Les données numériques: toutes des nombres! Algorithme kNN

Diane Lingrand

SI3

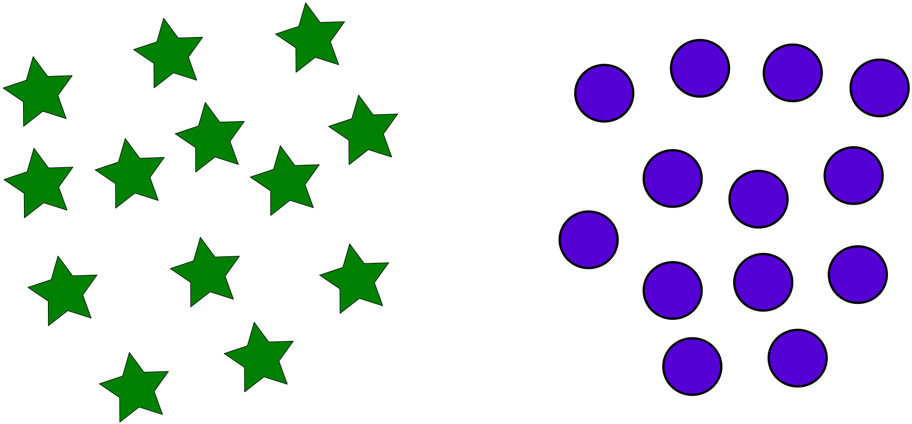
2022 - 2023

Outline

Outline

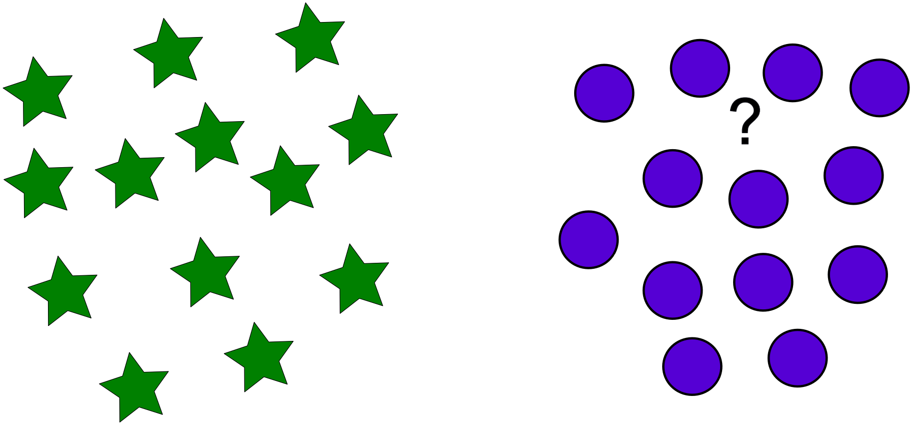
Context of classification

- training dataset
  - data $x_i \in \mathbb{R}^d$ with $0 \leq i \leq n$
  - labels or class index $y_i \in \mathbb{R}$ with $0 \leq i \leq n$
    - binary classification : positives and negatives
    - multi-class classification
- the goal is to be able to guess the class given the data
- metrics are computed on a test dataset

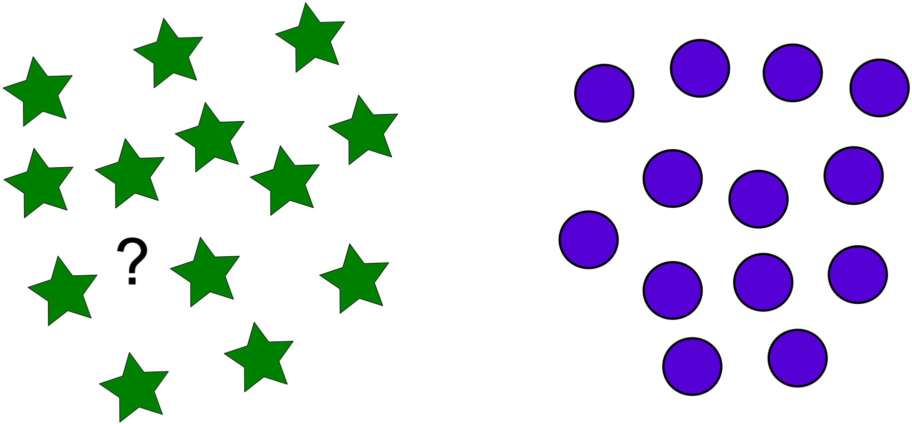


We observe that nearby objects belong to the same class.

