Clustering

K-means, DBSCAN

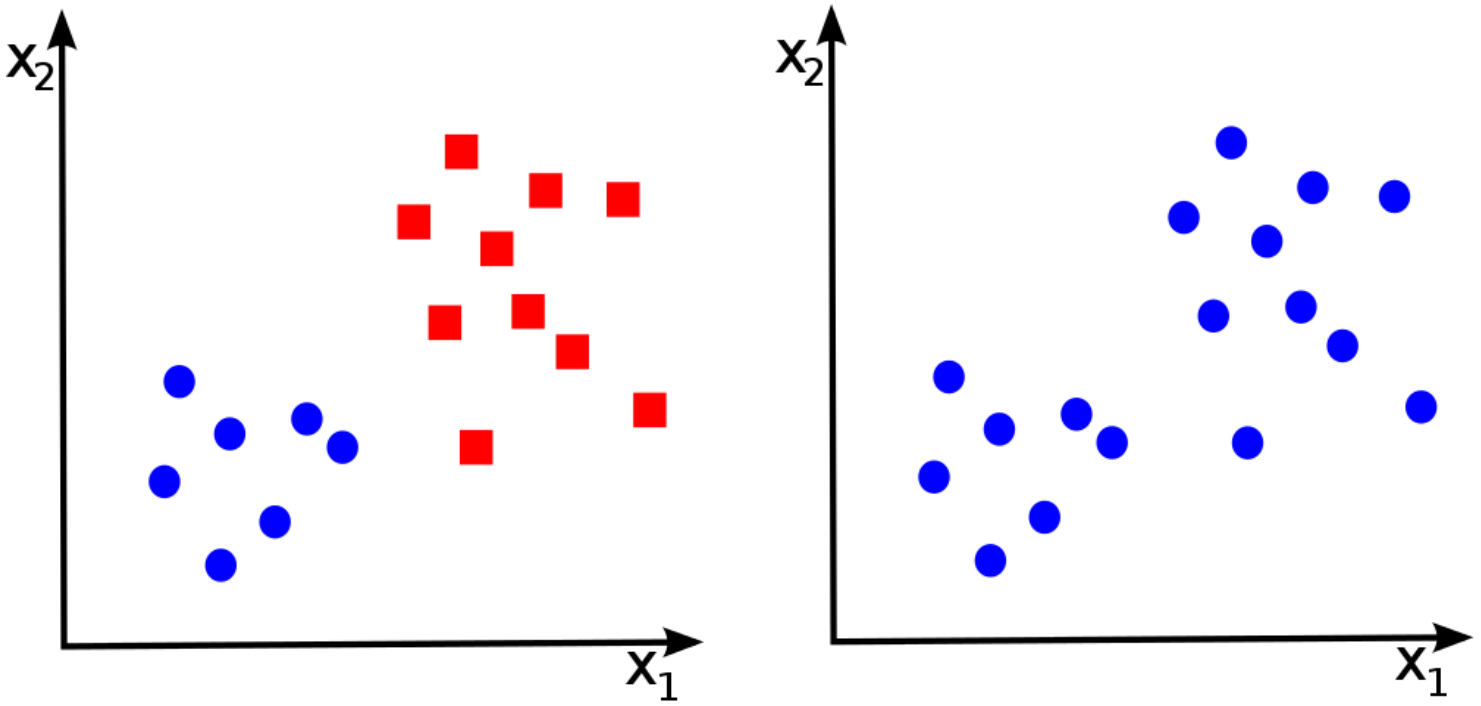
Pekka Marttinen

Aalto University

Outline

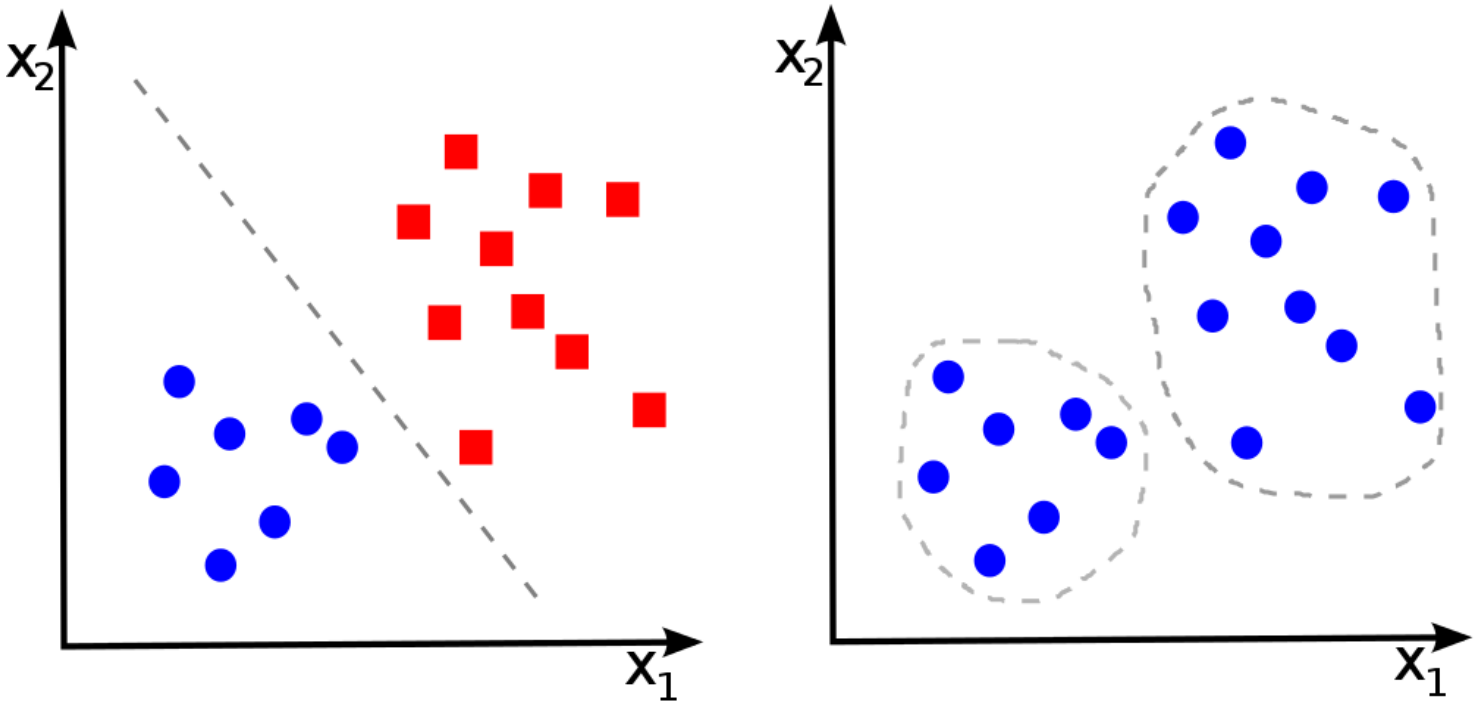
- Introduction
- K-means
- K-means application
- DBSCAN
- DBSCAN application

Unsupervised learning



- Classification, supervised:
  $\{(x_{11}, x_{12}, y_1), (x_{21}, x_{22}, y_2), \ldots, (x_{n1}, x_{n2}, y_n)\}$
- Clustering, unsupervised:
  $\{(x_{11}, x_{12}), (x_{21}, x_{22}), \ldots, (x_{n1}, x_{n2})\}$

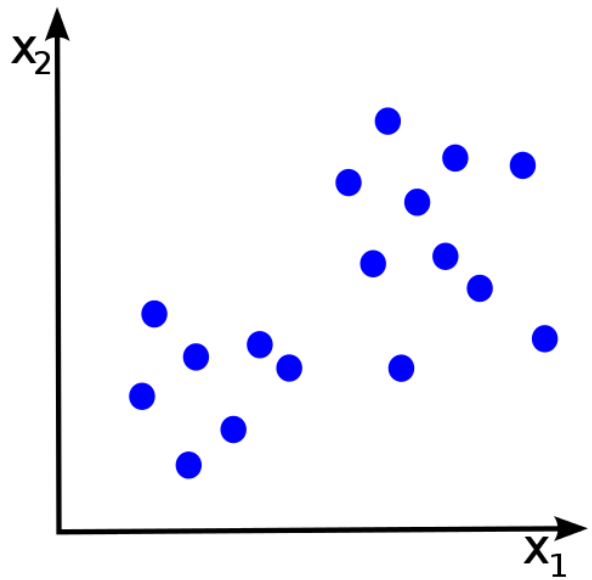
Unsupervised learning



- Classification, supervised:
  $\{(x_{11}, x_{12}, y_1), (x_{21}, x_{22}, y_2), \ldots, (x_{n1}, x_{n2}, y_n)\}$
- Clustering, unsupervised:
  $\{(x_{11}, x_{12}), (x_{21}, x_{22}), \ldots, (x_{n1}, x_{n2})\}$

