Lecture 14: High-dimensional Clustering

Felix Held, Mathematical Sciences

MSA220/MVE440 Statistical Learning for Big Data

17th May 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Administrative

- There will be **no** project 5!
- Only projects 1–4 will be part of the exam.
- However, the topics discussed today and next week could still be part of another exam question
- No office hours next week, but there will be some after the course is over (dates and times to be announced)

Short re-cap: Clustering

The goal of **clustering** is to find coherent groups in data. Clusters are generally considered to be dense, well-separated groups of data.

What shapes qualify as a cluster depends on the algorithm

- convex equally sized groups (e.g. k-means)
- convex or concave equally dense groups (e.g. DBSCAN, single-linkage hierarchical clustering)

Clustering is typically distance-based, making it difficult to use in **high-dimensions**

High-dimensional clustering

Remember: Nothing is close in high-dimensional spaces

Assume vectors $\mathbf{x} \in \mathbb{R}^p$ are uniformly distributed in the hypercube $[0,1]^p$. For $0 \leq t \leq 1$ consider

$$q = P(x_1 \leq t, \ldots, x_p \leq t) = t^p \quad \Rightarrow \quad t = q^{1/p}$$

To ensure that q percent of vectors are contained in the cube $[0,t]^p$ means that t has to be chosen as

	$\mathbf{q = 1\%}$	$\mathbf{q = 10\%}$
$\mathbf{p = 2}$	$t = 0.1$	$t = 0.32$
$\mathbf{p = 3}$	$t = 0.22$	$t = 0.46$
$\mathbf{p = 10}$	$t = 0.32$	$t = 0.56$
$\mathbf{p = 100}$	$t = 0.63$	$t = 0.79$

More than half of the unit cube needs to be covered to cover 1% of random vectors in high dimensions!

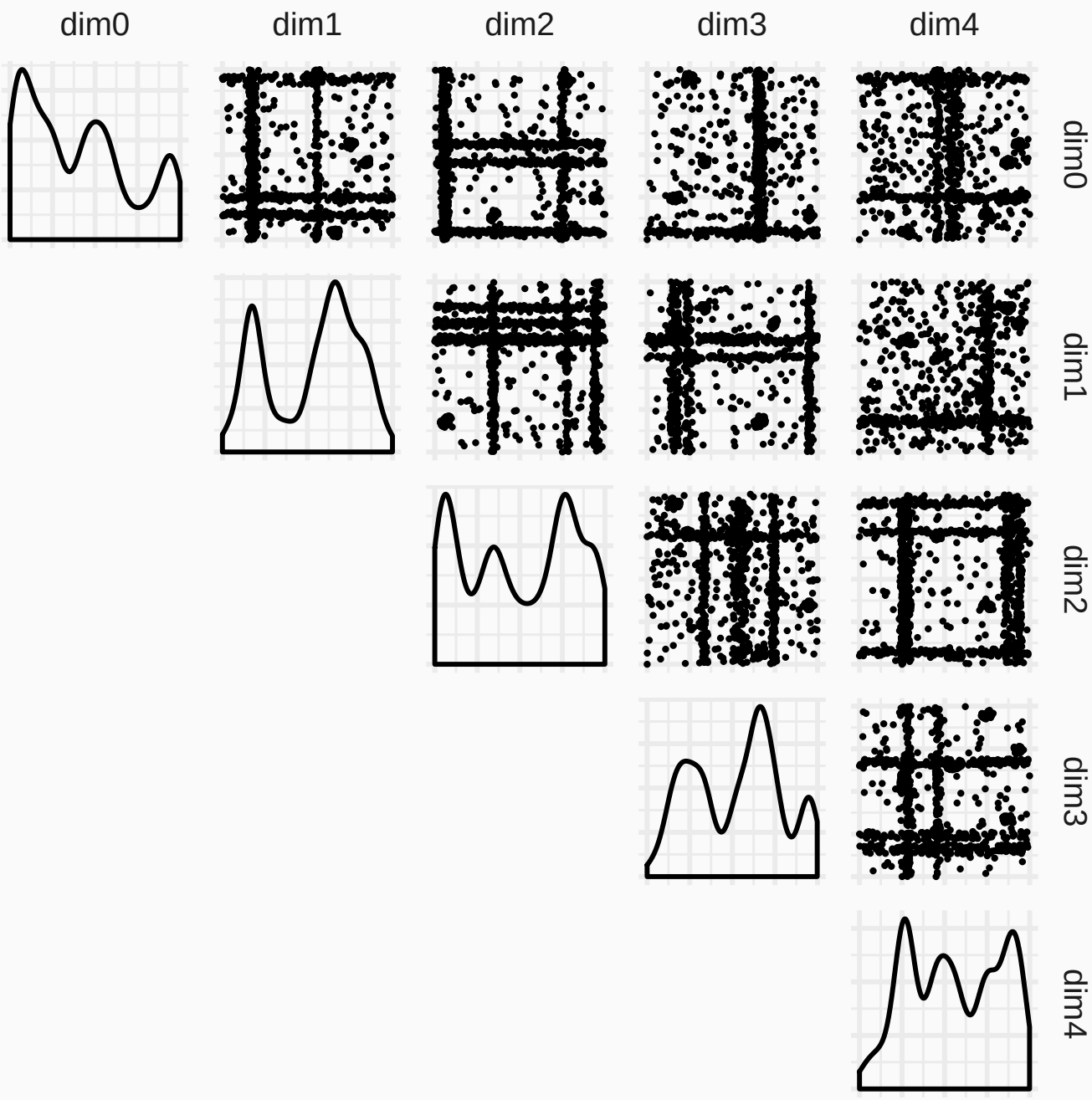
Solutions for high-dimensional clustering

