Clustering

Ulrike von Luxburg

July 2007

Overview

Rough plan of the lectures, might change as we go along:

1. Intuition and basic algorithms
2. Advanced algorithms (in particular spectral clustering)
3. What is clustering, after all? How can we define it? Some theoretic approaches
4. The number of clusters: different approaches in theory and practice

Clustering: intuition

What is clustering, intuitively?

Given:

- Data set of “objects”
- Some relations between those objects (similarities, distances, neighborhoods, connections, ...)

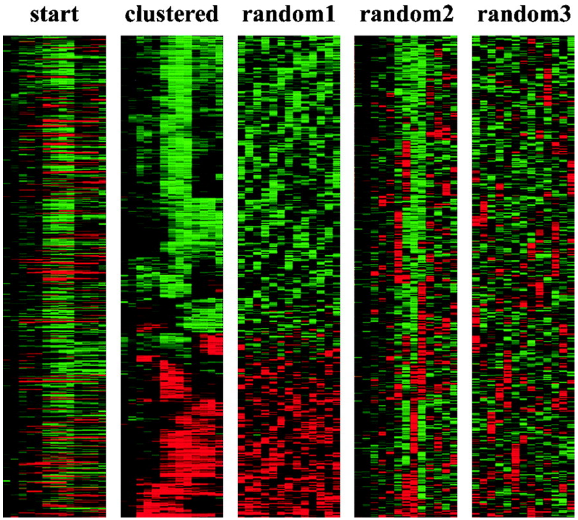
Intuitive goal: Find meaningful groups of objects such that

- objects in the same group are “similar”
- objects in different groups are “dissimilar”

Reason to do this:

- exploratory data analysis
- reducing the complexity of the data
- many more

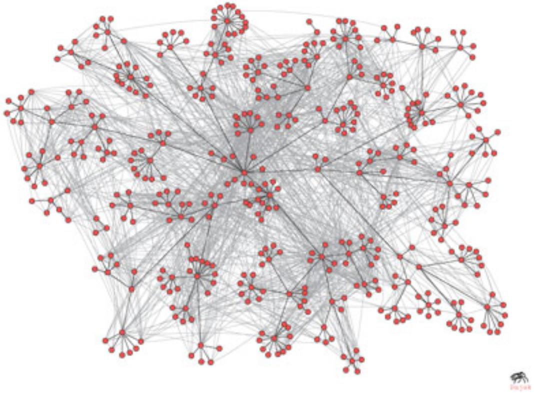
Example: Clustering gene expression data



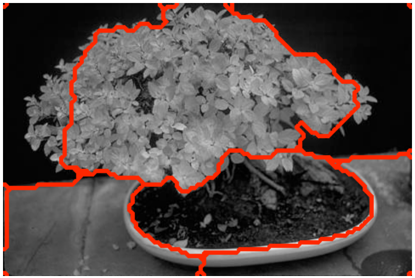
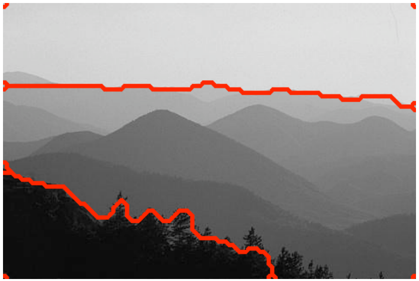
M. Eisen et al., PNAS, 1998

Example: Social networks

Corporate email communication (Adamic and Adar, 2005)

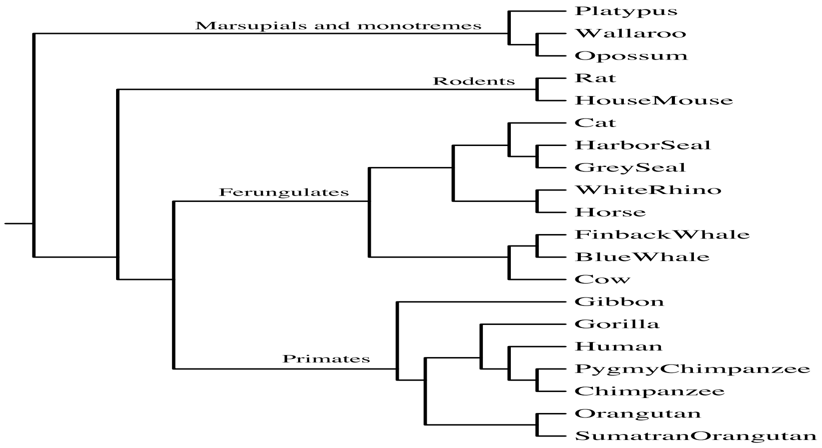


Example: Image segmentation

(from Zelnik-Manor/Perona, 2005)

Example: Genetic distances between mammals



cf. Chen/Li/Ma/Vitanyi (2004)

Two classes of algorithms

- "Flat clustering": Clustering as a partition of the data space. Want to construct "the most meaningful" partition of the data set into a fixed number of partitions.
- Hierarchical clustering:
  try not only to construct one partition, but a nested hierarchy of partitions.

The standard algorithm for “flat” clustering:

K-means

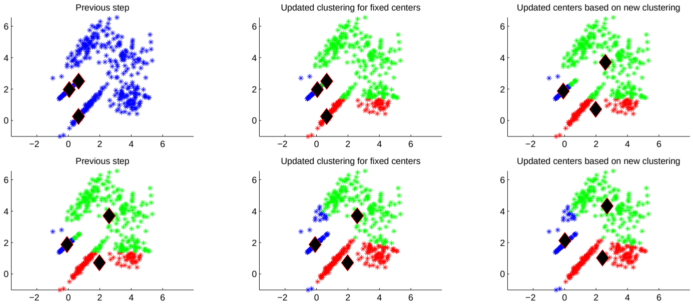
K-means – the algorithm

- Given data points $X_1, ..., X_n \in \mathbb{R}^d$.
- Want to cluster them based on Euclidean distances.

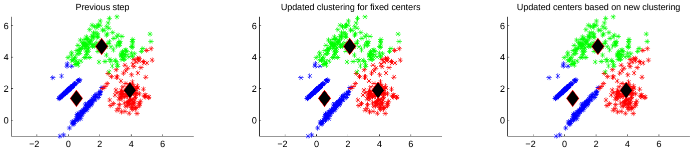
Main idea of the K-means algorithm:

- Start with randomly chosen centers.
- Assign all points to their closest center.
- This leads to preliminary clusters.
- Now move the starting centers to the true centers of the current clusters.
- Repeat this until convergence.

K-means – the algorithm (2)



...



K-means – the algorithm (3)

The K-means algorithm:

Input: Data points $X_1, ..., X_n \in \mathbb{R}^d$, number K of clusters to construct.

1. Randomly initialize the centers $m_1^{(0)}, ..., m_K^{(0)}$.
2. Iterate until convergence:
   - 2.1 Assign each data point to the closest cluster center, that is define the clusters $C_1^{(i+1)}, ..., C_K^{(i+1)}$ by

$$X_s \in C_k^{(i+1)} \iff \|X_s - m_k^{(i)}\|^2 \leq \|X - m_l^{(i)}\|^2, l = 1, ..., K$$

   - 2.2 Compute the new cluster centers by

$$m_k^{(i+1)} = \frac{1}{|C_k^{(i+1)}|} \sum_{s \in C_k} X_s$$

Output: Clusters $C_1, ..., C_K$

K-means – the algorithm (4)

matlab demo: `demo_kmeans()`

Justifying K-means

- Clustering as “data quantization”
- Tries to fit a mixture of Gaussians using a simplified EM-algorithm (this is what Joaquin showed you)
- Want to show now: K-means solves a more general problem

Find K groups $C_1, ..., C_K$ of points such that points in the same cluster are close together.

Minimizing within-cluster distances

For example we could choose the following objective function:

$$\min_{\{C_1,\ldots,C_K\}} \sum_{k=1}^{K} \frac{1}{|C_k|^2} \sum_{i \in C_k, j \in C_k} \|X_i - X_j\|^2 \qquad (OPT1)$$

How can we solve it?

Minimizing within-cluster distances (2)

Let's try to rewrite the optimization problem:

The optimization problem (OPT1) is equivalent to

$$\min_{\{C_1,\dots,C_K\}} \sum_{k=1}^{K} \sum_{i \in C_k} \|X_i - m_k\|^2 \qquad (OPT2)$$

where $m_k := \frac{1}{|C_k|} \sum_{j \in C_k} X_j$ is the mean of cluster C_k.

Minimizing within-cluster distances (3)

Proof: $\sum_{i \in C_k} \|X_i - m_k\|^2 =$

$$= \sum_{i \in C_k} \| \frac{1}{|C_k|} \sum_{j \in C_k} (X_i - X_j) \|^2$$

$$= \frac{1}{|C_k|^2} \langle \sum_{j \in C_k} (X_i - X_j), \sum_{s \in C_k} (X_i - X_s) \rangle$$

$$= \frac{1}{|C_k|^2} \Big(\sum_{i} \langle X_i, X_i \rangle - \sum_{i,s} \langle X_i, X_s \rangle - \sum_{j,i} \langle X_j, X_s \rangle + \sum_{j,s} \langle X_j, X_s \rangle \Big)$$

$$= \frac{1}{|C_k|^2} \Big(\sum_{i} \langle X_i, X_i \rangle - \sum_{i,j} \langle X_i, X_j \rangle \Big)$$

$$= \frac{1}{2} \frac{1}{|C_k|^2} \sum_{i,j} \|X_i - X_j\|^2$$

Minimizing within-cluster distances (4)

Ok, now want to solve:

$$\min_{\{C_1,\ldots,C_K\}} \sum_{k=1}^{K} \sum_{i \in C_k} \|X_i - m_k\|^2 \qquad (OPT2)$$

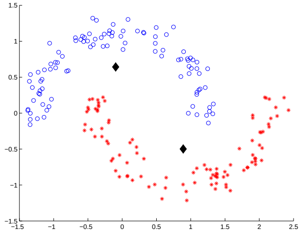
where $m_k := \frac{1}{|C_k|} \sum_{j \in C_k} X_j$ is the mean of cluster C_k.

Is it easier than (OPT1) we started with?

Minimizing within-cluster distances (5)

Not yet....

- Instead of optimizing over all possible partitions $C_1, ..., C_K$ (discrete!) would like to optimize over the means (continuous!).
- But at this stage: no rule how we can recompute the clusters from centers.

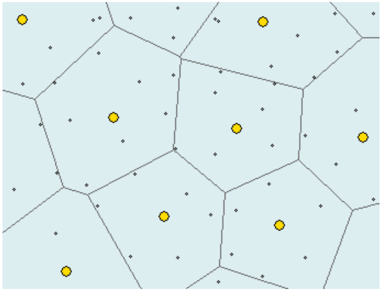


- Could "cheat" and only consider convex clusters
- The good and surprising news: this is not cheating.

Minimizing within-cluster distances (6)

Definition: Voronoi partition of a metric space into K cells:

- choose K points $m_1, ..., m_K$ in the space
- the k-th Voronoi cell is the set of points which are closer to m_k than to any other m_l



Minimizing within-cluster distances (7)

The clusters which solve optimization problem (OPT2) form a Voronoi partition of $\mathbb{R}^k$, and the centers of the cells coincide with the cluster means m_k.

