Advanced course in machine learning
582744
Lecture 4

Arto Klami

Outline

Clustering, basics

Hierarchical clustering and k-means (Sections 25.5 and 11.4.2.5)

Mixture models (Section 11)

Spectral clustering (Section 25.4)

Unsupervised learning

No distinction between inputs and outputs; we have some data $\mathbf{x}_n$ and want to understand the process generating them

Often solved with latent-variable models: Some unknown quantity for each data point becomes the new representation for that data

- Clustering: Data points represented by cluster index, similar points grouped together
- Dimensionality reduction: Data points represented by lower-dimensional vectors (linear or non-linear mapping)
- Missing value imputation: Estimate data elements that are unknown or censored

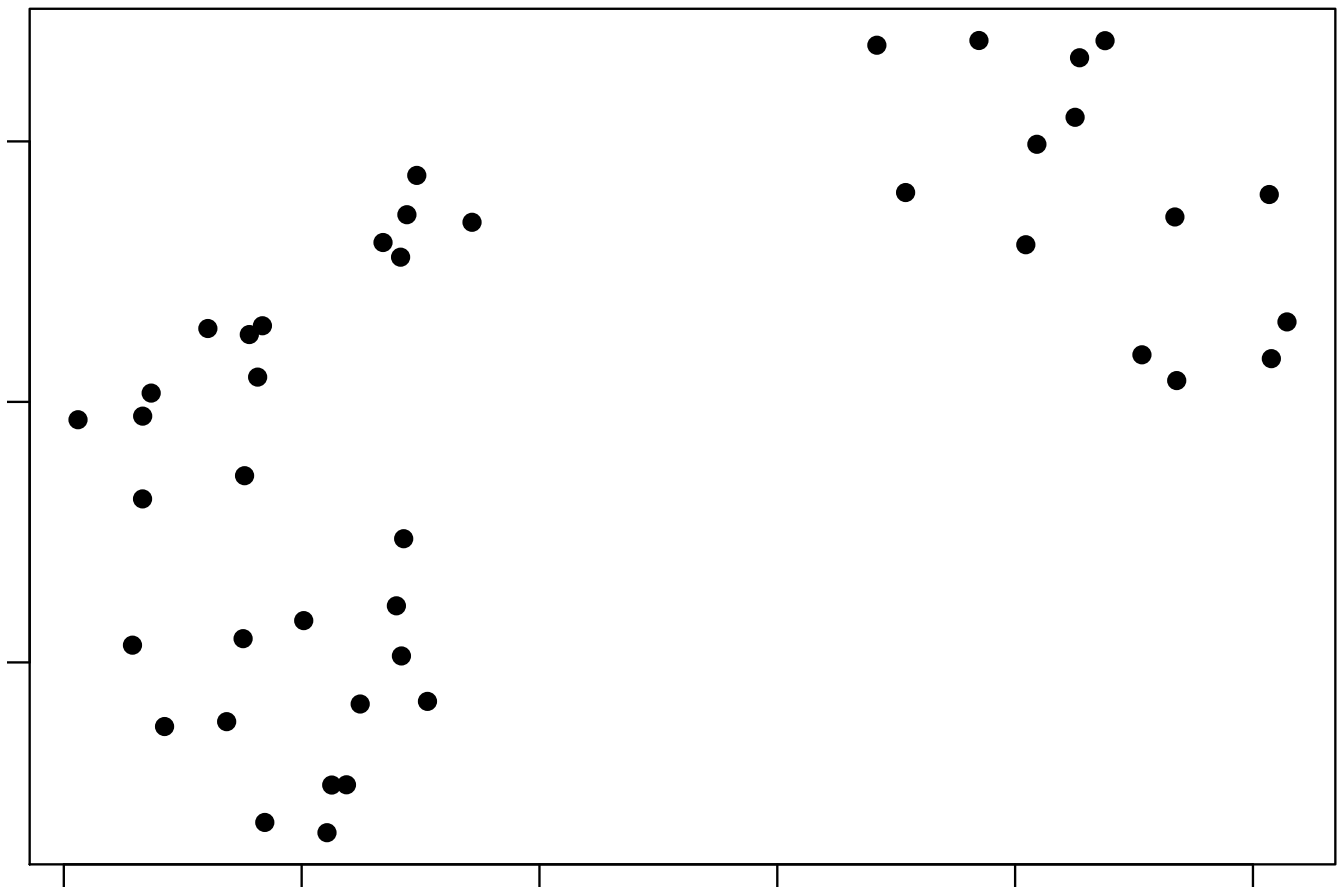
Clustering

Find groups of samples that are in some sense similar

Often: Distance between samples in a cluster is small, distance between samples in different clusters is big

Or: A cluster is maximally homogeneous

How many clusters?



Axiomatic formulation

A clustering function $f(d)$ that takes as input a distance function d and outputs a partitioning should satisfy

- *Scale-invariance*: Multiplying all distances with a scalar α should not change the result: $f(d) = f(\alpha d)$
- *Consistency*: Making within-cluster distances smaller and between-cluster distances bigger should not change the result: $f(d) = f(\tilde{d})$
- *Richness*: All possible data partitionings can be obtained with some distance d

Axiomatic formulation

A clustering function $f(d)$ that takes as input a distance function d and outputs a partitioning should satisfy

- *Scale-invariance*: Multiplying all distances with a scalar α should not change the result: $f(d) = f(\alpha d)$
- *Consistency*: Making within-cluster distances smaller and between-cluster distances bigger should not change the result: $f(d) = f(\tilde{d})$
- *Richness*: All possible data partitionings can be obtained with some distance d

Kleinberg's impossibility theorem for clustering: No function can satisfy all three (see `http://alexhwilliams.info/itsneuronalblog/2015/10/01/clustering2/` for some intuitions)

Axiomatic formulation

A clustering function $f(d)$ that takes as input a distance function d and outputs a partitioning should satisfy

- *Scale-invariance*: Multiplying all distances with a scalar α should not change the result: $f(d) = f(\alpha d)$
- *Consistency*: Making within-cluster distances smaller and between-cluster distances bigger should not change the result: $f(d) = f(\tilde{d})$
- *Richness*: All possible data partitionings can be obtained with some distance d

Kleinberg's impossibility theorem for clustering: No function can satisfy all three (see `http://alexhwilliams.info/itsneuronalblog/2015/10/01/clustering2/` for some intuitions)

Not as bad as it sounds: For pre-determined K they can be satisfied, and consistency is quite strong requirement anyway

Similarity-based vs feature-based

Two kinds of inputs:

- Feature-based: We are given samples $x \in \mathbb{R}^D$
- Similarity-based: We are only given pairwise dissimilarities $d(i, j) \geq 0$ (that might not satisfy triangle-inequality)

We can always compute distances based on features, but not the other way around

Partitional vs hierarchical

Partitional clustering methods divide the data space into K disjoint areas

Hierarchical clustering methods do that in a nested fashion, creating a tree of partitions

Evaluation

Evaluation of clustering models is hard

- Probabilistic models: The generative likelihood on test data
- Others: Bootstrap and other re-sampling techniques

Evaluation

Evaluation of clustering models is hard

- Probabilistic models: The generative likelihood on test data
- Others: Bootstrap and other re-sampling techniques
- Comparison against known labels or another clustering result: Purity, rand index, mutual information (Section 25.1.2)
- ...but that does not help in real clustering tasks, unless we know some samples that should be in the same/different cluster