What can be done about this dilemma?

1. **Feature selection:** Deciding on a subset of the original features
   - There is no response in clustering, making it harder to judge feature quality
   - Variance or entropy-based methods remain, but typically consider variance across all samples. Features very relevant to only a subset might get filtered out
2. **Feature transformation:** Combining existing features while reducing dimension
   - e.g. PCA, Isomap, tSNE, ...
   - Output is harder to interpret, e.g. what does it mean if observations cluster on their first and second tSNE coordinate?
   - Since features are transformed, it is not guaranteed that uninformative features are actually filtered out

Subspace clustering

Data in many subspaces



Subspace clustering

Instead of selecting single features, sometimes it would be better if we could select whole subspaces of features.

How can these be found?

- It is infeasible to look at all possible subspaces
- As in combinatorial clustering or stepwise selection methods in regression there are two approaches
  1. **Top-down:** Start with all dimensions and search for relevant dimensions
  2. **Bottom-up:** Start with a grid in each dimension and combine them step-wise

Examples:

- CLIQUE: Bottom-up algorithm; grid-based and density-based
- ProClus: Top-down algorithm; variant of k-medoids

The CLIQUE algorithm

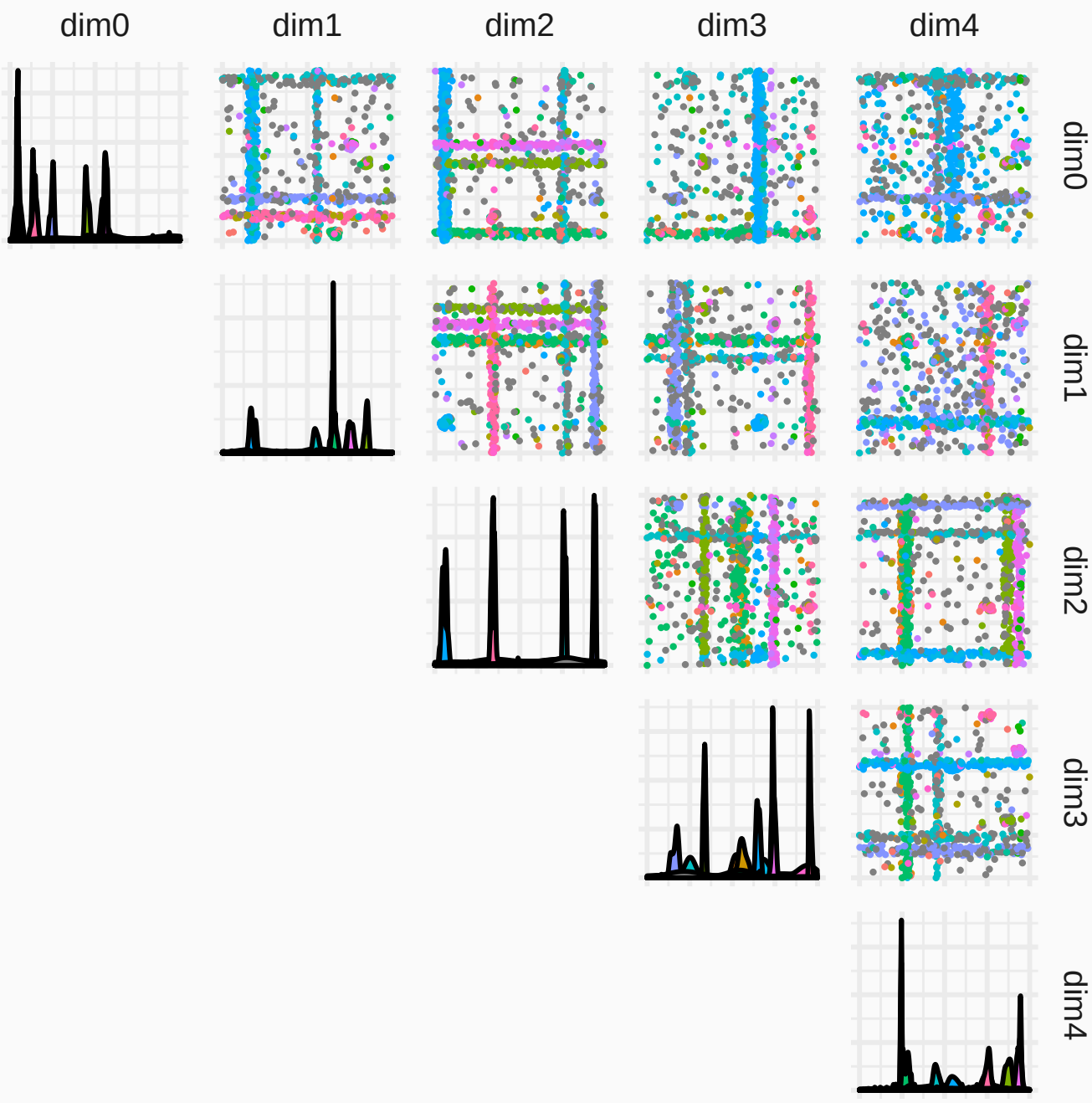
Apriori principle: A dense region in q dimensions should lead to dense regions in every $(q-1)$-dimensional projection.

Computational procedure

Input parameters: A positive integer m and $1 > \tau > 0$

1. **For 1D:**
   - 1.1 Partition each dimension $i = 1, \ldots, p$ into m intervals and assign the one-dimensional projections of the data
   - 1.2 Keep those with sample density $> \tau$
   - 1.3 Among the remaining intervals, merge neighbouring ones
2. **Moving from dimension $q-1$ to q:**
   - 2.1 Create volumes in q dimensions by combining those found in $q-1$ dimensions.
   - 2.2 Keep those with sample density $> \tau$
   - 2.3 Among the remaining volumes, merge neighbouring ones
3. **Post-processing:** Filter out remaining regions with low density and try to enlarge found clusters as much as possible

