Data Science

Unsupervised Learning – Part 1

Lionel Fillatre

Polytech Nice Sophia

lionel.fillatre@univ-cotedazur.fr

Join us!

- Moodle website

https://lms.univ-cotedazur.fr/2024/course/view.php?id=1850

Keyword : FH65hs32&

Classrooms for labs with Fillatre's Lectures

- In French
  - Teacher: Guillaume Gros
  - Students: MAM5

- In English
  - Teacher: Aneva Tsafack
  - Students: M2 IM-MSD + EIT Digital M2

Outlines

- Course Logistics
- General Introduction
- What is unsupervised learning?
- Covariance Estimate
- Density Estimate
- Clustering
  - K-means
  - DBSCAN
  - Hierarchical clustering
  - Performance Metrics
- Gaussian mixture models
- Decomposing signals in components
  - PCA
  - Kernel PCA
- Novelty and outlier detection
  - Robust covariance
  - One-class SVM
  - Isolation forest
  - Local outlier factor
- Manifold learning
  - Locally Linear Embedding
- Conclusion

Course logistics

A Brief Description

- This course offers an introduction to data science as well as various software tools.
- It provides a comprehensive presentation of business aspects of Data Science.
- It also provides an overview of machine learning approaches for Data Science
  - Unsupervised learning and metrics
  - Reinforcement learning

Timeline of this course

1. 16/09/2024 – 17h00:19h30: Data Science to work for business (Yann Gouedo)
2. 23/09/2024 – 13h30:17h30: Introduction and Unsupervised Learning – Part 1 (Lionel Fillatre)
3. 30/09/2024 – 17h00:19h30: IBM Watson Studio and Customer Analytics (Yann Gouedo)
4. 07/10/2024 – 13h30:17h15: Unsupervised Learning – Part 2 (Lionel Fillatre)
5. 14/10/2024 – 17h00:19h30: Risk and Fraud Management + quizz (Yann Gouedo)
6. 14/10/2024 – 14h00:15h00: Written exam (Lionel Fillatre)
7. 21/10/2024 – 13h30:17h15: Reinforcement Learning – Part 1 (Lionel Fillatre)
8. 04/11/2024 – 13h30:17h15: Reinforcement Learning – Part 2 (Lionel Fillatre)
9. 18/11/2024 – 13h30:17h15: Reinforcement Learning – Part 3 + graded lab (Lionel Fillatre)

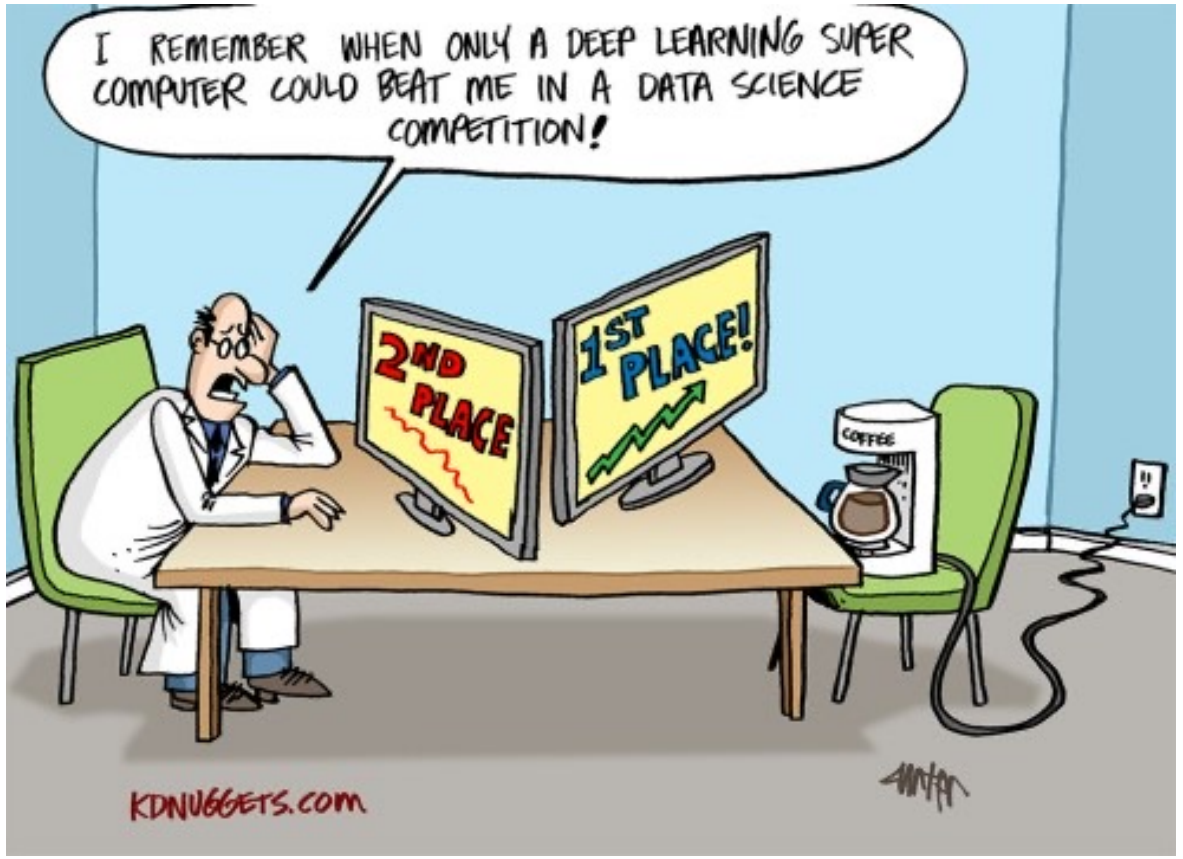
Grading

1. Written exam on unsupervised learning (Lionel Fillatre)
2. Quiz on business aspects (Yann Gouedo)
3. Lab on reinforcement learning (Lionel Fillatre)

- Final grade: average of the three scores (same weight).

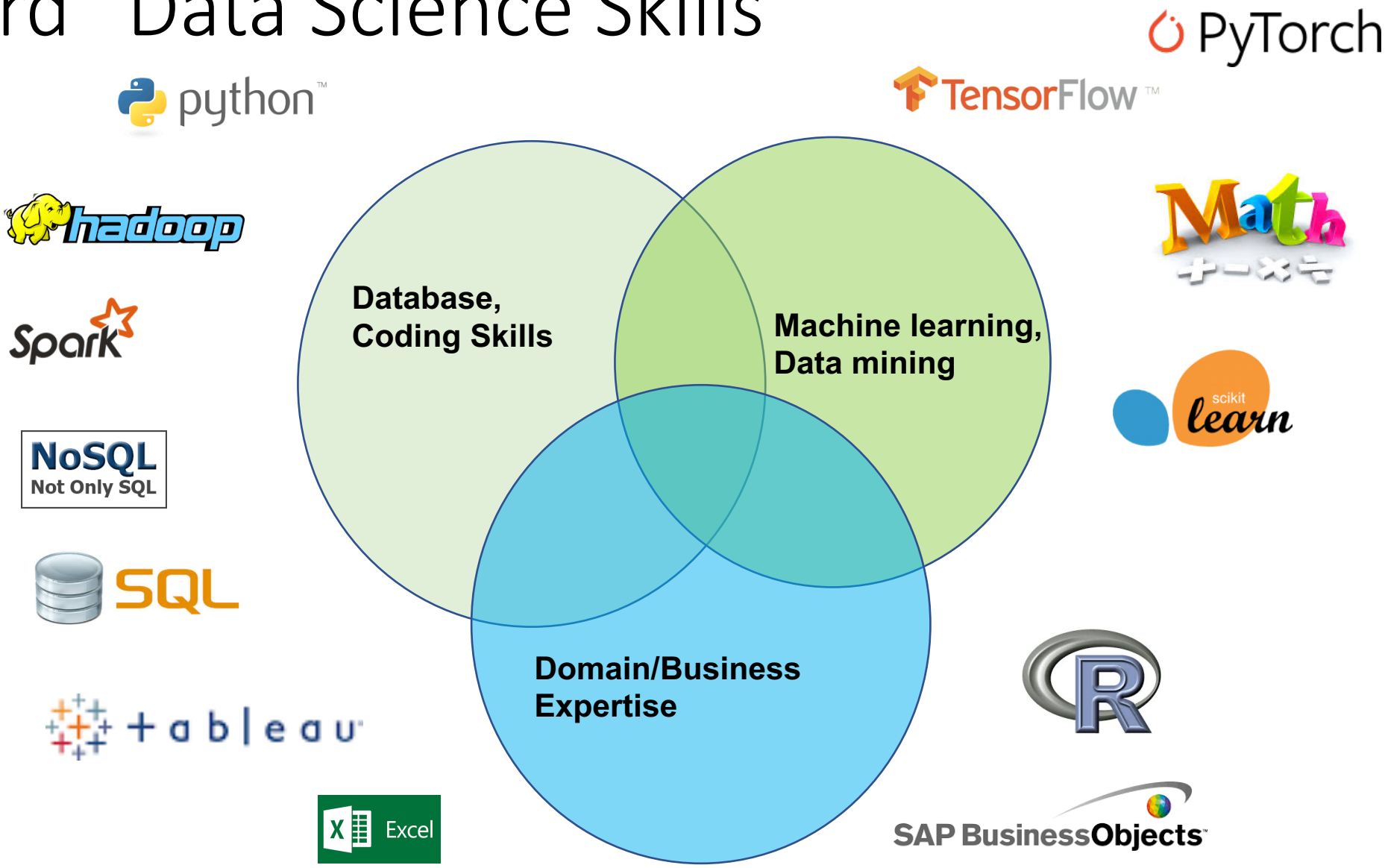
General introduction

Data Science Automation



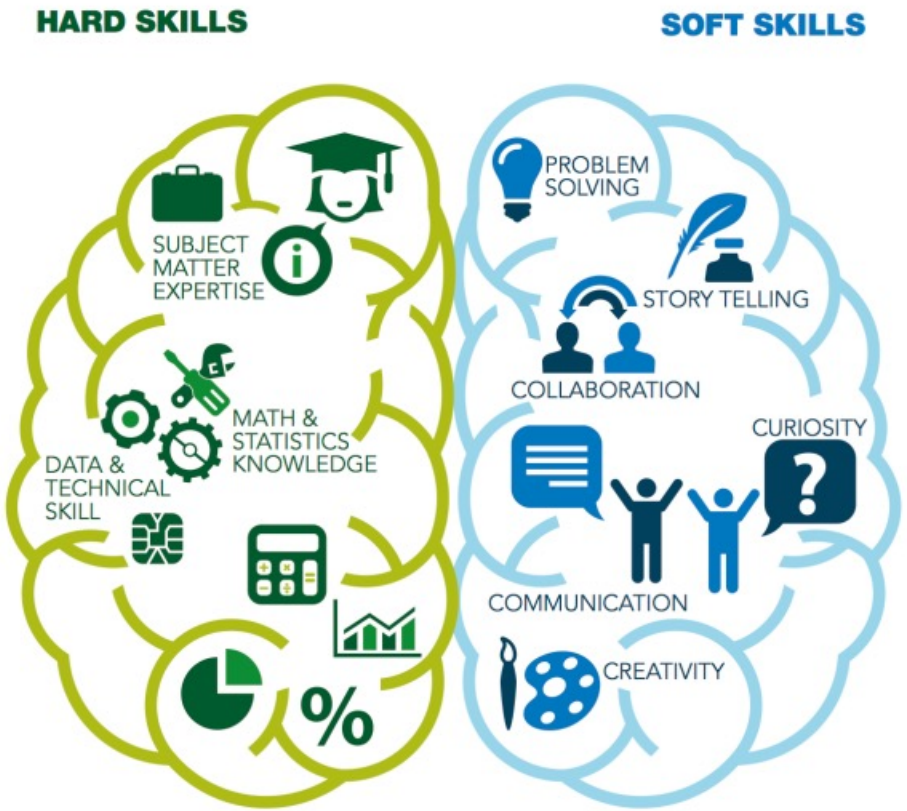
I remember when only a Deep Learning supercomputer could beat me in a Data Science competition