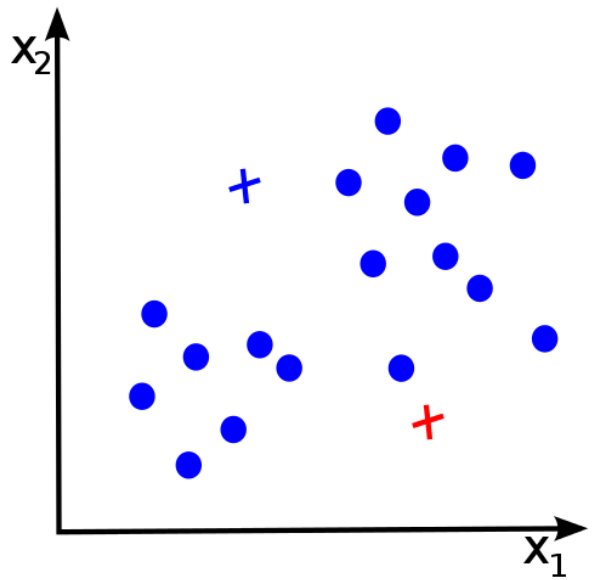
K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- Initialize by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - Assign each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - Update the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.



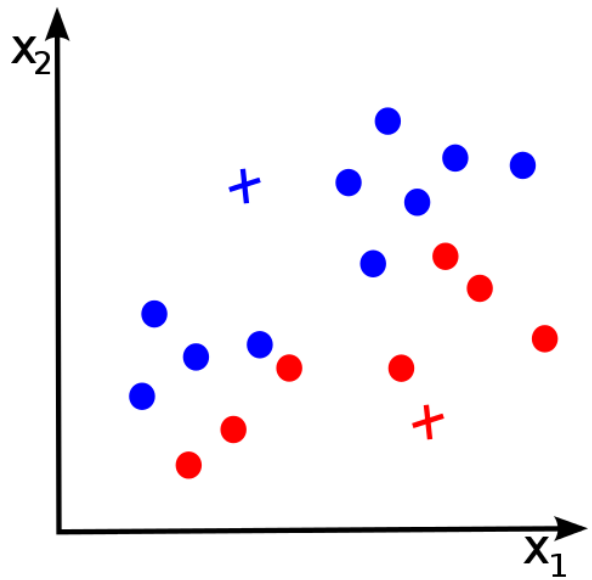
K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- **Initialize** by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - Assign each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - Update the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.



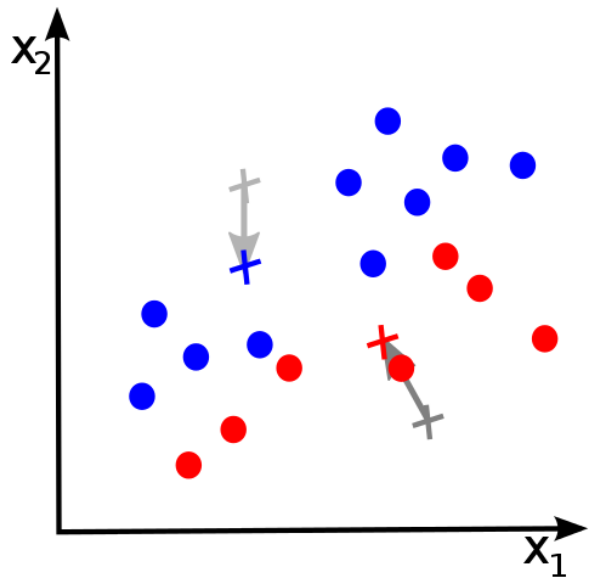
K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- Initialize by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - **Assign** each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - Update the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.



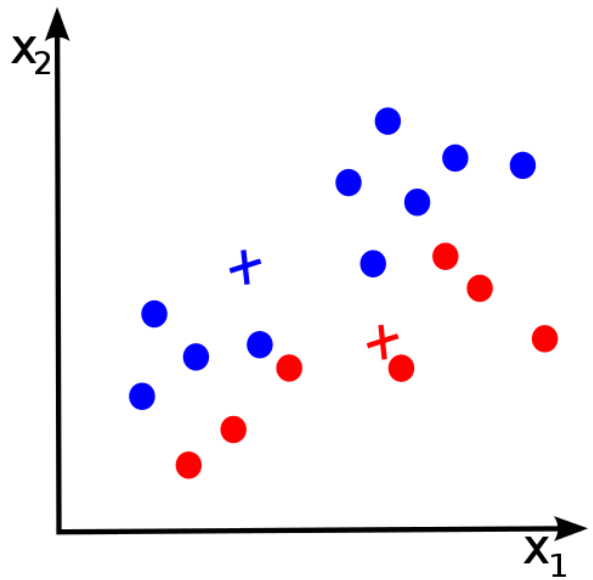
K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- Initialize by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - Assign each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - **Update** the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.



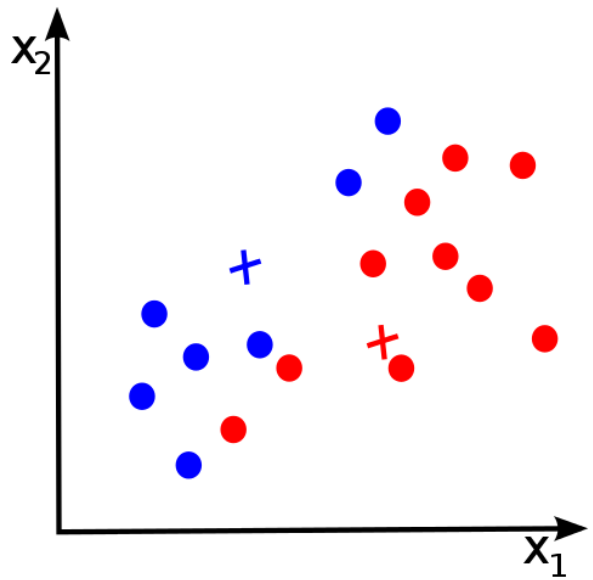
K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- Initialize by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - Assign each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - Update the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.



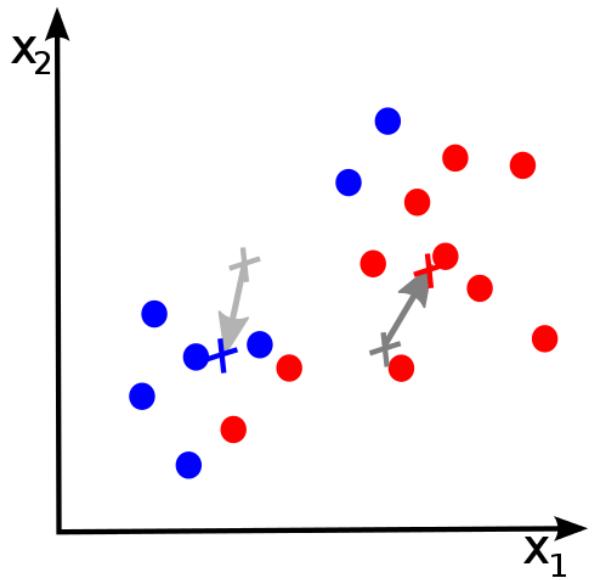
K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- Initialize by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - **Assign** each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - Update the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.



