Ludwig-Maximilians-Universität München
Lehrstuhl für Datenbanksysteme und Data Mining
Prof. Dr. Thomas Seidl

Knowledge Discovery and Data Mining 1

(Data Mining Algorithms 1)

Winter Semester 2019/20

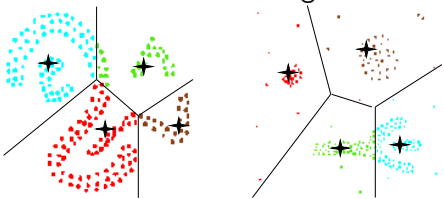
Agenda

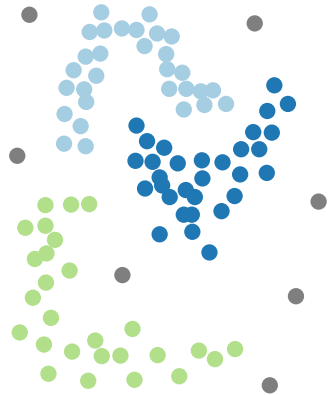
Density-Based Clustering

Basic Idea

Clusters are dense regions in the data space, separated by regions of lower density

Results of a k-medoid algorithm for $k = 4$:





Density-Based Clustering: Basic Concept

Note

Different density-based approaches exist in the literature. Here we discuss the ideas underlying the DBSCAN algorithm.

Intuition for Formalization

- For any point in a cluster, the local point density around that point has to exceed some threshold
- The set of points from one cluster is spatially connected

Density-Based Clustering: Basic Concept

Local Point Density

Local point density at a point q defined by two parameters:

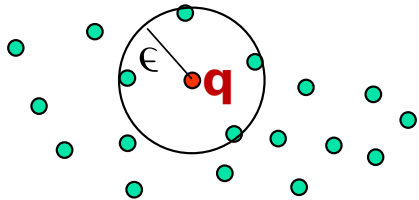
- ϵ-radius for the neighborhood of point q

$$N_\epsilon(q) = \{p \in D \mid dist(p, q) \leq \epsilon\} \tag{1}$$

In this chapter, we assume that $q \in N_\epsilon(q)$!

- *MinPts*: minimum number of points in the given neighbourhood $N_\epsilon(q)$.

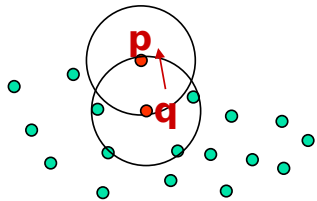
Density-Based Clustering: Basic Concept

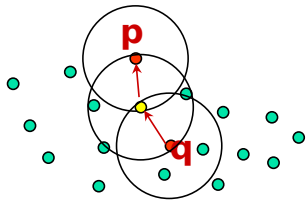


Core Point

q is called a core object (or core point) w.r.t. ϵ, *MinPts* if $|N_\epsilon(q)| \geq minPts$

Density-Based Clustering: Basic Definitions





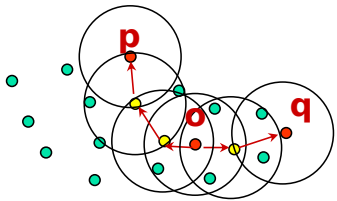
(Directly) Density-Reachable

p directly density-reachable from q w.r.t. ϵ, *MinPts* if:

1. $p \in N_\epsilon(q)$ and
2. q is core object w.r.t. ϵ, *MinPts*

Density-reachable is the transitive closure of directly density-reachable

Density-Based Clustering: Basic Definitions



Density-Connected

p is *density-connected* to a point *q* w.r.t. ϵ, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. ϵ, *MinPts*

Density-Based Clustering: Basic Definitions

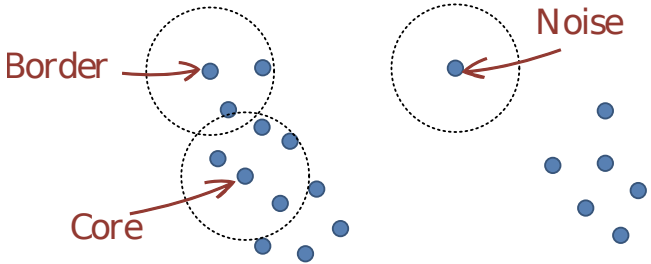
Density-Based Cluster

$\emptyset \subset C \subseteq D$ with database D satisfying:

Maximality: If $q \in C$ and p is density-reachable from q then $p \in C$

Connectivity: Each object in C is density-connected to all other objects in C

Density-Based Clustering: Basic Definitions



Density-Based Clustering

A partitioning $\{C_1, \ldots, C_k, N\}$ of the database D where

- $C_1, \ldots, C_k$ are all density-based clusters
- $N = D \setminus (C_1 \cup \ldots \cup C_k)$ is called the *noise* (objects not in any cluster)

Density-Based Clustering: DBSCAN Algorithm

Basic Theorem

- Each object in a density-based cluster C is density-reachable from any of its core-objects
- Nothing else is density-reachable from core objects.

Density-Based Clustering: DBSCAN Algorithm

Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise[1]

1: **for all** $o \in D$ **do**
2: **if** o is not yet classified **then**
3: **if** o is a core-object **then**
4: Collect all objects density-reachable from o and assign them to a new cluster.
5: **else**
6: Assign o to noise N

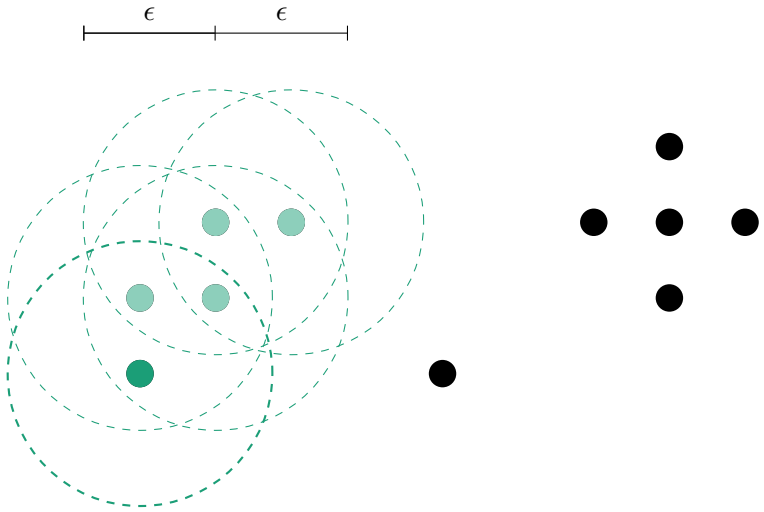
Note

Density-reachable objects are collected by performing successive ϵ-neighborhood queries.

[1] Ester M., Kriegel H.-P., Sander J., Xu X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", In KDD 1996 , pp. 226-231.