Proof:

- Assume contrary, i.e. cluster C_k is not a Voronoi cell.
- Then there exists a point $X \in C_k$ and index $\ell \neq k$ such that $\|X - m_\ell\| \leq \|X - m_k\|$
- Now want to show that moving X to cluster C_ℓ decreases the objective function.
- Note that moving X also changes the centers of the clusters! So this is not trivial (and in fact only works for squared Euclidean distances!)

Minimizing within-cluster distances (8)

- Will need following fact (*): the minimizer m^* of the problem

$$\min_m \sum_{i \in C_k} \|X_i - m\|^2$$

  is given by the mean $m^* = m_k$, the cluster mean of cluster C_k.
    - To see this, take the derivative with respect to m and set it to 0.
    - Note that here it is crucial that we use the squared norm, not the un-squared norm!
- Introduce the notation $\tilde{C}_\ell := C_\ell \cup \{X\}$
  $\tilde{C}_k := C_k \setminus \{X\}$
  m_k, m_ℓ : centers of clusters C_k, C_ℓ
  $\tilde{m}_k, \tilde{m}_\ell$ centers of clusters $\tilde{C}_k, \tilde{C}_\ell$

Minimizing within-cluster distances (9)

Now calculate: After moving point X the objective function is:

$$
\begin{aligned}
&\sum_{i \in C_k \setminus \{X\}} \|X_i - \tilde{m}_k\|^2 + \sum_{i \in C_\ell \cup \{X\}} \|X_i - \tilde{m}_\ell\|^2 \\
&\leq \text{ [using (*)] } \sum_{i \in C_k \setminus \{X\}} \|X_i - m_k\|^2 + \sum_{i \in C_\ell \cup \{X\}} \|X_i - m_\ell\|^2 \\
&= \sum_{i \in C_k \setminus \{X\}} \|X_i - m_k\|^2 + \sum_{i \in C_\ell} \|X_i - m_\ell\|^2 + \|X - m_\ell\|^2 \\
&< \sum_{i \in C_k \setminus \{X\}} \|X_i - m_k\|^2 + \sum_{i \in C_\ell} \|X_i - m_\ell\|^2 + \|X - m_k\|^2 \\
&= \sum_{i \in C_k} \|X_i - m_k\|^2 + \sum_{i \in C_\ell} \|X_i - m_\ell\|^2
\end{aligned}
$$

This was the objective function before moving the point. $\square$

Minimizing within-cluster distances (10)

- Have seen: solution of (OPT2) is always a Voronoi partition induced by the cluster centers.
- Great simplification: Instead of optimizing over all (exponentially many) partitions we "only" have to optimize over all Voronoi partitions of the sample.
- Side remark: for fixed K, the number of Voronoi partitions of n sample points is only polynomial in n!

  So in principle could search through all of them in polynomial time. But "polynomial" can still take pretty long ...
- But even better: we got a continuous parameterization of the optimization problem in terms of cluster means.

Minimizing within-cluster distances (11)

We can conclude:

Optimization problems (OPT1) and (OPT2) are equivalent to

$$\min_{m_1,\dots,m_K \in \mathbb{R}^d} \sum_{k=1}^{K} \sum_{i \in C_k} \|X_i - m_k\|^2 \qquad (OPT3)$$

- Instead of solving a discrete optimization problem such as (OPT2) we have a continuous one. Promising.
- Promising. We could use standard optimization techniques such as gradient descent.
- However, it is not convex ☹ (Why?)
- So still need heuristics ... K-means

K-means — properties of the algorithm

The algorithm terminates after a finite number of iterations.

Proof:

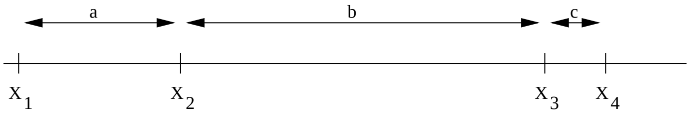
- In each step, the objective function is decreasing.
- There are only finitely many solutions we can inspect.

... exercise ...

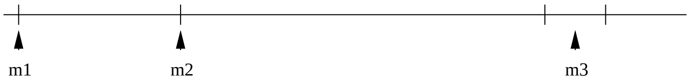
K-means — properties of the algorithm (2)

The algorithm only ends in a local optimum. This can be arbitrarily worse than the global one.

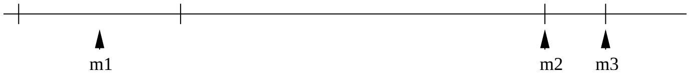
Data set: four points on the real line:



Optimal solution: (initialization: X1, X2, X3; value of solution: c^2 / 2)



Bad solution: (initialization: X1, X3, X4; value of solution: a^2 / 2)



By adjusting a and b and c we can achieve an arbitrarily bad ratio of global and local solution.

K-means — Implementation issues

Random initialization:

- Most common: randomly choose some data points as starting centers.
- Draw starting points randomly from $\mathbb{R}^d$.
- Initialize the centers using the solution of an even simpler clustering algorithm.
- Ideally have prior knowledge, for example that certain points are in different clusters.

Common problem for all those methods: empty clusters (centers to which no data point is assigned). Then best solution: restart...

K-means — Implementation issues (2)

Heuristics for improving the local minimum:

- Restart many times with different initializations.
- Swap individual points between clusters.
- Remove a cluster center, and introduce a completely new center instead.
- Merge clusters, and additionally introduce a completely new cluster center.
- Split a cluster in two pieces (preferably, one which has a very bad objective function). Then reduce the number of clusters again, for example by randomly removing one.

K-means — Implementation issues (3)

More efficient implementations:

- The bottleneck of the K-means algorithm is the computation of the nearest neighbors.
- This can be very costly, in particular if the dimension d of the space and the number K of clusters are high (e.g., data mining applications)
- Improvement: use more efficient data structures, for example kd-trees ...

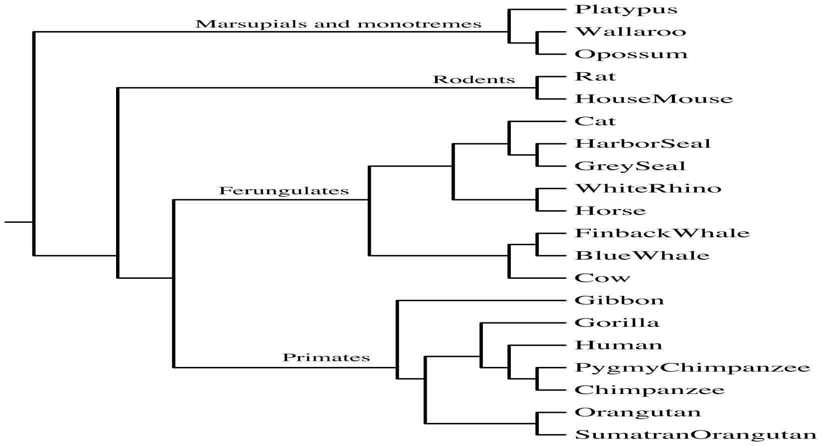
More variants of K-means

- K-median: here the centers are always data points. Can be used if we only have distances, but no coordinates of data points.
- weighted K-means: introduce weights for the individual data points
- kernel-K-means: the kernelized version of K-means (note that all boundaries between clusters are linear)
- soft K-means: no hard assignments, but "soft" assignments (often interpreted as "probability" of belonging to a certain cluster)
- Note: K-means is a simplified version of an EM-algorithm fitting a Gaussian mixture model.

The standard algorithm for hierarchical clustering: single linkage

Hierarchical clustering

Goal: obtain a complete hierarchy of clusters and sub-clusters in form of a dendrogram



cf. Chen/Li/Ma/Vitanyi (2004)

Simple idea

Agglomerative (bottom-up) strategy:

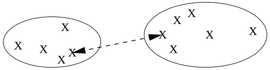
- Start: each point is its own cluster
- Then check which points are closest and "merge" them to form a new cluster
- Continue, always merge two "closest" clusters until we are left with one cluster only

The original article: S. C. Johnson. Hierarchical clustering schemes. Psychometrika, 2:241 - 254, 1967.
A complete book on the topic: N. Jardine and R. Sibson. Mathematical taxonomy. Wiley, London, 1971.
Nice overview with application in biology: J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. ISMB 1999.

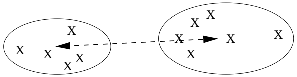
Simple idea (2)

To define which clusters are "closest":

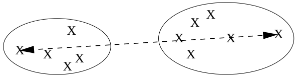
Single linkage: $dist(C, C') = \min_{x \in C, y \in C'} d(x, y)$



Average linkage: $dist(C, C') = \frac{\sum_{x \in C, yin C'} d(x,y)}{|C| \cdot |C'|}$



Complete linkage: $dist(C, C') = \max_{x \in C, y \in C'} d(x, y)$



Linkage algorithms – basic form

Input:

- Distance matrix D between data points (size $n \times n$)
- function *dist* to compute a distance between clusters (usually takes D as input)

```
Initialization: Clustering $\mathcal{C}^{(0)} = \{C_1^{(0)}, ..., C_n^{(0)}\}$ with $C_i^{(0)} = \{i\}$.
While the current number of clusters is $> 1$:
    ◦ find the two clusters which have the smallest distance to each other
    ◦ merge them to one cluster
```

Output: Resulting dendrogram

Examples

... show matlab demos ...

```
demo_linkage_clustering_by_foot()
demo_linkage_clustering_comparison()
```

Comments on linkage algorithms

- Single linkage tends to generate long "chains"
- Complete linkage tends to produce more "compact" clusters
- Linkage algorithms are very vulnerable to outliers
- one cannot "undo" a bad link
- Single linkage can also be described using the minimal spanning tree of data points (e.g., cutting the longest edge of an MST gives the first two single linkage clusters)
- Advantage of hierarchical clustering: do not need to decide on "the right" number of clusters
- There exist many more ways of generating different trees from a given distance matrix ...

Summary so far

We have see the two most widely used clustering algorithms:

- K-means and variants: tries to maximize within-cluster similarity
- Single linkage: builds a dendrogram

In 95% of all applications, one of those two algorithms is used...

Both are ...

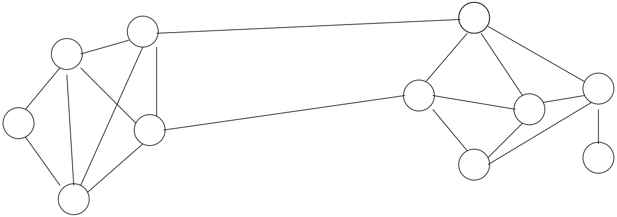
- simple heuristics, pretty ad hoc
- very easy to implement
- in practice, both algorithms often work "reasonable", are baseline for more advanced algorithms

Spectral Clustering

Spectral clustering on one slide

- Given: data points $X_1, ..., X_n$, pairwise similarities $s_{ij} = s(X_i, X_j)$

- Build similarity graph: vertices = data points, edges = similarities



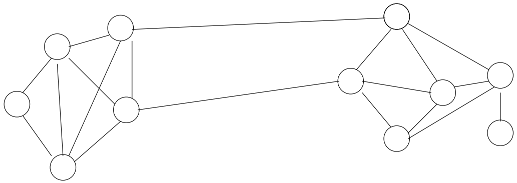
- clustering = find a cut through the graph
  - define a cut objective function
  - solve it

$\rightsquigarrow$ **spectral clustering**

Graph notation

Always assume that similarities s_{ij} are symmetric, non-negative
Then graph is undirected, can be weighted

- $S = (s_{ij})$ similarity matrix
- $d_i = \sum_j s_{ij}$ degree of a vertex
- $D = diag(d_1, \dots, d_n)$ degree matrix
- $|A| =$ number of vertices in A
- $\mathrm{vol}(A) = \sum_{i \in A} d_i$



In the following: vector $f = (f_1, ..., f_n)$ interpreted as function on the graph with $f(X_i) = f_i$.