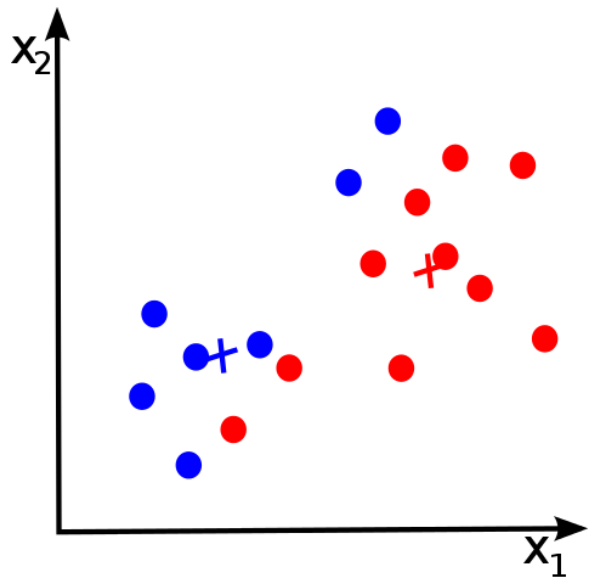
K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- Initialize by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - Assign each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - **Update** the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.



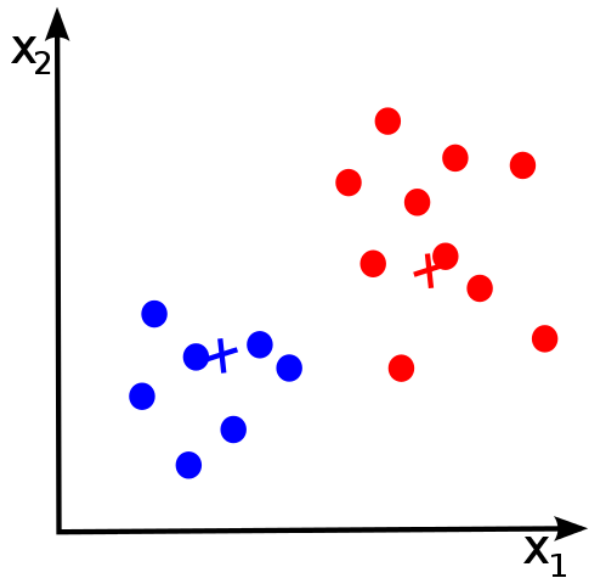
K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- Initialize by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - Assign each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - Update the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.



K-means clustering

- Iteratively find k clusters $\mathcal{C}_j$, $j = 1, \ldots, k$ in the data.
- Initialize by choosing k random points as cluster centroids $c_j = 1, \ldots, k$.
- Repeat:
  - **Assign** each data point $x_i, i = 1, \ldots, n$, to a cluster $\mathcal{C}_j$ whose centroid c_j is closest to x_i.
  - Update the cluster centroids c_j to the average of points x_i in $\mathcal{C}_j$.

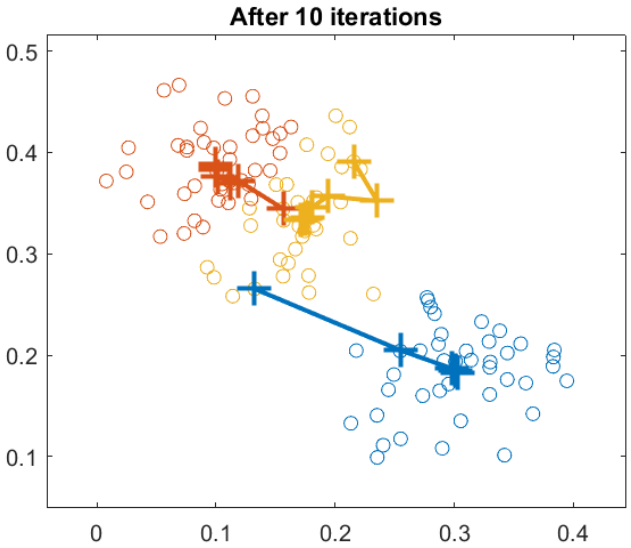


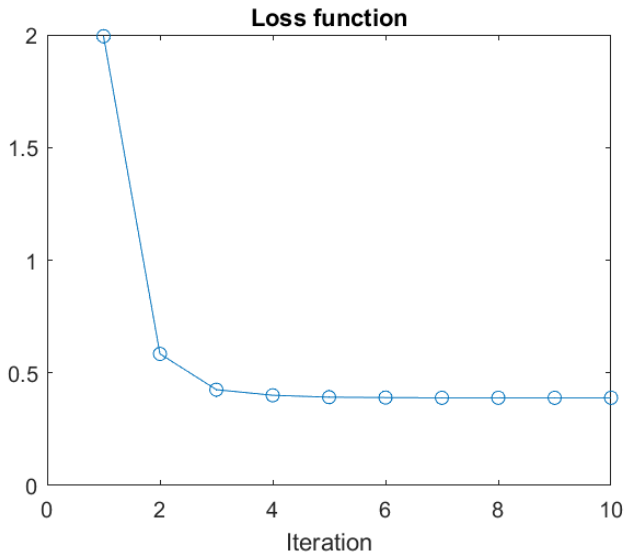
K-means loss function

- K-means tries to minimize a loss function: the sum of squared distances from data points to cluster centroids

$$J(\{c_1, \ldots, c_k\}) = \sum_{i=1}^{N} \min_j ||x_i - c_j||^2.$$

- Reminder: $||x_i - c_j||^2 = (x_{i1} - c_{j1})^2 + (x_{i2} - c_{j2})^2$.



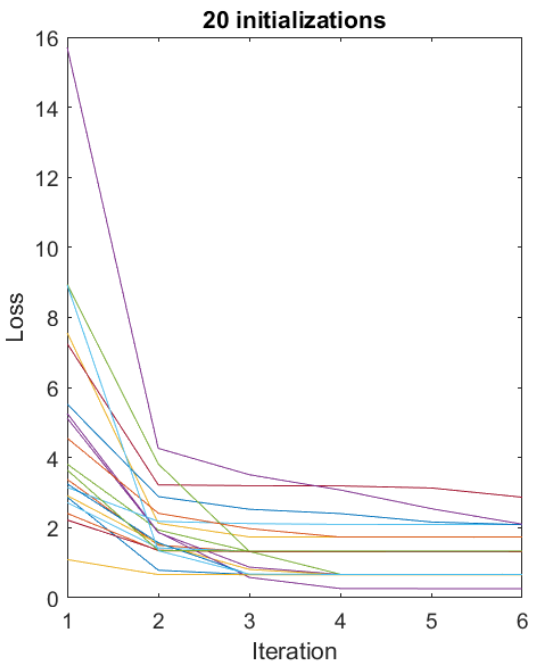


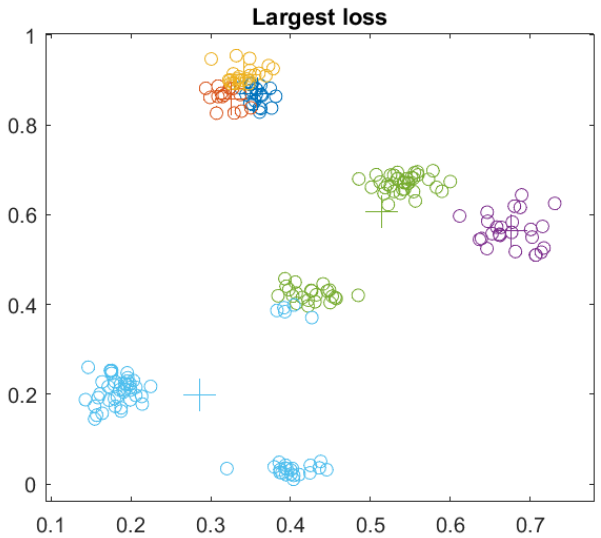
Initializing the K-means clustering

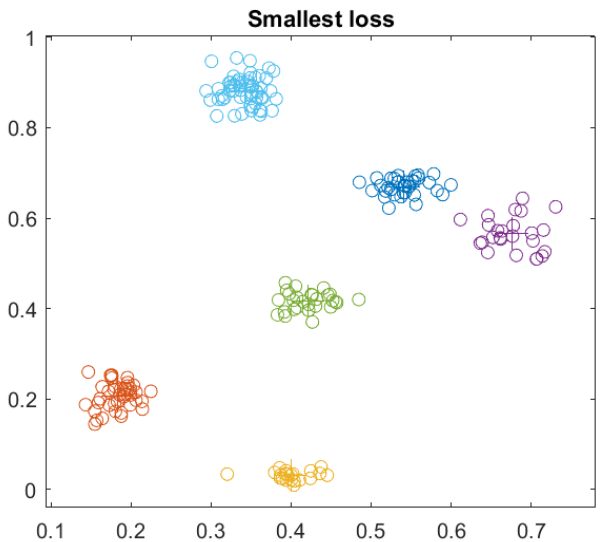
- K-means uses local updates to minimize the loss
  - Assign each point to the closest cluster centroid.
  - Update centroids to the average of nearby points.
  - -> Finding the global optimum is not guaranteed.
- K-means may converge to different solutions for different initializations.
- Common choice: Initialize k cluster centroids randomly from the set of data points.
- Important: Run multiple runs and monitor the loss function.