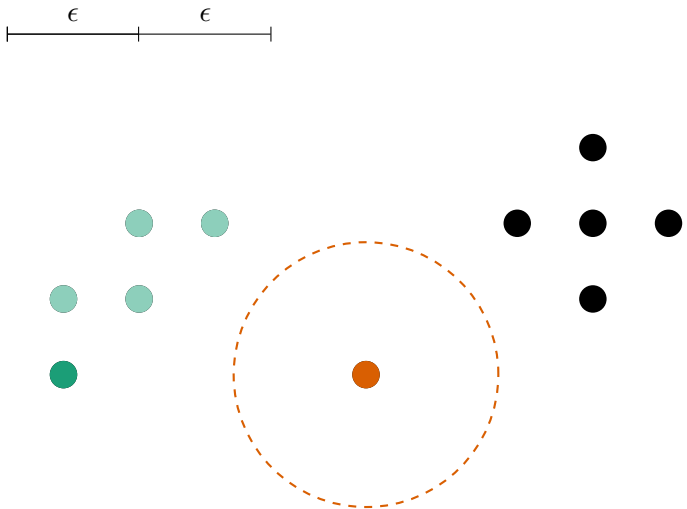
DBSCAN: Example

Parameters: $\epsilon = 1.75$, $minPts = 3$. Clusters: C_1



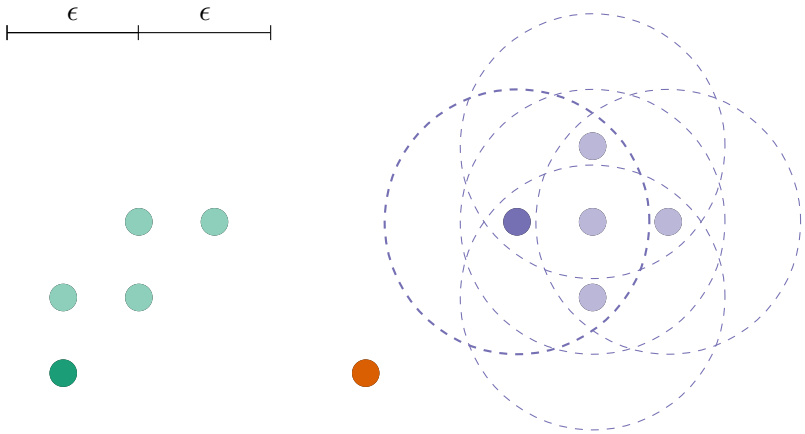
DBSCAN: Example

Parameters: $\epsilon = 1.75$, $minPts = 3$. Clusters: C_1; Noise: N



DBSCAN: Example

Parameters: $\epsilon = 1.75$, $minPts = 3$. Clusters: C_1, C_2; Noise: N



Determining the Parameters ϵ and *MinPts*

Recap

Cluster: Point density higher than specified by ϵ and *MinPts*

Idea

Use the point density of the least dense cluster in the data set as parameters.

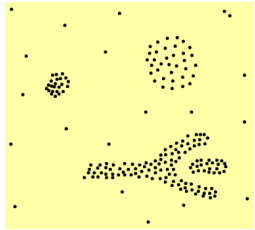
Problem

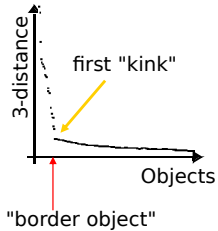
How to determine this?

Determining the Parameters ϵ and *MinPts*

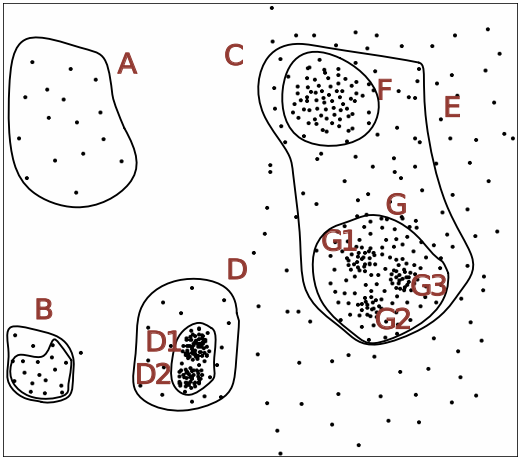
Heuristic

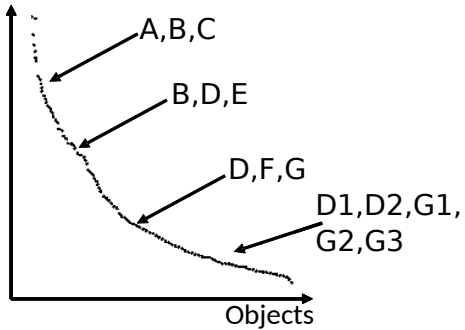
1. Fix a value for *MinPts* (default: $2d - 1$ where d is the dimension of the data space)
2. Compute the k-distance for all points $p \in D$ (distance from p to the its k-nearest neighbor), with $k = minPts$.
3. Create a k-distance plot, showing the k-distances of all objects, sorted in decreasing order
4. The user selects "border object" o from the *MinPts*-distance plot: ϵ is set to *MinPts*-distance(o).





Determining the Parameters ϵ and *MinPts*: Problematic Example





Database Support for Density-Based Clustering

Standard DBSCAN evaluation is based on recursive database traversal. Böhm et al.[2] observed that DBSCAN, among other clustering algorithms, may be efficiently built on top of similarity join operations.

