University of Warsaw
Faculty of Economic Sciences

Unsupervised Learning

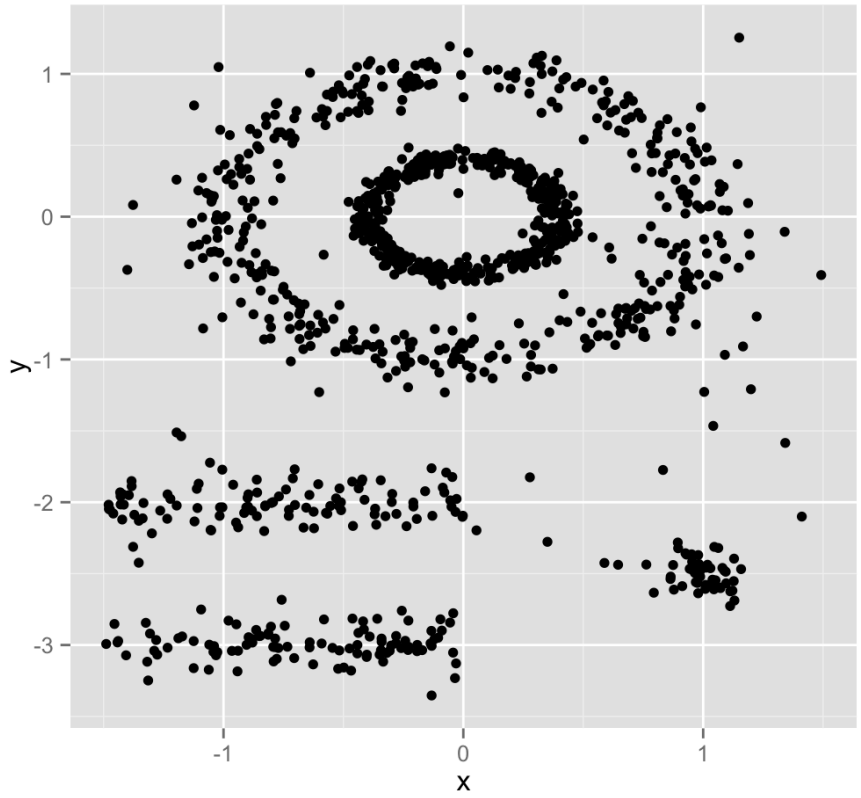
Winter Semester, 2025/2026

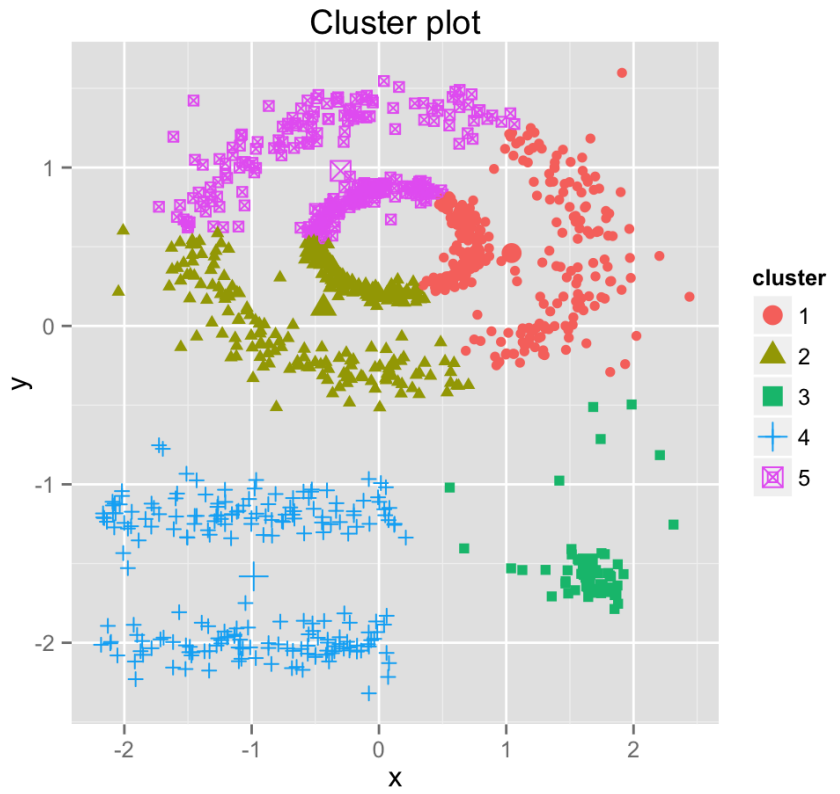
Unsupervised learning: DBSCAN

DBSCAN (STHDA)