Evaluation

Evaluation of clustering models is hard

- Probabilistic models: The generative likelihood on test data
- Others: Bootstrap and other re-sampling techniques
- Comparison against known labels or another clustering result: Purity, rand index, mutual information (Section 25.1.2)
- ...but that does not help in real clustering tasks, unless we know some samples that should be in the same/different cluster
- Silhouettes (Intro): Measures the quality based on distances, but if we want this then we could optimize directlu for this criterion

Evaluation

Evaluation of clustering models is hard

- Probabilistic models: The generative likelihood on test data
- Others: Bootstrap and other re-sampling techniques
- Comparison against known labels or another clustering result: Purity, rand index, mutual information (Section 25.1.2)
- ...but that does not help in real clustering tasks, unless we know some samples that should be in the same/different cluster
- Silhouettes (Intro): Measures the quality based on distances, but if we want this then we could optimize directlu for this criterion
- In practice we often need some external criteria

Evaluation

Evaluation of clustering models is hard

- Probabilistic models: The generative likelihood on test data
- Others: Bootstrap and other re-sampling techniques
- Comparison against known labels or another clustering result: Purity, rand index, mutual information (Section 25.1.2)
- ...but that does not help in real clustering tasks, unless we know some samples that should be in the same/different cluster
- Silhouettes (Intro): Measures the quality based on distances, but if we want this then we could optimize directlu for this criterion
- In practice we often need some external criteria

The same issue applies to choosing the number of clusters, since that is based on comparing models with different K

Practical algorithms

- K-means (Intro)
- Hierarchical clustering (Intro)
- Mixture models
- Spectral clustering

K-means

Minimize $\sum_n (\mathbf{x}_n - \boldsymbol{\mu}_{z_n})^2$, where z_n is the cluster index of $\mathbf{x}_n$

Alternate between:

- $z_n = \arg\min_k (\mathbf{x}_n - \boldsymbol{\mu}_k)^2$
- $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{z_n=k} \mathbf{x}_n$

Good practical initialization (K-means++):

- Pick one data sample at random as the first center
- For each data point compute the smallest distance $d(\mathbf{x}_n)$ from that point to any of the centers
- Pick the next center at random with probabilities $p \propto d(\mathbf{x}_n)^2$

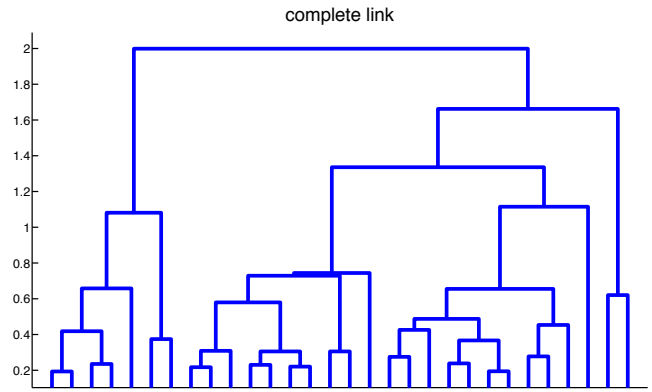
Hierarchical clustering

Construct a tree by always merging two most similar clusters using one of three criteria (agglomerative clustering):

- Single link: $d_C = \min d_{i,j}$
- Complete link: $d_C = \max d_{i,j}$
- Average link: $d_C = \frac{1}{N_i N_j} \sum d_{i,j}$

...or by splitting clusters (divisive clustering), starting from one big one

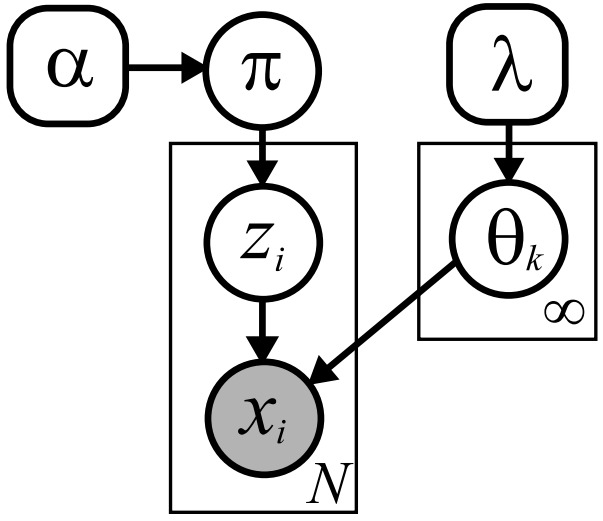
Produces a *dendrogram*, needs to be cutted to provide a partitioning – but sometimes the tree is actually what we want



Probabilistic clustering models

Generative models have very natural solution for clustering problems: Each cluster generates a collection of data points

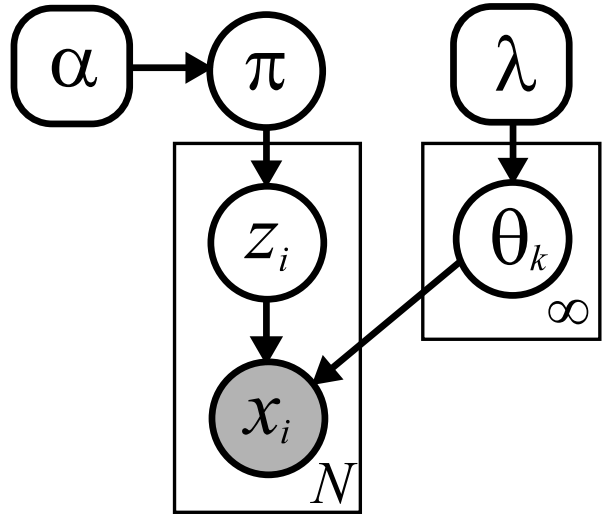
$$p(z_n = k) = \pi_k$$
$$p(\pi_k) = \text{Dirichlet}(\alpha_k)$$
$$p(\mathbf{x}_n | z_n = k, \theta) = p_k(\mathbf{x}_n | \theta_k)$$
$$p(\boldsymbol{\theta}_k | \lambda) = p(\boldsymbol{\theta}_k)$$



Probabilistic clustering models

Generative models have very natural solution for clustering problems: Each cluster generates a collection of data points

$$
\begin{aligned}
p(z_n = k) &= \pi_k \\
p(\pi_k) &= \text{Dirichlet}(\alpha_k) \\
p(\mathbf{x}_n | z_n = k, \theta) &= p_k(\mathbf{x}_n | \theta_k) \\
p(\boldsymbol{\theta}_k | \lambda) &= p(\boldsymbol{\theta}_k)
\end{aligned}
$$

