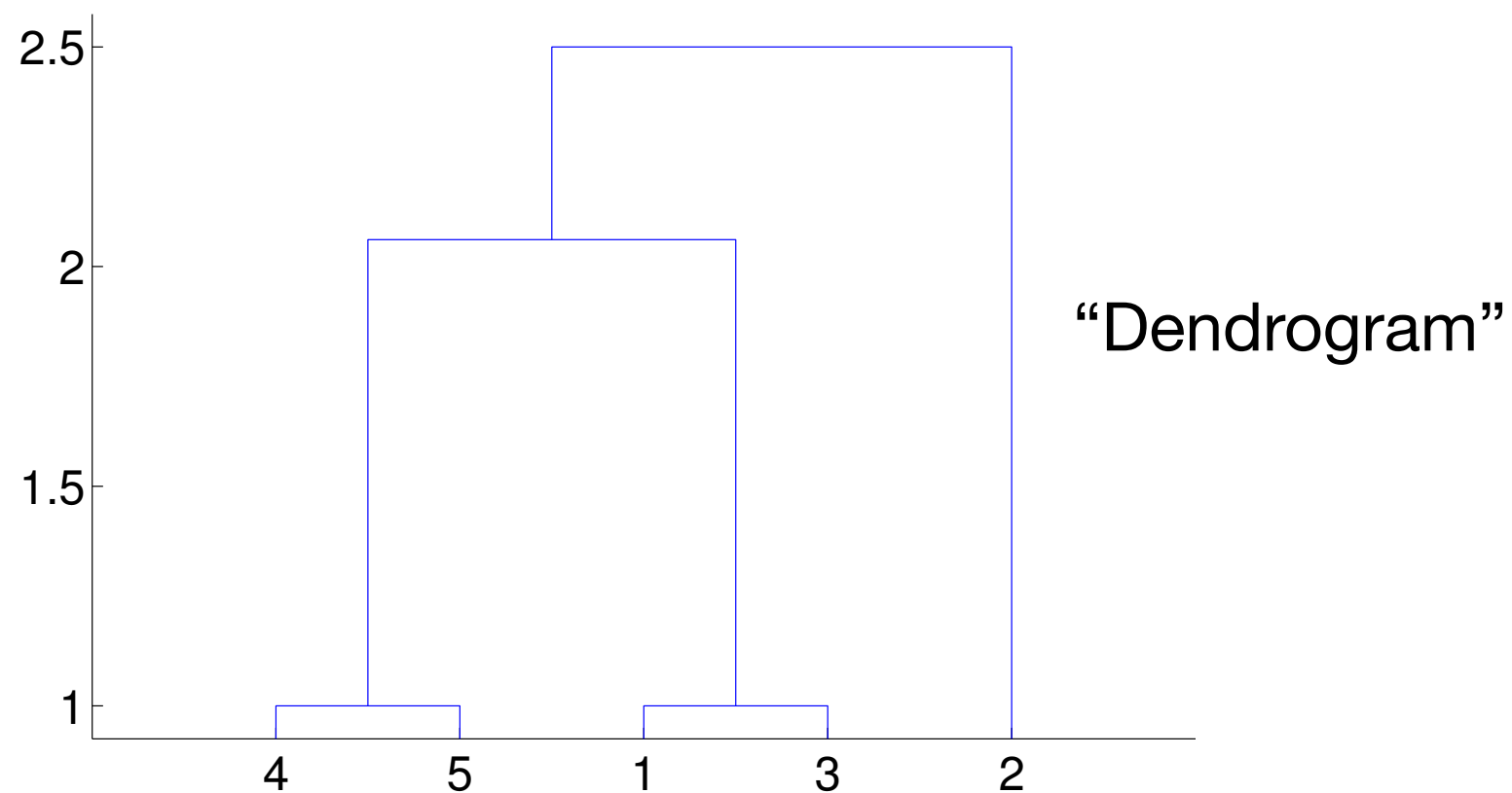