- Partitioning methods (K-means, PAM clustering) or hierarchical clustering are suitable for finding spherical-shaped clusters or convex clusters
- They work well for compact and well separated clusters
- They are also severely affected by the presence of noise and outliers in the data

Partitioning methods and nonconvex data/noise/outliers (STHDA)

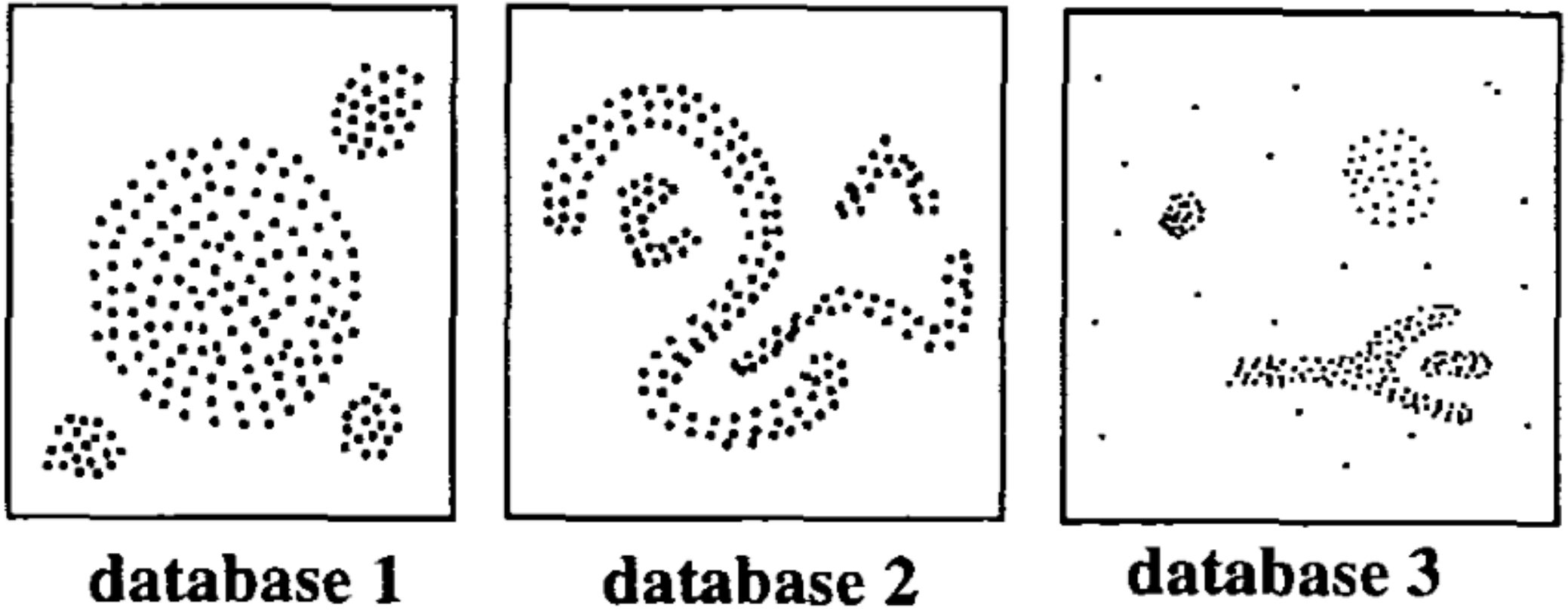




DBSCAN (STHDA)

- DBSCAN as a density-based clustering algorithm
  - introduced in Ester et al. 1996
  - can be used to identify clusters of any shape in data set containing noise and outliers
  - stands for Density-Based Spatial Clustering and Application with Noise
  - does not require the user to specify the number of clusters to be generated
  - can find any shape of clusters
  - can identify outliers pretty efficiently
  - derived from a human intuitive clustering method
  - it is a challenge to classify border points into clusters that, with a weakly variable density, can belong to different clusters, and cross-referencing can combine two clusters into one spatial structure.
  - the choice of distance metrics is important for the result, and as advanced works show, with multivariate data and the use of Euclidean distance, finding ε is difficult or even impossible
  - there may also be a problem with the selection of radius ε at highly heterogeneous local point densities