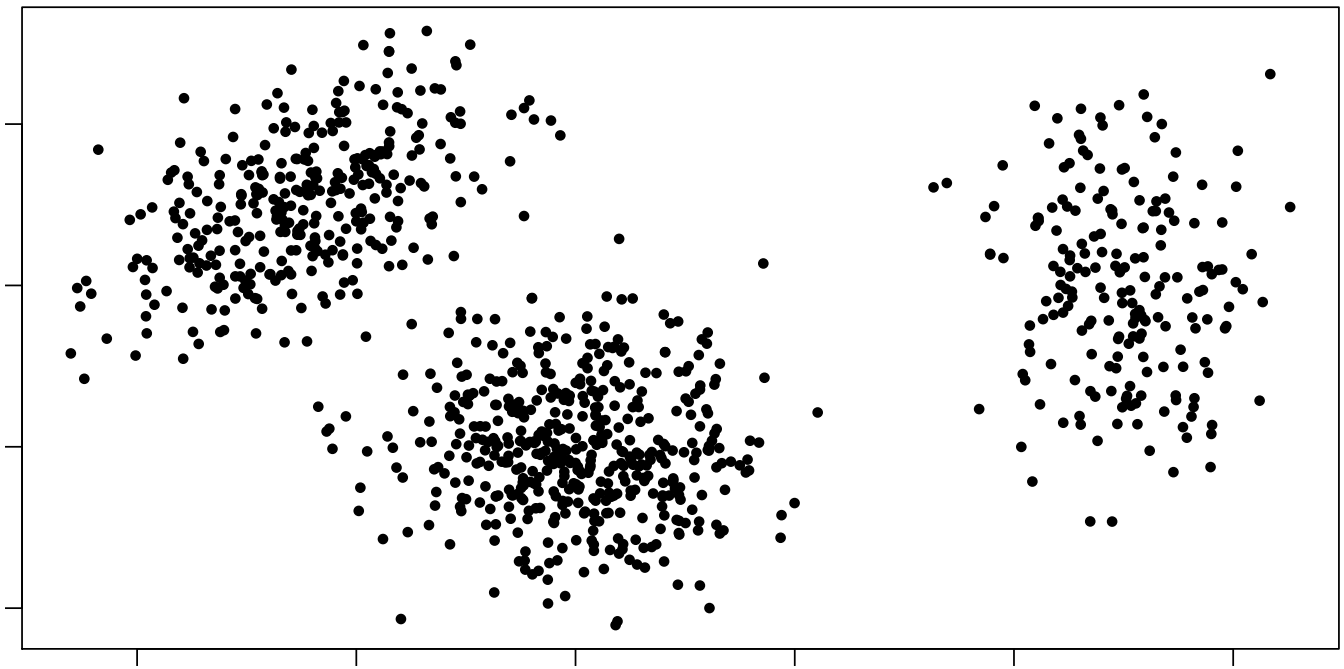


These are called *mixture models*, since the marginal density is

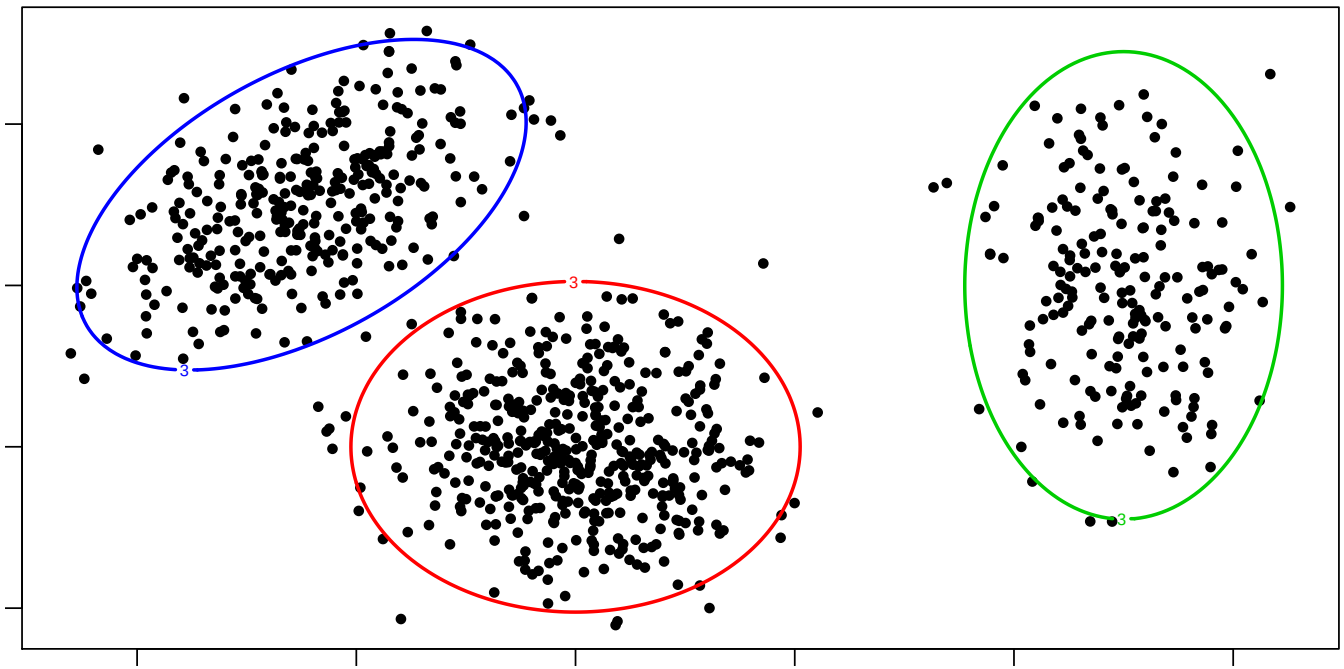
$$
p(\mathbf{x}_n | \boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}_n | \boldsymbol{\theta}_k),
$$