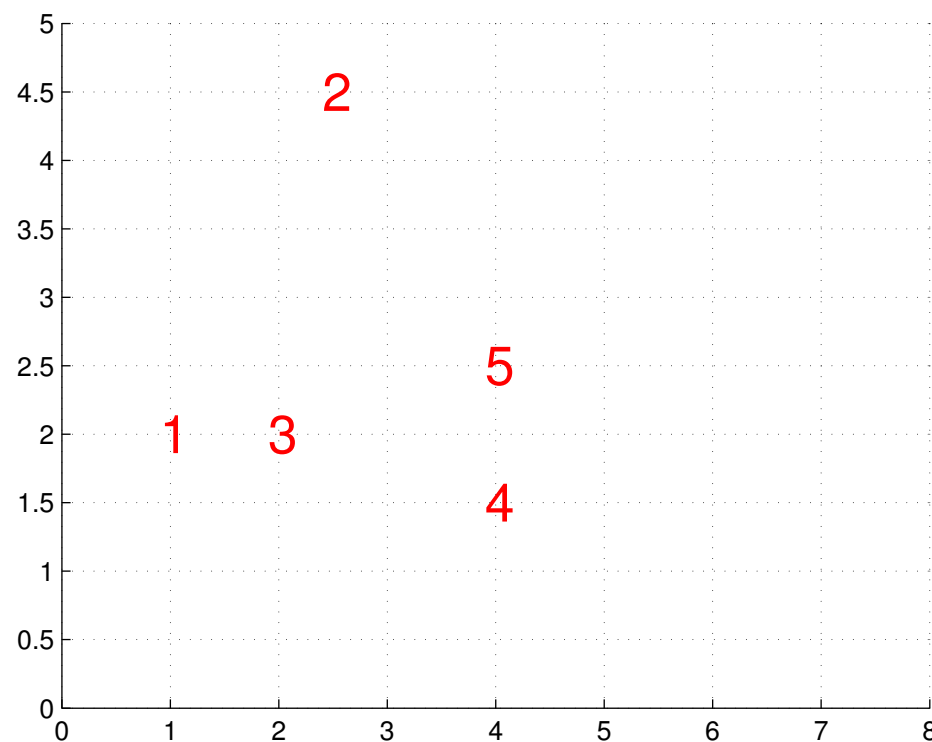
# Hierarchical Clustering