Subspace clustering: CLIQUE



The ProClus algorithm

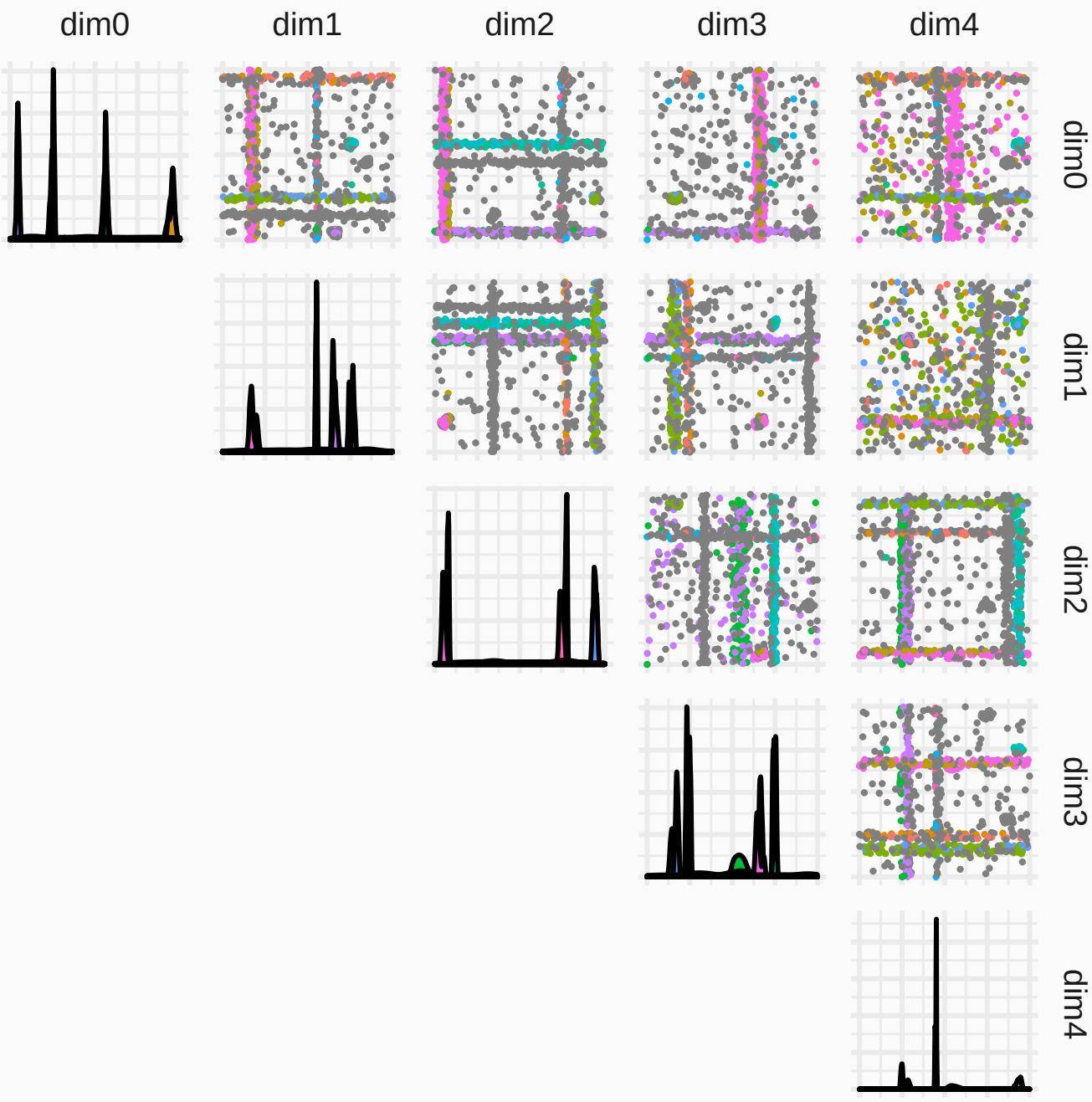
Computational procedure

Input parameters: The number of K and the average number of dimensions $d > 0$ in which clusters reside

1. **Initialising:** Find $M > K$ medoids in a greedy fashion and randomly sample K of these
2. **Iterate:** Until no change within some threshold
   - 2.1 For each medoid: Find best dimensions (≥ 2) where data is dense (dimensions in which average distance to the medoid is smaller than the overall average distance)
   - 2.2 Assign data points to medoids using the ℓ_1 norm restricted to the selected dimensions
   - 2.3 Evaluate clustering quality and remove medoids with small numbers of points. Replace them with others from the initial set of medoids
3. **Refining:** Determine the best dimensions once more for each medoid and assign points to clusters

Aggarwal et al. (1999) Fast algorithms for projected clustering. SIGMOD '99 Proceedings of the 1999 ACM SIGMOD international conference on Management of data

Subspace clustering: ProClus



Notes on subspace clustering

- **Pro:** Can deal with complex structures and high-dimensions
- **Pro:** The variable selection/subspace discovery that is performed per cluster can lead to mechanistic insight into a problem
- **Con:** Hard to tune in high-dimensions since it is difficult to get an understanding for the data (e.g. grid-size, average number of subspace dimensions, ...)
- Some adaptive algorithms exist to e.g. estimate optimal grid-size from data

Spectral methods

Starting point

- Many clustering methods focus on **global behaviour** of the data (e.g. GMM, k-means, hierarchical clustering with complete linkage)
- To adapt to **local behaviour** hierarchical clustering with single linkage and the group of **density-based** algorithms (e.g. DBSCAN, OPTICS) showed some success
- In dimension reduction **building a graph of the data** based on k nearest neighbours helped to capture **local behaviour**
- **Idea:** Build a graph representing local behaviour in the data and find good cut points to partition the graph into K clusters.

A graph from data

Recall: In the first step of Isomap, a **weighted undirected graph** was built based on the k nearest neighbours of a data point.