“Hard” Data Science Skills

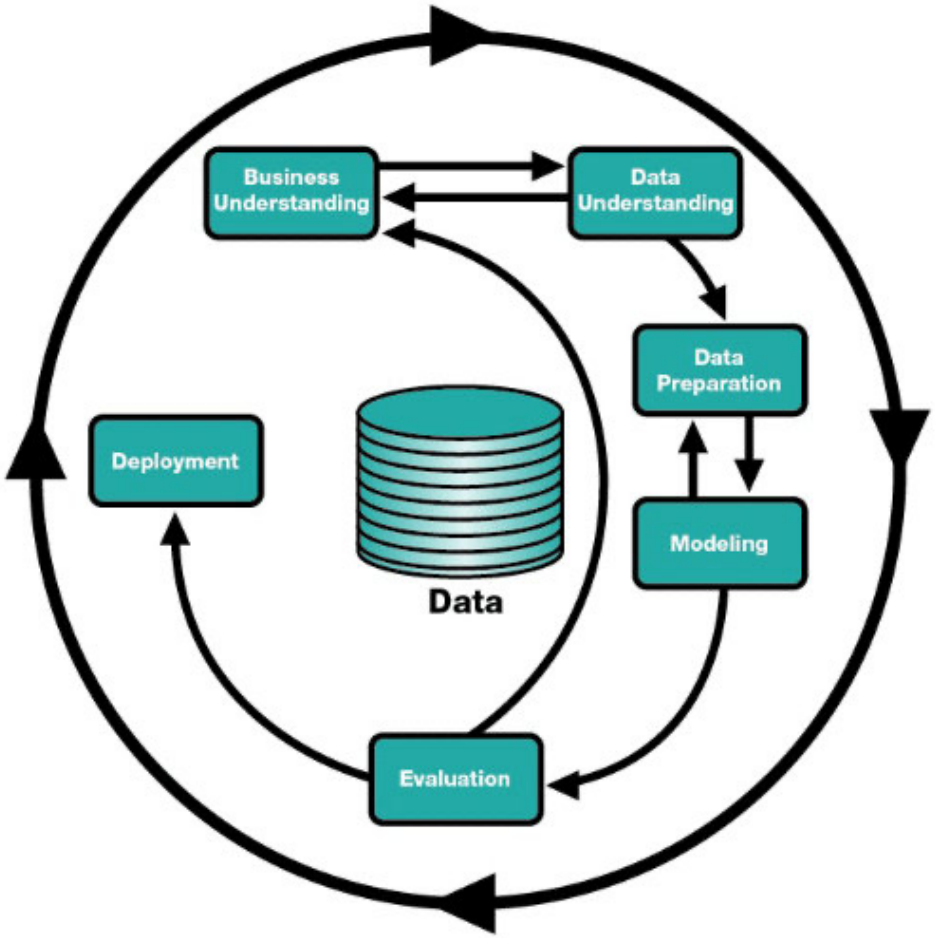


“Soft” Data Science Skills: Harder to Automate

- Curiosity
- Intuition
- Business Knowledge
- Selecting a good metric
- Posing the right question
- Presentation Skills



CRISP-DM: Iterative, Circular Process



CRISP-DM
Cross Industry Standard Process
for Data Mining

CRISP-DM, 1998

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

Data Engineering Takes The Bulk of Time

- Building Machine Learning/Predicting Models is the key (and most fun) part, but only a small part of the whole process

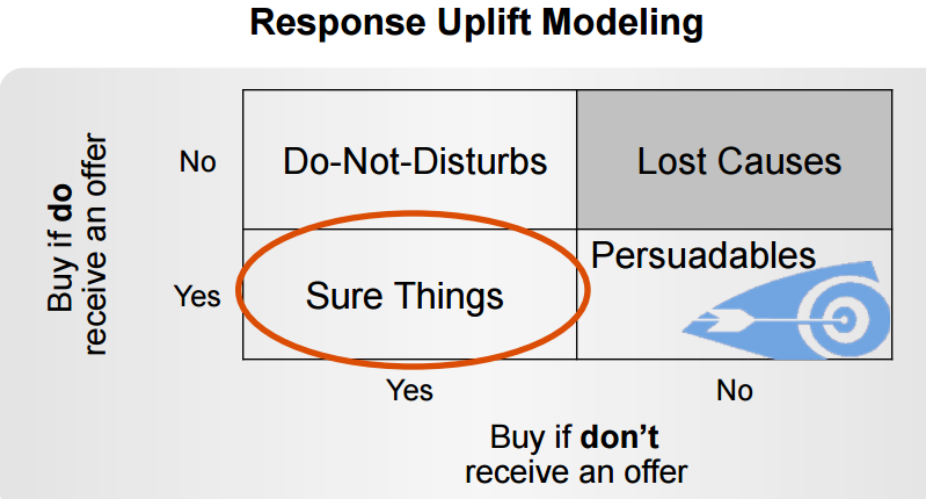
- 60-80% spent on Data Preparation/Engineering

Modeling

- No Free Lunch Theorem – no method is universally the best (Wolpert)

Focus on the Right Metric - Actionable

- Don’t model if consumer will buy
- Model if consumer will buy **in response to an offer**



- In Obama 2012 Campaign

www.thefiscaltimes.com/Articles/2013/01/21/The-Real-Story-Behind-Obamas-Election-Victory

Tell a story

- Combine facts into a story
- Combine visual and text presentation
- Explanation gives credibility
- Dynamic / Interactive

Limits to Predicting Human Behavior?

- Inherent randomness, complexity in human behavior
- Individual predictions have limited accuracy (but can still be better than random and very useful for consumer analytics)
- Aggregate predictions (e.g., who will win the election?) more accurate, because individual randomness cancels out

Deployment & Maintenance

- Netflix Prize winning algorithm not deployed

> … the additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment. Also, our focus on improving Netflix personalization had shifted to the next level by then.
>
> http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

- In real-world, simpler is usually better
- Is model explainable ? (legal, acceptance reasons)
- Deployment Test and Monitor
  - Monitor assumptions
    - Do fields have the same value distributions?
  - Detect when model is no longer valid, needs rebuilding
  - Automatic model re-build

What is unsupervised learning?

Machine Learning

- **Supervised:** We are given input samples $\boldsymbol{x}$ and output samples y of a function $y = f(\boldsymbol{x})$. We would like to "learn" f, and evaluate it on new data.
  - **Classification:** y is discrete (class labels).
  - **Regression:** y is continuous, e.g. linear regression.

- **Unsupervised:** Given only samples $\boldsymbol{x}$ of the data, we compute a function f such that $y = f(\boldsymbol{x})$ is "simpler" with y unkown (and often undefined).
  - **Clustering:** y is discrete
  - **Matrix factorization, Kalman filtering, unsupervised neural networks**: y is continuous

A few examples

- **Supervised:**
  - Is this image a cat, dog, car, house?
  - How would this user score that restaurant?
  - Is this email spam?

- **Unsupervised**
  - Cluster some hand-written digit data into 10 classes.
  - What are the top 20 topics in Twitter right now?
  - How to define the manifold in which this dataset lives in?

Lecun's Cake

- Reinforcement learning (cherry)
  - The machine predicts a scalar reward given once in a while
  - A few bits for some samples
- Supervised learning (icing)
  - The machine predicts a category or a few numbers for each input
  - 10->10,000 bits per sample
- Unsupervised learning (cake)
  - The machine predicts any part of its input for any observed part
  - Million of bits per sample

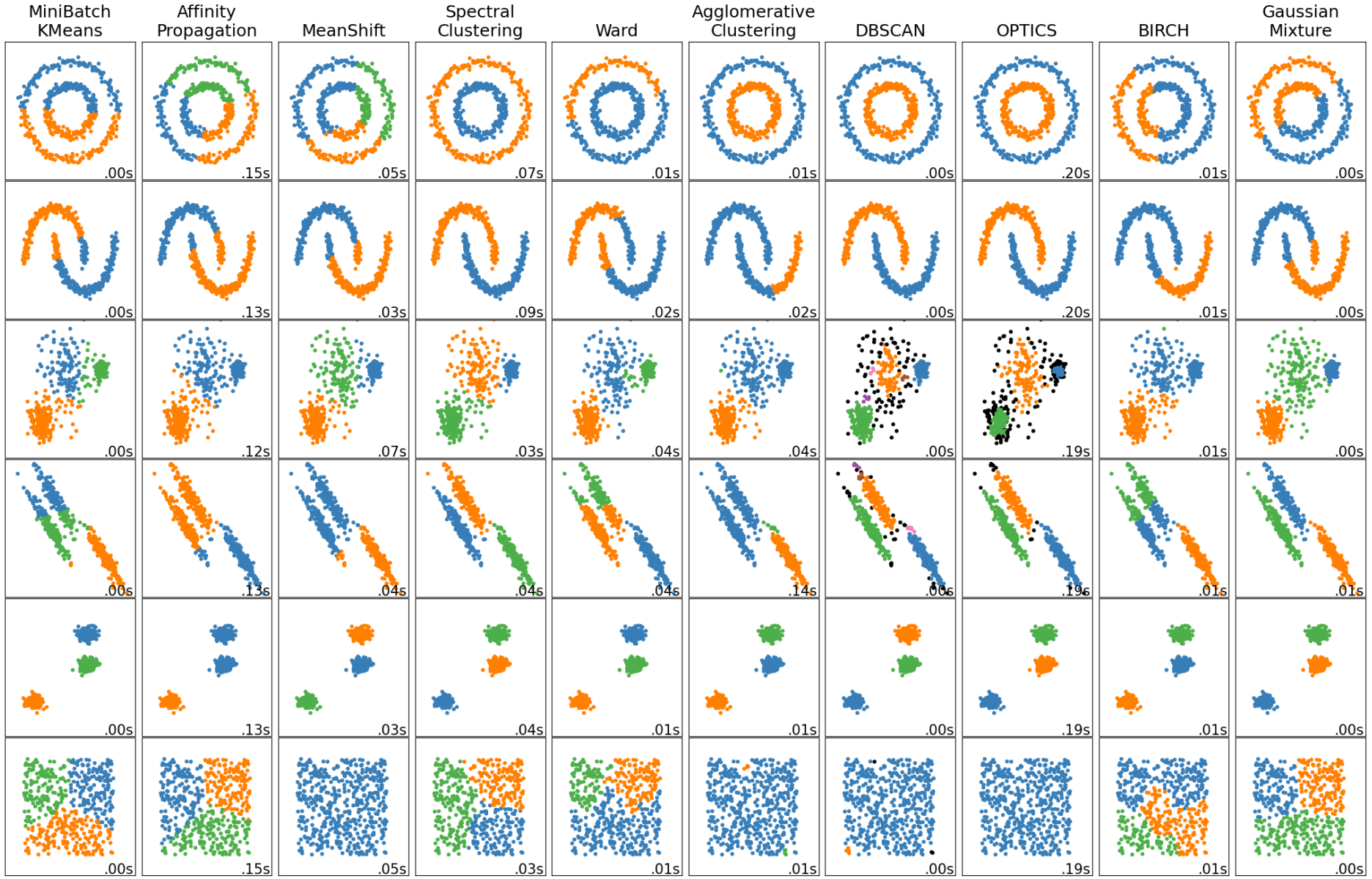
Unsupervised Learning

- **Why unsupervised learning?**
  - Creating labeled datasets for each task is an expensive, time-consuming, tedious task
  - Requires hiring human labelers, preparing labeling manuals, creating GUIs, creating storage pipelines, etc.
  - High quality annotations in certain domains can be particularly expensive (e.g., medicine)
  - Unsupervised learning takes advantage of the vast amount of unlabeled data on the internet (images, videos, text)
  - Rich discriminative features can be obtained by training models without actual labels

- **Main challenges for unsupervised learning**
  - There is no gold standard for comparison of learned feature representations
  - Selecting a suitable loss functions, since there is no single objective as the test set accuracy in supervised learning

Main approaches in unsupervised learning

- Covariance estimation
- Density estimation
- Clustering
- Biclustering
- Gaussian mixture models
- Decomposing signals in components (matrix factorization, PCA, SVD)
- Novelty and outlier detection
- Manifold learning
- Neural networks



https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py

Covariance Estimate

Motivation

- Several cases:
  - Many statistical problems require the estimation of a population's covariance matrix (PCA, etc.)
  - When you assume your data samples follow a Gaussian distribution, the estimation of the covariance matrix is crucial

- Such an estimation has to be done on a sample whose properties (size, structure, homogeneity) have a large influence on the estimation's quality.