Unnormalized graph Laplacian

Defined as

$$L = D - S$$

Key property: for all $f \in \mathbb{R}^n$

$$\begin{aligned} f'Lf &= f'Df - f'Sf \\ &= \sum_i d_i f_i^2 - \sum_{i,j} f_i f_j s_{ij} \\ &= \frac{1}{2}\left(\sum_i (\sum_j s_{ij}) f_i^2 - 2 \sum_{ij} f_i f_j s_{ij} + \sum_j (\sum_i s_{ij}) f_j^2 \right) \\ &= \frac{1}{2} \sum_{ij} s_{ij} (f_i - f_j)^2 \end{aligned}$$

Unnormalized graph Laplacian (2)

Where does the name "graph Laplacian" come from?

$$f'Lf = \frac{1}{2} \sum s_{ij}(f_i - f_j)^2$$

Interpret $s_{ij} \sim 1/d(X_i, X_j)^2$

$$f'Lf = \frac{1}{2} \sum ((f_i - f_j)/d_{ij})^2$$

looks like a discrete version of the standard Laplace operator

$$\langle f, \Delta f \rangle = \int |\nabla f|^2 dx$$

Hence the graph Laplacian measures the variation of the function f along the graph: $f'Lf$ is low if close points have close function values f_i -

Unnormalized graph Laplacian (3)

Spectral properties:

- L is symmetric (by assumption) and positive semi-definite (by key property)
- Smallest eigenvalue of L is 0, corresponding eigenvector is $\mathbb{1}$
- Thus eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.

First relation between spectum and clusters:

- Multiplicity of eigenvalue $0 =$ number k of connected components $A_1, ..., A_k$ of the graph.
- eigenspace is spanned by the characteristic functions $\mathbb{1}_{A_1}, ..., \mathbb{1}_{A_k}$ of those components (so all eigenvecotrs are piecewise constant).

Proof: Exercise

Normalized graph Laplacians

Row sum (random walk) normalization:

$$L_{\mathrm{rw}} = D^{-1}L \;=\; I - D^{-1}S$$

Symmetric normalization:

$$L_{\mathrm{sym}} = D^{-1/2}LD^{-1/2} \;=\; I - D^{-1/2}SD^{-1/2}$$

Spectral properties similar to L:

- Positive semi-definite, smallest eigenvalue is 0
- Attention: For L_{rw}, eigenspace spanned by $\mathbb{1}_{A_i}$ (piecewise const.) but for L_{sym}, eigenspace spanned by $D^{1/2}\mathbb{1}_{A_i}$ (not piecewise const).

Normalized graph Laplacians (2)

$$\begin{aligned} w \text{ eig of } L_{\text{sym}} &\iff L_{\text{sym}} w = \lambda w \quad | \cdot D^{-1/2} \\ &\iff D^{-1/2} D^{-1/2} L D^{-1/2} w = \lambda D^{-1/2} w \\ &\iff v = D^{-1/2} w \text{ eig of } L_{\text{rw}} \end{aligned}$$

Spectral clustering - main algorithms

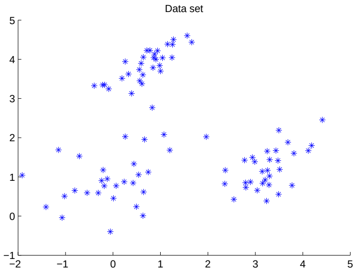
Input: Similarity matrix S, number k of clusters to construct

- Build similarity graph
- Compute the first k eigenvectors $v_1, \ldots, v_k$ of the problem matrix

$$\begin{cases} L & \text{for unnormalized spectral clustering} \\ L_{\text{rw}} & \text{for normalized spectral clustering} \end{cases}$$

- Build the matrix $V \in \mathbb{R}^{n \times k}$ with the eigenvectors as columns
- Interpret the rows of V as new data points $Z_i \in \mathbb{R}^k$
- Cluster the points Z_i with the k-means algorithm in $\mathbb{R}^k$.

Toy example with three clusters



- Data set in $\mathbb{R}^2$
- similarity function $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/\sigma^2)$ with $\sigma = 0.5$
- Use completely connected similarity graph
- Want to look at clusterings for k = 2,...,5 clusters

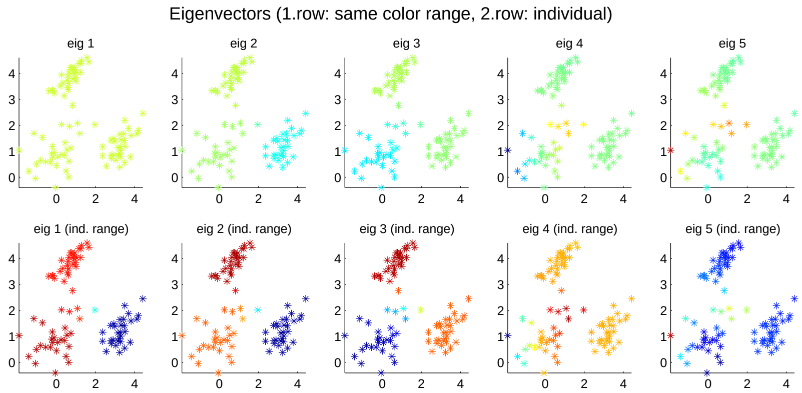
Toy example with three clusters (2)

Compute first 5 eigenvectors of L.

0.1054	-0.1445	0.0188	-0.0006	0.0017
0.1054	-0.1449	0.0191	-0.0023	-0.0039
0.1054	-0.1452	0.0193	-0.0043	-0.0112
0.1054	0.0987	0.0664	-0.0086	-0.0221
0.1054	-0.1429	0.0176	0.0029	0.0084
0.1054	0.0513	-0.1878	-0.1289	0.0082
0.1054	0.0986	0.0657	-0.0057	-0.0123
0.1054	0.0986	0.0658	-0.0055	-0.0102
0.1054	0.0511	-0.1820	0.0028	-0.0299
0.1054	0.0988	0.0667	-0.0099	-0.0266
0.1054	0.0984	0.0651	-0.0015	0.0048
0.1054	0.0509	-0.1830	0.0237	-0.0585
0.1054	0.0508	-0.1837	0.0497	-0.0951
0.1054	0.0974	0.0611	0.0113	0.0481
0.1054	0.0514	-0.1893	-0.2036	0.0565
0.1054	-0.1446	0.0189	-0.0010	0.0008
0.1054	0.0511	-0.1873	-0.0151	-0.1170
0.1054	0.0504	-0.1807	0.0681	-0.0640
0.1054	0.0986	0.0661	-0.0066	-0.0147
0.1054	0.0508	-0.1812	0.0444	-0.0509
0.1054	-0.1447	0.0190	-0.0021	-0.0040
0.1054	-0.1454	0.0195	-0.0053	-0.0148
0.1054	0.0987	0.0664	-0.0082	-0.0200
0.1054	0.0989	0.0671	-0.0123	-0.0358
0.1054	-0.1449	0.0191	-0.0021	-0.0025
0.1054	0.0518	-0.1171	0.2080	0.2655
0.1054	-0.1446	0.0189	-0.0008	0.0015
0.1054	-0.1449	0.0191	-0.0022	-0.0025
0.1054	-0.1457	0.0197	-0.0084	-0.0289
0.1054	0.0983	0.0649	-0.0017	0.0021
0.1054	0.0989	0.0671	-0.0124	-0.0364
0.1054	0.0976	0.0623	0.0049	0.0178

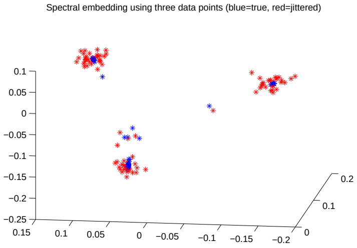
Toy example with three clusters (3)

Each eigenvector is interpreted as a function on the data points:
$x_j \mapsto j$-th coordinate of the eigenvectors
This mapping is plotted in a color code:

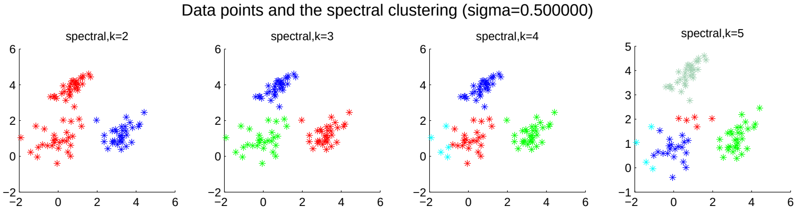


Toy example with three clusters (4)

The spectral embedding:

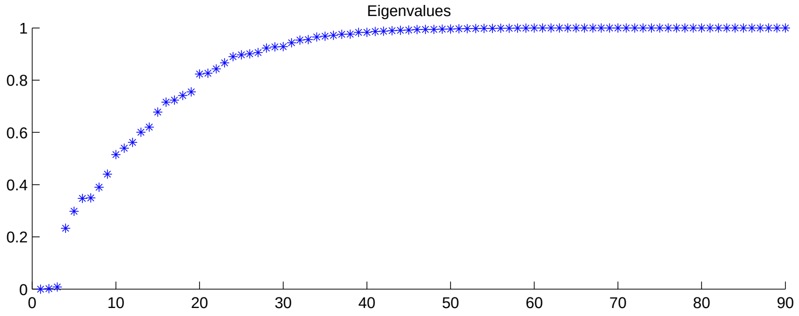


Resulting clustering using kmeans on embedded points (k=2,...,5):



Toy example with three clusters (5)

The eigenvalues (plotted i vs. λ_i):



Toy example with three clusters (6)

... show matlab demo ...

```
demo_spectral_clustering_fancy()
```

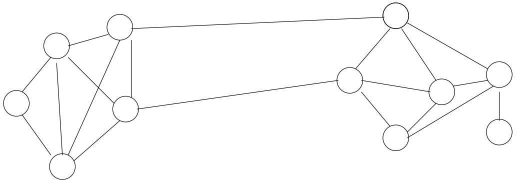
Graph cut explanation of spectral clustering

Clustering: within-similarity high, between similarity low

$$\text{minimize } \mathrm{cut}(A,B) := \textstyle\sum_{i\in A, j\in B} s_{ij}$$

Balanced cuts:

$$\mathrm{RatioCut}(A,B) := \mathrm{cut}(A,B)\left(\frac{1}{|A|}+\frac{1}{|B|}\right)$$
$$\mathrm{Ncut}(A,B) := \mathrm{cut}(A,B)\left(\frac{1}{\mathrm{vol}(A)}+\frac{1}{\mathrm{vol}(B)}\right)$$



Mincut can be solved efficiently, but RatioCut or Ncut is NP hard.

Spectral clustering: relaxation of RatioCut or Ncut, repectively.