A weighted undirected graph can be constructed from a **weighted adjacency matrix $\mathbf{W}$**.

1. For a data point $\mathbf{x}_l$, find the k nearest neighbours.
2. Set $w_{ll_i} = g(\|\mathbf{x}_l - \mathbf{x}_{l_i}\|_2)$ where g is a non-negative monotone function and $\mathbf{x}_{l_i}$, $i = 1, \dots, k$ are the neighbours of $\mathbf{x}_l$. In addition, set all $w_{lm} = 0$ for $m \notin \{l_1, \dots, l_k\}$ (in particular $w_{ll} = 0$).
3. Construct a graph where each node represents a data point $\mathbf{x}_l$ and there is a weighted edge between $\mathbf{x}_l$ and $\mathbf{x}_m$ if $w_{lm} > 0$.

In Isomap: $g(z) = z$. In the following: $g(z) = \exp(-z^2/c)$, which for $c = \infty$ is interpreted as $g(0) = 0$ and $g(z) = 1$ for $z > 0$.

Node degree and the graph Laplacian

Given the weighted adjacency matrix $\mathbf{W}$, the **degree of node** l describes how well-connected a node is

$$d_l = \sum_{m=1}^{n} w_{lm}$$

and the **degree matrix** is $\mathbf{D} = \mathrm{diag}(d_1, \dots, d_n)$.

Define now the **graph Laplacian** as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

Interpretation: If heat were to be distributed from node to node with flow speed described by $\mathbf{W}$, then $\mathbf{L}$ takes the role of the discretised **Laplacian operator** ∇^2 in the **heat equation** for the heat distribution $\boldsymbol{\phi}$

$$\frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} + k\mathbf{L}\boldsymbol{\phi} = \mathbf{0}$$

Graph cutting

A **good separation of the graph** into two parts A and B is one where flow between the parts is minimized and neither is chosen too small, i.e.

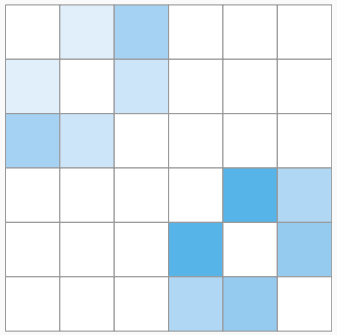
$$\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right) \sum_{l \in A, m \in B} w_{lm} \to \min$$

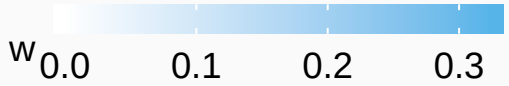
where

$$\mathrm{vol}(A) = \sum_{l \in A} \sum_{m=1}^{n} w_{lm} = \sum_{l \in A} d_l$$

Raw data and graph

Graph edge weights





Clustering result

Finding good cut points

Finding the best cut point would require to check all possible cuts and is an **NP-hard problem**.

Observations and theorem

1. The graph Laplacian is symmetric and positive semi-definite, since $\mathbf{y}^T \mathbf{L}\mathbf{y} = \sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^2$
2. If there are K connected components of the graph, then the set of eigenvectors of $\mathbf{L}$ with eigenvalue 0 is spanned by $\mathbf{1}_{A_k}$ for $k = 1, \ldots, K$, where $\mathbf{1}_{A_k} = (\mathbb{1}(i \in A_k))_i$.

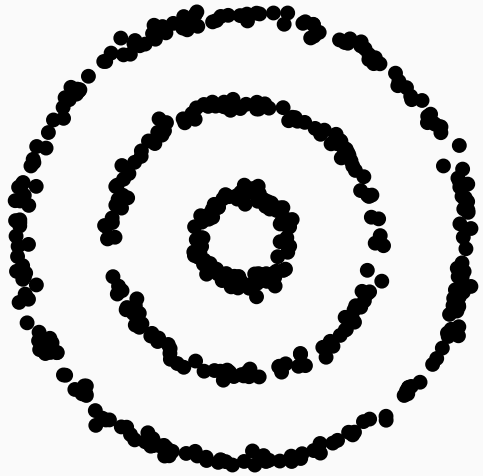
In practice

- There will not be K separate connected components
- However, if K clusters exist, the smallest K eigenvalues will be near zero and the and corresponding eigenvalues close to indicator vectors.