Matrix of data $X = (X_{i,j}) \in \mathbb{R}^{p \times N}$

- Data $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$
- $\boldsymbol{x}_i \in \mathbb{R}^p$

$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	…	$\boldsymbol{x}_N$
X_{11}	X_{12}		X_{1N}
X_{21}	X_{22}		X_{2N}
X_{p1}	X_{p2}		X_{pN}

Sample covariance matrix

- Given data $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, compute the covariance matrix estimate $\hat{\Sigma}$ **(maximum likelihood estimate)**

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T$$

$$\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i$$

- Matrix form (observation vectors as the columns of a matrix X):

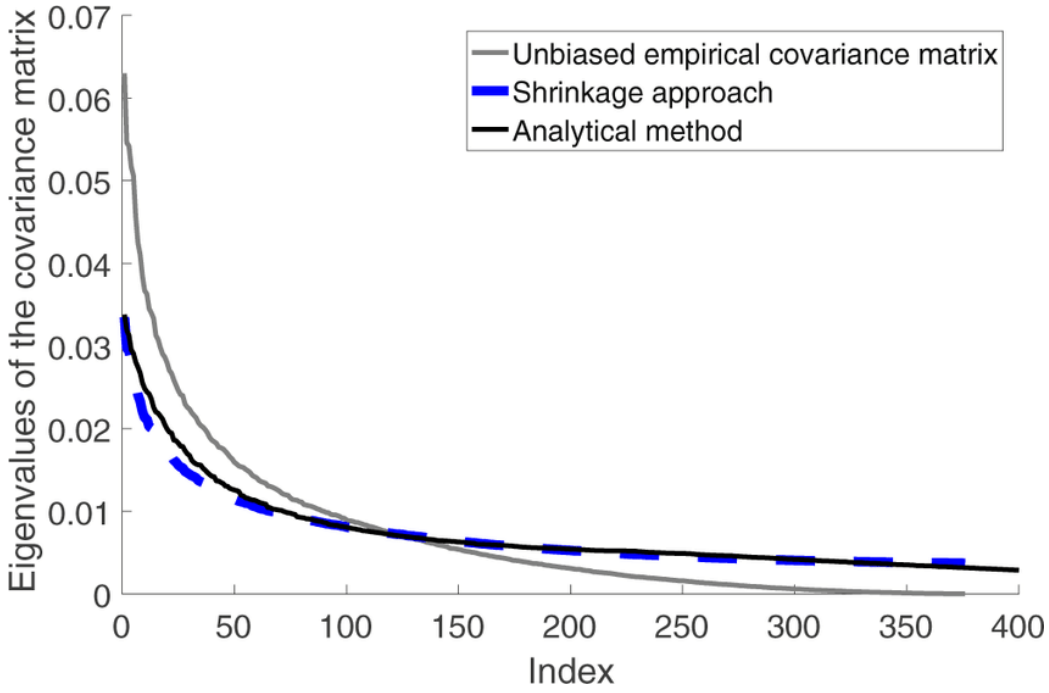
$$\hat{\Sigma} = \frac{1}{N}(X - \overline{\boldsymbol{x}}1_N^T)(X - \overline{\boldsymbol{x}}1_N^T)^T = \frac{1}{N}\tilde{X}\tilde{X}^T$$

$\tilde{X} = X - \overline{\boldsymbol{x}}1_N^T$ is the matrix of centered data

$1_N \in \mathbb{R}^N$ is a vector of 1's.

Regularization

- The Maximum Likelihood Estimator $\hat{\Sigma}$ is an asymptotically unbiased estimator of the covariance matrix
- But $\hat{\Sigma}$ is not a good estimator of the eigenvalues of the covariance matrix.
- Sometimes, $\hat{\Sigma}$ cannot be inverted for numerical reasons.
- A transformation of the empirical covariance matrix has been introduced: the **shrinkage**.
- When the number of samples is much larger than the number of features, one would expect that no shrinkage would be necessary



Shinkage covariance estimate

- The general idea of shrinkage is to stabilise a poor matrix estimate by blending it with a stable known matrix

$$\hat{\Sigma}_s = (1-\alpha)\hat{\Sigma} + \alpha \frac{\text{trace}(\hat{\Sigma})}{p} I_p$$

- $\hat{\Sigma}_s$ is invertible provided that $0 < \alpha \leq 1$
- $\frac{\text{trace}(\hat{\Sigma})}{p}$ is the average variance (since $\text{trace}(\hat{\Sigma})$ is the sum of the diagonal elements)

Ledoit-Wolf shrinkage

- Let the dot product for a $N \times N$ matrix

$$\langle A|A\rangle = \frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{N} A_{i,j}$$

- Let the Frobenius norm $\|A\|_F = \langle A|A\rangle$

- O. Ledoit and M. Wolf propose a formula to compute the optimal shrinkage coefficient α that minimizes the Mean Squared Error between the estimated and the real covariance matrix

$$\min_{\hat{\Sigma}_{LW}} \left\|\hat{\Sigma}_{LW} - \hat{\Sigma}\right\|_F^2$$

- The solution $\hat{\Sigma}_{LW}$ is

$$\hat{\Sigma}_{LW} = (1-\alpha)\hat{\Sigma} + \alpha\mu I_p$$

with $\mu = \left\langle \hat{\Sigma} \middle| I_p \right\rangle$

Other methods

- A lot of methods exist to regularize the covariance estimate
- It depends on your practical assumptions
  - Sparsity,
  - Gaussian distributed samples,
  - etc.

Density Estimate

Kernel density estimation methods

- Consider a simple estimator of the cumulative distribution function:

$$F_N(x) = \frac{\#\{x_i \leq x\}}{N}$$

- Derivative gives an estimator of the density function, but this is just a set of delta peaks.

- Derivative is defined as

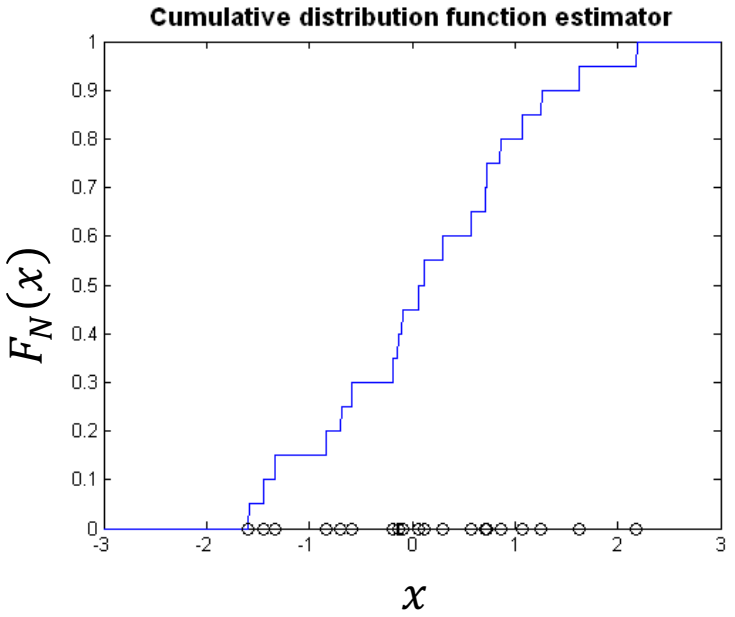
$$f(x) = \lim_{h \to 0} \frac{F_N(x+h) - F_N(x-h)}{2h}$$

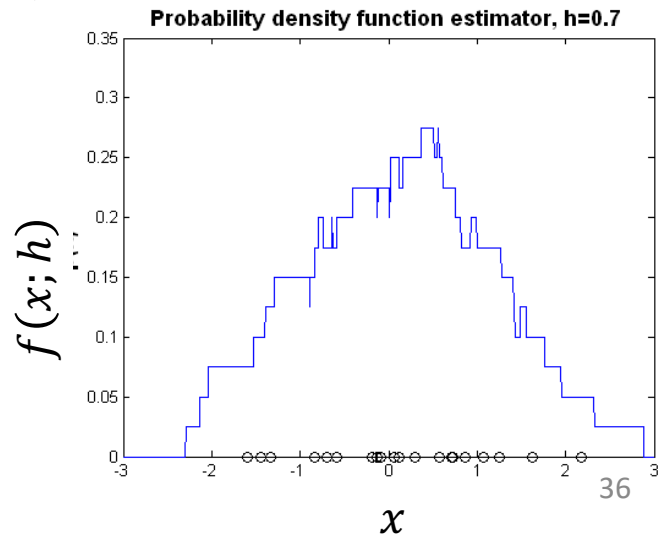
- Consider a non-limiting value of *h*:

$$f(x;h) = \frac{F_N(x+h) - F_N(x-h)}{2h} = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right)$$

$$K(z) = \begin{cases} 0 : |z| > 1 \\ \frac{1}{2} : |z| \leq 1 \end{cases}$$

- Each data point adds $\frac{1}{2Nh}$ in region of size h around it, sum of "blocks" gives estimate





Smoothing

- The idea of smoothing is to replace an observation at x with a smooth (differentiable) local kernel function $K(x) \geq 0$.
- The functions should satisfy

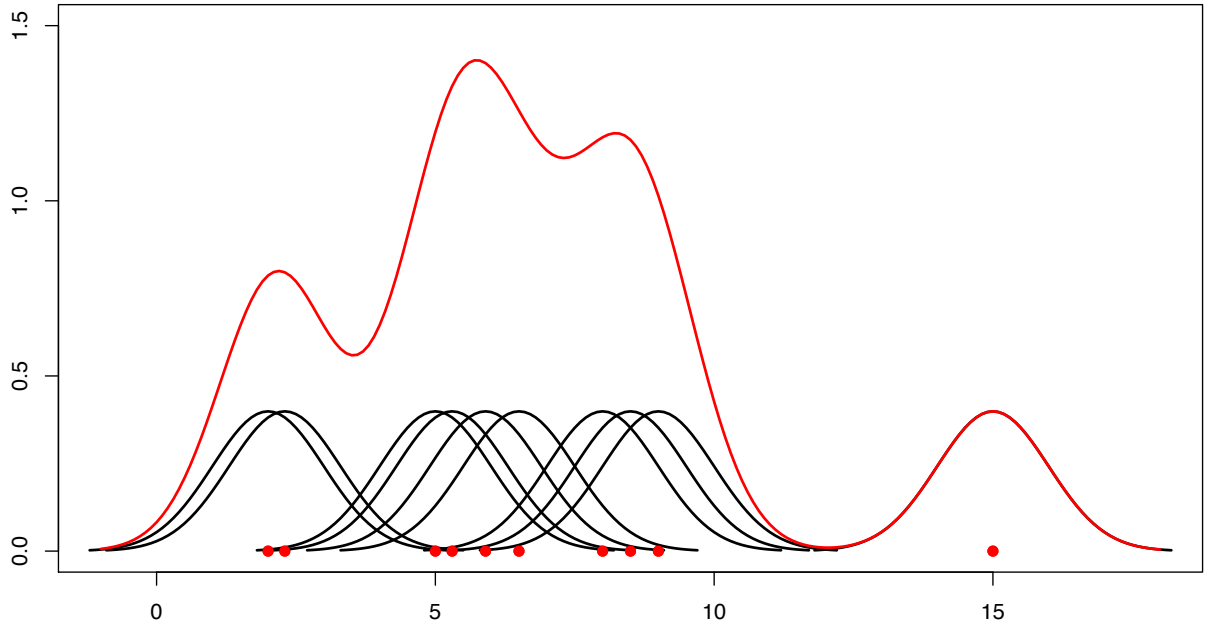
$$\int K(x)dx = 1$$

$$\int xK(x)dx = 0$$

$$\sigma_K^2 = \int x^2 K(x)dx < \infty$$

Kernel density estimates

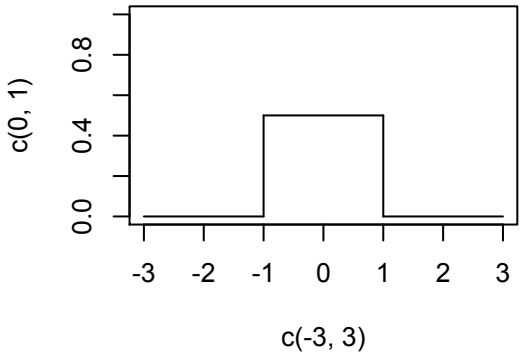
$$\hat{f}(x;h) = \frac{1}{h}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right)$$