ϵ-Similarity Join

An ϵ-*similarity join* yields all pairs of ϵ-similar objects from two data sets Q, P:

$$Q \bowtie_\epsilon P = \{(q, p) \in Q \times P \mid dist(q, p) \leq \epsilon\}$$

SQL Query

SELECT $*$ FROM Q, P WHERE $dist(Q, P) \leq \epsilon$

[2] Böhm C., Braunmüller, B., Breunig M., Kriegel H.-P.: *High performance clustering based on the similarity join*. CIKM 2000: 298-305.

Database Support for Density-Based Clustering

ϵ-Similarity Self-Join

An ϵ-similarity *self* join yields all pairs of ϵ-similar objects from a database D.

$$D \bowtie_\epsilon D = \{(q, p) \in D \times D \mid dist(q, p) \leq \epsilon\}$$

SQL Query

SELECT $*$ FROM D q, D p WHERE $dist(q, p) \leq \epsilon$

Database Support for Density-Based Clustering

The relation "directly ϵ, *MinPts*-density reachable" may be expressed in terms of an ϵ-similarity self join (abbreviate *minPts* with μ):

$$\begin{aligned} ddr_{\epsilon,\mu} &= \{(q,p) \in D \times D \mid q \text{ is } \epsilon,\mu\text{-core-point} \wedge p \in N_\epsilon(q)\} \\ &= \{(q,p) \in D \times D \mid dist(q,p) \leq \epsilon \wedge \exists_{\geq \mu} p' \in D : dist(q,p') \leq \epsilon\} \\ &= \{(q,p) \in D \times D \mid (q,p) \in D \bowtie_\epsilon D \wedge \exists_{\geq \mu} p'(q,p') \in D \bowtie_\epsilon D\} \\ &= \sigma_{|\pi_q(D \bowtie_\epsilon D)| \geq \mu}(D \bowtie_\epsilon D) =: D \bowtie_{\epsilon,\mu} D \end{aligned}$$

SQL Query

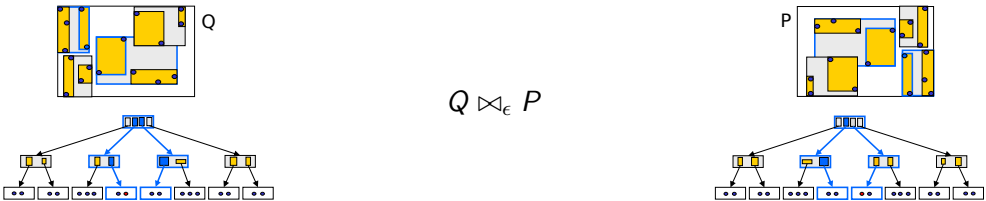
SELECT $*$ FROM D q, D p WHERE $dist(q,p) \leq \epsilon$ GROUP BY $q.id$ HAVING $count(q.id) \geq \mu$

Afterwards, DBSCAN computes the connected components of $D \bowtie_{\epsilon,\mu} D$.

Efficient Similarity Join Processing

For very large databases, efficient join techniques are available

- Block nested loop or index-based nested loop joins exploit secondary storage structure of large databases.
- Dedicated similarity join, distance join, or spatial join methods based on spatial indexing structures (e.g., R-Tree) apply particularly well. They may traverse their hierarchical directories in parallel (see illustration below).
- Other join techniques including sort-merge join or hash join are not applicable.



DBSCAN: Discussion

Advantages

- Clusters can have arbitrary shape and size; no restriction to convex shapes
- Number of clusters is determined automatically
- Can separate clusters from surrounding noise
- Complexity: N_ϵ-query: $\mathcal{O}(n)$, DBSCAN: $\mathcal{O}(n^2)$.
- Can be supported by spatial index structures ($\rightsquigarrow$ N_ϵ-query: $\mathcal{O}(\log n)$)

Disadvantages

- Input parameters may be difficult to determine
- In some situations very sensitive to input parameter setting