Initializing the K-means clustering

20 trainings with the correct number of clusters $k = 6$, with different initializations.

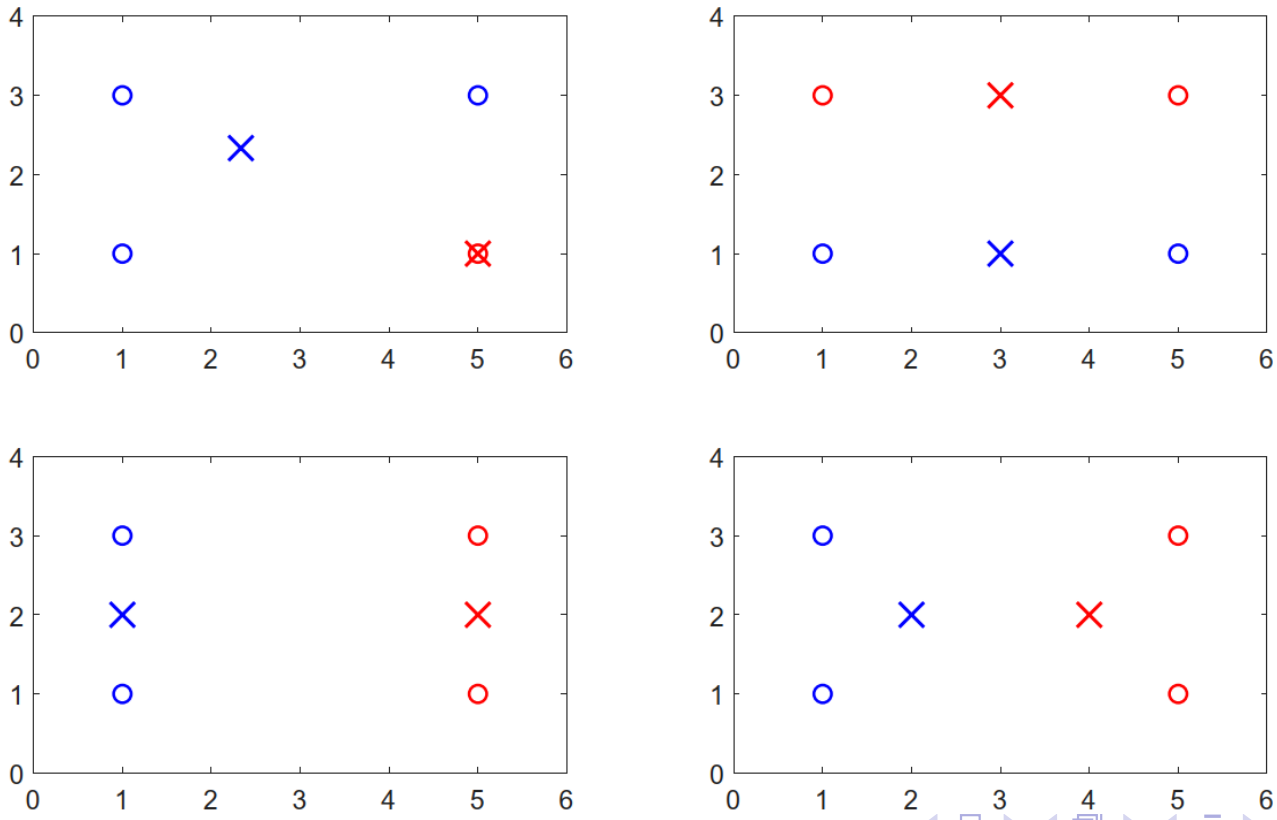






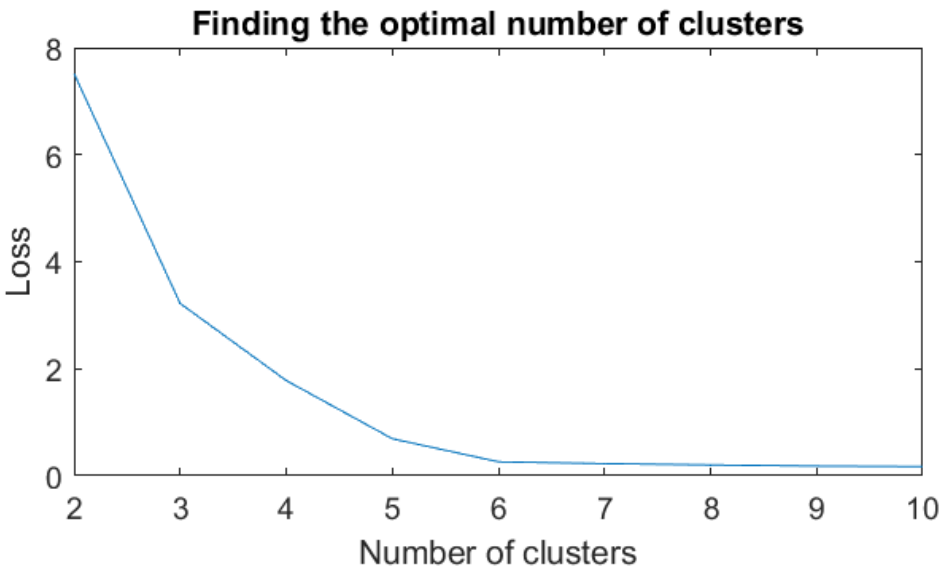
K-means, problem

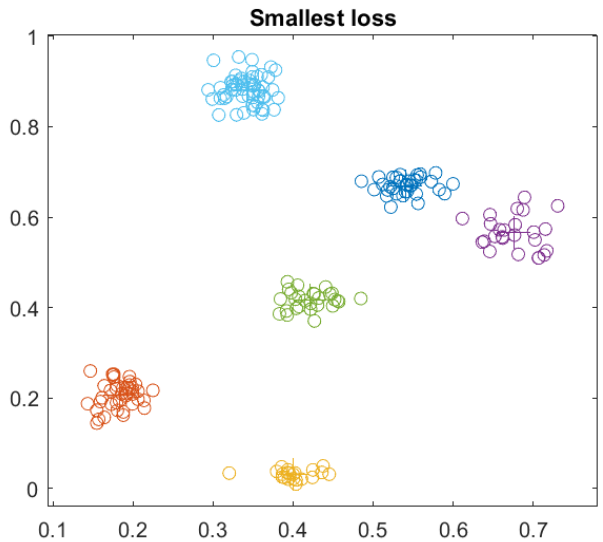
- The figure shows states from 4 different runs of K-means for 4 data points (circles).
- Centroids of two clusters (red, blue) are shown with crosses.
- In which figures the algorithm has converged, i.e., running more iterations will not improve the loss?



Choosing the number of clusters

- Elbow criterion:
  - Compare the smallest loss (over multiple runs) for each k.
  - Select k such that loss does not 'decrease much' by increasing k further.
  - Below, $k = 6$ seems like a good choice.



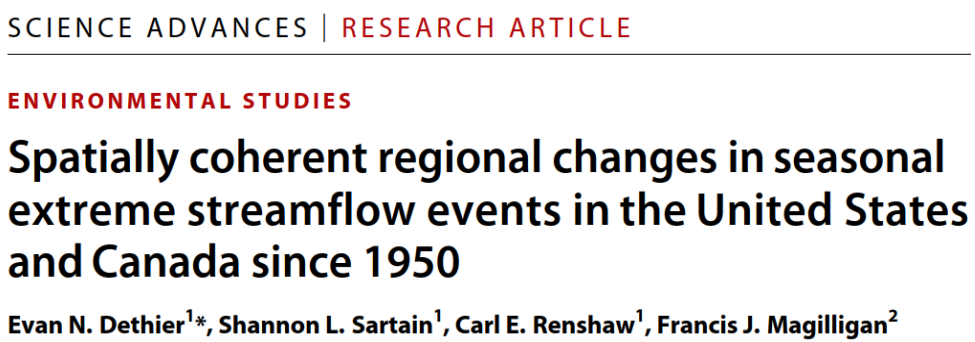


Applications of K-means

- K-means is extremely popular as a general-purpose clustering method.
- Over 52000 results in Google Scholar in 2020-2021 for "K-means"
  - Applications (biomedicine, climate change, social science, psychology, ...).
  - New methods and theoretical results.