Kernels

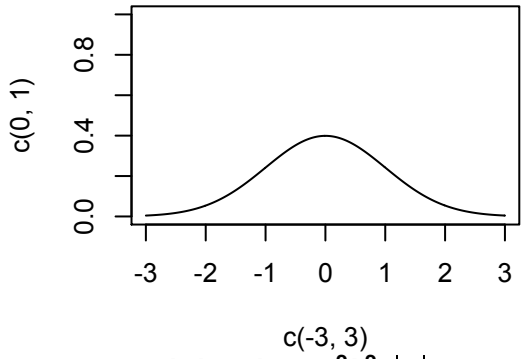
$$K(x) = \frac{1}{2}\mathbf{1}\{|x| \leq 1\}$$

Rectangular



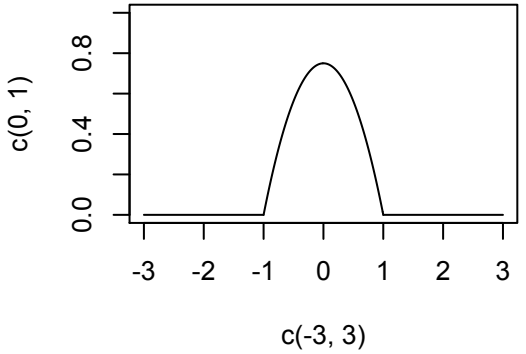
$$K(x) = \frac{1}{\sqrt{2}}e^{-\frac{x^2}{2}}$$

Gaussian



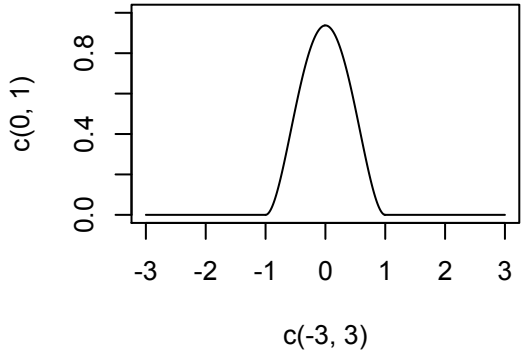
$$K(x) = {}^{3}/_{4}\,(1 - x^2)\mathbf{1}\{|x| \leq 1\}$$

Epanechnikov

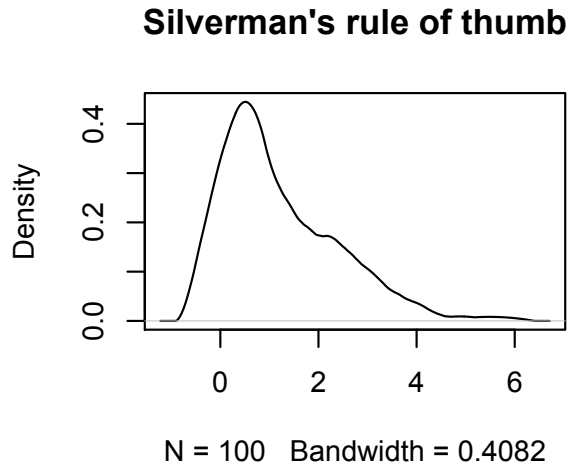


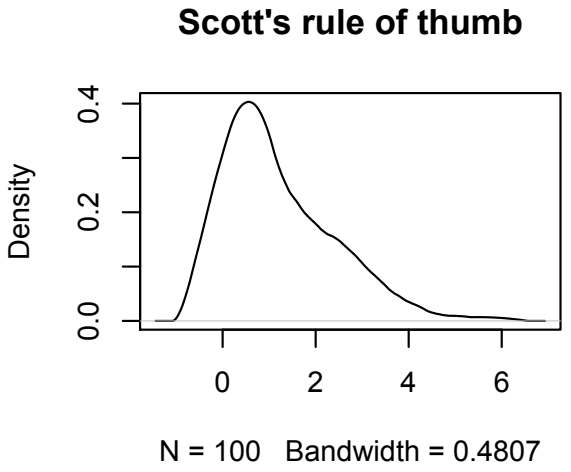
$$K(x) = (1 - x^2)^2\mathbf{1}\{|x| \leq 1\}$$

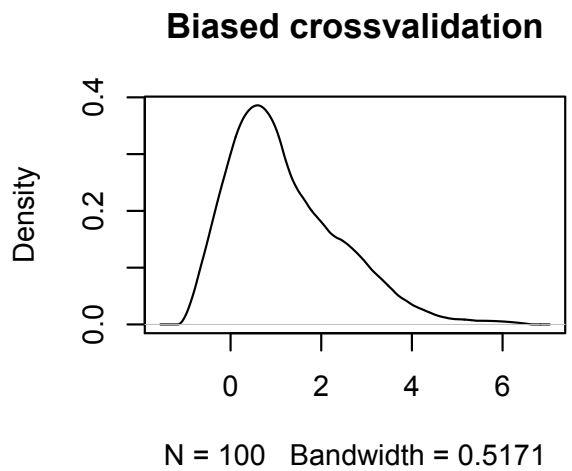
Biweight

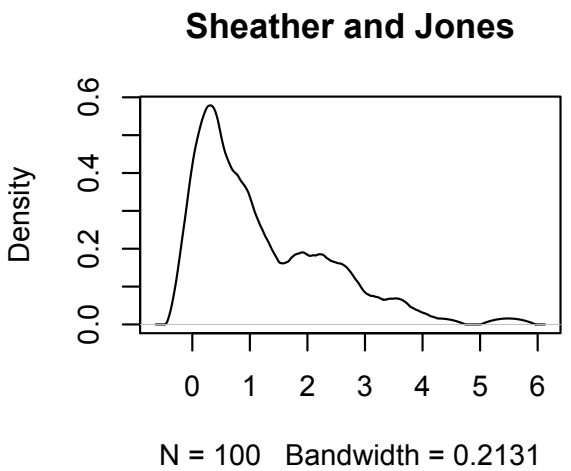


Bandwidth differences









Large dataset

- When N is large, the estimation lead to a burden
- **But due to the smoothing, the estimation depends only on neighboring samples**
- Idea: use an appropriate storage to get easily the neighbors of a given sample
- Solutions:
  - Kd-tree
  - Ball-tree

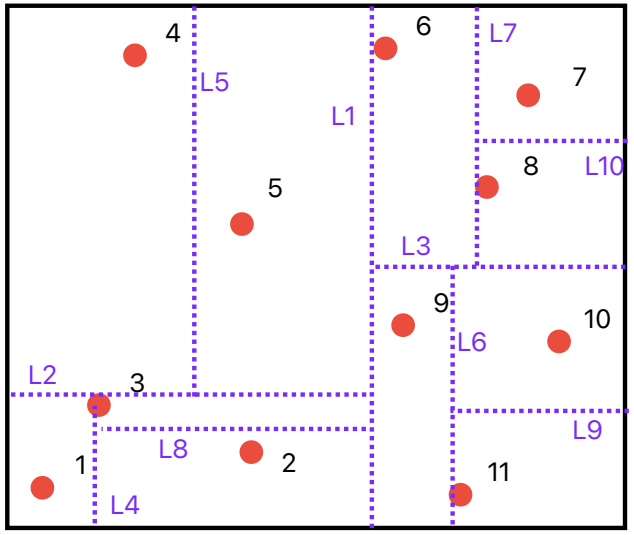
2-dimensional kd-trees

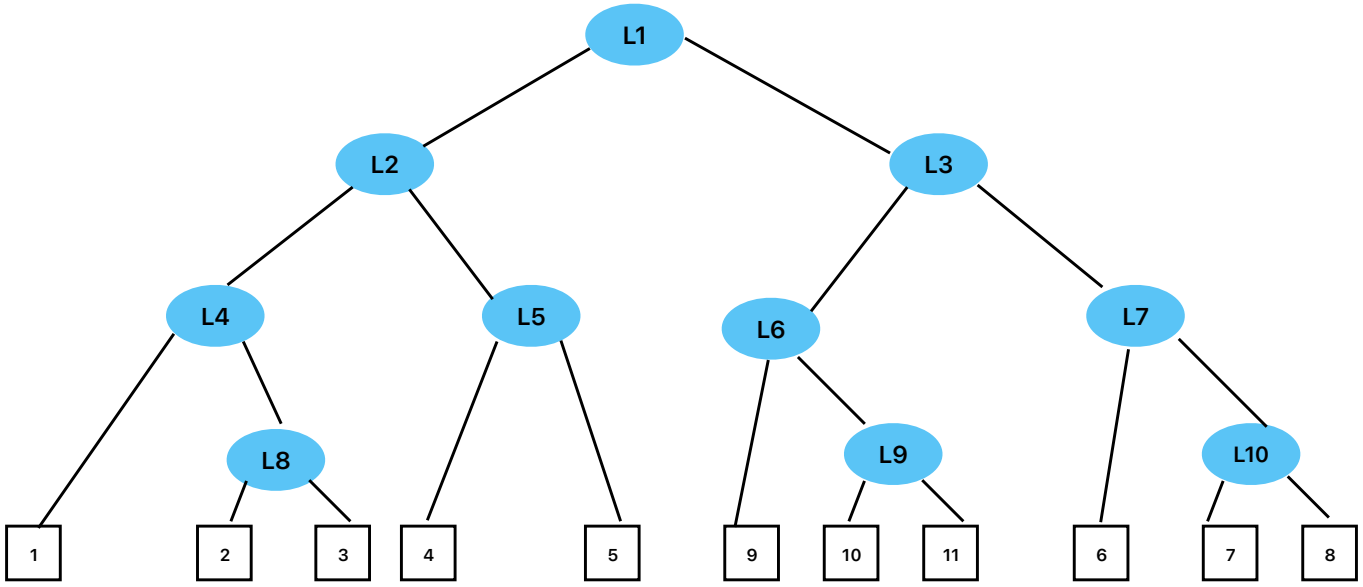
- A data structure to support nearest neighbor and rangequeries in $\mathbb{R}^2$.
  - Not the most efficient solution in theory.
  - Everyone uses it in practice.

- Algorithm
  - Choose x or y coordinate (alternate).
  - Choose the median of the coordinate; this defines a horizontal or vertical line.
  - Recurse on both sides until there is only one point left, which is stored as a leaf.