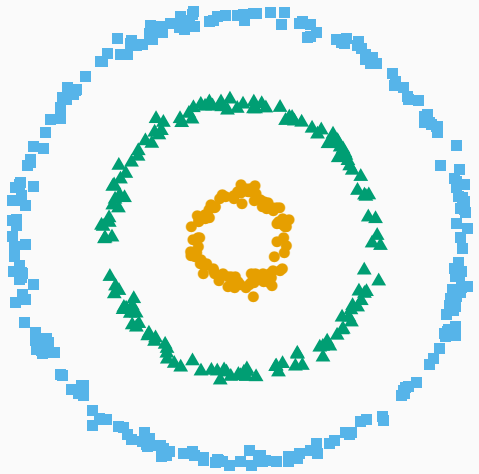
Spectral Clustering

1. Determine the weighted adjacency matrix $\mathbf{W}$ and the graph Laplacian $\mathbf{L}$
2. Find the K smallest eigenvalues of $\mathbf{L}$ that are near zero and well separated from the others
3. Find the corresponding eigenvectors $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_K) \in \mathbb{R}^{n \times K}$ and use k-means on the rows of $\mathbf{U}$ to determine cluster membership

Raw data (n = 500



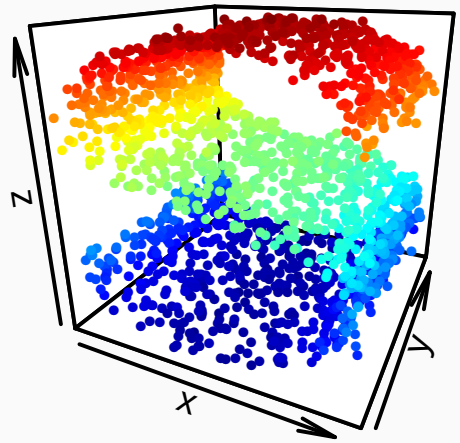
Spectral clustering



Laplacian Eigenmaps for dimension reduction

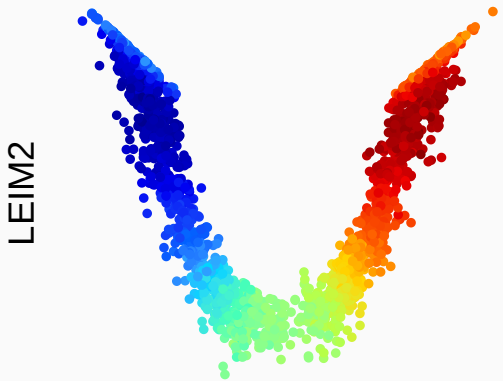
- In addition to clustering, the eigenvectors of the Laplacian can also be used for **dimension reduction**.
- For each component, use the q eigenvectors corresponding to the q smallest non-zero eigenvalues as an embedding of the original data.
- Laplacian Eigenmaps can be shown to **optimally preserve the local behaviour on average**, but not necessarily global behaviour.

S curve (n = 2000)



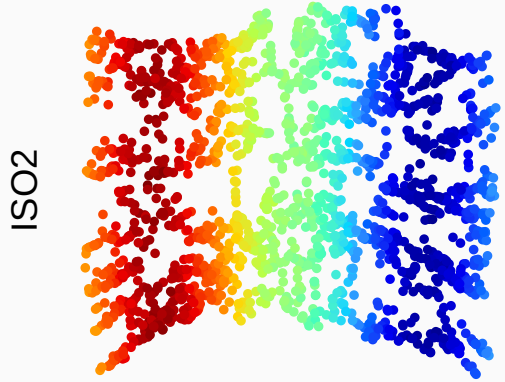
Laplacian Eigenmaps

knn = 6, c = ∞



Isomap

knn = 6



Network graphs

Network Graphs

Some common **network graphs** that are estimated from data on one set of variables (the **nodes**) are

- **Correlation graph:** Undirected edges weighted by the correlation between variables.
  Caveat: Correlation can be due to a common ancestor.
- **Partial correlation graph:** Undirected edges weighted by the correlation between variables **given all other variables**. This measures how much correlation is left once all other variables are controlled for.
- **Directed acyclic graphs:** Weighted directed egdes without cycles. Can describe causality but are much harder to estimate.

As in the linear models class, note that

Correlation does not imply causality.

Partial correlation graphs

Let each feature/variable denote a node in a graph, e.g. $x_1, \ldots, x_p$ for feature vectors $\mathbf{x} \in \mathbb{R}^p$.