DBSCAN (Ester et al. 1996)

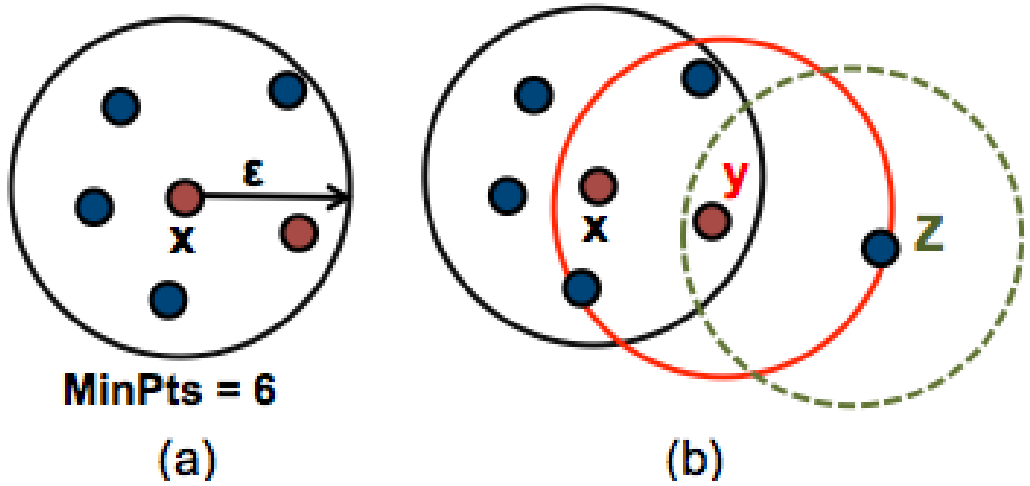


DBSCAN (STHDA)

- Clusters are dense regions in the data space, separated by regions of lower density of points
- The density of points in a cluster is considerably higher than the density of points outside the cluster (“areas of noise”)
- The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points
- The goal is to identify dense regions, which can be measured by the number of objects close to a given point

DBSCAN (STHDA)

- The parameter eps defines the radius of neighborhood around a point x. It's called called the \(\epsilon\)-neighborhood of x
- The parameter MinPts is the minimum number of neighbors within "eps" radius
- Any point x in the dataset, with a neighbor count greater than or equal to MinPts, is marked as a core point.
- x is border point, if the number of its neighbors is less than MinPts, but it belongs to the \(\epsilon\)-neighborhood of some core point z.
- If a point is neither a core nor a border point, then it is called a noise point or an outlier.
- Example on the rhs: x is a core point, y is a border point, z is a noise point



DBSCAN (STHDA)

- A density-based cluster is defined as a group of density connected points
- Essential conditions
  - Direct density reachable: A point “A” is directly density reachable from another point “B” if: i) “A” is in the \(\epsilon\)-neighborhood of “B” and ii) “B” is a core point (All these points in the radius ε from the core point (ε-neighborhood))
  - Density reachable: A point “A” is density reachable from “B” if there are a set of core points leading from “B” to “A (There is a sequence of points that are directly accessible to each other, which in practice means going from point to point through other points, always within a radius of ε)
  - Density connected: Two points “A” and “B” are density connected if there are a core point “C”, such that both “A” and “B” are density reachable from “C”.
  - Points are classified as noise when they are outside the radius of core and boundary points.