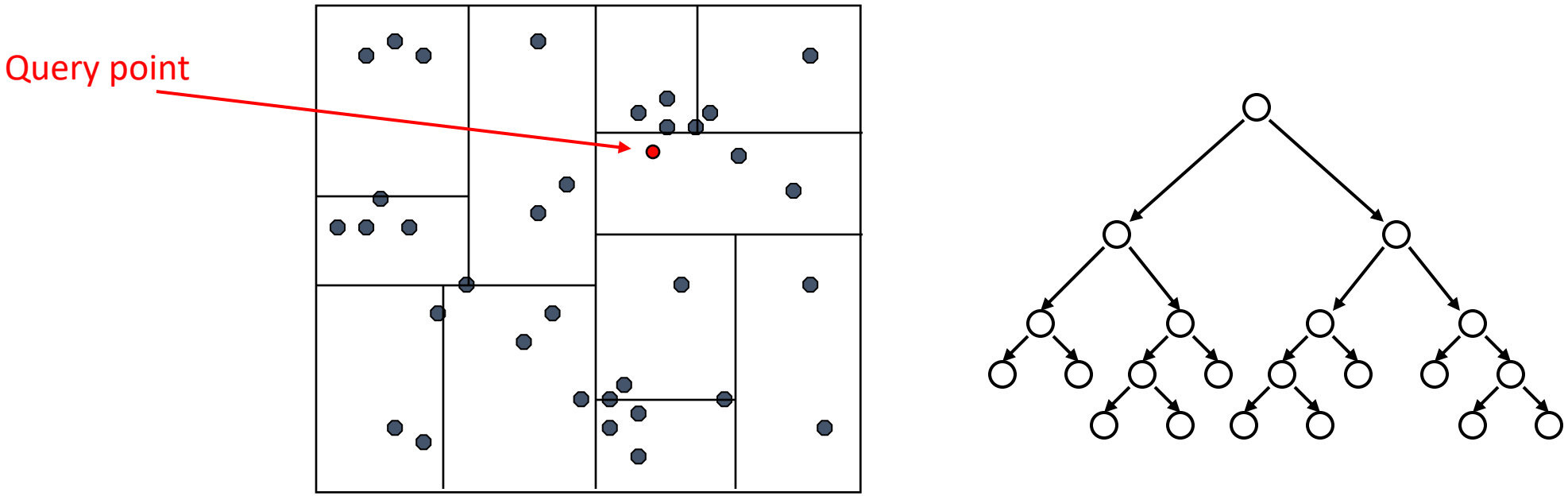
- We get a binary tree
  - Size $O(N)$.
  - Construction time $O(N \log N)$
  - Depth $O(\log N)$.
  - K-Nearest Neighbors query time: $O\left(k + \sqrt{N}\right)$ where k is the number of neighbors

Kd-trees


4
L5
6
L7
7
L1
8
L10
5
L3
9
10
L6
L2
3
L9
L8
2
1
11
L4

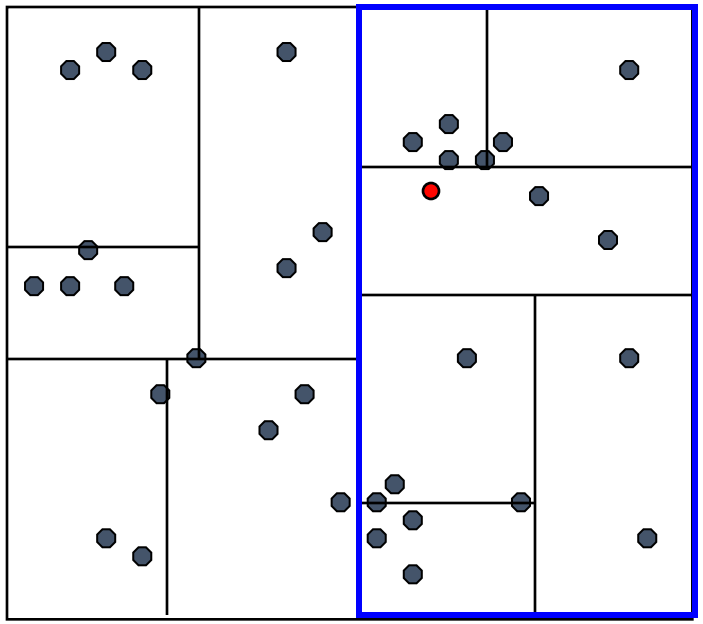

L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
1
2
3
4
5
9
10
11
6
7
8

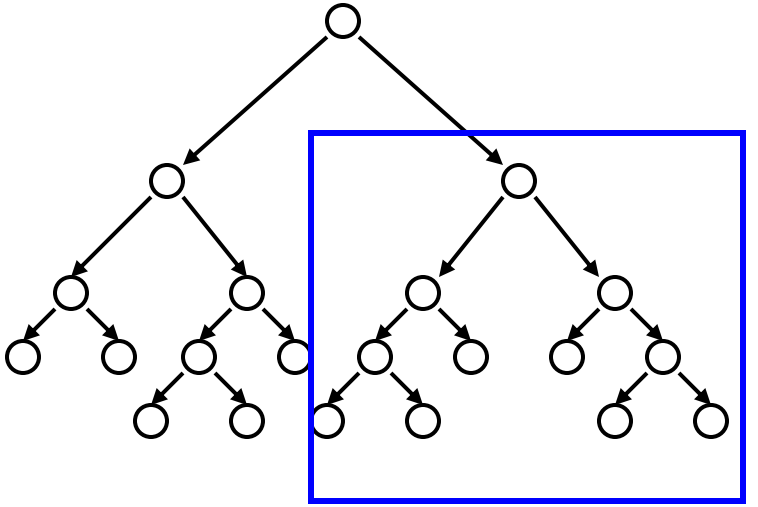
Nearest Neighbor with KD Trees



- Starting at the root node, we traverse the tree looking for the nearest neighbor of the query point.

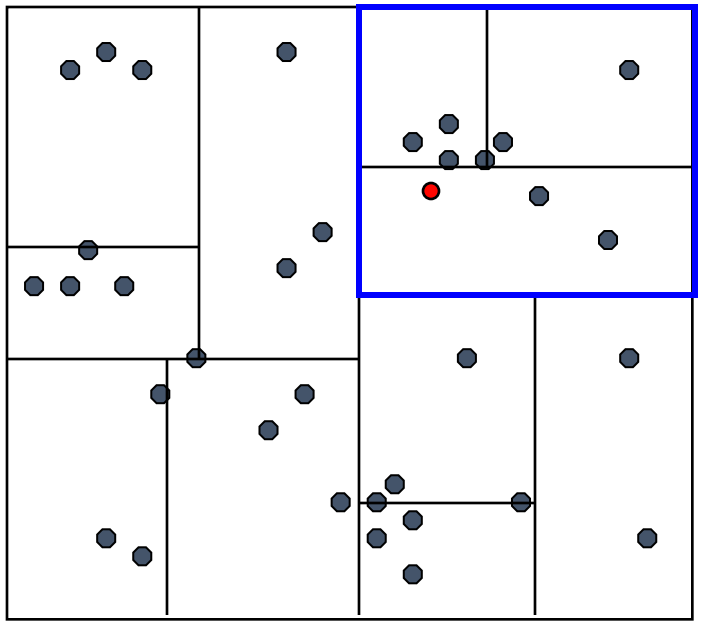
Nearest Neighbor with KD Trees

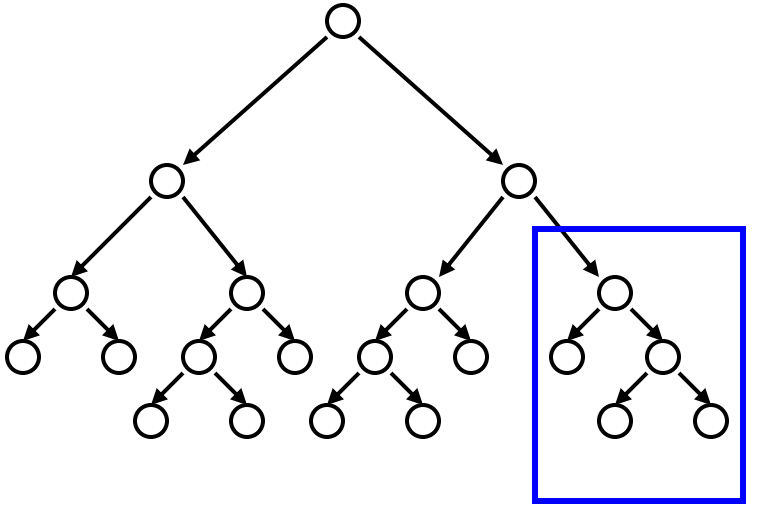




- Explore the branch of the tree that is closest to the query point first.

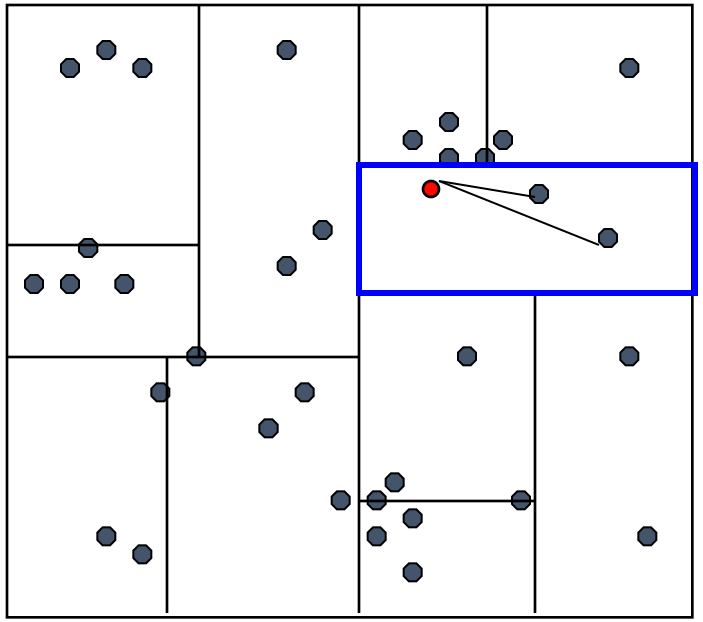
Nearest Neighbor with KD Trees

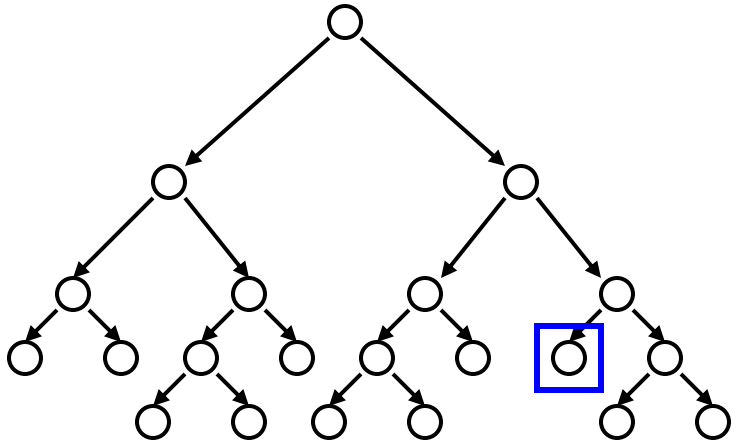




- Explore the branch of the tree that is closest to the query point first.