The weight of the edge between variables x_i and x_j is the **partial correlation**

$$\mathrm{Corr}(x_i, x_j | x_k, k \neq i, j) = \rho_{ij} = \rho_{ji}$$

- If $\rho_{ij} = 0$ there is no edge between x_i and x_j.
- ρ_{ij} captures the information left after controlling for x_k for $k \neq i, j$, i.e. the correlation that cannot be explained through a common ancestor or connection

Partial correlation and the normal distribution

Assume that feature vectors are distributed as

$$\mathbf{x} \sim N(\mathbf{0}, \mathbf{\Sigma})$$

them $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ is called the **precision matrix**. It can be shown that with $\mathbf{\Omega} = (\omega_{ij})_{ij}$

- $\rho_{ij} = -\dfrac{\omega_{ij}}{\sqrt{\omega_{ii}\omega_{jj}}}$
- With $\mathbf{x}_{-i} = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_p)$

$$p(x_i|\mathbf{x}_{-i}) = N\left(-\sum_{j\neq i}\frac{\omega_{ij}}{\omega_{ii}}x_j, \frac{1}{\omega_{ii}}\right)$$

It is therefore enough to estimate the precision matrix and shows that if $0 = \omega_{ij} = \rho_{ij}$ then there is no dependence of x_i on x_j, and vice versa.

Estimating the precision matrix

It can be shown that the likelihood of the precision matrix $\mathbf{\Omega}$ given the empirical covariance matrix

$$l(\mathbf{\Omega}) = \log(|\mathbf{\Omega}|) - \mathrm{tr}(\widehat{\mathbf{\Sigma}}\mathbf{\Omega})$$

- Can be used to estimate $\mathbf{\Omega}$ with iterative methods.
- In general, all entries will be non-zero and therefore all edges will be present in the resulting network.

Precision matrix and graph structure

1. For a **known graph structure Γ** where $\Gamma_{ij} \in \{0, 1\}$ the constrained problem

$$\arg\min_{\boldsymbol{\Omega}} -l(\boldsymbol{\Omega}) \quad \text{subject to} \quad \omega_{ij} = 0 \Leftrightarrow \gamma_{ij} = 0$$

   can be solved.
2. If the **graph structure** is **unknown** lasso regularisation can help to remove some edges. This leads to

$$\arg\min_{\boldsymbol{\Omega}} -l(\boldsymbol{\Omega}) + \lambda \sum_{i<j} |\omega_{ij}|$$

   which can be solved with neighbourhood regression-based lasso or gradient-based lasso. This is called the **Graphical lasso (glasso)**.

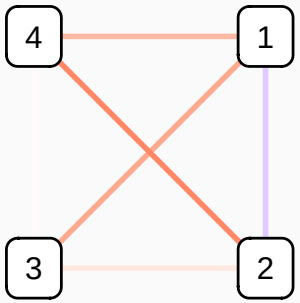
   Note: In the R packages they use ρ for the regularisation parameter

Example of network estimation

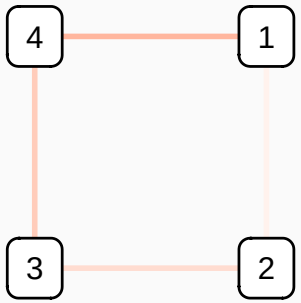
Assume the following (made-up) empirical covariance matrix and graph structure

$$\widehat{\mathbf{\Sigma}} = \begin{pmatrix} 10 & 1 & 5 & 4 \\ 1 & 10 & 2 & 6 \\ 5 & 2 & 10 & 3 \\ 4 & 6 & 3 & 10 \end{pmatrix} \quad \text{and} \quad \mathbf{\Gamma} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Direct estimate