Introduction : 1NN



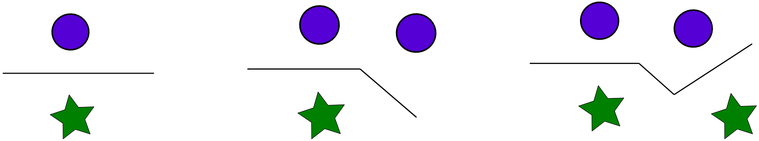
This new object will be assign to the class of purple rounds.



This new object will be assign to the class of green stars.

1NN

- intuition : nearest point
- Voronoi tesselation
  - points at same distance from two different points



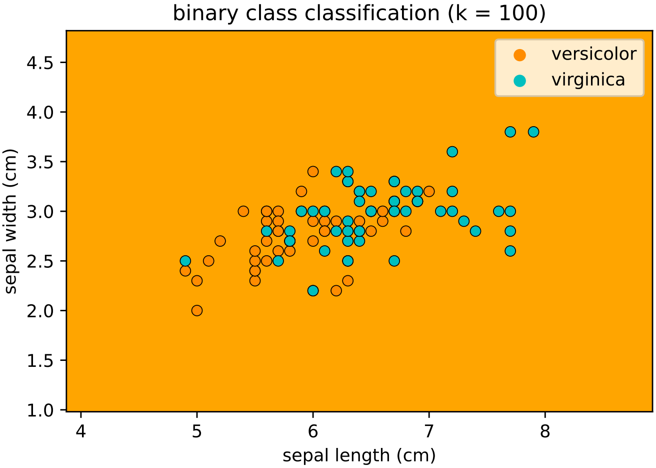
- complex decision boundary

kNN

- 1NN is very sensitive to outliers
  - idea : use more than one nearest neighbor to make decision
    - use k nearest neighbors
- algorithm
  - compute distance to every training sample x_i
  - select k closest instances
  - output the class that is most frequent

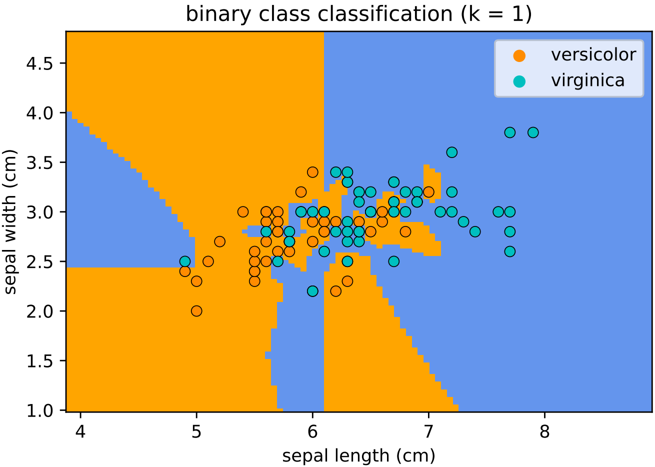
How to choose value k ?

- large value : everything classified as the most probable class



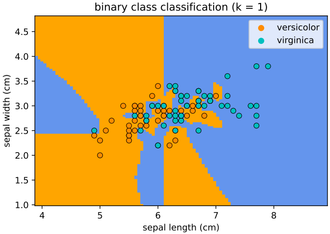
How to choose value k ?

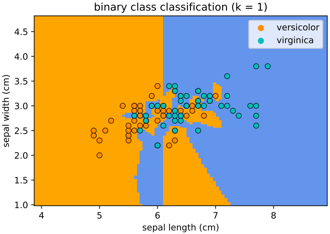
- large value : everything classified as the most probable class
- small value : highly variable, unstable decision boundaries



How to choose value k ?