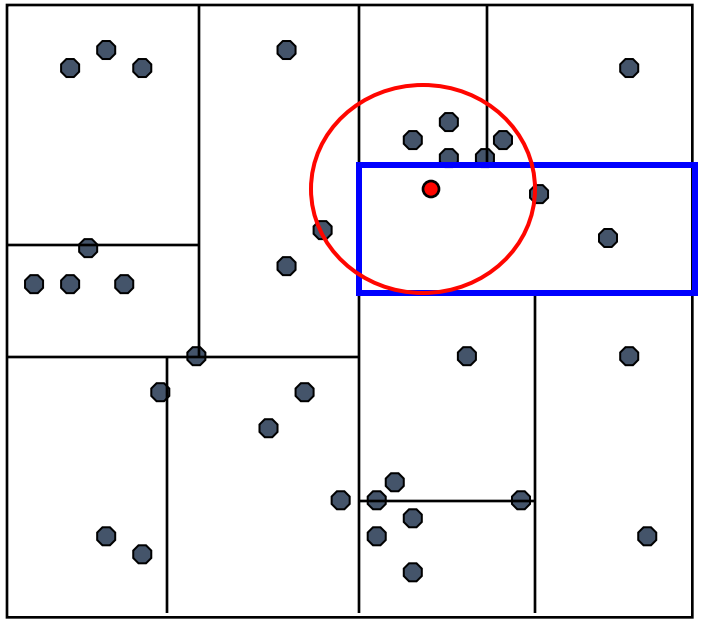
Nearest Neighbor with KD Trees

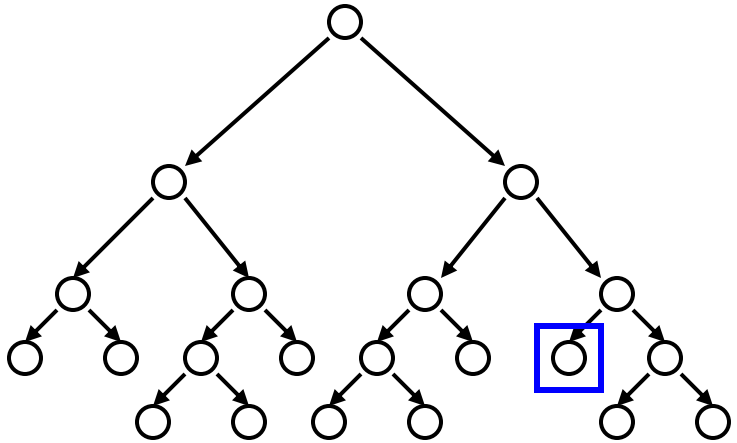




- When we reach a leaf node, compute the distance to each point in the node.
- The exploration is ended.

Nearest Neighbor with KD Trees

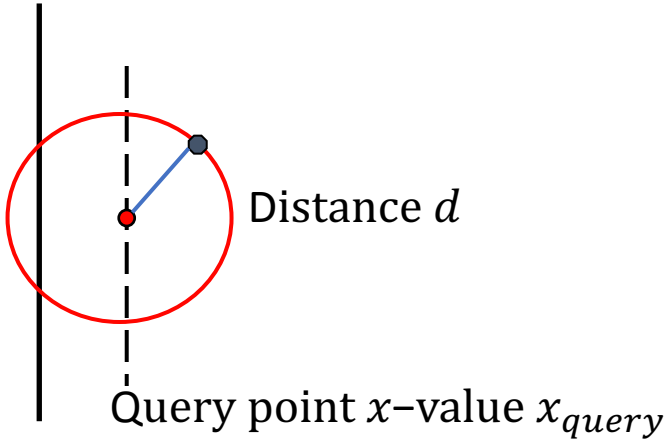




- Next, we are checking if a hypersphere around the query point with a radius equal to the current nearest distance intersects the splitting hyperplane of the node.
- This comparison is fast and easy because of the KD tree splitting strategy (the hyperplanes are all axis-aligned).

Why does k-d tree work?

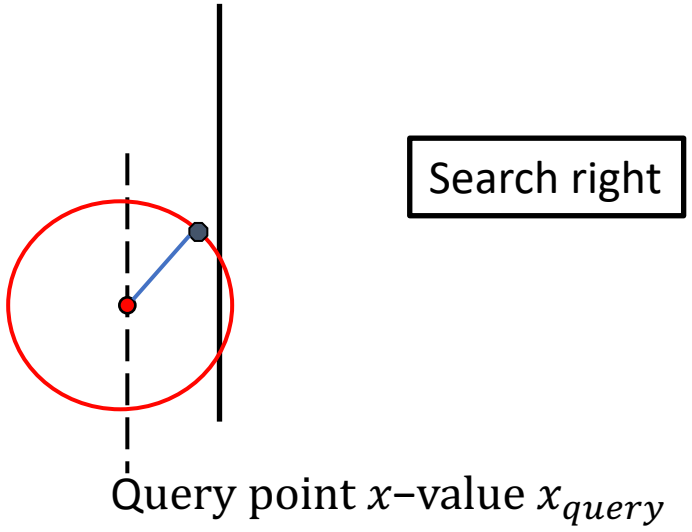
Search left



Splitting axis x-value x_{axis}

$x_{query} - d \leq x_{axis}$
means the circle overlaps with the left subtree.

Search right

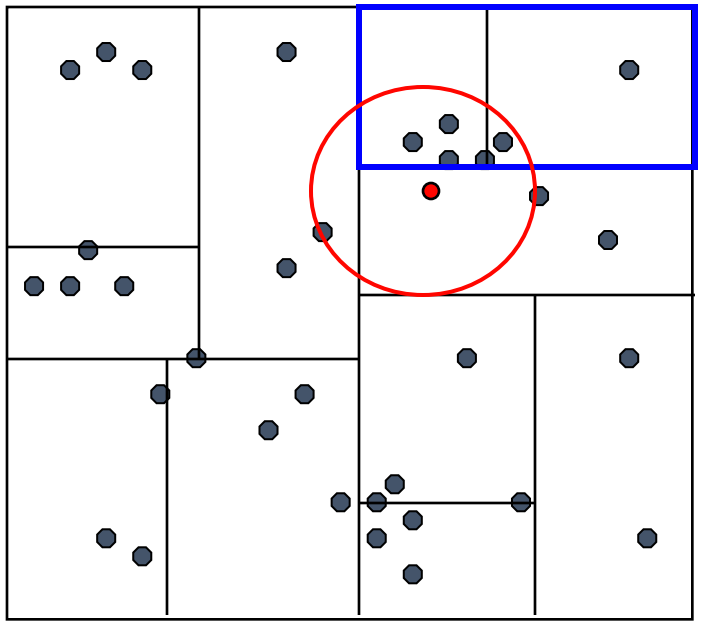


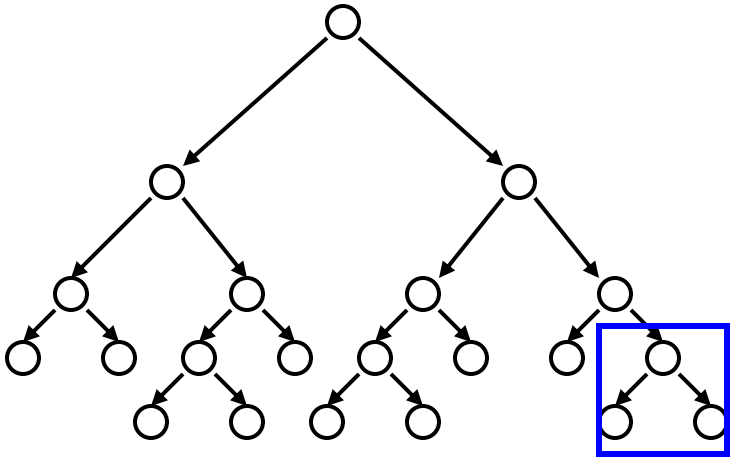
Splitting axis x-value x_{axis}

$x_{query} + d > x_{axis}$
means the circle overlaps with the right subtree.

http://andrewd.ces.clemson.edu/courses/cpsc805/references/nearest_search.pdf

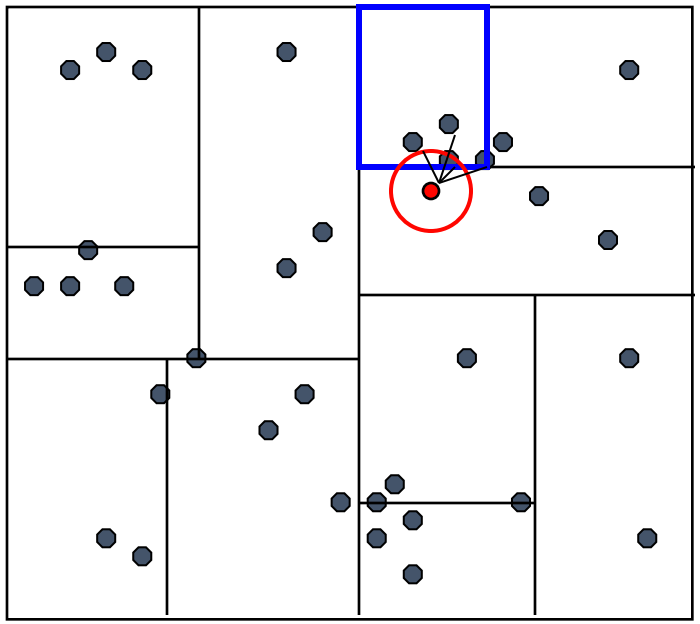
Nearest Neighbor with KD Trees

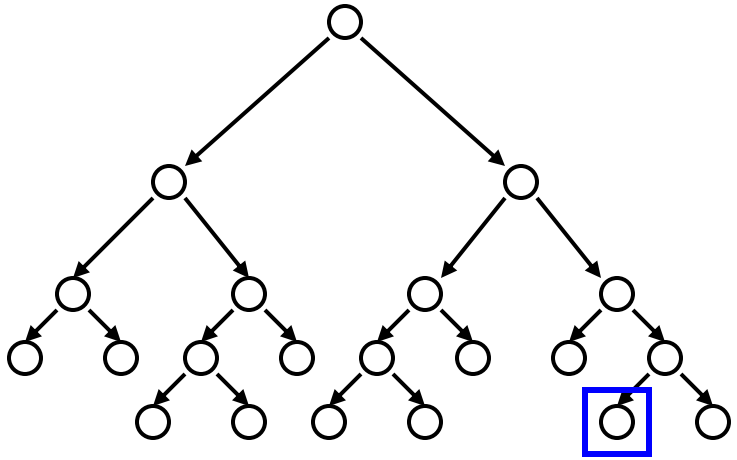




- If the hypersphere intersects the plane, there may be nearer points on the other side of the plane.
- Then we can backtrack and try the other branch at each node visited, looking for closer points and following the same recursive process as the entire search.

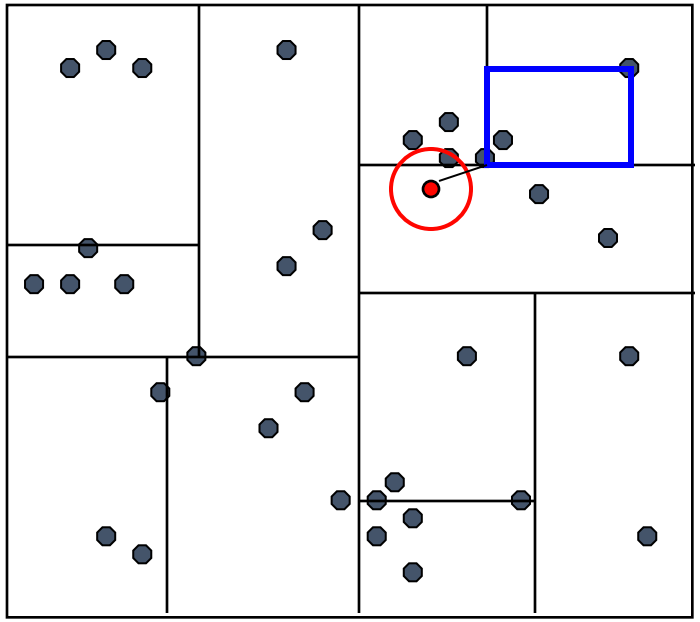
Nearest Neighbor with KD Trees

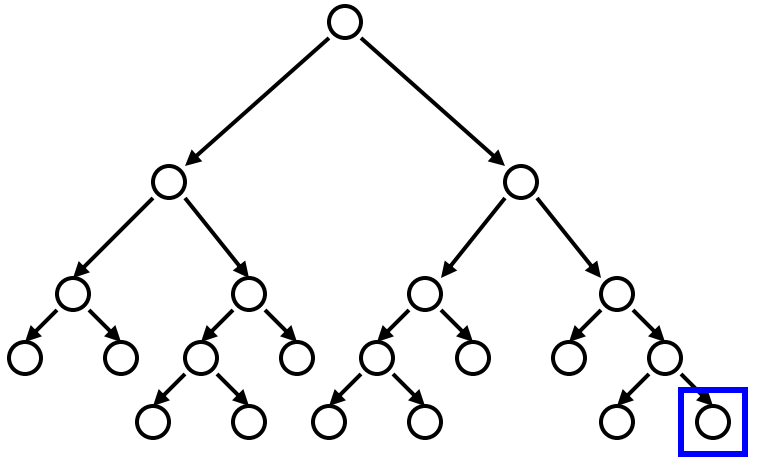




- Each time a new closest node is found, we can update the distance bounds.