Agenda

Iterative Mode Search

Idea

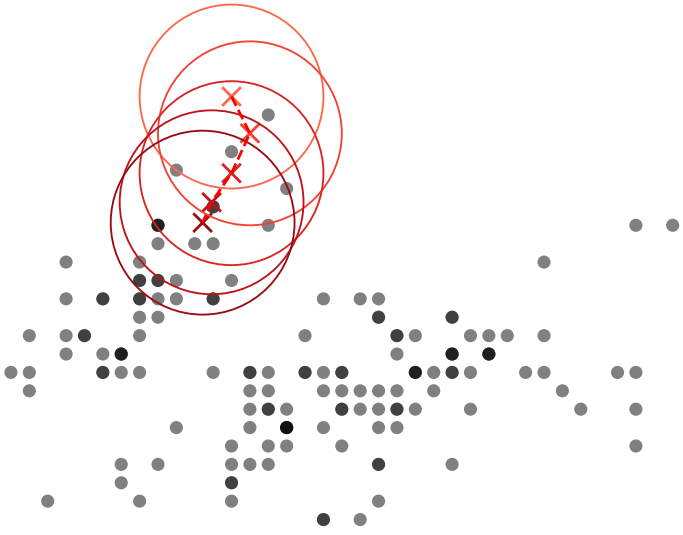
Find modes in the point density.

Algorithm[3]

1. Select a window size ϵ, starting position m
2. Calculate the mean of all points inside the window $W(m)$.
3. Shift the window to that position
4. Repeat until convergence.

[3] K. Fukunaga, L. Hostetler: *The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition*, IEEE Trans Information Theory, 1975

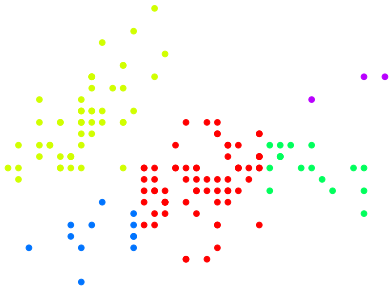
Iterative Mode Search: Example



Mean Shift: Core Algorithm

Algorithm[4]

Apply iterative mode search for each data point. Group those that converge to the same mode (called *Basin of Attraction*).



[4] D. Comaniciu, P. Meer. *Mean shift: A robust approach toward feature space analysis*. IEEE Trans. on pattern analysis and machine intelligence, 2002

Mean Shift: Extensions

Weighted Mean

Use different weights for the points in the window, with weights w_x, resp. calculated by some kernel κ:

$$m^{(i+1)} = \frac{\sum_{x \in W(m^{(i)})} w_x \cdot x}{\sum_{x \in W(m^{(i)})} w_x} \quad \rightarrow \quad m^{(i+1)} = \frac{\sum_{x \in W(m^{(i)})} \kappa(x) \cdot x}{\sum_{x \in W(m^{(i)})} \kappa(x)}$$

Binning

First quantise data points to grid. Apply iterative mode seeking only once per bin.

Mean Shift: Discussion

Disadvantages

- Relatively high complexity: N_ϵ-query (=windowing): $\mathcal{O}(n)$. Algorithm: $\mathcal{O}(tn^2)$

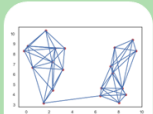
Advantages

- Clusters can have arbitrary shape and size; no restriction to convex shapes
- Number of clusters is determined automatically
- Robust to outliers
- Easy implementation and parallelisation
- Single parameter: ϵ
- Support by spatial index: N_ϵ-query (=windowing): $\mathcal{O}(\log n)$. Algorithm: $\mathcal{O}(tn \log n)$

Agenda

General Steps for Spectral Clustering I

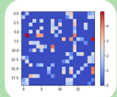
Construct Graph out of Data 1



using:
- kNN
- ε-neighborhood
- fully-connected graph

2

- (weighted) adjacency matrix W
- degree matrix D



- laplacian matrix L:
  unnormalized (D – W)
  normalized

3

problem to solve:

$$fLf^T = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 = \min$$

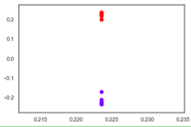
Solution:
calculate eigenvalues λ ,
eigenvectors v of matrix L

General Steps for Spectral Clustering II

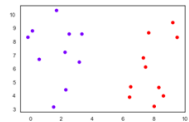
Choose usefull Number of Eigenvalues 4

- k smallest eigenvalues (k: #cluster)
- determine number by:
  - gap in eigenvalues
  - using eigenvectors
    (by matrix rotation and cost function)