- 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



# Linkage

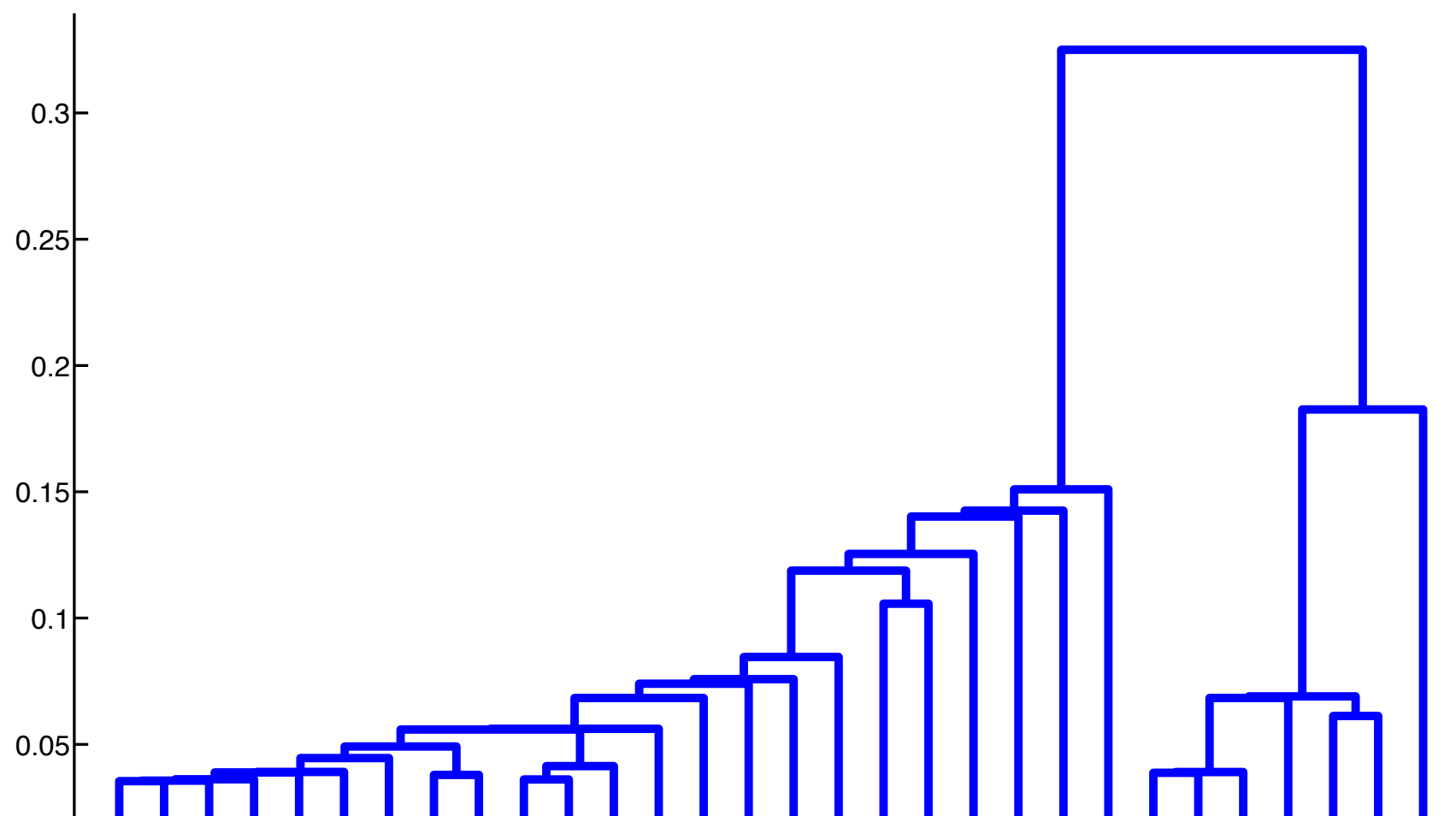
- In agglomerative clustering, it is important to define a distance measure between two clusters
- There are three different methods:
  - Single linkage: considers the two closest elements from both clusters and uses their distance
  - Complete linkage: considers the two farthest elements from both clusters
  - 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