a convex combination of the base distributions p_k

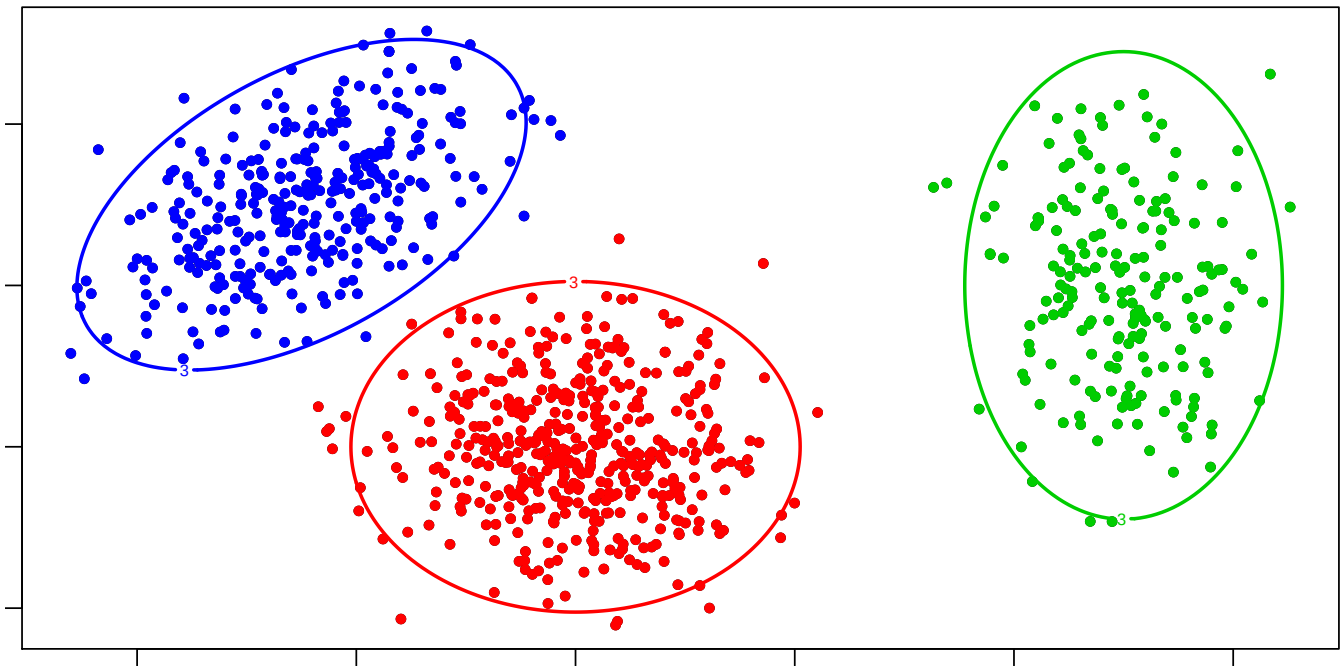
Mixture model example



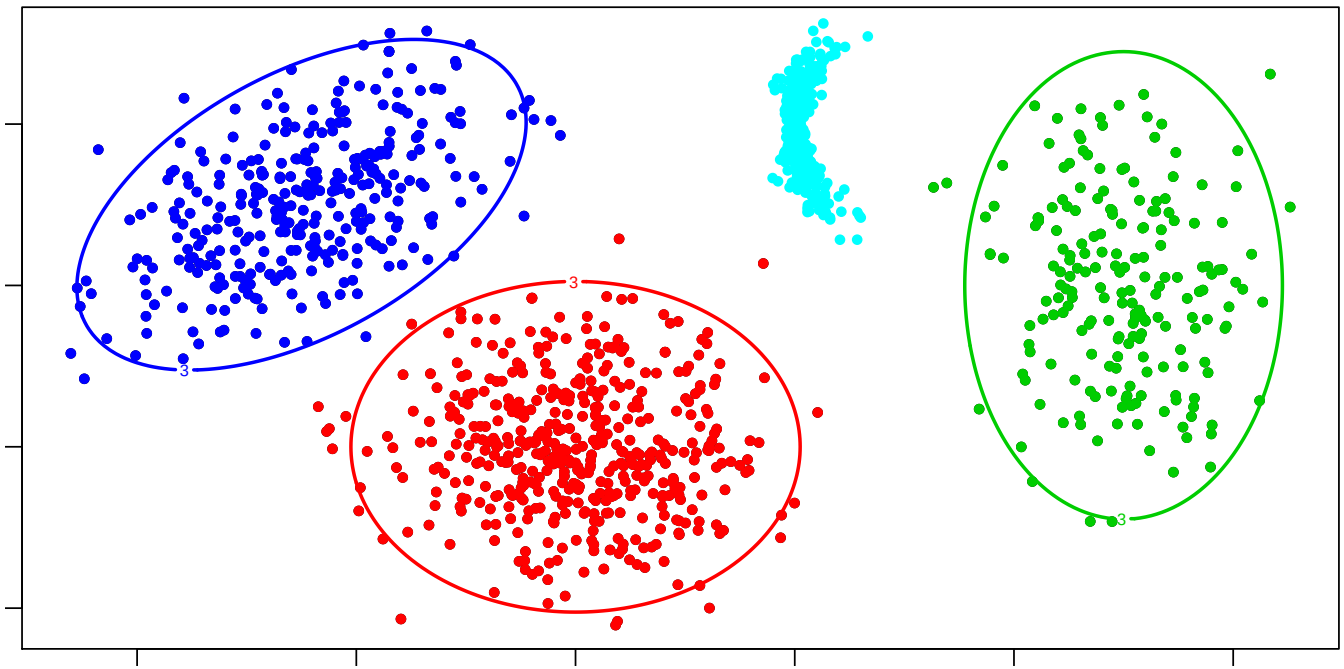
Mixture model example



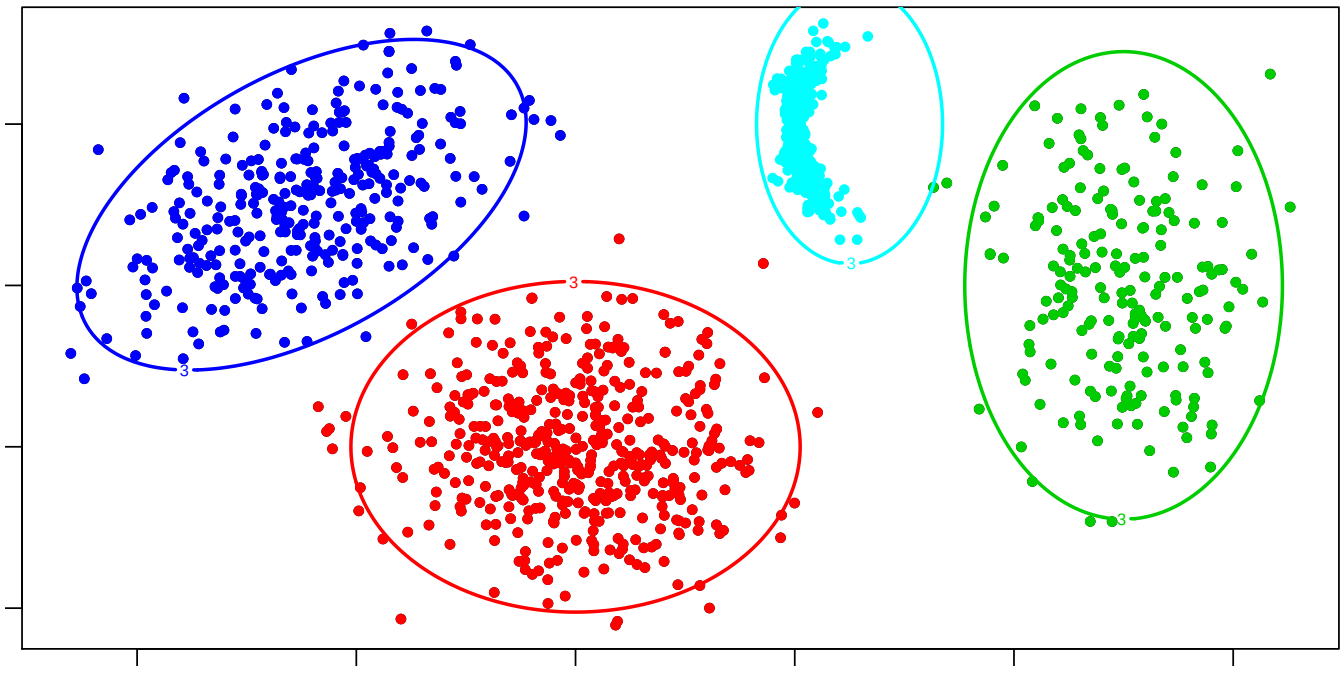
Mixture model example



Mixture model example



Mixture model example



Using mixture models for clustering

The joint pdf is $p(\mathbf{x}_n, z_n, \boldsymbol{\theta}, \boldsymbol{\pi}) = p(\mathbf{x}_n|z_n, \boldsymbol{\theta})p(z_n|\boldsymbol{\pi})$

If we know the parameters, then Bayes' rule gives

$$p(z_n = k|\mathbf{x}_n, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{p(\mathbf{x}_n|z_n = k, \theta_k)p(z_n = k|\boldsymbol{\pi})}{\sum_j p(\mathbf{x}_n|z_n = j, \theta_j)p(z_n = j|\boldsymbol{\pi})}$$

Hence, we get soft cluster assignments in form of a distribution

Inference

We could optimize $\log \prod_n p(\mathbf{x}_n, z_n|\boldsymbol{\theta}, \boldsymbol{\pi})$ directly, for example by alternating between gradient-updates for z, $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$

...but as argued before, marginalizing z_n out would be better, resulting in $\log p(D|\boldsymbol{\theta}) = \sum_n \log[\sum_z p(z_n, \mathbf{x}_n|\boldsymbol{\theta})]$. This is called *observed data log-likelihood*