Graph cut explanation of spectral clustering (2)

Relaxation for simple balanced cuts:

$$\min_{A,B} \operatorname{cut}(A,B) \text{ s.t. } |A| = |B|$$

Choose $f = (f_1, ..., f_n)'$ with $f_i = \begin{cases} 1 & \text{if } X_i \in A \\ -1 & \text{if } X_i \in B \end{cases}$

- $\operatorname{cut}(A,B) = \sum_{i \in A, j \in B} s_{ij} = \frac{1}{4} \sum_{i,j} s_{ij}(f_i - f_j)^2 = \frac{1}{4} f' L f$
- $|A| = |B| \implies \sum_i f_i = 0 \implies f^t \mathbb{1} = 0 \implies f \perp \mathbb{1}$
- $\|f\| = \sqrt{n} \sim const.$

$$\min_f f' L f \text{ s.t. } f \perp \mathbb{1},\ f_i = \pm 1,\ \|f\| = \sqrt{n}$$

Relaxation: allow $f_i \in \mathbb{R}$

By Rayleigh: solution f is the second eigenvector of L

Reconstructing solution: $X_i \in A \iff f_i >= 0$, $X_i \in B$ otherwise

Graph cut explanation of spectral clustering (3)

Similar relaxations work for the other balanced cuts:

- Relaxing RatioCut $\rightsquigarrow$ eigenvectors of L $\rightsquigarrow$ unnormalized spectral clustering
- Relaxing Ncut $\rightsquigarrow$ eigenvectors of L_{rw} $\rightsquigarrow$ normalized spectral clustering
- Case of $k > 2$ works similar, results in a trace min problem $\min_V \operatorname{Tr} H'LH$ where V is a $n \times k$ orthonormal matrix. Then again Rayleigh-Ritz.

Graph cut explanation of spectral clustering (4)

Notes on the relaxation approach:

- spectral clustering solves a relaxed version of graph cut problems.
- No guarantee whatsoever on the quality of the relaxation.
- using kmeans on new representation just for convenience, any other algorithm would work, too
- but Euclidean distance is “meaningful” im embedding space

Random walk explanations

General observation:

- Random walk on the graph has transition matrix $P = D^{-1}S$.
- note that $L_{\text{rw}} = I - P$

Specific observation about Ncut (Meila/Shi 2001):

- define $P(A|B)$ is the probability to jump from B to A if we assume that the random walk starts in the stationary distribution.
- Then: $Ncut(A, B) = P(A|B) + P(B|A)$
- Interpretation: Spectral clustering tries to construct groups such that a random walk stays as long as possible within the same group

Random walk explanations (2)

Commute distance: $c(X_i, X_j)$ is the expected time a random walk needs to travel from X_i to X_j and back

Can be expressed in terms of the pseudo-inverse of the unnormalized graph Laplacian: $c_{ij} = (e_i - e_j)' L^\dagger (e_i - e_j)$

Commute time embedding:

- decompose $L^\dagger = V' \Lambda^\dagger V$
- choose Z_i as rows of $(\Lambda^\dagger)^{1/2} V$.
- Then $\|Z_i - Z_j\|^2 = c_{ij}$.

Hand-waiving argument: spectral embedding $\approx$ commute time embedding $\implies$ spectral clustering is based on commute distance

The perturbation theory explanation

Ideal case: between-cluster similarities are exactly 0. Then:

$$S = \begin{pmatrix} S_1 & 0 \dots \\ 0 & S_2 \dots \end{pmatrix}$$

$$eig(L_{\text{rw}}) = \begin{pmatrix} 1 & 0 & \dots \\ 1 & 0 & \dots \\ 0 & 1 & \dots \\ 0 & 1 & \dots \end{pmatrix}$$

$$eig(L_{\text{sym}}) = \begin{pmatrix} d_1 & 0 & \dots \\ d_3 & 0 & \dots \\ 0 & d_4 & \dots \\ 0 & d_6 & \dots \end{pmatrix}$$

- For L or L_{rw}: all points of the same cluster are mapped on the identical point in $\mathbb{R}^k$
- Then spectral clustering finds the ideal solution.

The perturbation theory explanation (2)

Perturbation theory: "nearly" ideal case

$\Longrightarrow$ nearly ideal eigenvectors

$\Longrightarrow$ points of the same cluster are mapped on close points in $\mathbb{R}^k$

$\Longrightarrow$ spectral clustering identifies clusters

The perturbation theory explanation – caveats

Any block matrix leads to block eigenvectors

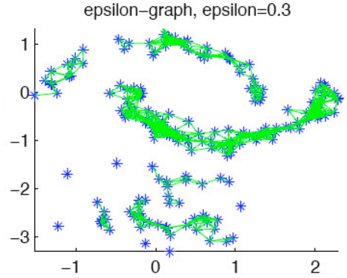
- But what is important is the *order* of eigenvectors: the first k eigenvectors need to contain a representative of each cluster
- This works out great for all Laplacians
- But this might not work for other matrices, e.g. S directly

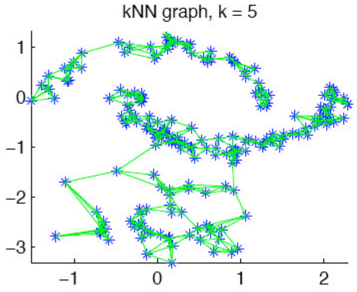
In ideal case, entries of eigenvectors should be "safely bounded away" from 0

- Otherwise, perturbing them leads to ambigous situations
- This might go wrong for L_{sym} if we have vertices with very small degrees

Implementation – similarity graph

vertices: data points X_i
edges: connect X_i and X_j if similarity s_{ij} is "high"





- k-nearest neighbor graph (mutual or symmetric),
- ε-neighborhood graph,
- completely connected graph for certain similarity functions
- ...

Implementation – similarity graph (2)

matlab demo: `demo_neighborhood_graphs(epsilon,k)`

Implementation – similarity graph (3)

k-nearest neighbor graph: **Should be first choice.**

- is sparse, can model data on different scales

ε-neighborhood graph:

- only use it if data similarities are "on same scale"

Completely connected graph:

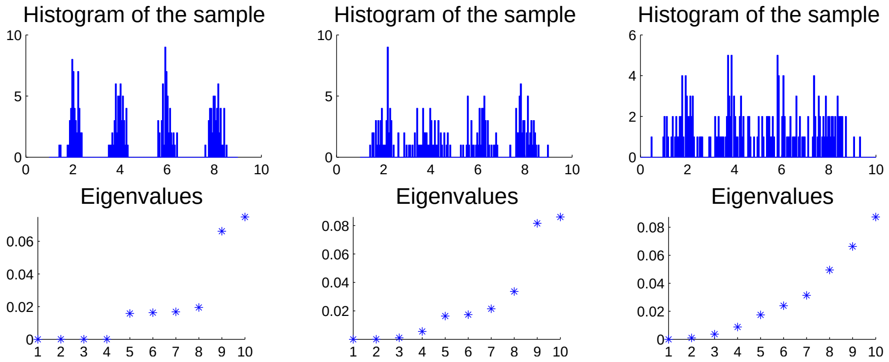
- together with Gaussian kernel $s_{ij} = \exp(-\|x_i - x_j\|^2/\sigma^2)$
- only use it if data is "on same scale"
- disadvantage: not sparse!

Choosing the parameter: points should be connected to their k-th nearest neighbors, where $k \sim \log n$

- sparse graphs
- graphs tend to be connected

Implementation – how many clusters?

- In general, all standard heuristics can be used (will discuss this later: gap statistic; stability methods)
- A special heuristic for spectral clustering: maximize the eigengap



Theoretical reasons: can bound Ncut-value by second eigenvalue ($\rightsquigarrow$ spectral graph theory)

Implementation – normalized or unnormalized?

First argument for normalized: clustering objectives

$\text{RatioCut}(A, B) := \text{cut}(A, B)(\frac{1}{|A|} + \frac{1}{|B|}) \rightsquigarrow$ unnormalized

$\text{Ncut}(A, B) := \text{cut}(A, B)(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}) \rightsquigarrow$ normalized

- between-similarity: $\sum_{i \in A, j \in B} s_{ij} = \text{cut}(A, B)$, minimized by both
- within-similarity is only maximized by Ncut, but not by Ratiocut: $\sum_{ij \in A} s_{ij} = \sum_{i \in A, j \in V} s_{ij} - \sum_{i \in A, j \in B} s_{ij} = \text{vol}(A) - \text{cut}(A, B)$

Second argument for normalized: Statistical consistency

- Unnormalized spectral clustering is statistically not consistent and can lead to artifacts!
- Normalized spectral clustering is always statistically consistent!

$\rightsquigarrow$ **always use normalized spectral clustering!!!**

The modularity approach

Pretty new and pretty popular algorithm for community detection in networks, ends in an algorithm very similar to spectral clustering:

- So far we simply “counted” whether there are few or many edges between two groups of vertices.
- If there are few edges between the groups, we tend to believe that the groups are clusters.
- But now assume we have a very large graph. Just by random chance there will be some groups between which there will be few edges. Having “few edges” is not meaningful enough.
- Idea is thus: only consider groups clusters if there are significantly fewer edges than expected.

M. Newman: Finding community structure in networks using the eigenvectors of matrices. 2006.

The modularity approach (2)

In fact, the modularity approach works with within-cluster similarity.
Want to have significantly more edges within a cluster than we would just expect by mere chance:

$$\max_A \sum_{i,j \in A} s_{ij} - E(s_{ij})$$

To define "$E(s_{ij})$" need random graph model. Simplest choice:

- put an edge between vertex i and j with probability $p_{ij} := d_i d_j / \operatorname{vol}(G)$ for all pairs i, j
- then $E(s_{ij}) = p_{ij}$
- expected degrees in the model graphs coincide with given degrees in our graph:
  $E(\text{degree i in model}) = \sum_j p_{ij} = d_i \sum_j d_j / \operatorname{vol}(G) = d_i$

- matlab demo: `demo_modularity_newman.m`

The modularity approach (3)

Have optimization problem:
$\max_A \sum_{i,j \in A} s_{ij} - E(s_{ij})$
$\max_A \sum_{i,j \in A} s_{ij} - d_i d_j / vol(G)$

Can also relax this to an eigenvector problem:

- Set indicator vector $f = (f_1, ..., f_n)'$ with $f_i = \begin{cases} 1 & \text{if } X_i \in A \\ -1 & \text{if } X_i \notin A \end{cases}$
- Define matrix $B = \bar{D} - S$ where $\bar{D}_{ij} = d_i d_j / \operatorname{vol}(G)$
- Can write:

$$\begin{aligned} \sum_{i,j \in A} s_{ij} - E(s_{ij}) &= \sum_{i,j=1}^{n} (s_{ij} - E(s_{ij}))(f_i f_j + 1) \\ &= \sum_{i,j=1}^{n} (s_{ij} - E(s_{ij})) f_i f_j + const. \\ &= -f' B f + const. \end{aligned}$$

The modularity approach (4)

Then optimization problem becomes:

$$\min_f f'Bf \text{ s.t. } f_i = \pm 1$$

Relax to $f_i \in \mathbb{R} \rightsquigarrow$ use first eigenvector of B

Note the similarity of B to the Laplace matrix $L = D - S$!

The modularity approach (5)

Properties:

- B usually has positive and negative eigenvalues, and always eigenvalue 0 with eigenvector $\mathbb{1}$
- Interpretation: if smallest eigenvalue is 0, then the graph does not contain any community as the cluster indicator vector is $\mathbb{1}$
- Hence can detect absence of clusters!
- Even better: number of clusters = number of negative eigenvalues of B.
- We need no balancing condition (as we look at individual communities).
- Random graph model chosen such that graphs are similar to the given graph. Thus take into account at least some features of the geometry of the current graph

Spectral clustering – pros and cons

Why is spectral clustering useful?