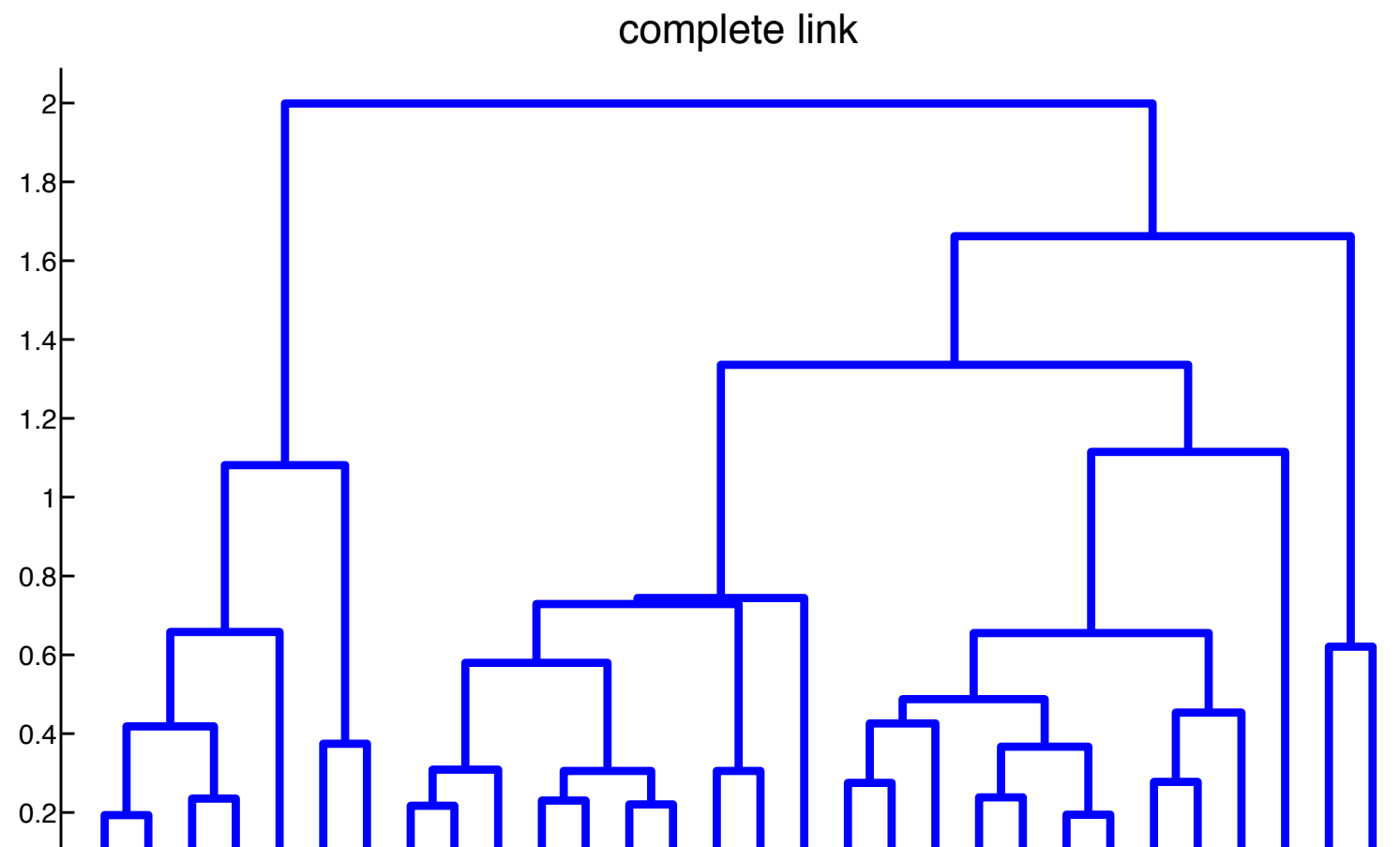
- The distance is based on  $d_{SL}(G, H) = \min_{i \in G, i' \in H} d_{i, i'}$
- The resulting dendrogram is a minimum spanning tree, i.e. it minimizes the sum of the edge weights
- Thus: we can compute the clustering in  $O(N^2)$  time