This loss is not convex. Gradient-based optimization is still possible, but somewhat tricky due to the constraints (see exercise 11.5 in the book)

Expectation maximization

The *complete data likelihood* for one data point can be re-written in a product form

$$p(\mathbf{x}_n, z_n) = \prod_k \left[p(\mathbf{x}_n|z_n = k)p(z_n = k)\right]^{\mathbb{I}(z_n=k)},$$

which means the whole log-likelihood becomes

$$L = \sum_n \sum_k p(z_n = k|\mathbf{x}_n, \boldsymbol{\theta}) \log(\pi_k p_k(\mathbf{x}_n|z_n, \boldsymbol{\theta}_k))$$

If we know the parameters $\hat{\boldsymbol{\theta}}$ then we can compute the *expected complete data log-likelihood*

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \sum_n \sum_k \mathbb{E}[p(z_n = k|\mathbf{x}_n, \hat{\boldsymbol{\theta}})] \log(\pi_k p_k(\mathbf{x}_n|z_n, \boldsymbol{\theta}_k)),$$

which is still a function of $\boldsymbol{\theta}$

Expectation maximization

EM is an alternating algorithm that:

- Computes the expected cluster assignments $\mathbb{E}[p(z_n = k|\mathbf{x}_n, \boldsymbol{\theta}^t)]$ for the current values of parameters $\boldsymbol{\theta}^t$
- Maximizes (or at least improves) the expected likelihood $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$: $\boldsymbol{\theta}^{t+1} = \arg\max Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$

...and iterates the above two steps until convergence

Applicable to all mixture models, and the expectation step is actually identical for all models

Formal treatment

Assume some arbitrary distribution $q(z_n)$ over the latent variables and re-write the observed data log-likelihood as

$$\sum_n \log \left[\sum_k p(\mathbf{x}_n, z_n) \right] = \sum_n \log \left[\sum_k q(z_n) \frac{p(\mathbf{x}_n, z_n)}{q(z_n)} \right]$$

Then create a *lower bound* for the log-likelihood using Jensen's inquality

$$\begin{aligned} Q(\boldsymbol{\theta}, q) &= \sum_n \sum_k q(z_n) \log \frac{p(\mathbf{x}_n, z_n)}{q(z_n)} \\ &= \mathbb{E}_{q(z_n)}[\log p(\mathbf{x}_n, z_n)] + \mathbb{H}(q(z_n)) \leq l(\boldsymbol{\theta}) \quad \forall \boldsymbol{\theta} \end{aligned}$$

This holds for all $q(z_n)$, so let's choose one that makes the bound as tight as possible

Formal treatment

Re-write the lower bound for one sample as

$$\sum_k q(z_n) \log \frac{p(\mathbf{x}_n, z_n)}{q(z_n)} = \sum_k q(z_n) \log \frac{p(z_n|\mathbf{x}_n)}{q(z_n)} + \sum_k q(z_n) \log p(\mathbf{x}_n)$$
$$= \mathrm{KL}(q(z_n), p(z_n|\mathbf{x}_n)) + \log p(\mathbf{x}_n)$$

The latter term is constant wrt $q(z_n)$ and the former term is zero if $q(z_n) = p(z_n|\mathbf{x}_n)$

Conditional on $\boldsymbol{\theta}^t$ the optimal choice is hence the posterior distribution of z_n, and for that choice the lower bound is identical for the true observed data likelihood

Formal treatment

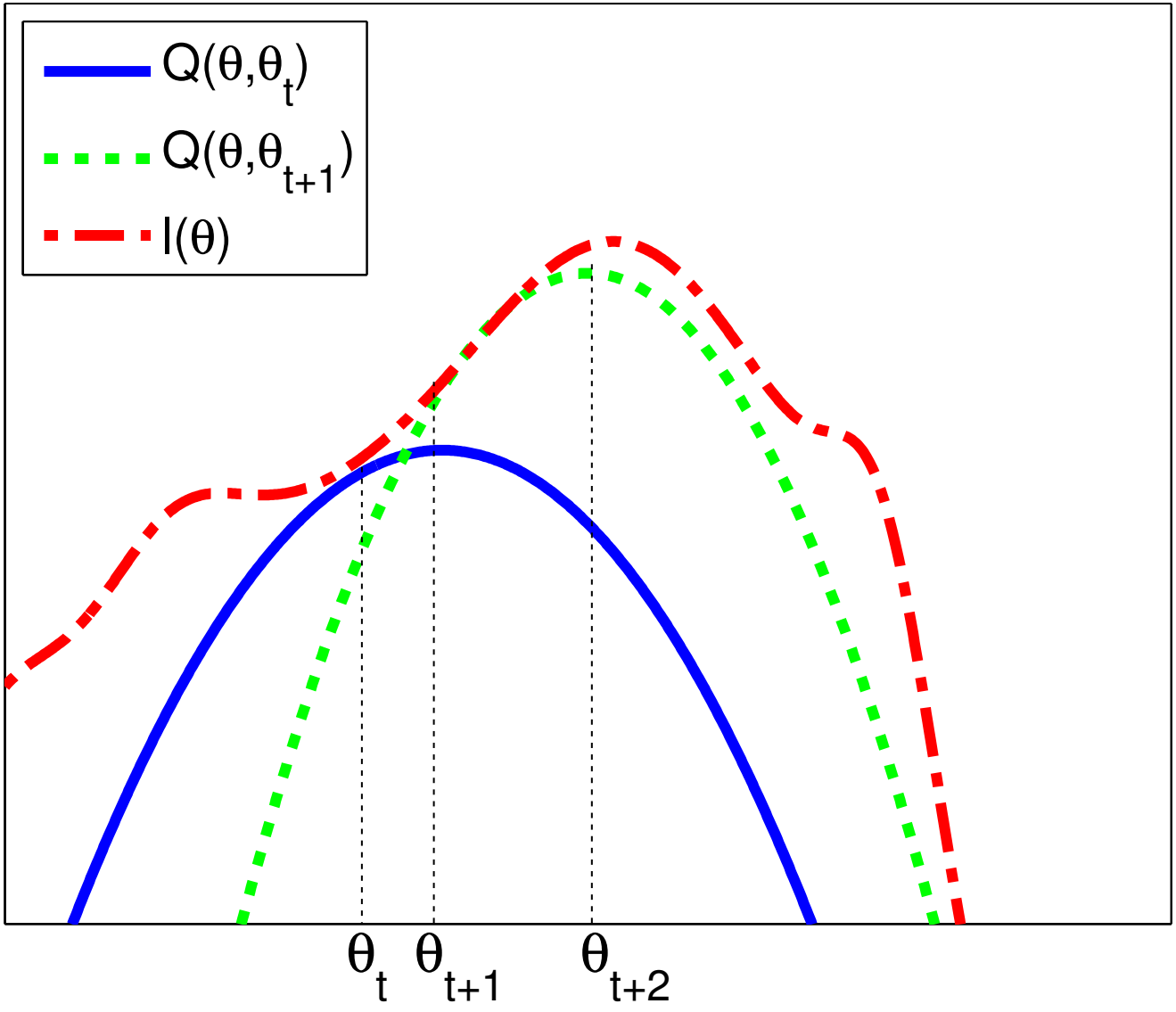
Now that we know what $q(z_n)$ is, we plug it in to the lower bound

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \mathbb{E}_{p(z_n|\mathbf{x}_n, \boldsymbol{\theta}^t)}[\log p(\mathbf{x}_n, z_n)] + \mathbb{H}(q(z_n))$$

and choose new $\boldsymbol{\theta}^{t+1}$ to maximize the bound (note that the entropy term is here constant)