Application: Impact of climate change on flooding/drought

- **Motivation**: to understand whether extreme streamflows (floods, droughts) are increasing in frequency due to climate change.
- **Data**: 541 rivers in the US and Canada, with 15 features
  - 12 normalized monthly high flows
  - latitude, longitude, mean watershed elevation



SCIENCE ADVANCES | RESEARCH ARTICLE

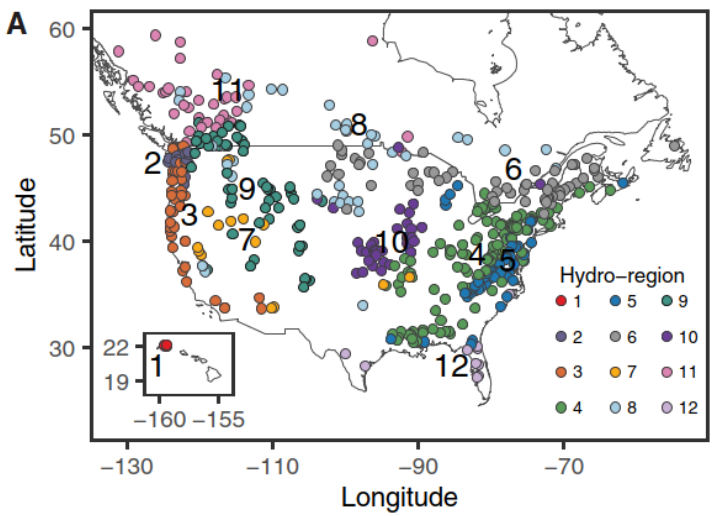
ENVIRONMENTAL STUDIES

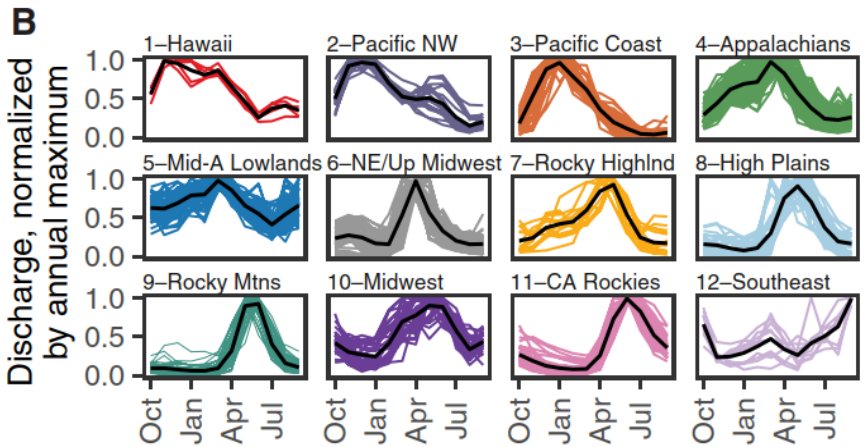
Spatially coherent regional changes in seasonal extreme streamflow events in the United States and Canada since 1950

Evan N. Dethier[1]*, Shannon L. Sartain[1], Carl E. Renshaw[1], Francis J. Magilligan[2]

Application: Results

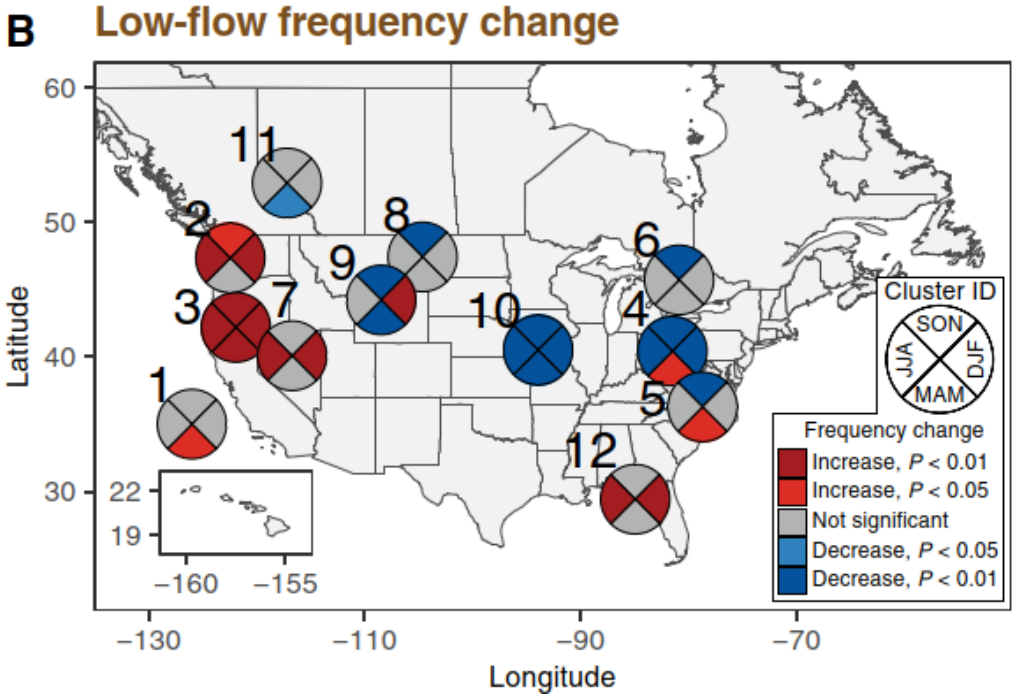
- Method: K-means to divide watersheds into hydrologically coherent 'hydroregions'.
- 15 clusters (hydroregions) found, with similar streamflow characteristics.





Application: Results

- Time series since 1910 shows that the number of extreme high- and low-flow events (floods/droughts) has increased.