Apply k-means on k Eigenvectors 5



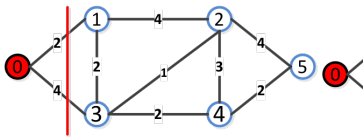
Map Results back to Original Data 6



Clustering as Graph Partitioning

Approach

- Data is modeled by a similarity graph $G = (V, E)$
  - Vertices $v \in V$: Data objects
  - Weighted edges $\{v_i, v_j\} \in E$: Similarity of v_i and v_j
  - Common variants: ϵ-neighborhood graph, k-nearest neighbor graph, fully connected graph
- Cluster the data by partitioning the similarity graph
  - Idea: Find global minimum cut
    - Only considers inter-cluster edges, tends to cut small vertex sets from the graph
    - Partitions graph into two clusters
  - Instead, we want a *balanced multi-way partitioning*
  - Such problems are NP-hard, use approximations



Spectral Clustering

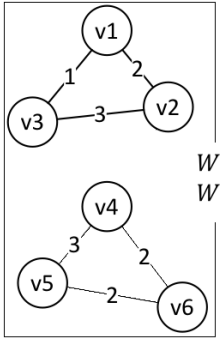
Given

Undirected graph G with weighted edges

- Let W be the (weighted) adjacency matrix of the graph
- And D its degree matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}$; other entries are 0

Aim

Partition G into k subsets, minimizing a function of the edge weights between/within the partitions.



$W[2,3] = 3$
$W[2,5] = 0$

2 connected components

Spectral Clustering

Idea

- Consider the *indicator vector* f_C for the cluster C, i.e.

$$f_C{}^{(i)} = \begin{cases} 1 & \text{if } v_i \in C \\ 0 & \text{else} \end{cases}$$

  and e.g. the *Laplacian* matrix $L = D - W$
- Further, consider the function $fLf^T = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(f_i - f_j)^2$ (derivation see exercise)
  - Small if f corresponds to a good partitioning
  - Given an indicator vector f_C, the function $f_C L f_C^T$ measures the weight of the inter-cluster edges! (see next slide)
  - Since L is positive semi-definite we have $fLf^T \geq 0$
  - Formulate a minimization problem on fLf^T

Connected Components and Eigenvectors

- General goal: find indicator vectors minimizing function fLf^T besides the trivial indicator vector $f_C = (1, \ldots, 1)$
- Problem: Finding solution is NP-hard (cf. graph cut problems)
- How can we relax the problem to find a (good) solution more efficiently?

Observations: For the special case with k connected components

- the k indicator vectors fulfilling $f_C L f_C^T = 0$ yield the perfect clustering
- The indicator vector for each component is an eigenvector of L with eigenvalue 0
- The k indicator vectors are orthogonal to each other (linearly independent)

Connected Components and Eigenvectors

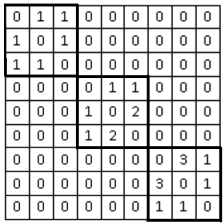
Lemma

The number of linearly independent eigenvectors with eigenvalue 0 for L equals the number of connected components in the graph.

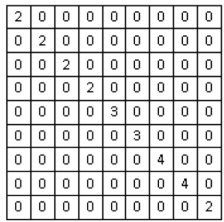
- the eigendecomposition on the Laplacian matrix can be done

Spectral Clustering: Example for Special Case

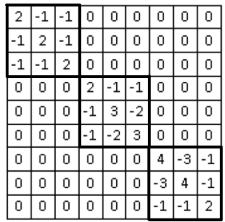
- Special case: The graph consists of k independent connected components (here: $k = 3$ and each consisting of 3 knots)
- The k components yield a "perfect" clustering (no edges between clusters), i.e. optimal clustering by indicator vectors $f_{C_1} = (1,1,1,0,0,0,0,0,0)$, $f_{C_2} = (0,0,0,1,1,1,0,0,0)$ and $f_{C_3} = (0,0,0,0,0,0,1,1,1)$



Adjacency matrix W



Degree matrix D



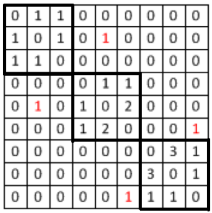
Laplacian matrix $L = D - W$

- Because of the block form of L, we get $f_C L f_C^T = 0$ for each component C, i.e. L has zero-eigenvectors.