Since the bound exactly matched the true likelihood at $\boldsymbol{\theta}^t$ and we know that it is a lower bound, we are guaranteed to increase also the true likelihood if we manage to improve $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$

Formal treatment



Mixture of Gaussians

Mixture of Gaussians is the most common mixture model with $p_k(\mathbf{x}_n|\boldsymbol{\theta}_k) = N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

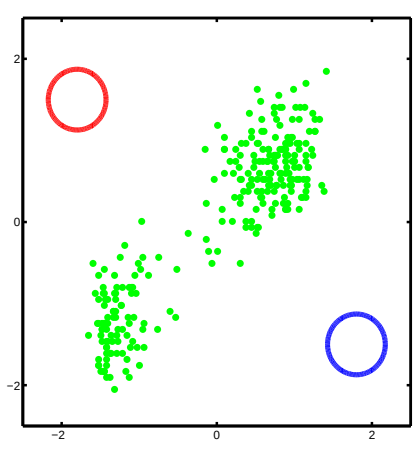
Full derivation shown in Section 11.4.2, but it is easy anyway:

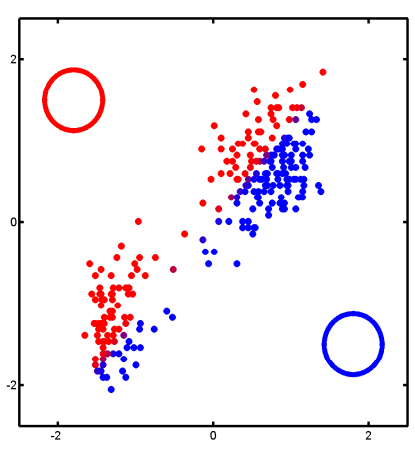
- The expected cluster allocations follow from the Bayes' rule showed earlier
- The ML estimate for the means and covariances was shown last lecture – now we simply need to re-weight every data point with the probability it has for belonging to that cluster

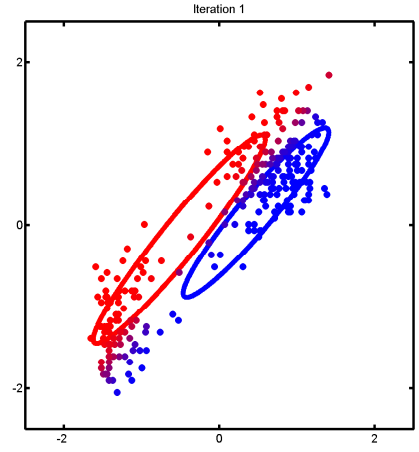
Practical details matter:

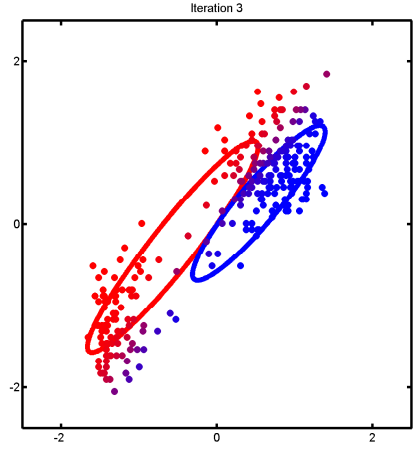
- GMM easily overfits; one cluster starts modeling a single data point and the ML estimate tends to zero variance
- Best way to regularize is to place good priors on the covariance
- $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$, $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2)$, $\boldsymbol{\Sigma}_k \approx \boldsymbol{\Sigma}$, ...