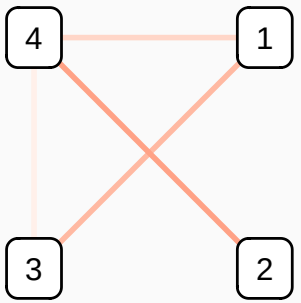


Known graph structure



Estimated graph structur

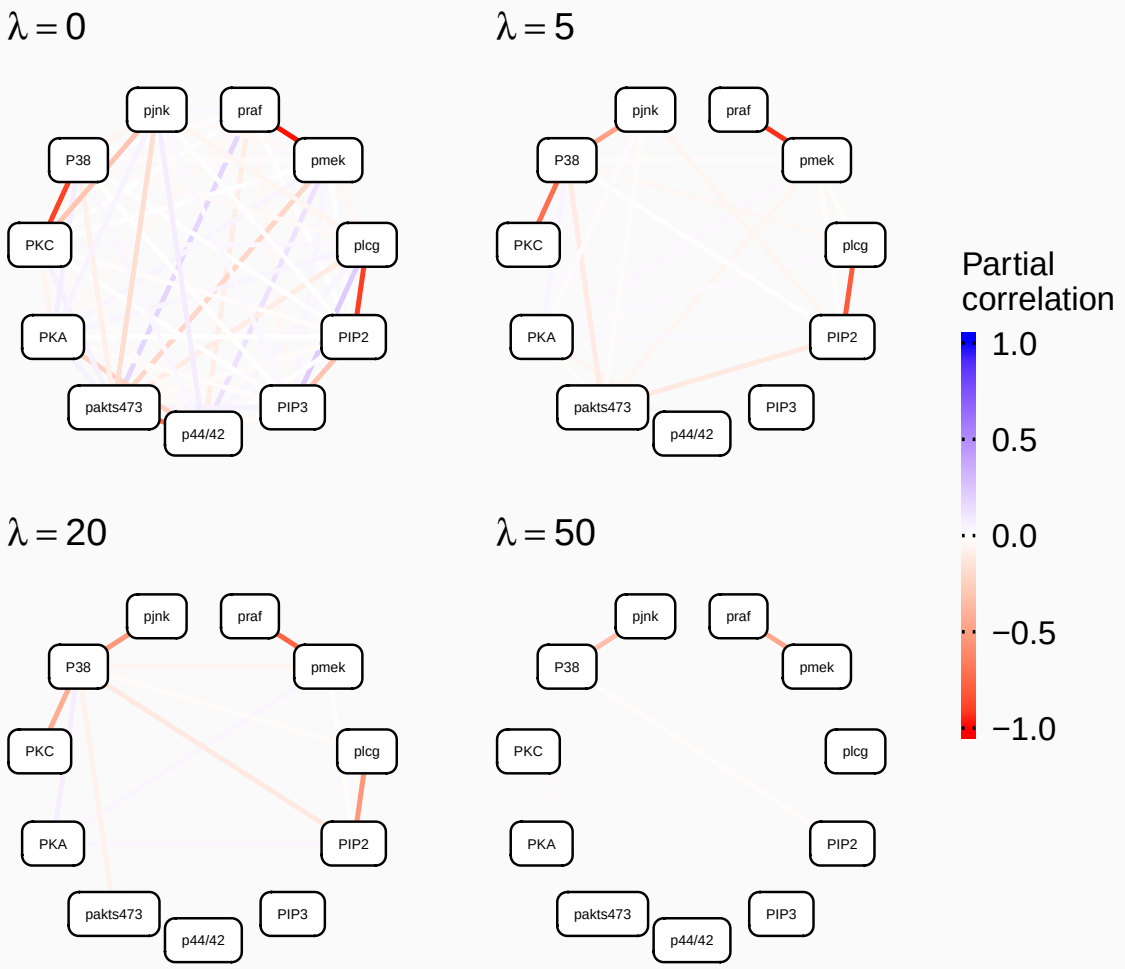
$\lambda = 1$



Graphical lasso: More advanced example

Protein flow cytometry data
($n = 7466$ cells,
$p = 11$ proteins)

Network was estimated repeatedly from the covariance matrix to show the effect of regularisation on structure estimation



Notes on Graphical lasso

- As with any lasso method, the two main caveats are
  1. If too many variables are highly correlated, the network graph cannot be identified
  2. Is the true data generating process sparse?
- Bootstrapping and re-running the network estimation often can help to get more stable results
- If data is too high-dimensional, pre-processing such as filtering can be necessary
- Another approach for big-n: If a network graph has fallen into K components for a certain λ_0, then the Graphical lasso can be run on each component separately for $\lambda > \lambda_0$.

Take-home message

- Subspace clustering is class of algorithms that try to find clusters that live in subspaces of features
- Spectral embedding and clustering uses similar ideas to Isomap but exploits the spectral/eigenvalue decomposition of the graph Laplacian
- Graph structures are highly versatile and can be used in different contexts
- Graphical lasso is one of many network estimation algorithms using the lasso to find a sparse network structure