DBSCAN (STHDA)

For each point \(x_i\), compute the distance between \(x_i\) and the other points. Finds all neighbor points within distance eps of the starting point (\(x_i\)). Each point, with a neighbor count greater than or equal to MinPts, is marked as core point or visited

For each core point, if it's not already assigned to a cluster, create a new cluster. Find recursively all its density connected points and assign them to the same cluster as the core point

Iterate through the remaining unvisited points in the dataset

Those points that do not belong to any cluster are treated as outliers or noise

DBSCAN (K. Kopczewska)

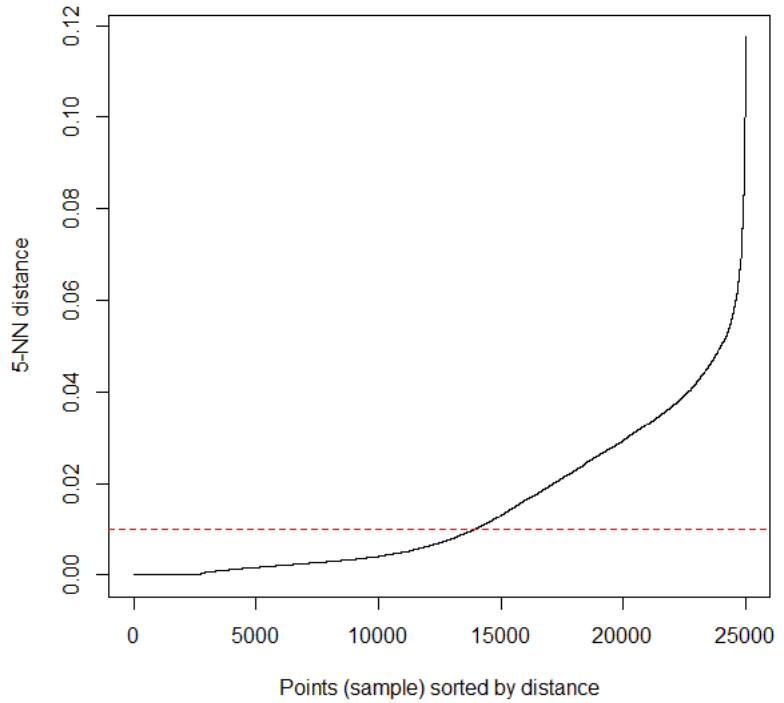
- The epsilon ε radius and the minimum number of points MinPts in this radius are crucial for the detection of clusters.
- They are not independent - for points with different density and assuming a high ε, it's harder to find a cluster for high MinPts than for low MinPts.
- Setting those parameters often requires an expert knowledge. MinPts should be treated as the desired minimum cluster size.
- In order to obtain any reasonable grouping, MinPts should be 3 or more (with MinPts = 1 each point creates its own cluster, while MinPts = 2 the basic hierarchical grouping appears).

DBSCAN (STHDA)

- One limitation of DBSCAN is that it is sensitive to the choice of \(\epsilon\), in particular if clusters have different densities.
  - If \(\epsilon\) is too small, sparser clusters will be defined as noise.
  - If \(\epsilon\) is too large, denser clusters may be merged together.
  - This implies that, if there are clusters with different local densities, then a single \(\epsilon\) value may not suffice.
- How to determine an optimal eps value?
  - Compute the he k-nearest neighbor distances in a matrix of points.
  - The idea is to calculate, the average of the distances of every point to its k nearest neighbors.
  - The value of k will be specified by the user and corresponds to MinPts.
  - Next, these k-distances are plotted in an ascending order.
  - The aim is to determine the “knee” (or „elbow”), which corresponds to the optimal eps parameter.
  - A knee corresponds to a threshold where a sharp change occurs along the k-distance curve.
  - The function kNNdistplot() [in dbscan package] can be used to draw the k-distance plot

DBSCAN (K. Kopczewska)

- This step is to find radius ε, in which one looks for given number of points
- we analyse distance to nearest neighbours on the graph - for each point in the dataset the distance to the knn-th neighbour
- The y-axis gives the distance between neighbours (it is ε), and the points from the dataset are indexed on the x-axis - they have been ordered in ascending order according to the value of ε.
- The number of index points on the x-axis is equal to the number of points in the dataset multiplied by the number of knn neighbours indicated.
- we set the ε value (for the dbscan() command) at the level at which the knn graph breaks down (the so-called knee/elbow)
- The chart can be created with the command kNNdistplot(), also from the dbscan:: package.
- One gives the value of MinPts – a number of points in radius, which will create clusters



DBSCAN (K. Kopczewska)

- In the diagnosis of k nearest neighbours of a given point, one can also use the command kNN() from the dbscan:: package. The points to be analyzed and the number of k nearest neighbors should be given as the input, and in the result the matrix of distances to the next neighbours is obtained in the $dist slot (first - nearest, second etc.) and in the $id slot the index matrix which observations are the nearest neighbours for a given point.