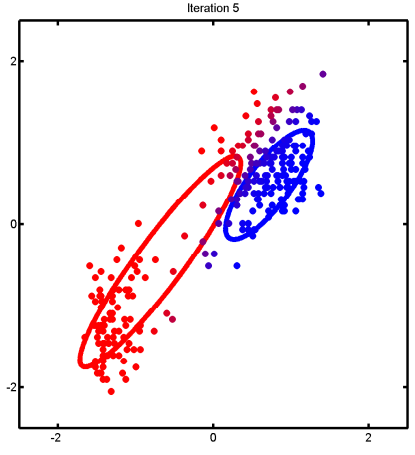
Example

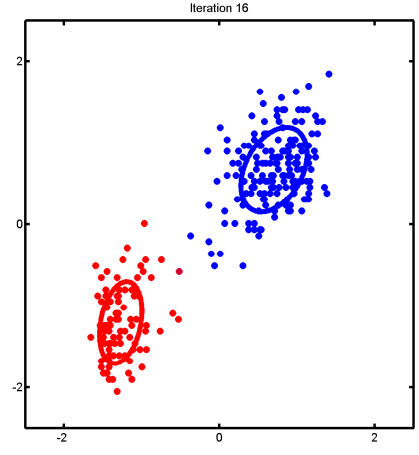










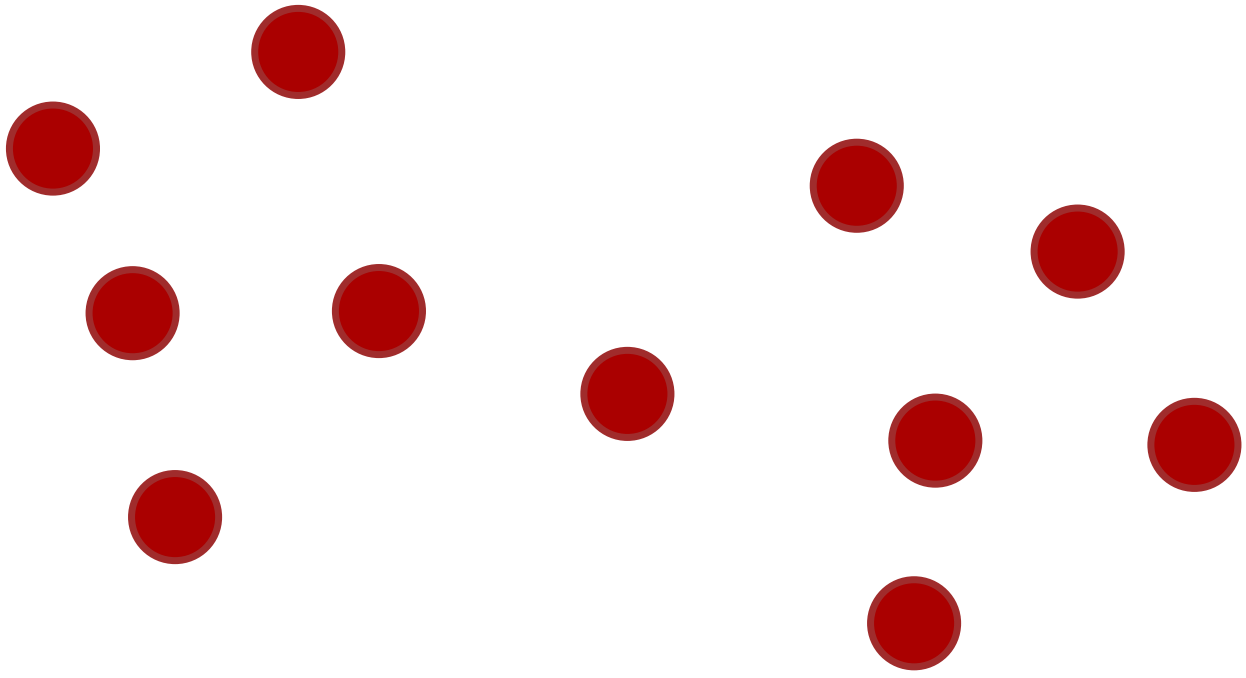


Clustering as graph partitioning

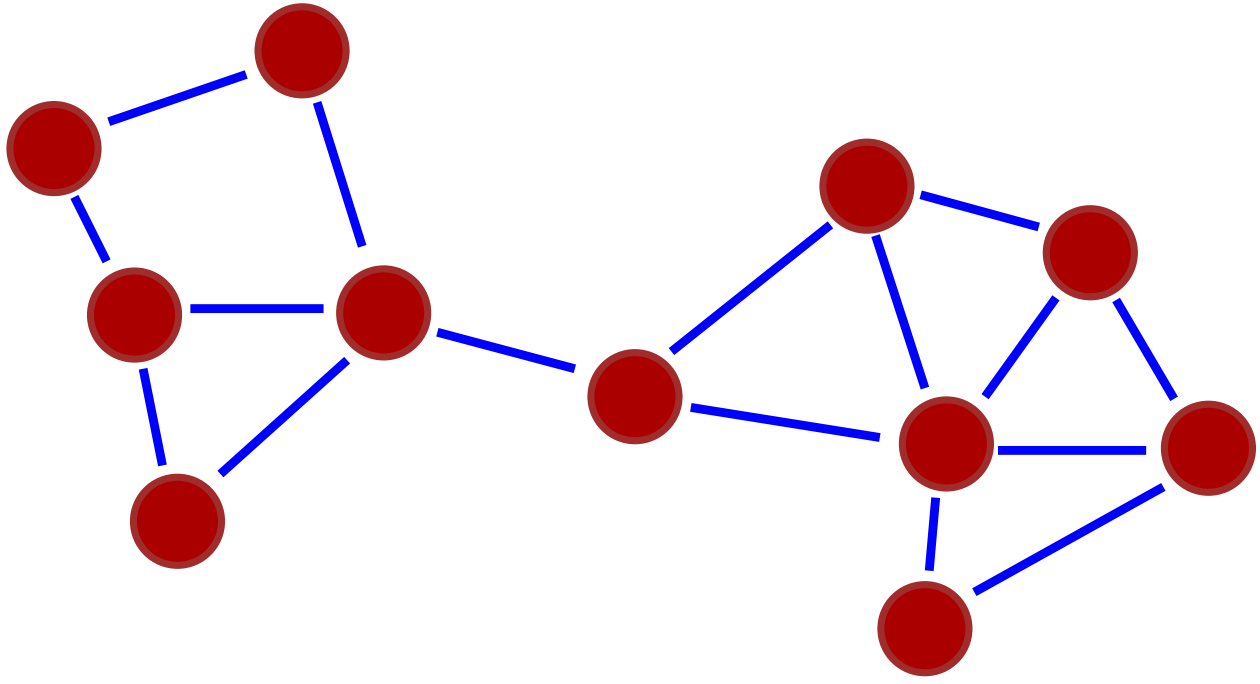
Any dissimilarity matrix can be converted into a graph: Each data point is a node and we have weighted edges between them, with weights depending on the similarity