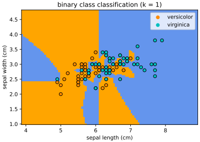
- large value : everything classified as the most probable class
- small value : highly variable, unstable decision boundaries

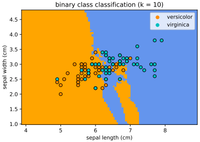


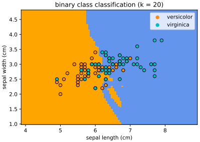


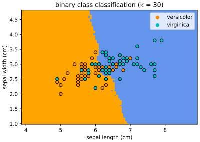
- small changes in the training set imply large changes in classification

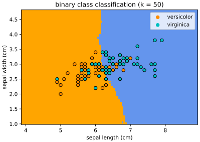
- large value : everything classified as the most probable class
- small value : highly variable, unstable decision boundaries
  - small changes in the training set imply large changes in classification
- affects smoothness of the boundary











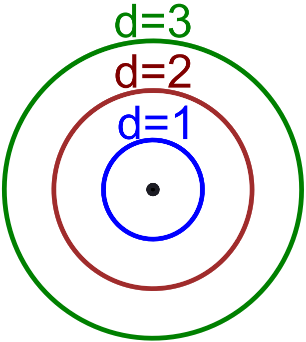
How to choose value k ?

- large value : everything classified as the most probable class
- small value : highly variable, unstable decision boundaries
  - small changes in the training set imply large changes in classification
- affects smoothness of the boundary
- select value of k using validation dataset
  - it is critical to use the validation dataset
    - if we use the training dataset, 1NN is the best (the nearest example is itself and has the correct class/value)
- we thus need training / validation / test dataset

- Euclidean :

$$d_2(i,j) = \sqrt{\sum_{k=1}^{m} |x_{ik} - x_{jk}|^2}$$

  - nice properties : symmetric, isotropic, ..

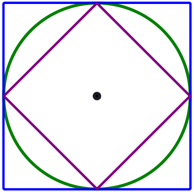


Distance functions : key component

- Minkowski (generalisation)

$$d_q(i,j) = \sqrt[q]{\sum_{k=1}^{m} |x_{ik} - x_{jk}|^q}$$

  - Manhattan ($q = 1$) : $d_1(i,j) = \sum_{k=1}^{m} |x_{ik} - x_{jk}|$
  - Euclidean ($q = 2$) : $d_2(i,j) = \sqrt{\sum_{k=1}^{m} |x_{ik} - x_{jk}|^2}$
  - max distance ($q = \infty$) : $d_\infty(i,j) = \max_k |x_{ik} - x_{jk}|$



Indetermination

- no majority class : equal number of neighbours for at least 2 classes
  - in case of binary classification : use odd k
  - random between the equal classes
  - prior : pick class with greater prior
  - nearest : let 1-NN decide

Variations

- Parzen windows
  - instead of choosing the number of neighbours, choose the size of the neighborhood
  - or use all the points with a decay function for distances (close to kernel that converts distances to numbers)

kNN pros and cons

- assumption : nearby (defined by distance fn) regions of space concern the same class
- almost no learning (except k)
- sensitive to class outliers
- sensitive to lots of irrelevant attributes
- computationnally expensive :
  - space : need to store all training samples
  - time : need to compute distances to all training samples
  - expensive at testing, no training time (which is bad)

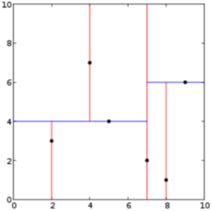
Outline

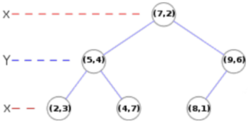
Reducing time

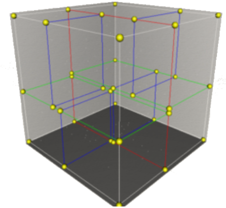
- time : $O(nd)$ where n is the nb of samples and d the number of features
- reduce d (feature selection)
- reduce n :
  - idea : quickly find $m << n$ potential near neighbors
  - compare only to those and pick k nearest neighbors $O(md)$ time
  - Kd tree : low dim, real valued data
    - $O(dlog_2(n))$, only if $d << n$, can miss neighbors
  - others methods we won't study in this class :
    - LSH (locality-sensitive hashing) for high dimensions
    - inverted lists
    - ball trees
    - ...

Kd-trees (Friedman etal 1977)

- k corresponds to the dimension of the data (d in this course)
- kd-trees are binary trees
- designed to handle spatial data in a simple way