- Neighbors can be sought not only as the nearest, but also within a given radius of ε. To do this, one can use the frNN() command from the dbscan:: package (fr is abbreviation for fixed radius). Similarly to kNN(), in frNN() output one gets in the $dist slot a matrix of distances to all neighbors indicated in a given radius and in the $id slot of the index matrix, which observations are the nearest neighbors for a given point. Neighbor connections found can be drawn with the plot() command, where the first argument is the result of the kNN() or frNN() commands, and the second argument is the set of analysed points.

- The command frNN() is supplemented by the pointdensity() command from the dbscan:: package, which counts neighbors in a given radius ε - the local density for each point. As arguments to this function, one gives the points for analysis, radius ε and the method of counting (usually counting as type = "frequency"). The result is a vector of the number of neighbors of each point under examination.

- One can also conduct a comparative analysis of the local density of points. The lof() function (Local Outlier Factor) from the dbscan:: package compares the local densities of the tested point and its k neighbors. If the local densities are similar (similar numbers of neighbors in a given radius), the lof is 1, and for outlier - noise, lof is significantly higher than 1. As arguments of this function, one gives the dataset of the points and neighborhood width – the number of k-nearest neighbors used for comparisons.

DBSCAN (K. Kopczewska)

- The extension of the DBSCAN algorithm is HDBSCAN, based on a hierarchical tree of clusters. This allows for a more stable solution.. In the dbscan:: package, the hdbscan() function is available, as well as extractFOSC(), which optimally selects clusters based on the output of the hdbscan() and glosh() commands comparing local densities with the global ones.
- Yet another extension is the OPTICS algorithm, available via the optics() command from the dbscan:: package, which is based on data sorting, and the given radius ε is treated as the upper limit. Parallel to the optics() command, the reachability() command specifying the availability structure is used.

Unsupervised learning: spectral methods

Spectral methods (A. Aoullay)

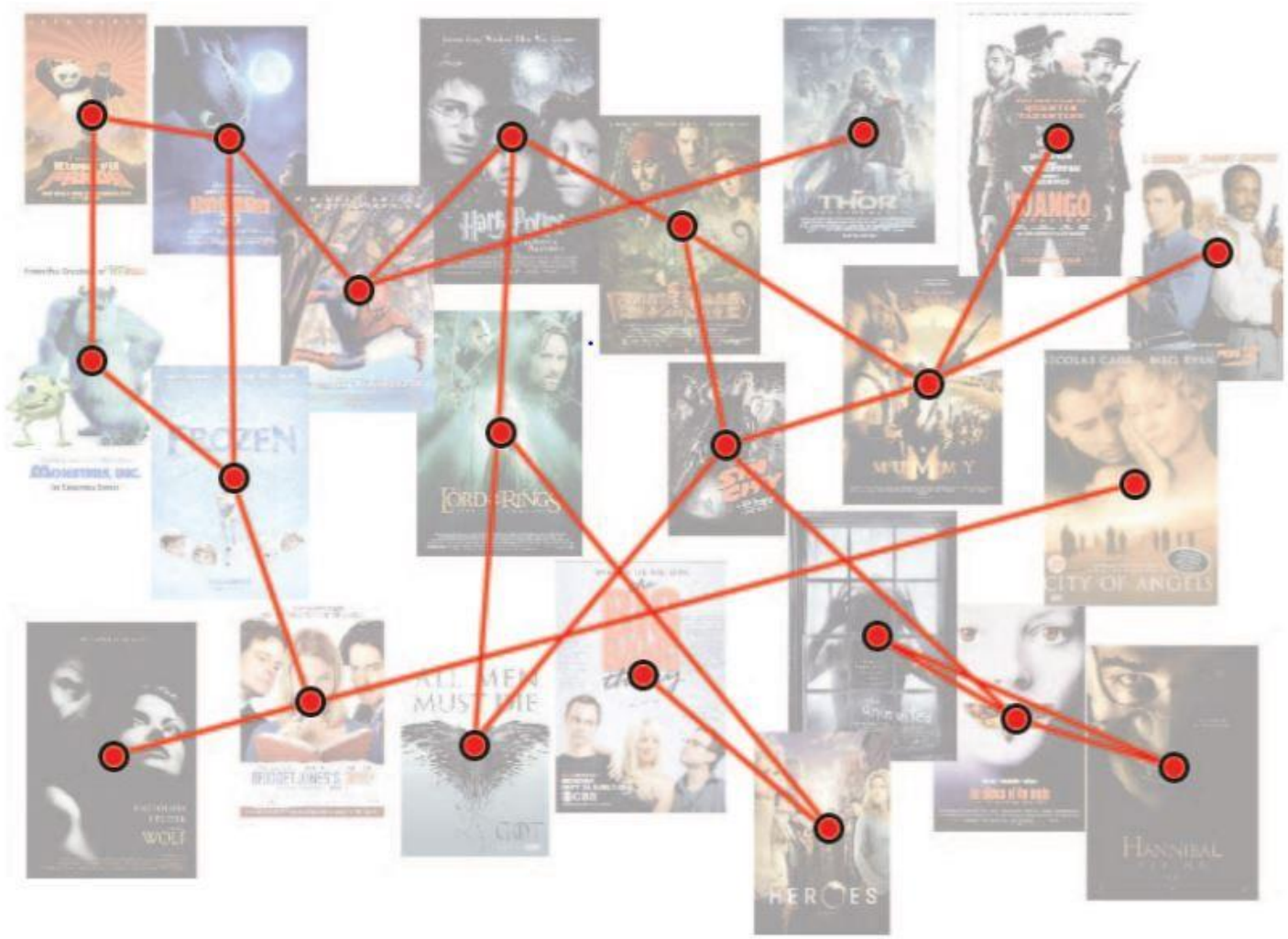
- Graph-based
- It can be solved efficiently by standard linear algebra software
- Very often outperforms traditional algorithms such as the k-means algorithm