Nearest Neighbor with KD Trees

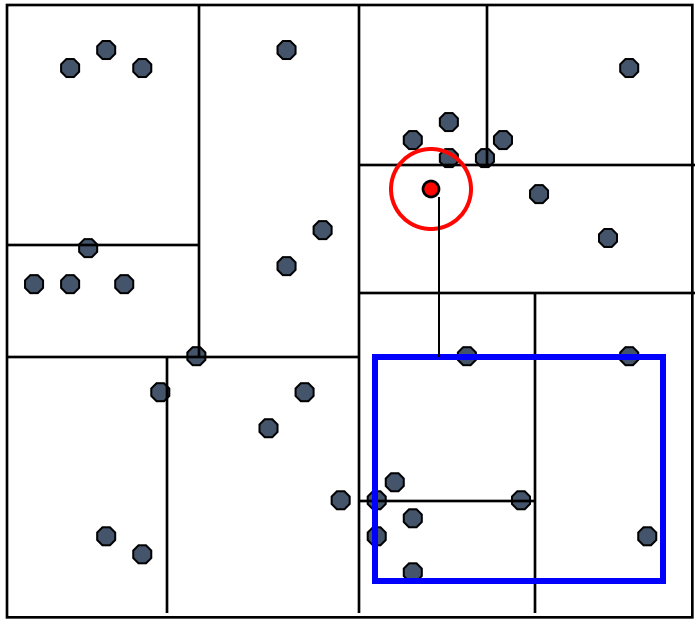


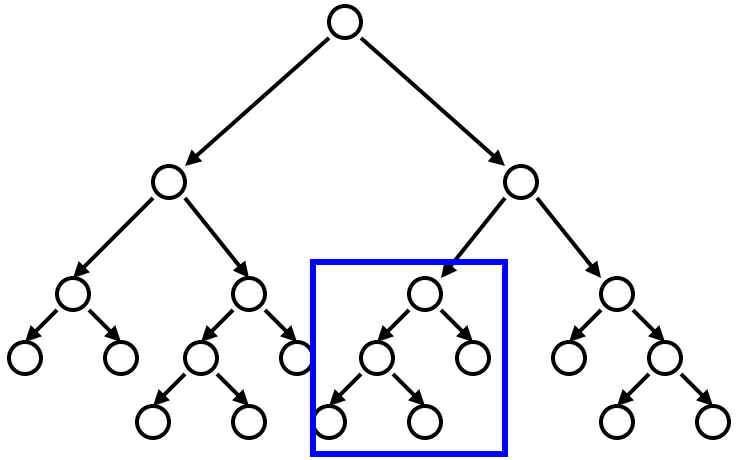


- Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

https://www.baeldung.com/cs/k-d-trees#:~:text=Nearest%20Neighbour%20Search,points%20in%20each%20leaf%20node.

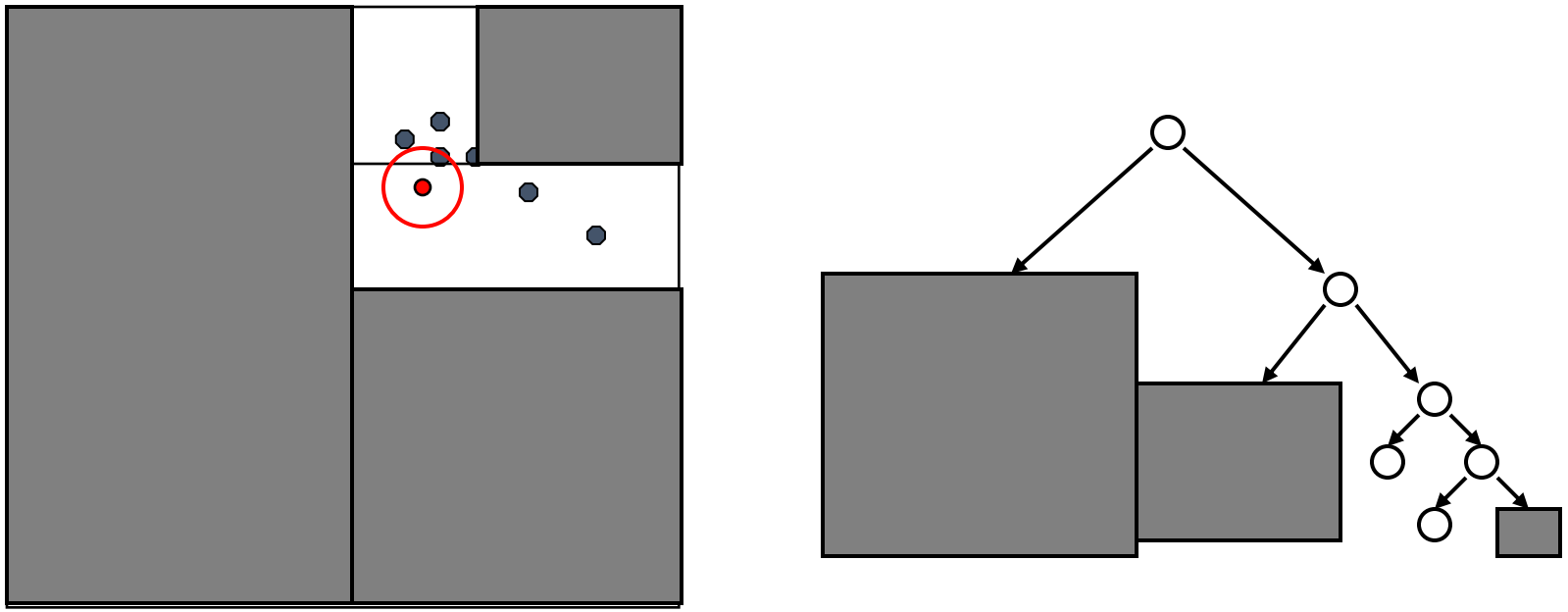
Nearest Neighbor with KD Trees





- Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Nearest Neighbor with KD Trees



- Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

K-Nearest Neighbor Search

- The algorithm can provide the *k*-Nearest Neighbors to a point by maintaining k current bests instead of just one.

- Branches are only eliminated when they can't have points closer than any of the k current bests.

d-dimensional kd-trees

- A data structure to support range queries in $\mathbb{R}^d$

- The construction algorithm is similar as in 2D
  - At the root we split the set of points into two subsets of same size by a hyperplane vertical to x_1-axis.
  - At the children of the root, the partition is based on x_2 coordinate and so on.
  - At depth d, we start all over again by partitioning on the first coordinate.
  - The recursion stops until there is only one point left, which is stored as a leaf.

- We get a binary tree
  - Size $O(N)$
  - Construction time $O(N \log N)$
  - Depth $O(\log N)$
  - K-Nearest Neighbors query time: $O\left(k + N^{1-\frac{1}{d}}\right)$ where k is the number of neighbors

Clustering

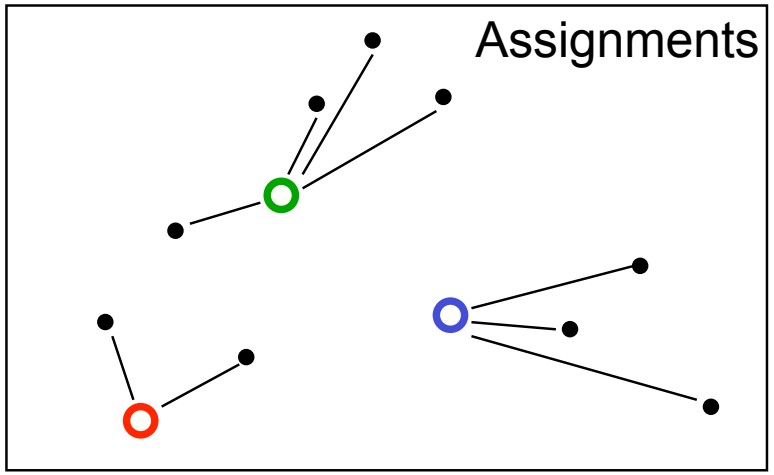
Main approaches

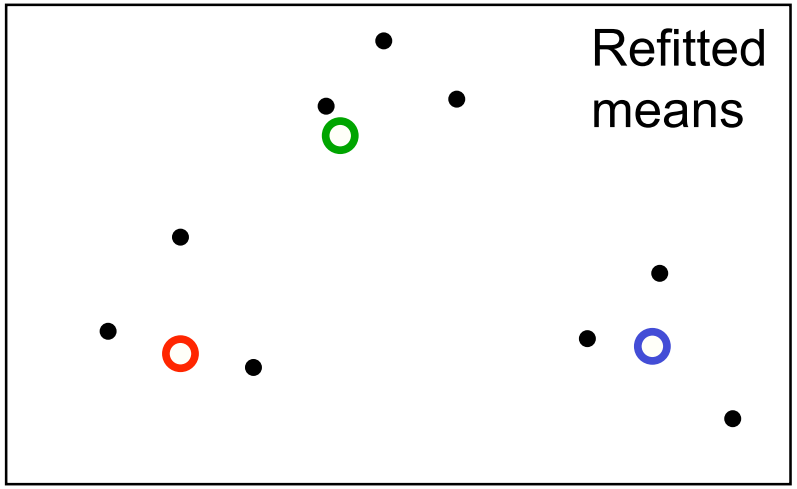
- K-means
- Hierarchical clustering
- DBSCAN
- OPTICS (not presented)
- BIRCH (not presented)
- Affinity Propagation (not presented)
- Mean Shift (not presented)
- Spectral clustering (not presented)

K-Means

K-means

- Initialization: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - Assignment step: Assign each data point to the closest cluster
  - Refitting step: Move each cluster center to the center of gravity of the data assigned to it





K-means Objective

- Find cluster centers $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K\}$ and assignments $\left\{r_k^{(i)}\right\}$ to minimize the sum of squared distances of data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ to their assigned cluster centers

$$\min_{\{\boldsymbol{m}_k\}, \left\{r_k^{(i)}\right\}} J\left(\{\boldsymbol{m}_k\}, \left\{r_k^{(i)}\right\}\right) = \min_{\{\boldsymbol{m}_k\}, \left\{r_k^{(i)}\right\}} \sum_{i=1}^{N} \sum_{k=1}^{K} r_k^{(i)} \|\boldsymbol{m}_k - \boldsymbol{x}_i\|_2^2$$

s.t. $\sum_{k=1}^{K} r_k^{(i)} = 1 \,, \forall i$, where $r_k^{(i)} \in \{0,1\}, \forall k, i$

- $r_k^{(i)} = 1$ means that $\boldsymbol{x}_i$ is assigned to cluster k (with center $\boldsymbol{m}_k$)
- The assignment and refitting steps were each doing coordinate descent on this objective.
- This means the objective improves in each iteration, so the algorithm can't diverge, get stuck in a cycle, etc.

L. Bottou and Y. Bengio. Convergence properties of the kmeans algorithm. In Advances in Neural Information Processing Systems. 1995.

Soft assignments

- Initialization : set K means centers $\{\boldsymbol{m}_1, \dots, \boldsymbol{m}_K\}$ to random values
- Repeat until convergence (until assignments do not change)
  - Assignment:

$$\hat{k}^{(i)} = \underset{k}{\operatorname{argmin}} \|\boldsymbol{m}_k - \boldsymbol{x}_i\|_2$$

$$r_k^{(i)} = 1 \Leftrightarrow \hat{k}^{(i)} = k$$

(hard assignment)

$$r_k^{(i)} = \frac{e^{-\beta \|\boldsymbol{m}_k - \boldsymbol{x}_i\|_2}}{\sum_j e^{-\beta \|\boldsymbol{m}_j - \boldsymbol{x}_i\|_2}}$$

(soft assignment)

  - Refitting:

$$\boldsymbol{m}_k = \frac{\sum_i r_k^{(i)} \boldsymbol{x}_i}{\sum_i r_k^{(i)}}$$

Mini-Batch K-means

- With the increasing size of the datasets being analyzed, K-means is losing its attractive because its constraint of needing the whole dataset in main memory.
- The mini batch K-means is faster but gives slightly different results than the normal batch K-means.
- Its main idea is to use small random batches of examples of a fixed size so they can be stored in memory.
- As the number clusters and the number of data increases, the relative saving in computational time also increases. The saving in computational time is more noticeable only when the number of clusters is very large.