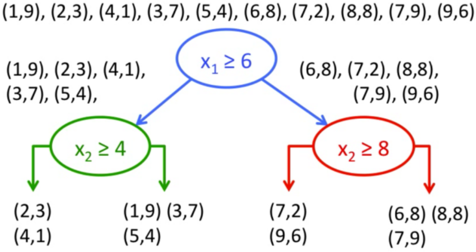


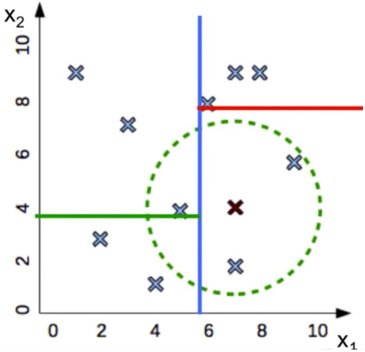




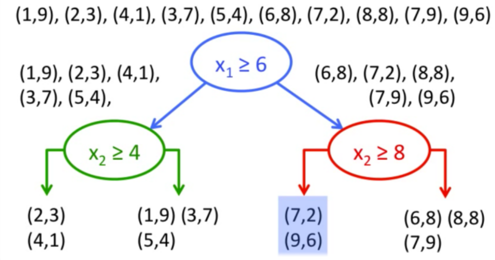
Kd-tree : construction

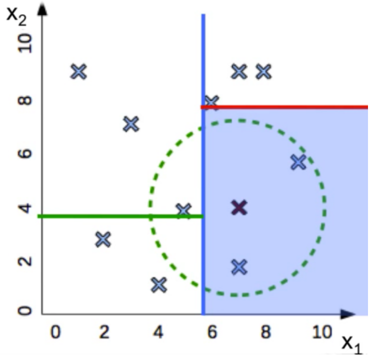
- Repeat :
  - Pick random dimension
  - Find median element
  - Split data





- Example : find NNs for new point (7,4)
  - Find region containing (7,4)
  - Find kNN amoung all the points in this region





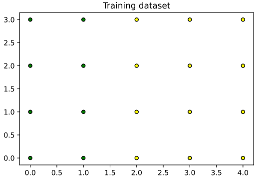
Outline

Code

- Many implementations are available :
  - scikit-learn library
  - scipy library
- But today, you will write YOUR implementation, in `python`
  - brute force (original kNN), Euclidean distance
  - kd-tree

Brute force implementation

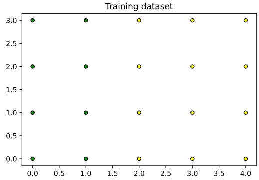
- start with a simple dataset



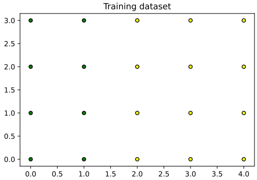
  - k in k-NN is a variable
  - start be searching for the class of one single new point

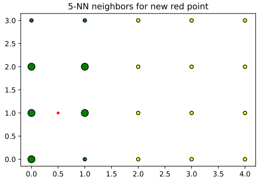
Brute force implementation

- start with a simple dataset

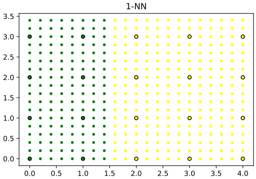


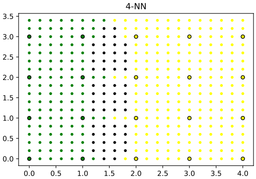
- k in k-NN is a variable
- start be searching for the class of one single new point
  - list of k neighbors
  - vote for majority class
  - decide what to do in case of ambiguity

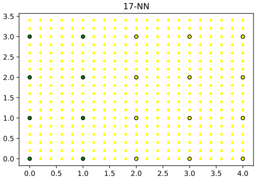




- start with a simple dataset
  - *k* in k-NN is a variable
  - start be searching for the class of one single new point
  - **then predit class for a set of new points**
    - varying *k*

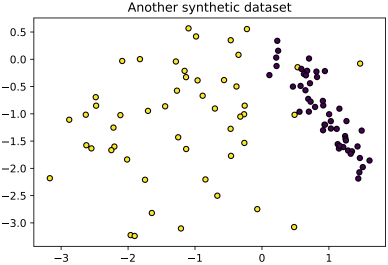


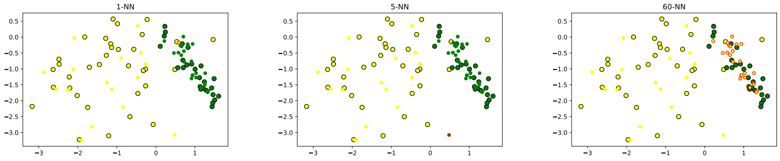




Brute force implementation (3)

- test on a more complex synthetic dataset





- Classification of 3 classes 'cat', 'dog' and 'bird' from the Speech Commands Dataset
  - 200 sounds per class for training, 200 other sounds per class for testing