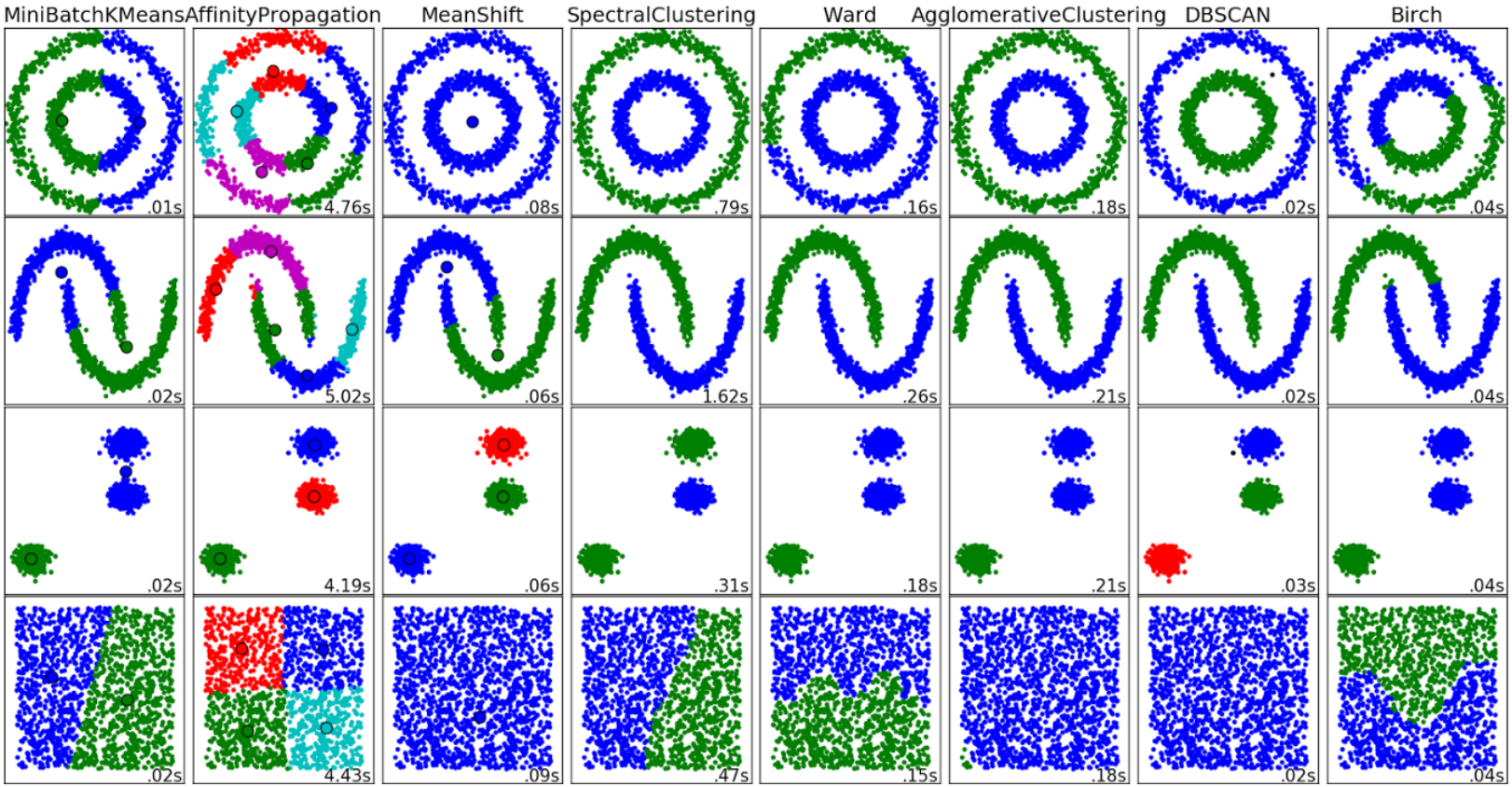
Application: Lessons learned

- What was gained by clustering?
- Alternatives
  - Analysis of indidual rivers $\longrightarrow$ Noisy results
  - Regions determined by location/political boundary/watershed bundary $\longrightarrow$ Do not reflect hydrologic boundaries $\longrightarrow$ Inconsistent results
- Clustering
  - An objective way to find meaningful clusters with similar flood regimes.
    $\longrightarrow$ Increased signal-to-noise ratio.
    $\longrightarrow$ Robust, generalizable results.

Different types of clustering algorithms

- A very large number of different clustering algorithms exist.
- Main groups
  - Representative -based (e.g. K-means)
  - Hierarchical
  - Probabilistic model -based
  - Density -based (including grid-based)
  - Graph -based
  - Matrix factorization based

Overview of clustering algorithms



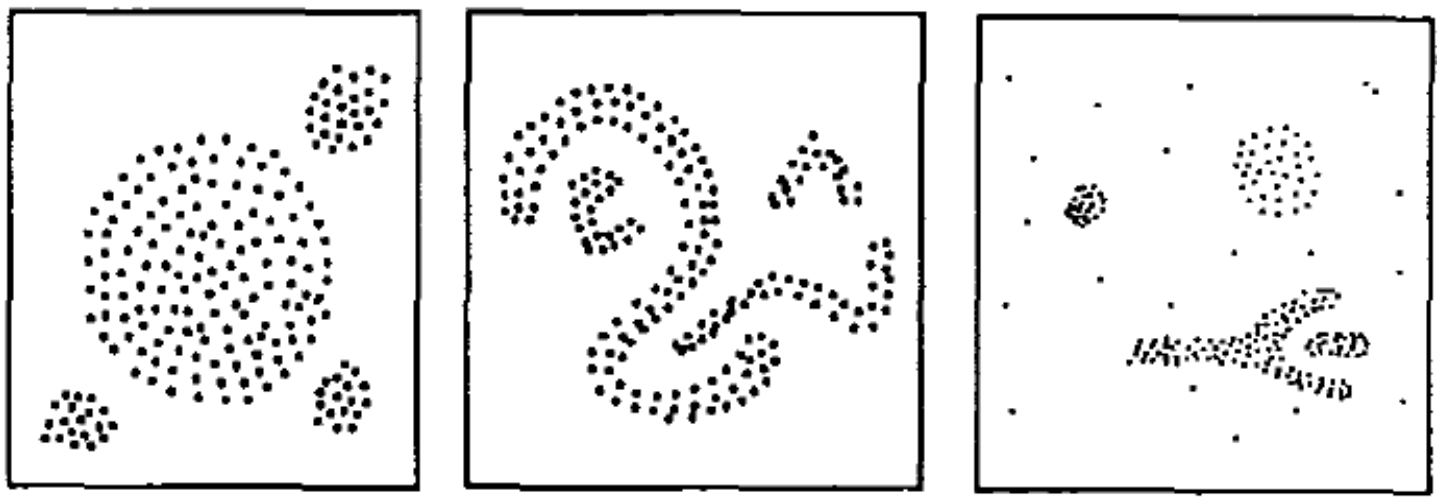
Scikit Learn - Plot Cluster Comparison

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

- Requirements for a clustering algorithm
  - Minimal required domain knowledge.
  - Good efficiency on large data sets.
  - Dicovery of clusters of arbitrary shape.

DBSCAN, intuition

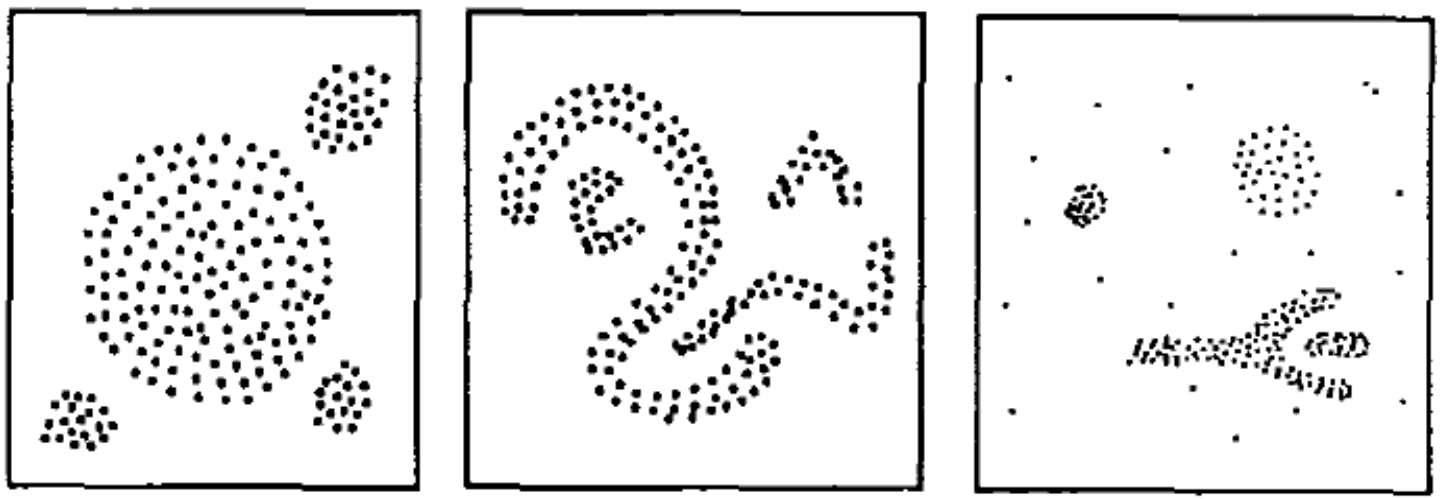
- Definition of a cluster:
  - Each cluster has a typical density of points which is considerably higher than outside the cluster[1].



[1] Ester et al. (1996). A Density-Based Algorithm for Discovering Clusters, KDD.