- Does not make strong assumptions on cluster shape
- Is simple to implement (solving an eigenproblem)
- Spectral clustering objective does not have local optima
- Is statistically consistent (normalized only!)
- Has several different derivations
- Successful in many applications

What are potential problems?

- Can be sensitive to choice of parameters (k in kNN-graph).
- Computational expensive on large non-sparse graphs
- Not really clear what it does on non-regular graphs (e.g. power law graphs)

Some selected literature on spectral clustering

Of course I recommend the following ☺

- U.von Luxburg. A tutorial on spectral clustering. Statistics and Computing, to appear. On my homepage.

The three articles which are most cited:

- Meila, M. and Shi, J. (2001). A random walks view of spectral segmentation. AISTATS.
- Ng, A., Jordan, M., and Weiss, Y. (2002). On spectral clustering: analysis and an algorithm. NIPS 14.
- Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation.IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888 - 905.

Nice historical overview on spectral clustering; and how relaxation can go wrong:

- Spielman, D. and Teng, S. (1996). Spectral partitioning works: planar graphs and finite element meshes. In FOCS, 1996

The modularity approach:

- M. Newman: Finding community structure in networks using the eigenvectors of matrices. 2006.

Even more clustering algorithms

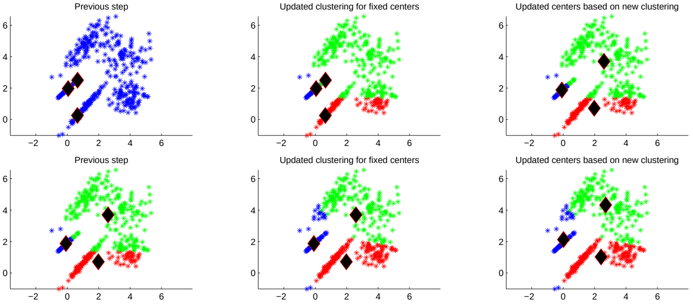
K-means

- Given data points $X_1, ..., X_n \in \mathbb{R}^d$.
- Want to cluster them based on Euclidean distances.

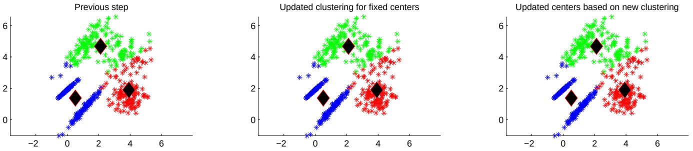
Main idea of the K-means algorithm:

- Start with randomly chosen centers.
- Assign all points to their closest center.
- This leads to preliminary clusters.
- Now move the starting centers to the true centers of the current clusters.
- Repeat this until convergence.

K-means (2)


Previous step
Updated clustering for fixed centers
Updated centers based on new clustering
Previous step
Updated clustering for fixed centers
Updated centers based on new clustering

. . .


Previous step
Updated clustering for fixed centers
Updated centers based on new clustering

K-means (3)

The K-means algorithm:

Input: Data points $X_1, ..., X_n \in \mathbb{R}^d$, number K of clusters to construct.

1. Randomly initialize the centers $m_1^{(0)}, ..., m_K^{(0)}$.
2. Iterate until convergence:
   - 2.1 Assign each data point to the closest cluster center, that is define the clusters $C_1^{(i+1)}, ..., C_K^{(i+1)}$ by

$$X_s \in C_k^{(i+1)} \iff \|X_s - m_k^{(i)}\|^2 \leq \|X - m_l^{(i)}\|^2, l = 1, ..., K$$

   - 2.2 Compute the new cluster centers by

$$m_k^{(i+1)} = \frac{1}{|C_k^{(i+1)}|} \sum_{s \in C_k} X_s$$

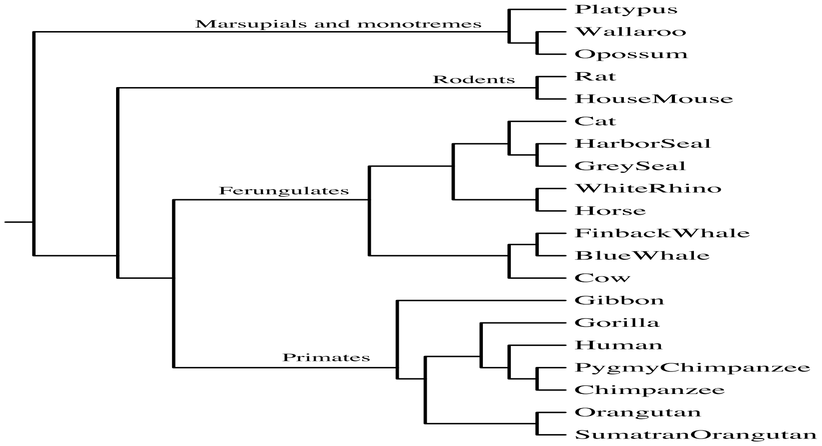
Output: Clusters $C_1, ..., C_K$

K-means (4)

matlab demo: `demo_kmeans()`

Linkage algorithms for hierarchical clustering

Goal: obtain a complete hierarchy of clusters and sub-clusters in form of a dendrogram



cf. Chen/Li/Ma/Vitanyi (2004)

Linkage algorithms for hierarchical clustering (2)

Linkage algorithms use an agglomerative (bottom-up) strategy:

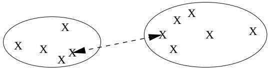
- Start: each point is its own cluster
- Then check which points are closest and "merge" them to form a new cluster
- Continue, always merge two "closest" clusters until we are left with one cluster only

The original article: S. C. Johnson. Hierarchical clustering schemes. Psychometrika, 2:241 - 254, 1967.
A complete book on the topic: N. Jardine and R. Sibson. Mathematical taxonomy. Wiley, London, 1971.
Nice overview with application in biology: J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. ISMB 1999.

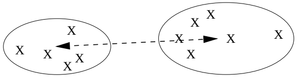
Linkage algorithms for hierarchical clustering (3)

To define which clusters are "closest":

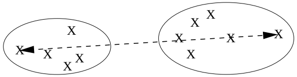
Single linkage: $dist(C, C') = \min_{x \in C, y \in C'} d(x, y)$



Average linkage: $dist(C, C') = \frac{\sum_{x \in C, y in C'} d(x,y)}{|C| \cdot |C'|}$



Complete linkage: $dist(C, C') = \max_{x \in C, y \in C'} d(x, y)$



Linkage algorithms for hierarchical clustering (4)

Input:

- Distance matrix D between data points (size $n \times n$)
- function *dist* to compute a distance between clusters (usually takes D as input)

Initialization: Clustering $\mathcal{C}^{(0)} = \{C_1^{(0)}, ..., C_n^{(0)}\}$ with $C_i^{(0)} = \{i\}$.

While the current number of clusters is > 1:

- ◦ find the two clusters which have the smallest distance to each other
- ◦ merge them to one cluster

Output: Resulting dendrogram

Linkage algorithms for hierarchical clustering (5)

... show matlab demos ...

```
demo_linkage_clustering_by_foot()
demo_linkage_clustering_comparison()
```

Linkage algorithms for hierarchical clustering (6)

- single linkage tends to generate long “chains”
- complete linkage tends to produce “round”, compact clusters
- linkage algorithms are very vulnerable to outliers
- one cannot “undo” a bad link
- Single linkage can also be described using the minimal spanning tree of data points (e.g., cutting the longest edge of an MST gives the first two single linkage clusters)
- Underlying model: Clustering = estimation of densities (see later lecture)
- Advantage of hierarchical clustering: do not need to decide on “the right” number of clusters
- There exist many more ways of generating different trees from a given distance matrix ...

Model-based clustering

Generative clustering (as opposed to discriminative):

- assume that data has been generated according to some probabilistic model
- want to estimate the parameters of the model
- Methods to do this: maximum likelihood, Bayesian approaches
- In practice: EM-algorithm
- Examples: Gaussian mixtures, hidden Markov models
- Advantage: lead to better interpretation of the clusters (as they come with a model)
- Disadvantage: how to specify the model? What if the model does not fit the data?

C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. JASA, 97:611 -631, 2002.
S. Zhong and J. Ghosh. A Unified framework for model-based clustering. JMLR, 4:1001 - 1037, 2003

Information-theoretic approaches

- Want to construct clusters such that as few “information” as possible is lost about the data
- Minimize some distortion function
- Purely based on probability counts, no similarity or dissimilarity needed
- Most popular approach: Information Bottleneck

N. Tishby, Fernando Pereira, and W. Bialek. The information bottleneck method. In Proceedings of the 37th Annual Al lerton Conference on Communication, Control and Computing, pages 368- 377, 1999.

METIS

A coarsen-refinement algorithm for partitioning graphs:

- Given a large graph
- "Coarsen" it by merging nodes of the graph to "super-nodes"
- Cluster the coarse graph
- Uncoarsen the graph again, thereby also refining the clustering

G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing, 20(1):359-392, 1999

Ensemble methods

(also called aggregation methods, or bagging)

- Repeatedly cluster the data using different algorithms, parameter settings, perturbations, ...
- Obtain an “ensemble” of different clusterings
- Implicitly contains the information which points might “really” belong in the same cluster, and about which points we are rather unsure
- Try to generate a final clustering of the ensemble.

Nice overview paper: A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. JMLR, 3:583-617, 2002.

Subspace clustering

- If data is very high-dimensional: different reasonable clusterings might exist in different "subspaces"
- Example: customer data base. Can cluster customers along shopping preferences, or along age, or along money they spend ...
- Goal is to come up with a list of subspaces and the corresponding clusters.

Review article: L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. SIGKDD Explor. Newsl., 6(1):90-105, 2004.
A completely different perspective on this problem: Paper by Tishby et al at NIPS 2005 about increasing feature dimension, **TO DO** look up

Co-clustering

also called bi-clustering

- Want to find clusters in different domains simultaneously
- Example: term/document co-occurrence matrix. Want to cluster words and texts simultaneously such that I identify groups of words which are typical for certain groups of texts
- Most approaches work iteratively: first cluster along one dimension; then use those clusters as input to cluster along the other dimension; repeat until convergence

Some references:
J. Hartigan. Direct clustering of a data matrix. JASA, 67(337):123- 129, 1972.
S. C. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: A survey. IEEE transactions on computational biology and bioinformatics, 1(1): 24-45, 2004
S. Climer and W. Zhang. Rearrangement clustering: Pitfalls, remedies, and applications. JMLR, 7:919-943, 2006

??? Thousands of clustering algorithms ???

- Have seen: there is a huge variety of clustering algorithms
- Is there a method to compare them?
- Which one should we choose?

What is clustering, after all???

What is clustering after all? Theoretic Approaches

What makes clustering so different from classification?

- Unsupervised: We don't get "hints" in form of labels. But this alone is not the problem (there exist other unsupervised problems which are easier to define, for example density estimation).
- There exists no ground truth. OK, that counts as a reason. This makes it very hard to *define* what clustering is. But once we came up with a definition?
- A clustering is a *global* property. In classification, we can eventually just look at local neighborhoods and optimize some function on this neighborhood, independently of what happens at other neighborhoods. In clustering, the results at some part of the space might affect our opinion on a completely different region. Thus very difficult from an computational point of view.