Now we can use graph theory to partition the samples

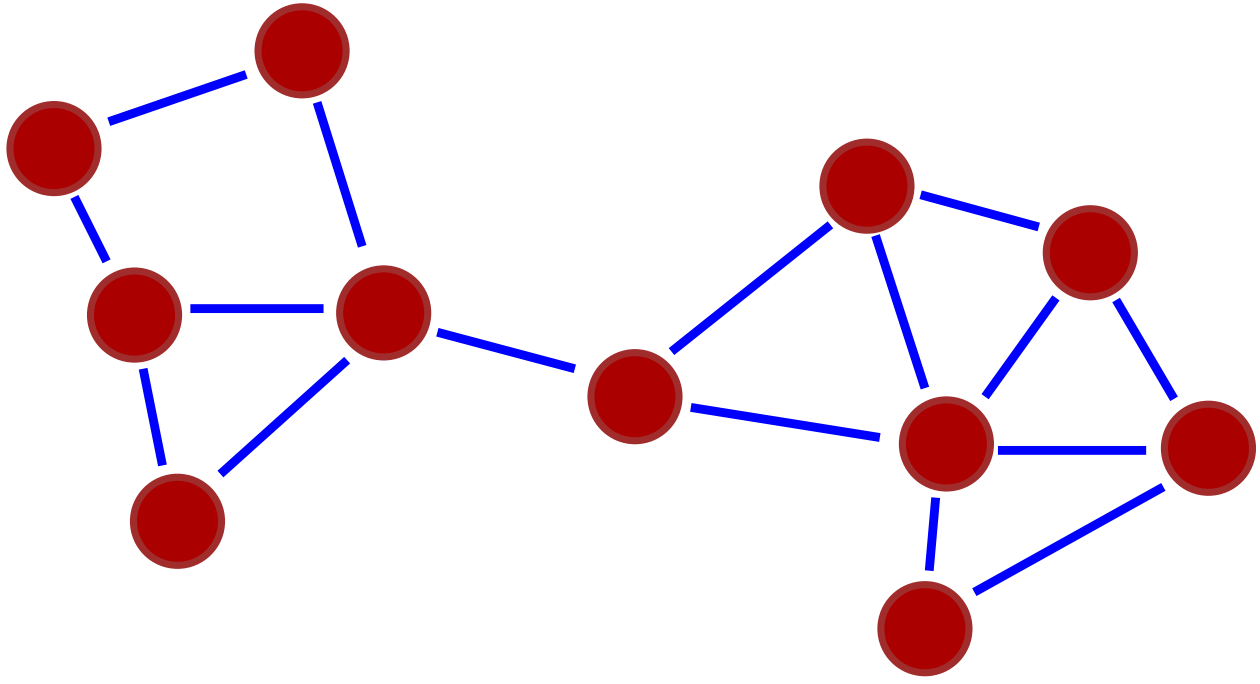
Intuition



Intuition

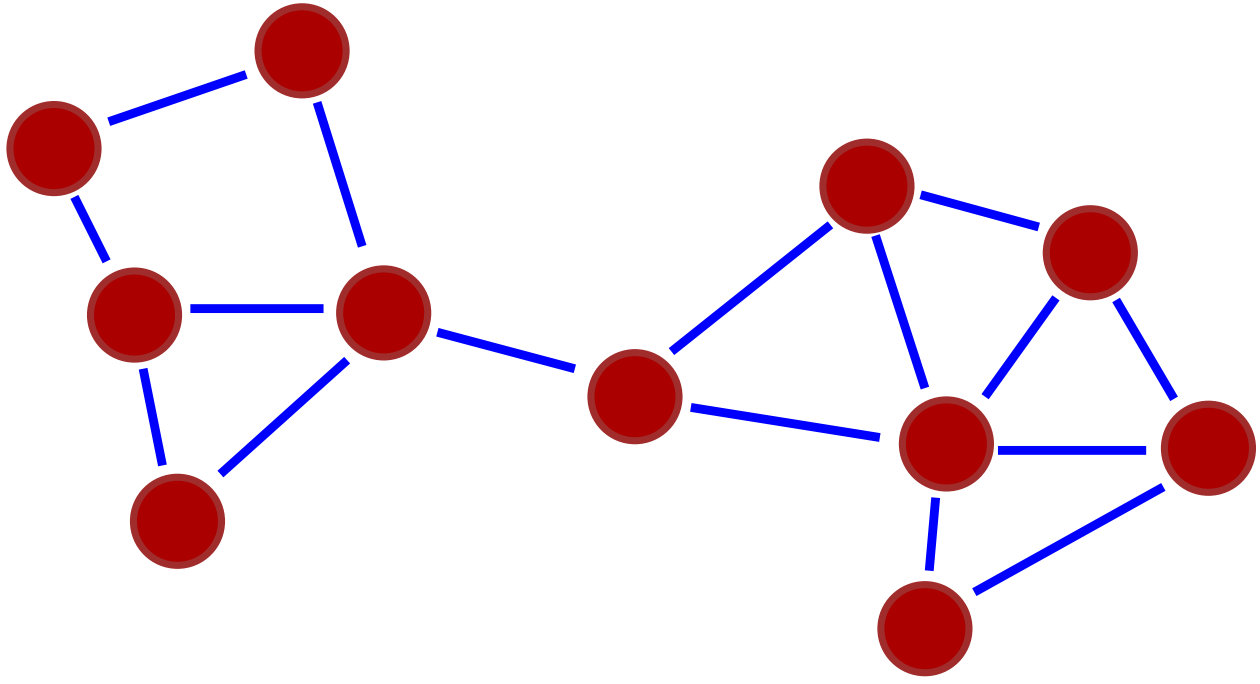


Intuition



Graph cut: Remove edges with minimal weight so that the graph is split into disjoint sets

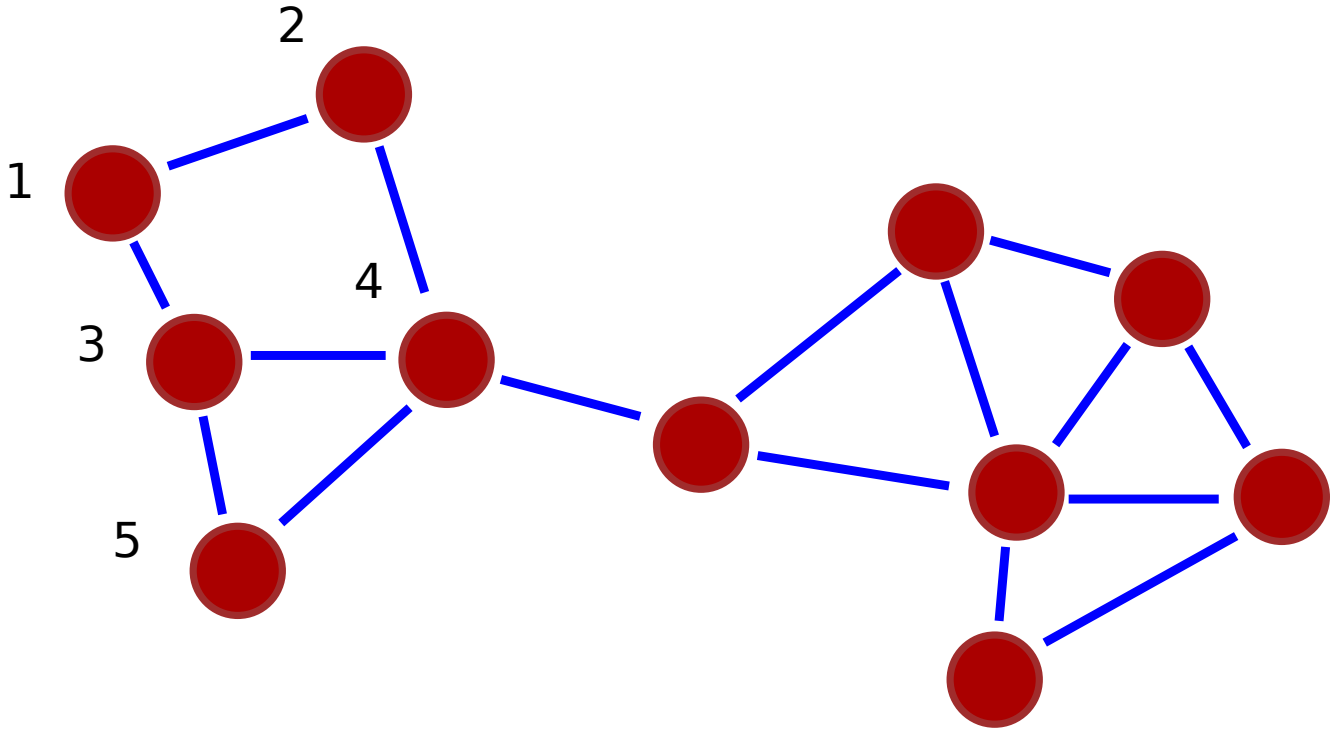
Intuition



Graph cut: Remove edges with minimal weight so that the graph is split into disjoint sets

In practice: Normalized graph cut to ensure the sets are large enough. NP-hard optimization problem, solved for example by relaxing the assignments into real values

Intuition



	1	2	3	4	5
1	2	0	0	0	0
2	0	2	0	0	0
3	0	0	3	0	0
4	0	0	0	4	0
5	0	0	0	0	2

D

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	0
3	1	0	0	1	1
4	0	1	1	0	1
5	0	0	1	1	0

W

	1	2	3	4	5
1	2	-1	-1	0	0
2	-1	2	0	-1	0
3	-1	0	3	-1	-1
4	0	-1	-1	4	-1
5	0	0	-1	-1	2

L=D-W

Graph Laplacian

The graph Laplacian $L = D - W$ has some interesting properties:

- It is symmetric and positive definite
- It has K eigenvalues of 0 if the graph has K disjoint components
- ...and the corresponding eigenvectors are binary vectors with ones for nodes in the corresponding component

Sketch of proof: $f^T L f = \frac{1}{2} \sum_{i,j} W_{i,j}(f_i - f_j)^2$ and $W > 0$, and hence $\lambda = 0$ has to correspond to $f_i = f_j \quad \forall i, j$

Hence: If the graph splits into K sets then the eigenvectors corresponding to the zero eigenvalues directly reveal the clusters!

Graph Laplacian

In practice we do not have disjoint sets, but perturbation theory says that for almost disjoint sets the eigenvalues and eigenvectors are somehow close to the ideal case

The eigenvectors no longer directly reveal the clustering, but we can use them to represent the data in a format that clusters easily

This results in a practical spectral clustering algorithm:

- Compute $L = D - W$ and its eigen-decomposition $L = V \Lambda V^{-1}$
- Represent the data as $Y = V[:, 1 : M]$
- Use any clustering algorithm for Y to reveal the final clustering; often K-means is used here

Spectral clustering in practice

In realistic cases there need not be any connection between the number of eigenvectors retained (M) and the number of clusters K; often $M > K$

Instead of the graph laplacian $L = D - W$, it is typically better to use some normalized version of it:

- $L_{rw} = D^{-1}L = I - D^{-1}W$
- $L_{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$

Why?

Distance-based clustering algorithm (like hierarchical clustering); does not require knowing the actual data points

Freedom in generating the graph; complex shapes typically easier to cluster this way rather than trying to write down the generative model

- ϵ-neighborhood: Connect all pairs for which $d(i,j) < \epsilon$
- k-nearest neighbors: Connect all samples to their k nearest neighbors
- Fully connected graph with some decreasing function $w = f(d(i,j))$

Somewhat involved theoretical connections to the graph cut intuition; see for example `http://www.cs.cmu.edu/~aarti/Class/10701/readings/Luxburg06_TR.pdf` if you are interested, but this is not necessary for the course