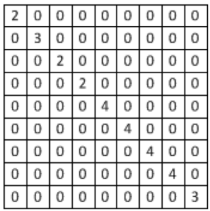
Spectral Clustering: General Case

- In general: L does not have zero-eigenvectors
  - One large connected component, no perfect clustering
  - Determine the (linear independent) eigenvectors with the *k smallest eigenvalues*!

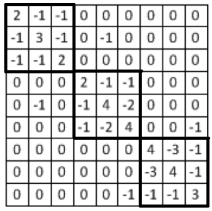
- Example: The 3 clusters are now connected by additional edges



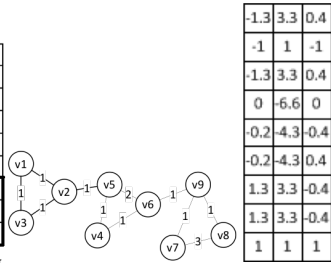
Adjacency matrix W



Degree matrix D



Laplacian matrix L

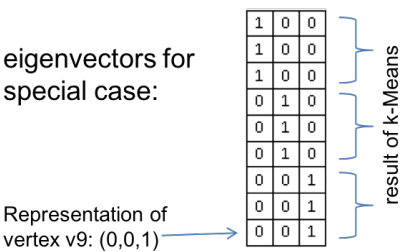


Eigenvectors of L

- Smallest eigenvalues of L: $(0.23, 0.70, 3.43)$

Spectral Clustering: Data Transformation

- How to find the clusters based on the eigenvectors?
  - Easy in special setting: 0-1 values; now: arbitrary real numbers
- Data transformation: Represent each vertex by a vector of its corresponding components in the eigenvectors
  - In the special case, the representations of vertices from the same connected component are equal, e.g. v_1, v_2, v_3 are transformed to $(1, 0, 0)$
  - In general case only *similar* eigenvector representations
- Clustering (e.g. k-Means) on transformed data points yields final result



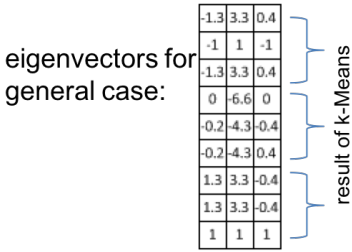
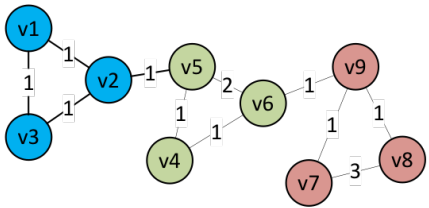
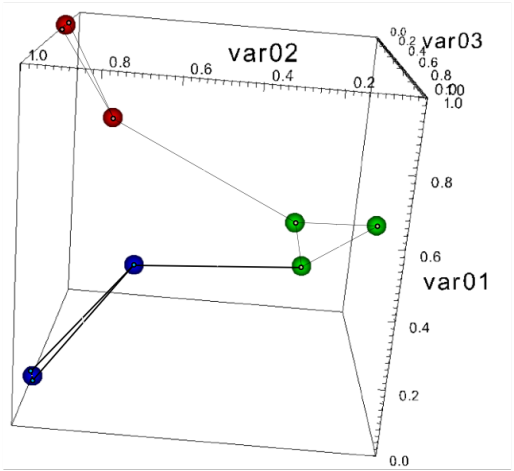


Illustration: Embedding of Vertices to a Vector Space

Spectral layout of previous example





Spectral Clustering: Discussion

Advantages

- No assumptions on the shape of the clusters
- Easy to implement

Disadvantages

- May be sensitive to construction of the similarity graph
- Runtime: k smallest eigenvectors can be computed in $\mathcal{O}(n^3)$ (worst case)
  - However: Much faster on sparse graphs, faster variants have been developed

- Several variations of spectral clustering exist, using different Laplacian matrices which can be related to different graph cut problems [1]

[1] Von Luxburg, U.: A tutorial on spectral clustering, in Statistics and Computing, 2007