Defining clustering by a quality function

Most straight forward way to define a clustering of a given data set:

- Define an objective function (clustering quality function) Q
- Find the partition which minimizes Q

Examples:

- K-means; spectral clustering

There exist many such quality functions!!! Most of them look at one or both of the following intuitive quantities:

- The within-cluster similarity (should be high)
- The between-cluster similarity (should be low)

Defining clustering by a quality function (2)

Advantages:

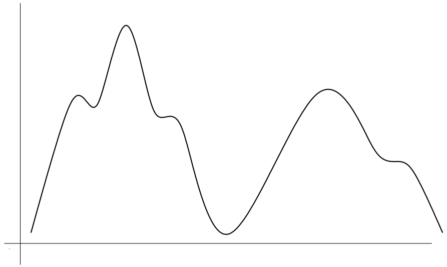
- Well-defined mathematical object
- Can use standard optimization techniques

Disadvantages:

- Choice of quality function is rather heuristic
- Optimization problems are usually NP hard

Defining clustering: high density areas

A cluster is a high density area. Different clusters are separated by low density areas. But this is still a bit fuzzy ....



What people often do to make definition more precise:

- Clusters are density level sets (here have to choose the level)
- Clusters correspond to modes of a density (here again: problem of scale)

Defining clustering: high density areas (2)

Example: single linkage (implicitly)

Advantages:

- Intuitively makes sense
- Well-defined mathematical object
- Can use standard statistical techniques for estimating densities, level sets, the support of density, modes of a density, ...

Disadvantages:

- Suggests that to perform clustering we first have to estimate the density :-(
- At any price, avoid estimating densities ... won't work even in case of a moderate number of dimensions
- Hope: often don't estimate the full density, but just some aspects (for example for level set estimation; single linkage)

Defining clustering: axiomatic view

A clustering is a function defined on the space of data sets which satisfies a certain set of axioms.

Such axioms include:

- Invariance with respect to rotation and translation
- Invariance with respect to isotropic scaling
- Moving points closer together which are in the same clustering does not change the clustering
- The class of all clusterings is "rich enough", i.e. it can generate many different clusterings

- ... many more ...

Defining clustering: axiomatic view (2)

Once axioms are defined, people usually show:

- that there exists a clustering method that satisfies all axioms
- hopefully, that method can even be made explicit
- ideally, it is unique
- even more ideally, people show that their algorithm always satisfies those axioms :-)

N. Jardine and R. Sibson. Mathematical taxonomy. Wiley, London, 1971.
W. Wright. A formalization of cluster analysis. Pattern Recognition, 5:273 - 282, 1973.
S. Ben-David. Can Clustering be Axiomatized? Rethinking Kleinberg's Impossibility. Talk given at the NIPS 2005 Workshop on Theoretical Foundations of Clustering, 2005.

Defining clustering: axiomatic view (3)

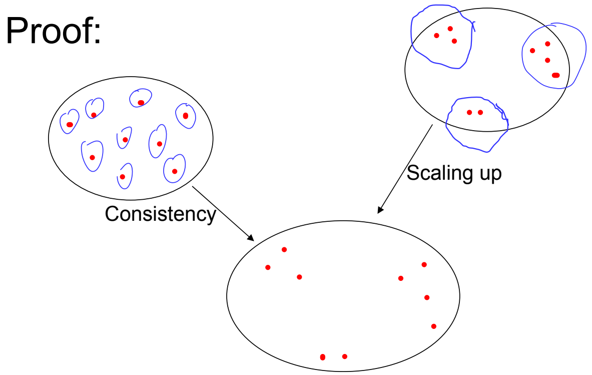
A really popular negative result: Kleinberg (2003)

J. Kleinberg. An impossibility theorem for clustering. NIPS 2002.

- Clustering: maps distance matrix to partition of $\{1, ..., n\}$
- Three axioms:
  - Scale invariance
  - "Richness": Any clustering of $\{1, .., n\}$ is possible: for any clustering there exists a distance matrix which induces this clustering.
  - "Consistency": If we shrink distances between points inside a cluster and expand distances between points in different clusters, then the clustering result does not change.
- All axioms sound perfectly harmless.
- But: one can prove that there does not exist any clustering function which can satisfy all three axioms!

Defining clustering: axiomatic view (4)

Original proof is pretty complicated, but here is a nice, simple argument:



S. Ben-David. Can Clustering be Axiomatized? Rethinking Kleinberg's Impossibility. Talk given at the NIPS 2005 Workshop on Theoretical Foundations of Clustering, 2005.

Defining clustering: axiomatic view (5)

Advantage of axiomatic approach:

- Avoids the question to define a clustering directly
- Clustering is just defined via properties
- Clean way to mathematically define clustering

Disadvantages:

- Which set of axioms??? Even though it sounds fundamental, it is pretty ad hoc!
- In most cases, not really helpful for practice.

Defining clustering: model based approach

We assume that the data has been generated by some probabilistic model. Implicitly this model defines what a clustering is.

Most popular example: Gaussian mixtures

- Assume that data comes from a distribution of the form $p(x) = \sum_k \pi_k N(\mu_k, \Sigma_k)$
- Clusters are simply mixture components: a point belongs to cluster k if it has been generated by $N(\mu_k, \Sigma_k)$

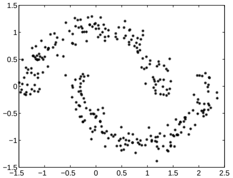
Advantages:

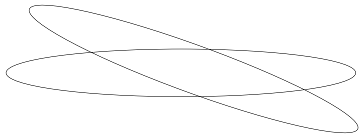
- Have a clear model of the data, and a clear interpretation what a cluster is
- Can use standard techniques to estimate the parameters (maximum likelihood, EM, Bayesian approaches, ...)

Defining clustering: model based approach (2)

Disadvantages:

- Parameter estimation is often not so easy...
- Often makes very strong assumptions which might not be satisfied.
  Example: Gaussian mixtures on the “two moons data set”
- Often, model is not really designed towards clusters.
  Example: Gaussian mixtures find two components on top of each other, but intuitively we think this is one cluster.





Defining clustering: information theoretic view

Clustering is a lossy compression of the information contained in the original data set.

- Either find the optimal clustering for a given “code length”
- Or find the optimal clustering for a specified “amount of loss” we are willing to accept.
- Often formulated in terms of rate-distortion theory.

Example: Information bottleneck approach.

Defining clustering: information theoretic view (2)

Advantages:

- Sounds quite natural, has a nice interpretation.

Disadvantages:

- Often not so clear what "original information" we are referring to (the coordinates of the points? Particular features?)
- Often not so easy to implement (minimize some distortion which has many local minima, often done by some simulated annealing scheme)

The best definition of clustering????

- Have seen many different definitions of what “clustering” is.
- None of the definitions can cover the full range of applications of clustering.
- For a given task, need to choose the one which fits best (which often is also not so easy ...)
- There are some tasks where it might not be so crucial which definition exactly one chooses (e.g., in applications where clustering is just one preprocessing step to reduce the amount of data)

One has to make a choice here, and no default exists!

Theory about clustering

Once one has chosen a definition for clustering, what theoretical statements can we make about it?

- First of all need to devise an algorithm which can implement this kind of clustering.
- (Often, this works the other way round: you start with an algorithm and then "define" clustering just to be the thing your algorithm does).

Once you have an algorithm:

- Can you prove that it really does what you want?
- Under which assumptions?
- Guarantees?
- Reliability?

Very little is known about those questions ...

Clustering in a statistical setting

- Assume that the data points have been sampled from some underlying probability distribution P on some space $\mathcal{X}$
- Ultimate goal: construct a partition of the underlying space $\mathcal{X}$
- To this end: Define a goal of clustering on the space $\mathcal{X}$! This takes into account P.
- Example 1: clusters are density level sets.
  Example 2: minimize average distance to class centers (average refers to P)

- Now we are only given a finite sample $X_1, ..., X_n$
- Have to estimate what a good partition of $\mathcal{X}$ would be.

U. von Luxburg and S. Ben-David. Towards a statistical theory of clustering. In PASCAL workshop on Statistics and Optimization of Clustering, London, 2005.

Clustering in a statistical setting (2)

In the following: want to look at statistical properties of clustering algorithms.

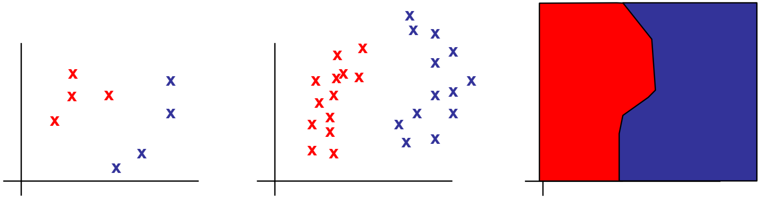
- Generally, we investigate whether a certain algorithm survives some minimal "sanity checks"
- Consider this as necessary conditions, not as sufficient ones!!!
- Just tells us that a clustering algorithm is not completely stupid. Does not guarantee anything on a particular data set.

- First property: want that the partitions converge.
- Second property: want that the "limit partition" makes sense.
- Third property: want that results on finite samples are stable (do not vary too much).

Consistency of clustering algorithms

Consistency, intuitively:

- The more data points I get, the more accurate can I estimate the "true" partition.
- Those estimated partitions "converge" to the true partition.



Not so easy to define ... and even more difficult to prove.

What is known on consistency of clustering algorithms? In fact, very little!!!

Consistency of clustering algorithms (2)

Consistency of K-means:

- On the finite sample: minimize $Q_n = \sum_k \sum_{i \in C_k} \|X_i - m_k\|^2$
- In the limit of $n \to \infty$, this corresponds to minimizing $Q = \sum_k \int \mathbb{1}_{X \in C_k} \|X - m_k\|^2 \, dP(X)$.
- Assume that we can always discover the global optimum of Q_n and Q.
- Then one can prove: the centers corresponding to the empirical global minimum of Q_n converge to the centers corresponding to the true (limit) global minimum of Q.
- Problem: the K-means *algorithm* does not find the global minimum. So this theorem is nice, but does not apply to the actual algorithm :-(

D. Pollard. Strong consistency of k-means clustering. Annals of Statistics, 9(1):135 - 140, 1981.

Consistency of clustering algorithms (3)

Consistency of single linkage:

- Can prove some "fractional consistency": A positive fraction of disjoint clusters will eventually be separated
- This is not really consistency :-(

Does not hold for average or complete linkage.

Limit results: J. Hartigan. Consistency of single linkage for high-density clusters. JASA, 76(374):388 - 394, 1981.
Finite sample analysis: Dasgupta. Performance guarantees for hierarchical clustering. COLT 2002. Related literature: random geometric graphs.

Consistency of clustering algorithms (4)

Consistency of spectral clustering:

- Normalized spectral clustering is usually consistent.
- It converges to a nice limit clustering.

- Unnormalized spectral clustering is not always consistent.
- Can converge to trivial solutions.
- It is possible to characterize the situation when this happens.

- This is an example where one can seen why consistency analysis of algorithms is important!!!

As a consequence: use normalized spectral clustering only.

U. von Luxburg, M. Belkin, O. Bousquet. Consistency of spectral clustering. Annals of Statistics, to appear. See also technical report.