D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, ACM, 2010

Mini Batch K-Means algorithm

- **Given:** number of clusters k, mini-batch size b, iterations t, data set $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, $f(\mathcal{C}, \boldsymbol{x})$ that returns the nearest cluster center $c \in \mathcal{C}$ using Euclidean distance where $\mathcal{C}$ is a given cluster
- Initialize each $c \in \mathcal{C}$ with a $\boldsymbol{x}$ picked randomly from $\mathcal{D}$
- $v \leftarrow 0$
- For $i \leftarrow 1$ to t do

  $M \leftarrow b$ examples picked randomly from $\mathcal{D}$

  For $\boldsymbol{x} \in M$ do

    $d[\boldsymbol{x}] \leftarrow f(\mathcal{C}, \boldsymbol{x})$ // Cache the center nearest to $\boldsymbol{x}$

  End

  For $\boldsymbol{x} \in M$ do

    $c \leftarrow d[\boldsymbol{x}]$ // Get cached center for this $\boldsymbol{x}$

    $v[\boldsymbol{c}] \leftarrow v[\boldsymbol{c}] + 1$ // Update per-center counts

    $\eta \leftarrow \frac{\mathbf{1}}{v[\boldsymbol{c}]}$ // Get per-center learning rate

    $\boldsymbol{c} \leftarrow (1 - \eta)\boldsymbol{c} + \eta\boldsymbol{x}$ // Take gradient step

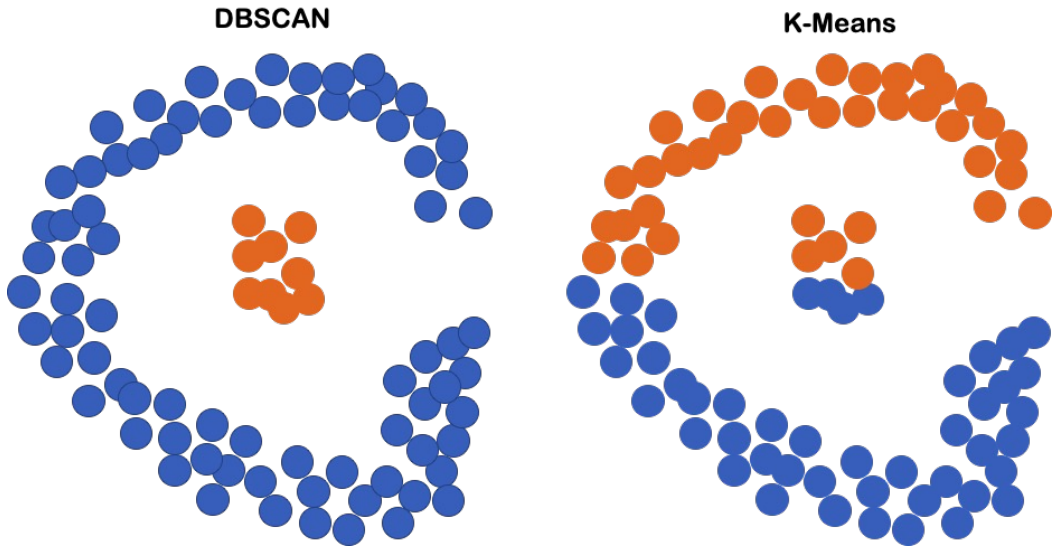
  End

End

DBSCAN

Density-based clustering

- Yet another approach to clustering is built on the assumption that clusters can be defined by identifying areas of high density in the dataset
- Produces a clustering of data points which are in dense areas, but typically leaves some data points un-labeled



DBSCAN

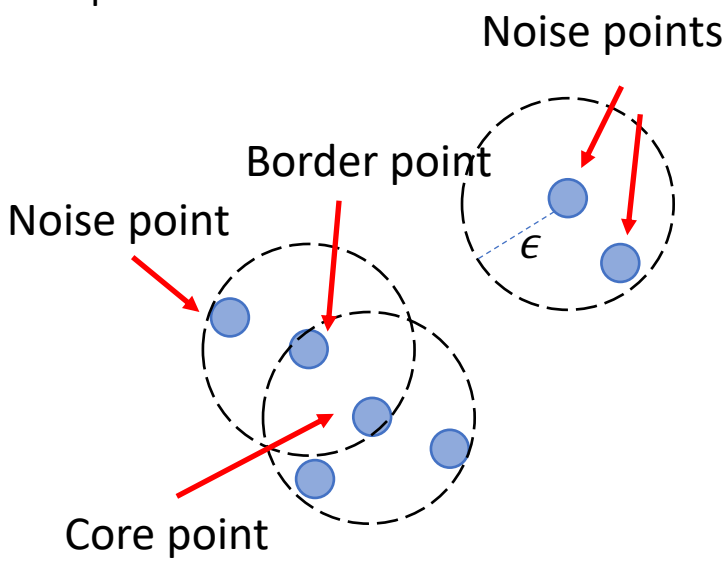
- Density-based spatial clustering of applications with noise (DBSCAN) is probably the most popular density-based clustering algorithm
- Divides the dataset into three categories of points
  - Core points: points in dense regions
  - Reachable points: points which are near a core point
  - Outliers: everything left over
- DBSCAN has two parameters ($p_{\min}$ and ϵ) which help quantify how dense a cluster needs to be
- Note: Does not require a priori determination of the number of clusters

Core points, border points and noise points

- A point $\boldsymbol{x}$ is a core point if there are at least $p_{\min}$ points in the set

$$\mathcal{N}_{\epsilon}(\boldsymbol{x}) = \{\boldsymbol{y}: \|\boldsymbol{y} - \boldsymbol{x}\|_2 \leq \epsilon\}$$

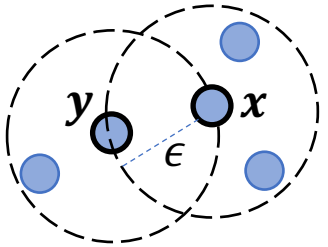
- A border point has fewer than $p_{\min}$ within ϵ, but is in the neighborhood of a core point.
- A noise point or an outlier is any point that is not a core point nor a border point.



$p_{\min} = 4$

Reachability

- A point $\boldsymbol{y}$ is directly reachable from $\boldsymbol{x}$ if 1) point $\boldsymbol{x}$ is a core point and 2) point $\boldsymbol{y}$ is within distance ϵ from $\boldsymbol{x}$.
  - In the illustration, $\boldsymbol{y}$ is directly reachable from $\boldsymbol{x}$ but $\boldsymbol{x}$ is not directly reachable from $\boldsymbol{y}$



$p_{\min} = 4$

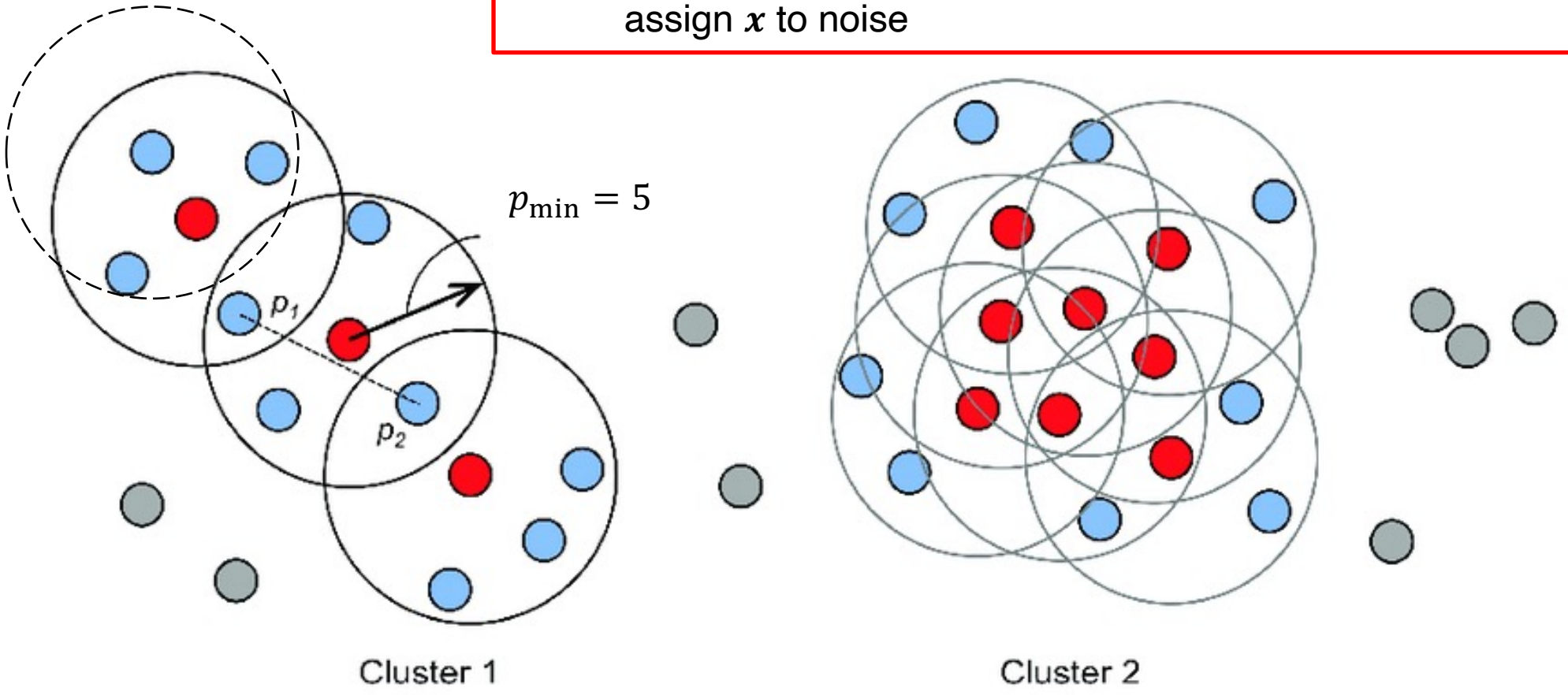
- A point $\boldsymbol{y}$ is reachable from $\boldsymbol{x}$ if there is a sequence of points satisfying $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_\ell$
  - $\boldsymbol{p}_1 = \boldsymbol{x}$ and $\boldsymbol{p}_\ell = \boldsymbol{y}$
  - all $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{\ell-1}$ must be core points
  - each $\boldsymbol{p}_k$ must **be directly reachable from** $\boldsymbol{p}_{k-1}$

- If $\boldsymbol{x}$ is a core point, then it forms a cluster together with all points (core and non-core) that are reachable from $\boldsymbol{x}$

Illustration

Algorithm in brief:

for each point $\boldsymbol{x}$ do
  if $\boldsymbol{x}$ is not yet classified then
    if $\boldsymbol{x}$ is a core-point then
      collect all points reachable from $\boldsymbol{x}$. and assign them to a new cluster.
    else
      assign $\boldsymbol{x}$ to noise



A cluster consists of core points (red) and border points (blue).

Remarks

- Clustering can be computed in a simple iterative fashion
  - Find a core point
  - Add its reachable points
  - Iterate until all points are either clustered or labeled as outliers
- Algorithm can lead to slightly different clusterings depending on the order in which the points are processed
- « Border points » can sometimes be assigned to multiple clusters, depending on which one is grown first
- Non-assigned points are declared as outliers
- Can struggle with clusters of varying density – see extensions, e.g., OPTICS