DBSCAN, intuition

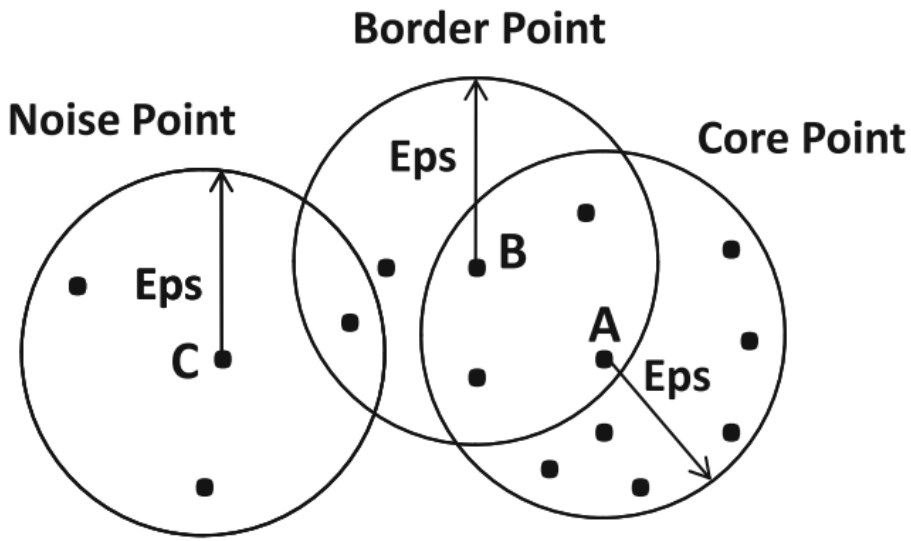
- Definition of a cluster:
  - Each cluster has a typical density of points which is considerably higher than outside the cluster.



- Density in the **neighborhood** has to exceed some threshold for a point to be considered inside a cluster.

Core, border, and noise points

- Hyperparameters: neighborhood size ϵ and density τ
- Divide the points into:
  - **Core point**: its $\epsilon-$neighborhood contains at least τ points.
  - **Border point**: not a core point, but its neighborhood contains at least one core point.
  - **Noise point**: neither a core nor a border point.

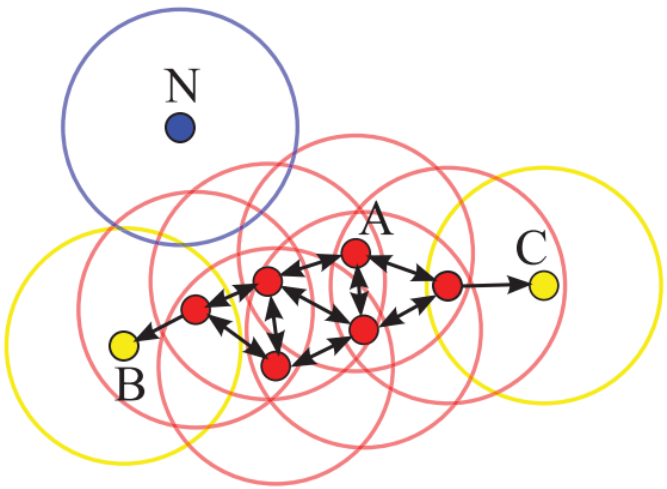


Aggarwal, Data Mining, Fig 6.16. Here, $\tau = 10$.

DBSCAN: Abstract algorithm[2]

ALGORITHM 2: Abstract DBSCAN Algorithm

```
1 Compute neighbors of each point and identify core points      // Identify core points
2 Join neighboring core points into clusters                    // Assign core points
3 foreach non-core point do
4 |   Add to a neighboring core point if possible               // Assign border points
5 |   Otherwise, add to noise                                   // Assign noise points
```



Here the radius of a circle shows ϵ, and $\tau = 4$.

[2] Schubert et al. (2017). DBSCAN Revisited, revised: Why and How You Should (Still) Use DBSCAN

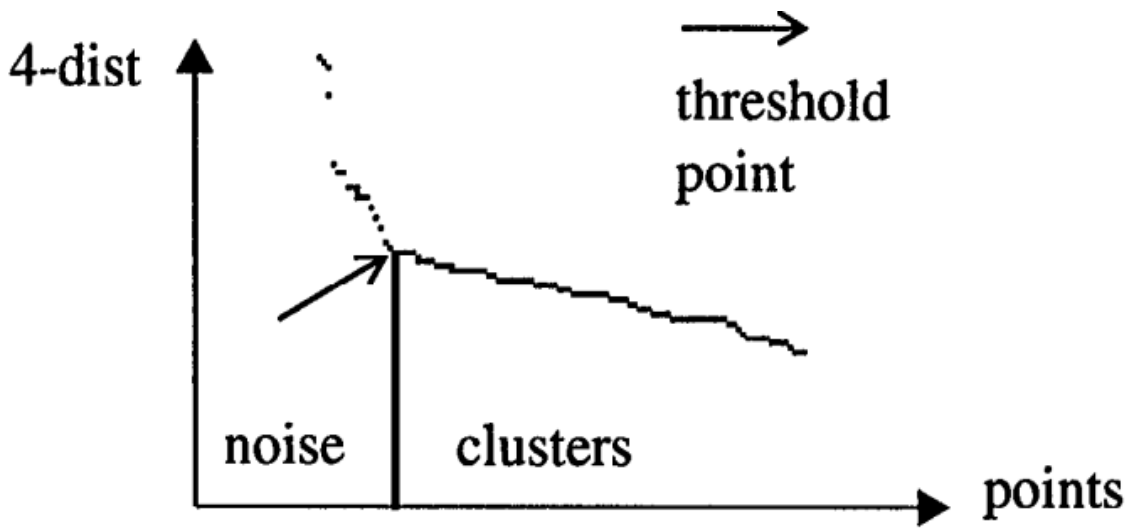
DBSCAN: Sequential algorithm

- Scan the database linearly once for objects which have not yet been processed.
  - Non-core points are assigned to noise.
  - When a core point is discovered, its neighbors are iteratively expanded and added to the cluster.
  - Objects that have been assigned to a cluster will be then be skipped when encountered later by the linear scan.[3]
- Illustration: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

[3] Full pseudocode in Schubert et al. (2017).

DBSCAN: Selecting hyperparameters[4]

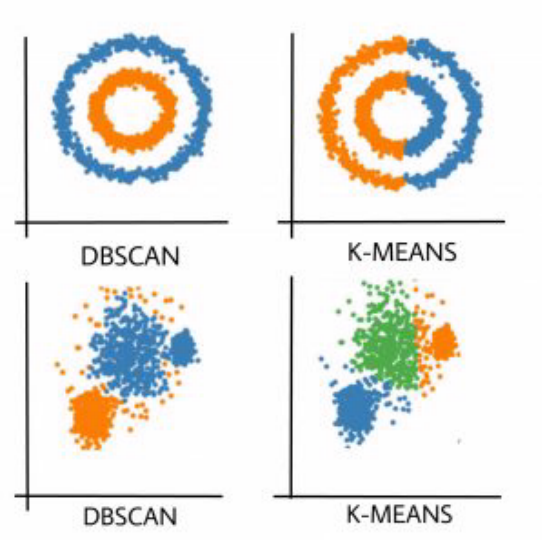
- Set $\tau = 4$ and compute the 4-distance of each point (distance to the 4th closest neighbor)
- Set ϵ to a threshold value where there is a 'valley' in the 4-distance graph.
  - Points to the left of the threshold are noise
  - Points to the right belong to clusters.

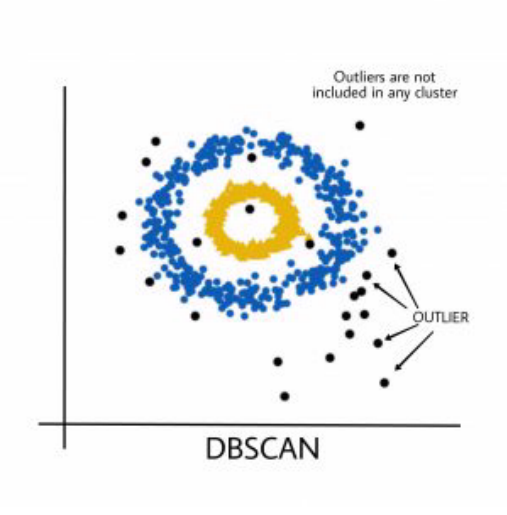


[4] From the original article Ester et al. (1996). See also Schubert et al. (2017) for further hints and discussion.