Consistency of clustering algorithms (5)

Consistency of model based clustering

- Here we can rely on the statistics literature:
- Need to estimate parameters of models.
- In principle, by using maximum likelihood or Bayesian approaches this can often be done in a consistent way.
- In practice, we have the same problem as in K-means: the algorithms usually don't globally minimize their objective function.
- Moreover, even if we have consistency: in case the data has not been generated by the class of models under investigation, the clustering result might not be very meaningful.

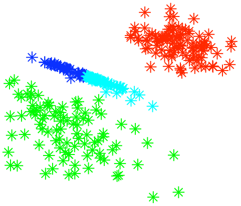
Theoretical foundations of clustering – summary

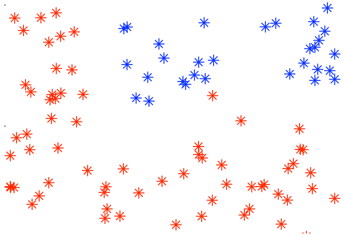
- Cannot avoid to make choices depending on task: goal of clustering, algorithm to achieve this, ...
- Very little theoretical guidelines in this process.
- Once we made choices, would like to have theoretical guarantees on the behavior of the algorithm. Often do not exist ...
- My point of view: instead of making guarantees, make sanity checks. Does the algorithm at least behave correctly for some theoretical properties (convergence, stability, complexity, ...)?
- Lots of research waiting to be done :-)

Selecting the number of clusters: practice and some destructive theory

How many clusters are there?

- Most flat clustering algorithms need to get the number K of clusters as input
- No matter what the structure in the data is, they will return K clusters as output
- But what if we don't know how many clusters we are looking for???





Answer depends on application...

Example: Using clustering for image segmentation

- Clusters should represent "meaningful" (parts of) objects in the image.
- Ultimate goal is to obtain a correct segmentation.
- So far, this can only be judged by a human, and I don't see a clear mathematical way to do it.
- Use heuristics, need experience to do this ...

Answer depends on application... (2)

If clustering is just used as a preprocessing step to reduce the complexity of the data, for example in a classification task:

- Choose K pretty large (in order not to loose too much information about your data)
- but only as large as you can afford later on (trade-off between computational costs of next steps and number of clusters)
- might also be a trade-off between number of clusters and overfitting, once K is too large.
- Try to evaluate the clustering parameters with cross-validation over your final classification result.

Answer depends on application... (3)

In a statistical setting (we will focus on this in the following):

- Assume that data has been drawn according to some probability distribution
- How can we find out what "the true number of clusters" is?
- Note: ultimate goal is to learn a partition of the underlying space
- But have to do this based on finite amount of samples.

- Often can assume a statistical setting: bioinformatics, social networks, customers, astronomy, ...
- Here clustering is not just a preprocessing step, we really want to evaluate the result!

First attempt to define "the correct K"

Define: Given the distribution P, what is the correct clustering?

- Sometimes this already implies what "the true K" is.
  - Example: "Clusters are disconnected components of the density."
  - Here K is uniquely defined.
- In most cases, have a parameter which directly or indirectly controls the number of clusters.
  - Example: "Clusters are disconnected components of level sets of the density." Here depending on the level t, different number of K possible.
  - Example: K-means
  - **Then we need a second definition: Given P, what is the "correct" number of clusters?**

Given a finite sample, now want to estimate K by some K_n.
Have to prove that estimator converges to the correct one.

Choosing K using an objective function

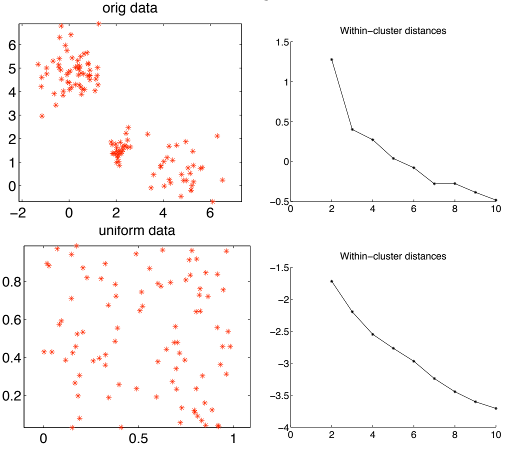
First idea: use an objective function such as K-means.

- For each k in a reasonable range, let Q_k be the value of the objective function of K-means with k clusters.
- Choose $K = argmin_k Q_k$

This has a big problem: which one?

Choosing K using an objective function (2)

The values Q_k usually scale with k.



Need to correct (normalize) the values Q_k!!!

Choosing K using an objective function (3)

Normalizing the value Q_k: compare to unclustered "uniform" data.

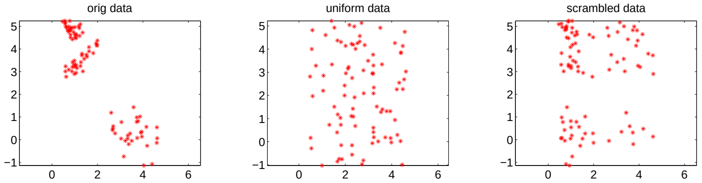
- Generate "unclustered" reference data on the same domain
  - Use uniform distribution on data domain (Problematic, in particular in high dimensions. Why?)
  - Scramble the data: permute each feature randomly (also problematic, see next slide)
- Then cluster reference data for different values of k and compute $Q_{k,uniform}$
- Then consider the curve of $Q_k / Q_{k,uniform}$ and take the minimum $\rightsquigarrow K$

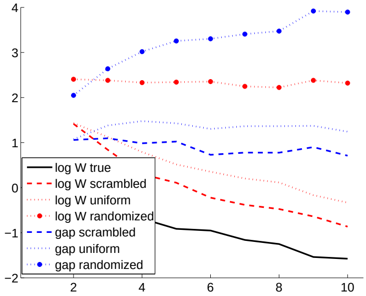
This is called the gap statistics.

Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. J. Royal. Statist. Soc. B, 63(2):411-423, 2001.

Choosing K using an objective function (4)

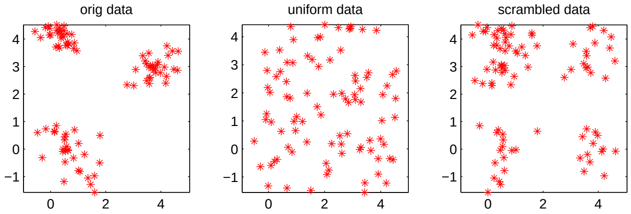
Example 1: gap statistics

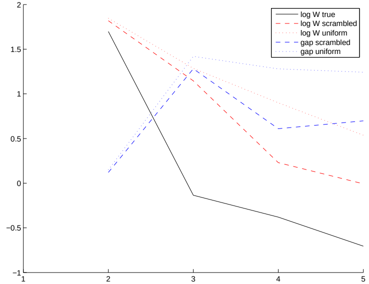




Choosing *K* using an objective function (5)

Example 2: gap statistics





How many clusters – stability approach

Scientific results should be reproducible!

Thus necessary requirement: use algorithms which are "stable".

Idea: evaluate clusterings indirectly using stability.

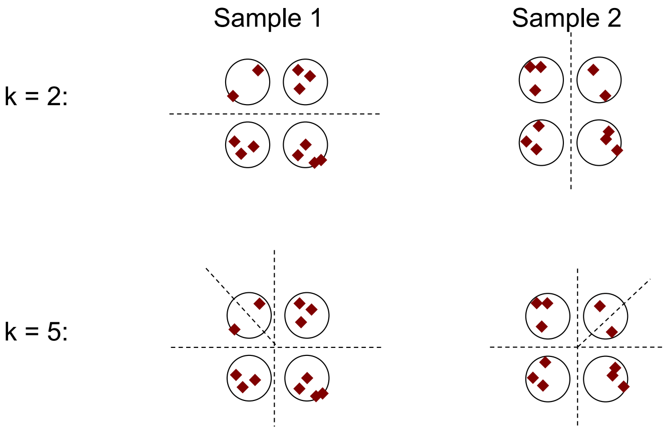
- Want that our results are stable.
- Hence, choose parameter for which the result is most stable.
- In practice, this often "works"
- However, it is not really clear what it really does.

Stabilty – the general principle

- Given a data set $X_1, ..., X_n$, a clustering algorithm $\mathcal{A}$
- For different values of k (=number of clusters):
  - draw subsamples of the given data
  - cluster them in k clusters using $\mathcal{A}$
- compare the resulting “bootstrap” clusterings
  - define some distance between clusterings
  - compute some notion of “stability” depending on how much the clustering distances vary
- choose the parameter k which gives the “best” stability (where “best” is defined in different ways)

The toy figure in favor of stability

How many clusters?



Distances between clusterings: same data set

Want to define a distance between two clusterings $f^{(1)}$ and $f^{(2)}$ of the same data set.

For $i, j \in \{0, 1\}$ define N_{ij} = number of pairs of points for which $f^{(1)}(X, Y) = i$ and $f^{(2)}(X, Y) = j$

Many possible distance or similarity functions functions:

- Rand index: $(N_{00} + N_{11})/(n(n-1))$
- Jacard index: $N_{11}/(N_{11} + N_{01} + N_{10})$
- Hamming distance (L_1-distance on pairs):
  $(N_{01} + N_{10})/(n(n-1))$

Distances between clusterings: same data set (2)

- Variation of information distance: information theoretic distance, measures how much we know about clustering2 if we already know clustering1:

  Entropy(clust1) + entropy(clust2) - MutInf(clust1,clust2)

M. Meila. Comparing clusterings: an axiomatic view. In Proceedings of the International Conference of Machine Learning (ICML), pages 577-584, 2005.

Can see: Many distances/similarities are possible ...
The actual choice is often not sooo important!

Distances between clusterings: different data sets

More general:

- have a clustering $\mathcal{C}_1$ of the first data set $X_1, ..., X_n$
- and a second clustering $\mathcal{C}_2$ of a second data set $X'_1, ..., X'_m$
- and we assume that both data sets come from the same domain

Two ways to define a distance between the clusterings: using restriction or extension operator.

Distances between clusterings: different data sets (2)

Restriction operator:

- compute the joint domain $S = \{X_1, ..., X_n\} \cap \{X'_1, ..., X'_m\}$ of both clusterings
- Restrict both clusterings to $S \rightsquigarrow \mathcal{C}'_1, \mathcal{C}'_2$
- Compute distance between $\mathcal{C}'_1$ and $\mathcal{C}'_2$ (easy as now defined on same domain)
- Note that this only makes sense if the two domain have a reasonable overlap.

Distances between clusterings: different data sets (3)

Extension operator:

- Extend both clusterings from their domain to the domain of the other clustering (or even to the whole underlying space)
- Then compute a distance between the resulting clusterings (easy as now defined on same domain)
- For some algorithms there exist natural extensions:
  - K-means (just assign new points to the closest cluster center)
  - single linkage (assign new points to the same cluster as the closest data point belongs to)
  - spectral clustering (using theory of integral operators, not trivial)
- If one only needs to extend to a few new points: greedy heuristic.
- Otherwise: use a classifier as extension operator!

Stability approaches in the literature

- Stability is used in many different flavors.
- Want to introduce the most common ones.