Spectral methods (A. Aoullay)

Create a similarity graph between our N objects to cluster

Compute the first k eigenvectors of its Laplacian matrix to define a feature vector for each object

Run k-means on these features to separate objects into k classes

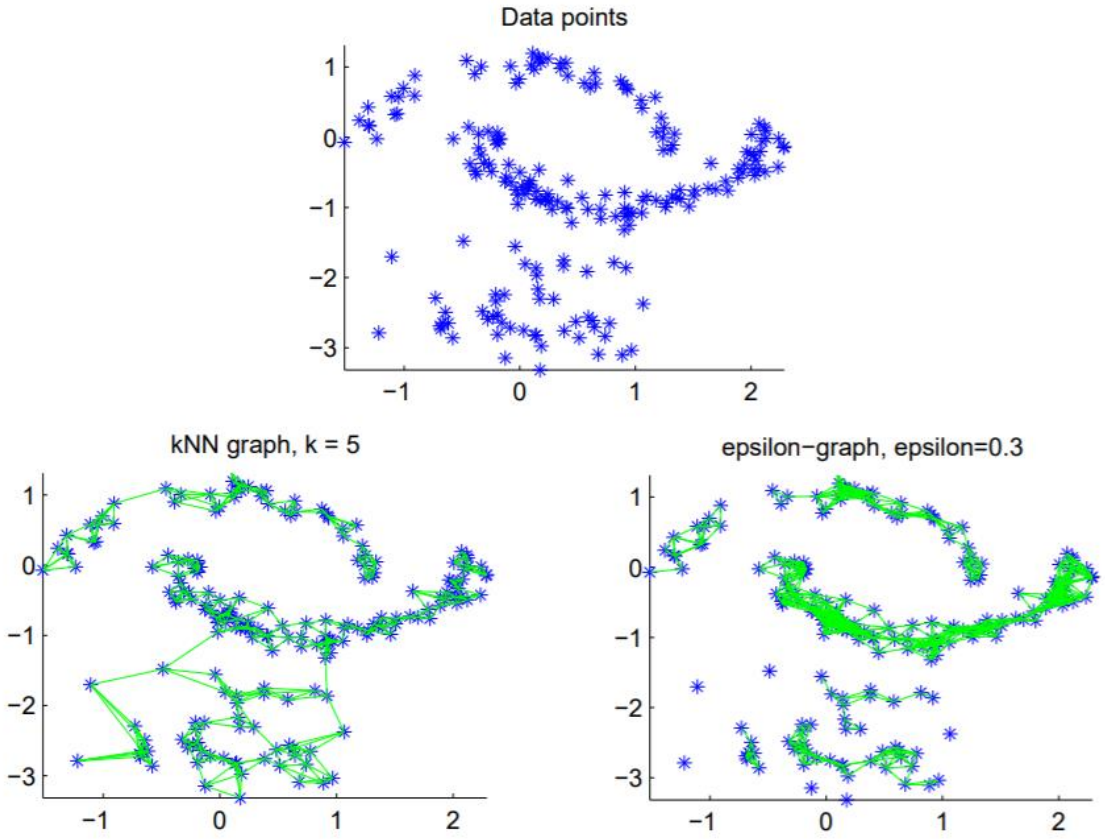


Spectral methods (A. Aoullay)

Spectral methods (A. Aoullay)