DBSCAN, pros and cons

+ Does not require the number of clusters (unlike K-means)
+ Finds clusters of arbitrary shape
+ Is robust to outliers (these are treated as noise)
- A single density does may not work well for all clusters
- Selecting the neighborhood size can be difficult

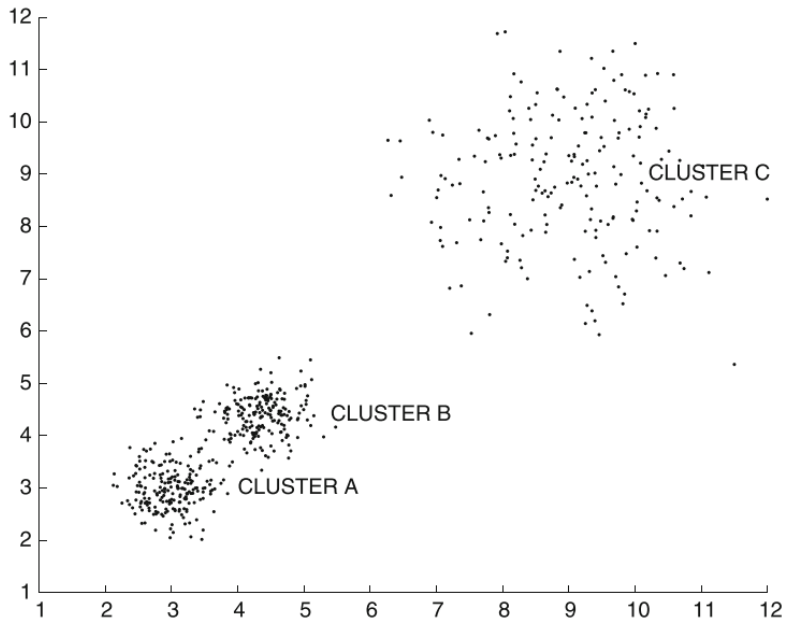




www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

DBSCAN, problem

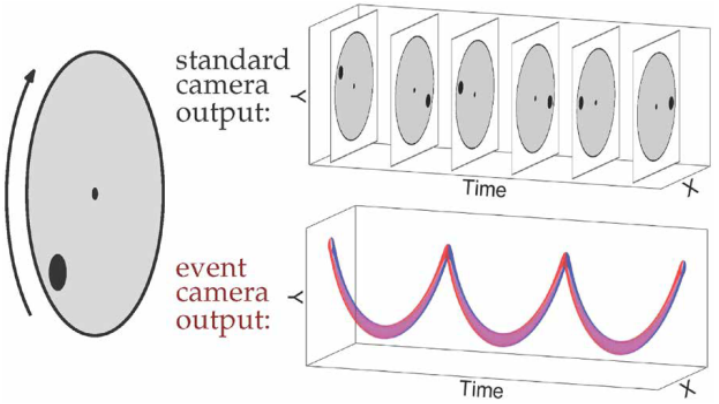
- What do you think will happen with DBSCAN for the shown data if the density parameter is selected to be
  1. too small?
  2. too large?



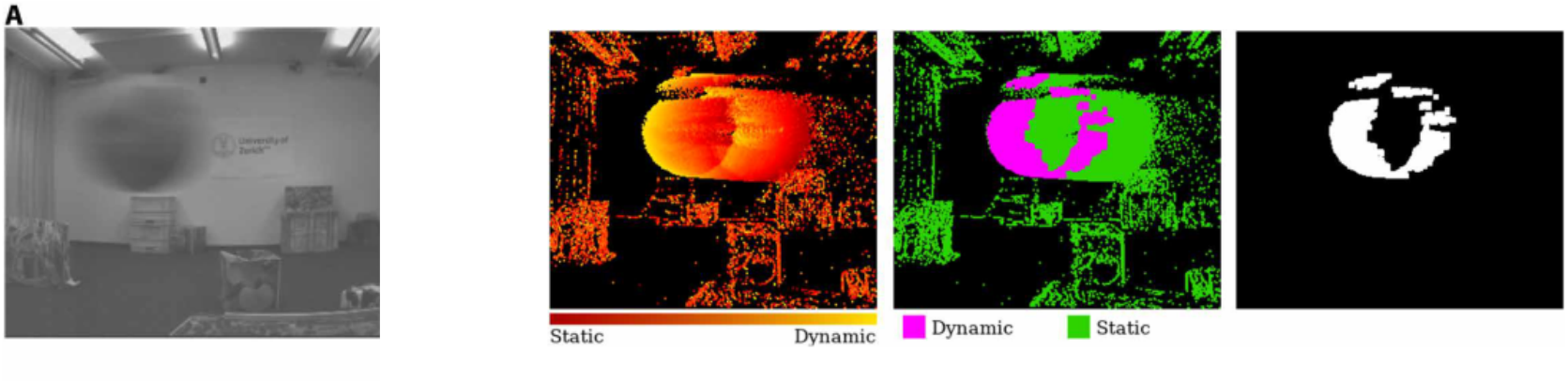
Aggarwal, Data Mining, Fig. 6.14

DBSCAN application: Motivation

- Falanga et al. (2020). **Dynamic obstacle avoidance quadrotors with event cameras**. *Science Robotics*.



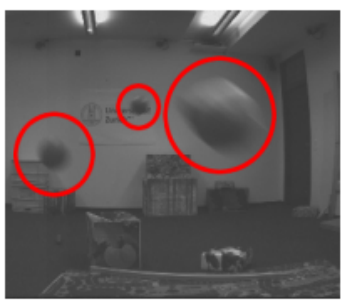
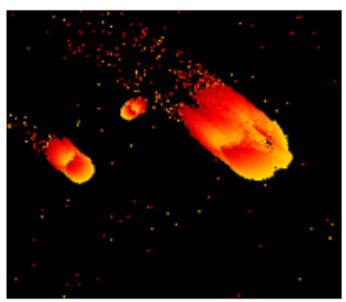
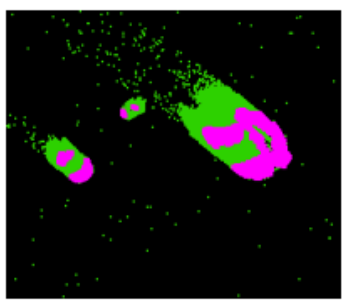
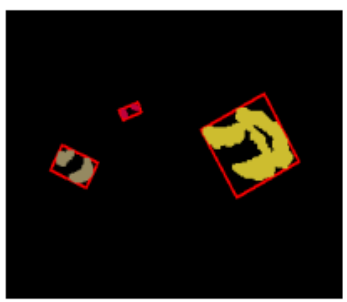
DBSCAN application: Approach



- Pixels are colored by the average event times within a time slice.
  - Uniformly distributed events are from static background.
  - Pixels with events accumulated at a certain time interval are from dynamic objects.
- Pixels from dynamic objects are clustered into objects using DBSCAN.

DBSCAN application: Why dbscan

- DBSCAN
  - can handle an arbitrary number of objects
  - can detect objects (clusters) of various shapes
  - is robust against noise
  - is very fast

DBSCAN application: Implementation details

- To cluster image pixels, DBSCAN needs a distance measure.
- In the application, the following distance is used:

$$C_{ij} = w_p \underbrace{\|\mathbf{p}_i - \mathbf{p}_j\|}_{\Delta\text{location}} + w_v \underbrace{\|\mathbf{v}_i - \mathbf{v}_j\|}_{\Delta\text{velocity}} + w_\rho \underbrace{|\rho_i - \rho_j|}_{\Delta\text{event times}}$$

- Two image pixels are likely clustered together if:
  - they are close to each other in space
  - if their image plane velocities are similar
  - if the event time stamps are similar

Summary

- Clustering is an unsupervised method where the goal is to find groups of similar data points in unlabeled data.
- Different algorithms with different properties exist:
  - **K-means** tries to find cluster centroids and assign nearby point to the centroids, often resulting in 'spherical' clusters.
  - **DBSCAN** combines points that belong to the same connected dense region, and can identify clusters of various shapes.
- Selecting hyperparameters carefully and ensuring convergence are necessary for good clustering performance. Always investigate the sensitivity of your results w.r.t. the hyperparameters and run the algorithm with different initializations (when applicable).

Recommended reading

- Aggarwal: Data Mining - The Text book
  - K-means, Section 6.3
  - DBSCAN, Section 6.6