Stability: Levine and Domany (2001)

For each value of k in some reasonable range:

- Cluster the full data set
- Repeatedly (for $r = 1, \ldots, r_{max}$):
  - Draw a subsample and cluster it
  - Compute a distance d_r between the clustering of the full set and the one of the subset using restriction operator
- Compute stability $stab(k) = mean_r(d_r)$

Choose the parameter k for which $stab(k)$ is minimal

(apparently do not normalize the values $stab_k$;
this is problematic as $stab_k$ always scales with k)

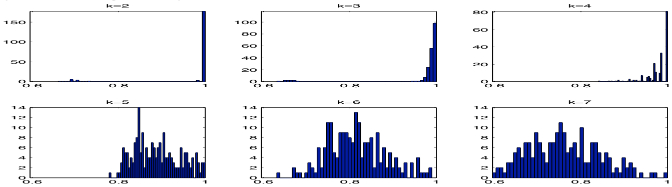
E. Levine and E. Domany. Resampling Method for Unsupervised Estimation of Cluster Validity. Neural Computation 13 (11), 2001.

Stability: Ben-Hur, Elisseeff and Guyon (2002)

For each value of k in some reasonable range:

- Repeatedly (for $r = 1, \ldots, r_{max}$):
    - Draw subsamples $\mathcal{S}_1$ and $\mathcal{S}_2$ and cluster them
    - Compute a similarity s_r using restriction operator.
- Generate histogram of the similarities s_r between the clusterings.

Choose the parameter k for which histogram is most concentrated (by looking for jumps in area under cumulative distribution).



A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In Pacific Symposium on Biocomputing, pages 6 - 17, 2002.

Stability: Fridlyand and Dudoit (2001)

For each value of k in some reasonable range:

- Repeatedly (for $r = 1, \dots, r_{max}$)
  - Randomly split the given data set into two halves.
  - Cluster both halves independently.
  - Extend both clusterings from "their" half to the other half using a classification algorithm.
  - Compute distance d_r between both clusterings
- Compute stability $stab(k) = mean_r(d_r)$
- For normalization:
  - Randomly generate b_{max} reference data sets under a suitable null hypothesis (e.g., uniform distribution).
  - Repeat the procedure outlined above for those data sets and compute the stability values $stab_b(k)$.

Stability: Fridlyand and Dudoit (2001) (2)

With a bootstrap test: choose k for which the difference of $stab(k)$ to $mean_b(stab_b(k))$ is most significant.

J.Fridlyand and S.Dudoit. Applications of resampling methods to estimate the number of clusters and to improve the accuracy of a clustering method. Technical Report 600, 2001.

Stability: Lange, Roth, Braun, Buhmann (2004)

For each value of k in some reasonable range:

- Repeatedly (for $r = 1, \dots, r_{max}$)
    - Randomly split the given data set into two halves.
    - Cluster both halves independently.
    - Extend both clusterings from "their" half to the other half using a classification algorithm.
    - Compute distance d_r between both clusterings
- Compute the stability $stab(k) = mean_r(d_r)$.
- For normalization:
    - Consider the same splits as above
    - Instead of clustering each split, assign random labels to points
    - Extend both "random clusterings" as above, compute distances, and evaluate stability: $\rightsquigarrow stab_{rand}(k)$

Choose the parameter k such that $stab(k)/stab_{rand}(k)$ is minimal.

LanRotBraBuh04T. Lange, V. Roth, M. Braun, and J. Buhmann. Stability-based validation of clustering solutions. Neural Computation, 16(6):1299 - 1323, 2004.

matlab demo on gap statistics and stability

Theoretical statements?

Under certain assumptions on the distribution P,
and for certain basis algorithms $\mathcal{A}$,

- ... what can we prove about the parameter K chosen with stability?
- Is it reasonable?
- In simple examples, does it select "the right" K?
- How large does the sample need to be?
- Does it "converge"? Confidence statements?
- Are there any fundamental differences between the approaches of the literature?

Critical view on the stability approach

Formalize what we mean by stability:

- Consider a clustering algorithm $\mathcal{A}$ which minimizes some empirical cluster quality function Q_{emp}.
- Assume that the algorithm always finds a global minimum of Q_{emp} (no convergence issues).
- Denote by S_n, $\tilde{S}_n$ two independent samples of size n drawn i.i.d. according to probability distribution $\mathbb{P}$.
- Let d be a distance function between clusterings.
- Define stability of algorithm $\mathcal{A}$ with respect to sample size n:

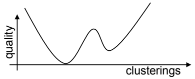
$$stab(\mathcal{A}, n) := \mathbb{E}_{S,\tilde{S}} \; d(\mathcal{A}(S_n), \mathcal{A}(\tilde{S}_n))$$

Critical view on the stability approach (2)

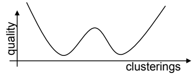
Theorem:

- Assume that Q has a unique global minimum. Then any clustering algorithm A which minimizes Q_{emp} in some consistent way is stable for large n, that is $\limsup_{n\to\infty} stab(A, n) = 0$.
- Assume that the global minimum of Q is not unique. Then A is not stable.

▶ If q has a unique global minimum, then A is stable.



▶ If q has several global minima, then A is instable!



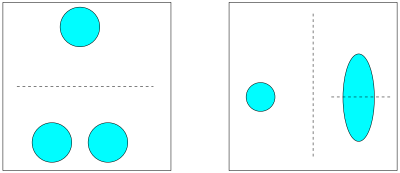
Ben-David, Luxburg, Pal. A sober look at clustering stability. COLT 2006.

Pal, Ben-David, Simon: Stability of K-means clustering. COLT 2007.

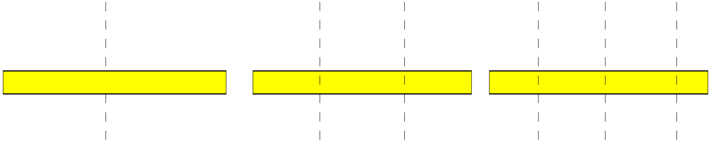
Critical view on the stability approach (3)

The counter-stability toy figures: stability even for "wrong" k

- Non-symmetric distribution: stability even for "wrong" k

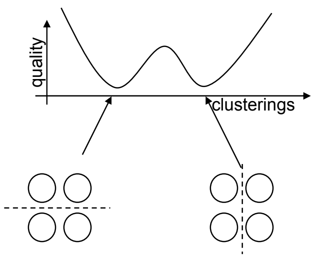


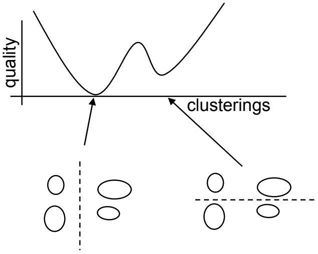
- Uniform distribution on $[0, 1]$: k-means is stable for all k



Critical view on the stability approach (4)

- Several global minima are often induced by symmetry
- Natural distributions are usually not perfeclty symmetric
- In this case for large n: every k is stable!





For large *n* we usually have stability for every *k*!!!

Critical view on the stability approach (5)

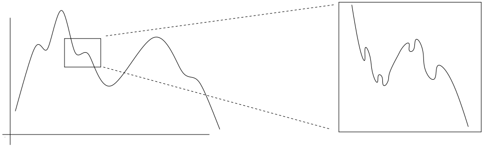
- Stability just tells us whether we can reliably find the global optimum
- In the results are instable, this can have many different reasons (e.g., symmetry).
- Those reasons are not necessarily related to the number of clusters.

Summary gap statistics and stability

- both try to determine the unique "correct" K
- in practice, both of them "kind of work"
- both are heuristics
- both have no theoretical guarantees whatsoever
- for lack of better alternatives: reasonable to use them
- but don't be surprised if weird things happen
- never do this as a black box without looking at the results ...

The "correct" K, second approach

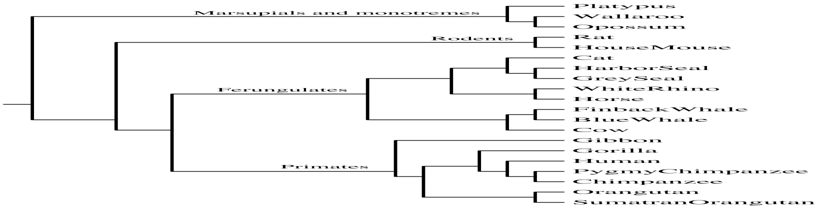
- Even if we know P, we can justify different numbers of clusters, depending on the "the scale" or "the resolution" we use to look at the distribution.
  Example: "Clusters correspond to modes of the density."



- We can even have infinitely many clusters of P.
- Now goal is different: On the finite sample, construct as many reliable clusters as possible
  - If n is small, only look for major clusters.
  - The larger n, the more clusters should be constructed.
  - But always make sure that the clusters are not just sampling artifacts.

The “correct” K, second approach (2)

- Hierarchical clustering is a standard tool in this context
- Avoids the question what the “correct” clustering is, just outputs a complete hierarchy.
- Only issue might be: want to know at which level in the hierarchy clusters are reliable and at which level we are just fitting noise ...



The “correct” K, second approach (3)

- Example for a way to find out: Still and Bialek (2004):
  - Clustering = data compression
  - Want to minimize some distortion function $dist(clustering, data)$
  - Distortion depends on the underlying distribution, and is usually minimized by $K = \infty$.
  - Now take into account uncertainty when estimating distortion function on finite sample.
  - Minimize something like “empirical distortion” + “safety term”
  - Turns out: is minimized for a particular K

S. Still and W. Bialek. How many clusters? an information-theoretic perspective. Neural Comput., 16(12):2483 - 2506, 2004.

The “correct” K, second approach (4)

- I believe that stability is a good tool for this second approach
- intuitively, stability just measures whether we are fitting noise
- then one would say: any number of clusters is reasonable, as long as the results do not get really unstable
- but of course we need some theory

That is a topic I am working on right now ...

How many clusters? Summary

- Well, it is really not simple!!!
- Depending on your goal:
  - Might be very hard to define mathematically (eg image segmentation)
  - Want to find a fixed K
  - Want to look at clusters at different resolutions
- There exist methods which work sometimes (often? in clear cases?): gap statistic, stability
- Other than that: play around ...

Wrapping up

Wrapping up

What I didn't talk about:

- Choosing the similarity function between the data points: is crucial, but of course not simple ...
- Issues in high-dimensional data: distances can become meaningless; density estimation is impossible; can find many clusters when looking at different dimensions; ...
- Efficient implementations: exist for most algorithms, see literature ...
- Online algorithms, algorithms for treating data streams, clustering "moving targets", ...

Wrapping up (2)

There are lots of clustering algorithms out there!
There are not much theoretical guidelines out there!

If you use clustering for exploratory data analysis:

- Play around with different algorithms, parameters, ...
- Try not to use algorithms which make too strong assumptions – unless you know that those assumptions hold in your case.
- Go for the more powerful algorithms (kernel K-means, spectral clustering, ...)
- You always need to check manually whether your results make sense.

- Playing around cannot really be automated.

Wrapping up (3)

If you use clustering as a pre-processing step:

- Usually one wants to find a rather high number of clusters
- In this case, use simple algorithms (eg K-means)
- Sometimes the choice of parameters can be evaluated using cross-validation, e.g. on the final classification result
- But often sub-optimal parameters do not completely kill your application.

Wrapping up (4)

Finally:

- Clustering as a field of research is really fun to work on!
- There is not much theory done yet.
- You still have the freedom to make "your theory", as no standards exist.

Actually, we are looking for PhD students and postdocs ;-)