ε-neighborhood graph: Each vertex is connected to vertices falling inside a ball of radius ε where ε is a real value that has to be tuned in order to catch the local structure of data

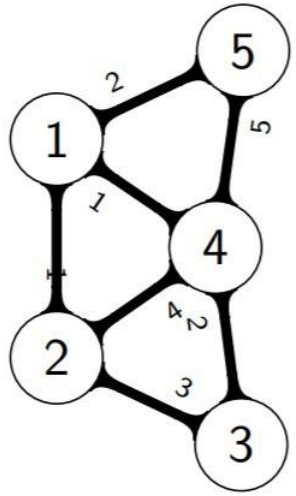
k-nearest neighbor graph: Each vertex is connected to its k-nearest neighbors where k is an integer number which controls the local relationships of data



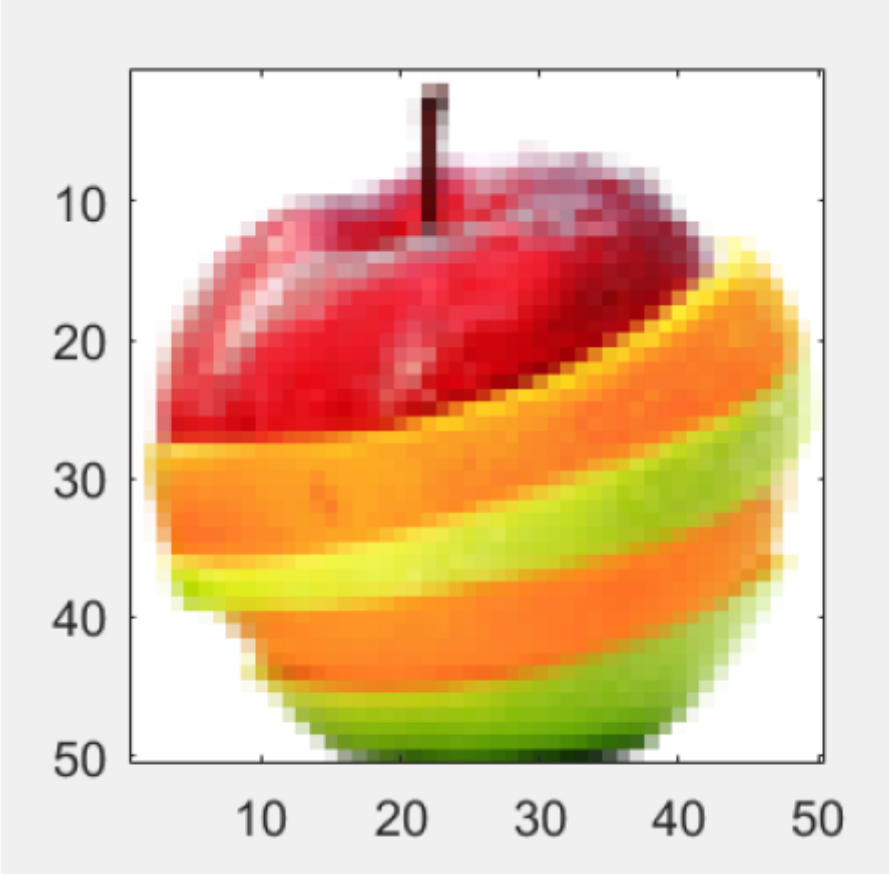
Spectral methods (A. Aoullay)