single link



# Complete Linkage

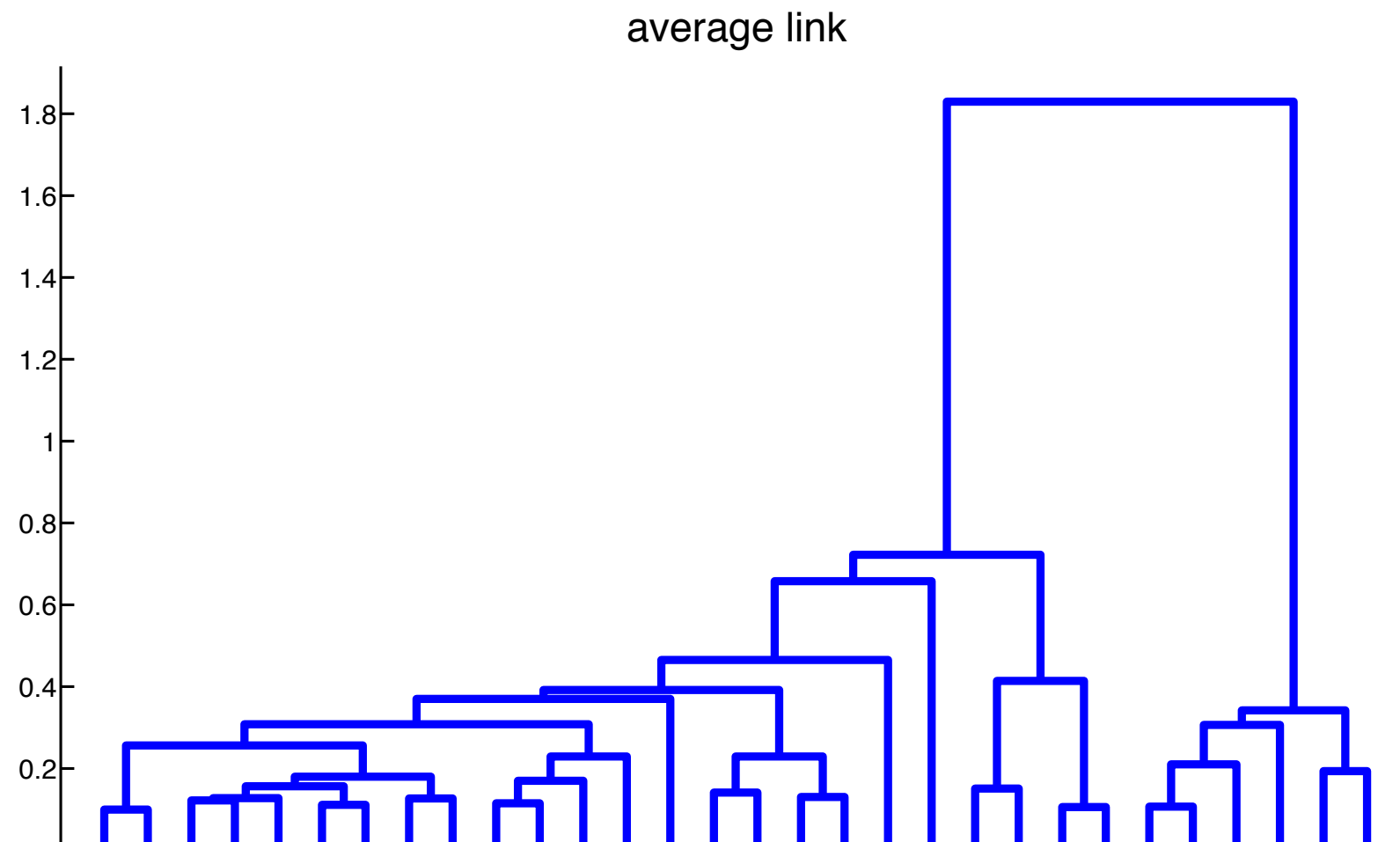
- The distance is based on  $d_{CL}(G, H) = \max_{i \in G, i' \in H} d_{i, i'}$
- Complete linkage fulfills the **compactness property**, i.e. all points in a group should be similar to each other
- Tends to produce clusters with smaller diameter





# 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)



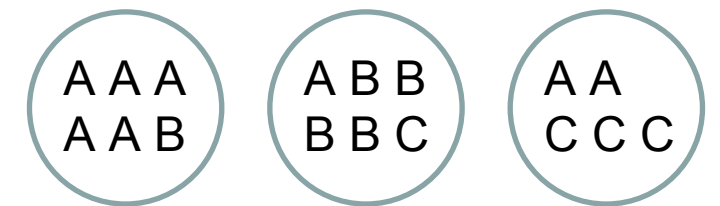
# Evaluation of Clustering Algorithms

- 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  $i$
- $p_{ij} = \frac{N_{ij}}{N_i}$   $p_i = \max_j p_{ij}$  “Purity”
- overall purity  $\sum_i \frac{N_i}{N} p_i$
- Purity ranges from 0 (bad) to 1 (good)
- But: a clustering with each object in its own cluster has a purity of 1



Purity = 0.71



# 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  $u_i$   $p_U(i) = \frac{|u_i|}{N}$

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

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



# Summary

- Several Clustering methods:
  - Dirichlet process mixture model does not require the number of clusters to be known; full Bayesian
  - Affinity Propagation: iterative approach where exemplars are determined as cluster centers
  - Spectral clustering uses the graph Laplacian and performs an eigenvector analysis
  - Hierarchical approaches can be bottom-up or top-down
- Evaluation methods for Clustering are hard to find
- Some are based on purity or mutual information