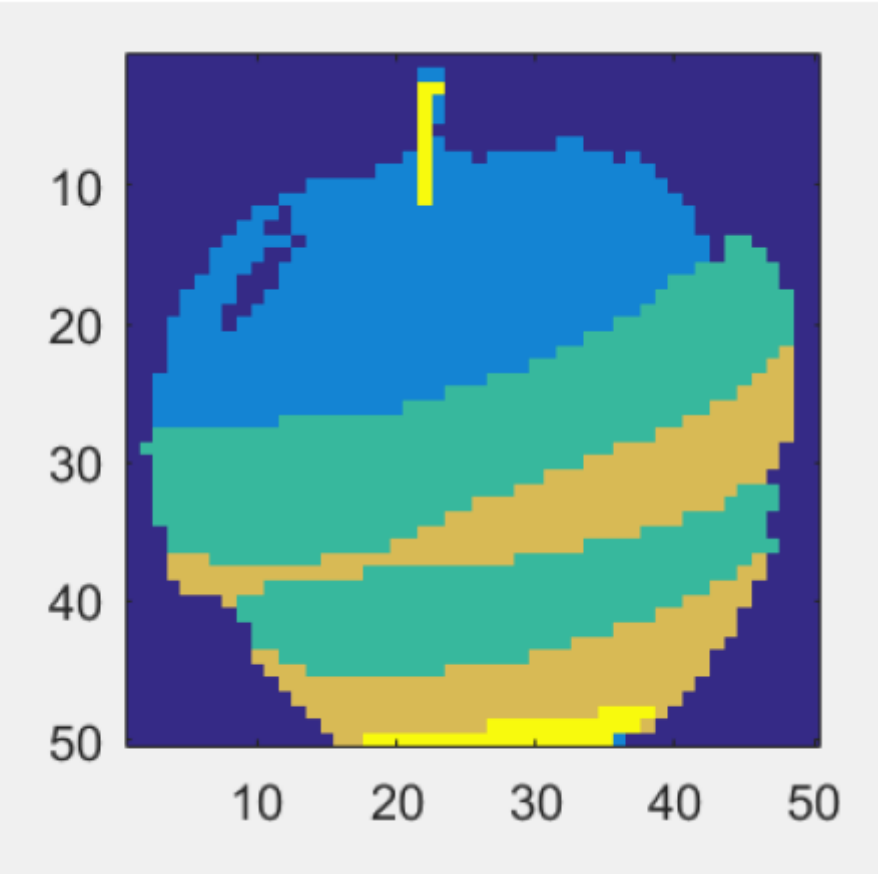
$\mathbf{A}$	adjacency matrix
$\mathbf{W}$	weight matrix
$\mathbf{D}$	(diagonal) degree matrix
$\mathbf{L} = \mathbf{D} - \mathbf{W}$	graph **Laplacian** matrix

$$\mathbf{L} = \begin{pmatrix} 4 & -1 & 0 & -1 & -2 \\ -1 & 8 & -3 & -4 & 0 \\ 0 & -3 & 5 & -2 & 0 \\ -1 & -4 & -2 & 12 & -5 \\ -2 & 0 & 0 & -5 & 7 \end{pmatrix}$$



Spectral methods (A. Aoullay)





Spectral methods (A. Aoullay)

- By projecting the points into a non-linear embedding and analyzing the eigenvalues of the Laplacian matrix one can deduce the number of clusters present in the data
- When the similarity graph is not fully connected, the multiplicity of the eigenvalue $\lambda = 0$ gives us an estimation of k

Spectral methods

- More materials on the topic
  - https://www.mygreatlearning.com/blog/introduction-to-spectral-clustering/
  - http://www.di.fc.ul.pt/~jpn/r/spectralclustering/spectralclustering.html
  - https://towardsdatascience.com/spectral-clustering-aba2640c0d5b
  - https://link.springer.com/article/10.1007/s11222-007-9033-z
  - https://books.google.pl/books?hl=pl&lr=&id=JU_SBQAAQBAJ&oi=fnd&pg=PA177&dq=spectral+clustering&ots=LC6KkwE0EJ&sig=H07K7mQSQ6COi6K5UDQEFRRf7Ww&redir_esc=y#v=onepage&q=spectral%20clustering&f=false

Thank you!