Supervised Learning: The classification problem

B. Manca

A.a. 2025/26

The Classification Problem

- In supervised learning we consider a set of labeled points $\mathcal{X} \times \mathcal{Y}$ and we want to determine a criterion that distinguishes points with different labels and apply it also to point not included in $\mathcal{X}$
- If the set $\mathcal{Y}$ is discrete and finite we talk about Classification
  - If $|\mathcal{Y}| = 2$ we talk about Binary Classification
  - If $|\mathcal{Y}| > 2$ we talk about Multiclass Classification

Random classifier

- Given the labeled points $\mathcal{X} \times \mathcal{Y}$, the simplest classification criterion we can use is to assign a label randomly
- For every $x \in \mathcal{X}$ generate a random number $r \in [0, 1]$ and assign one label if $r < 0.5$ and the other one otherwise
- If we want we can introduce an hyper-parameter $t \in [0, 1]$ and use it as a threshold for the classification criterion:

$$\hat{y} = \begin{cases} -1 & \text{if } r \leq t \\ +1 & \text{otherwise} \end{cases} \tag{1}$$

- The parameter t is not determined by the algorithm, and therefore one can ask how to choose the best value of t.

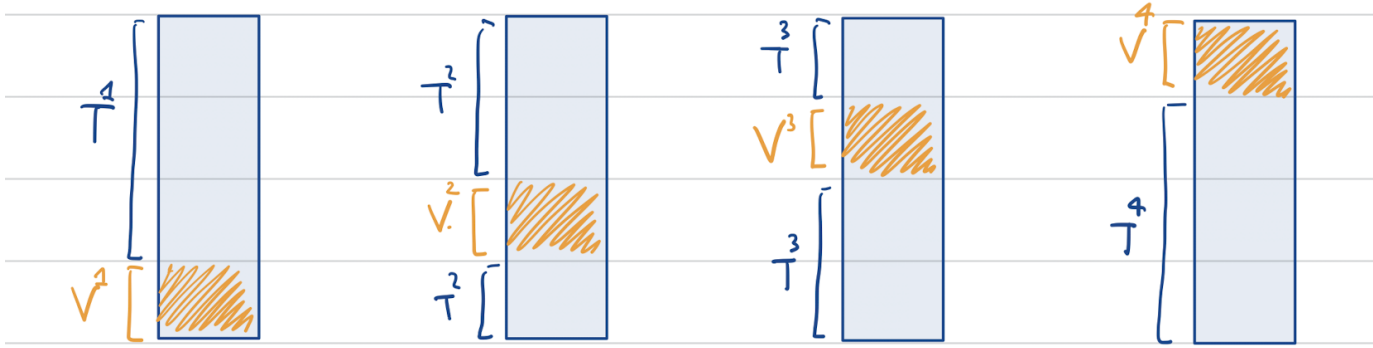
Estimate the best hyper-parameters

- In the random classifier, we have one hyper-parameter t which determine how the classifier behaves
- With different values of the hyper-parameter we obtain a different classifier and we want to determine the best value that correspond to the highest classification score
- Assume to have a classifier with two hyper-parameters $C_1 \in \{C_1^1, \ldots, C_1^{m_1}\}, C_2 \in \{C_2^1, \ldots, C_2^{m_2}\}$
- To estimate the best values for C_1 and C_2, we use a grid-search, i.e., we consider the grid with all the possible combination of C_1, C_2:

C_1 \ C_2	C_2^1	$\cdots$	$C_2^{m_2}$
C_1^1	(C_1^1, C_2^1)	$\cdots$	$(C_1^1, C_2^{m_2})$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
C_1^m	$(C_1^{m_1}, C_2^1)$	$\cdots$	$(C_1^{m_1}, C_2^{m_2})$

Gridsearch with k-fold cross validation

- Given the hyper-parameters grid and a classification score $\mathcal{S}(C_1, C_2, X)$, in order to determine the maximum value of $\mathcal{S}$ we consider a k-fold cross validation:
- We split the training sample $\mathcal{X}_{train}$ in k disjoint subsets and we use k-1 of them to train the model and 1 to validate it:



- For every pair (C_1^i, C_2^j) we train the model on T^ℓ and we compute the score $\mathcal{S}_\ell^{i,j} := \mathcal{S}(C_1^i, C_2^j, V^\ell)$ on V^ℓ
- The final classification score for the pair (C_1^i, C_2^j) is given by

$$\mathcal{S}^{i,j} = \frac{1}{k} \sum_k \mathcal{S}_\ell^{i,j}$$

Gridsearch with k-fold cross validation

- Once we have computed the value $\mathcal{S}^{i,j}$ for every i and j we consider

$$i^*, j^* = \arg\max_{i,j} \mathcal{S}^{i,j}$$

- The best hyper-parameters of the classifier are $C_1^{i^*}$ e $C_2^{j^*}$ and can be used to train the model on $\mathcal{X}_{train}$

Implementation of the Random Classifier

- Implement a python class for the random classifier containing three modules:
  - __init__: initialization of the hyper-parameters;
  - fit: training phase of the model;
  - predict: prediction phase of the model.
- Generate a simple dataset using the sklearn function make_classification
- split the dataset into training and test sample and use the train sample for the training phase and the test sample to validate the model
- Print the classification report (sklearn function) to observe the scores
- Using the gridsearch function of sklearn estimate the best value for the hyper-parameter t of the random classifier

Homework

- Implement a custom version of the k-fold cross validation gridsearch for the random classifier
- Compare the results with the ones obtained using the sklearn gridsearch function

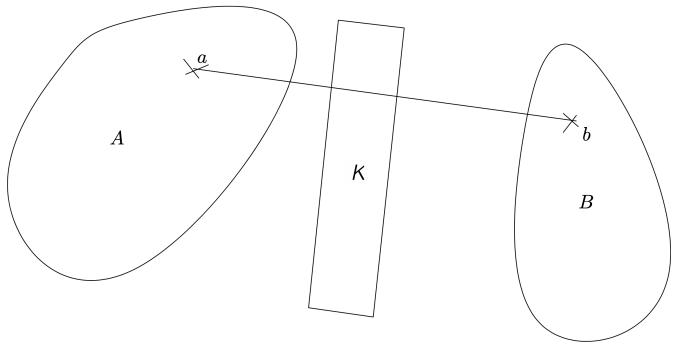
From separation to classification (binary case)

- Assuming that we are representing $\mathcal{X}$ as a subset of $\mathbb{R}^n$, we can determine a classification criterion using linear algebra
- One idea is to find a criterion that separates the ambient space $\mathbb{R}^n$ in two areas, each one corresponding to one of the classes:

Definition

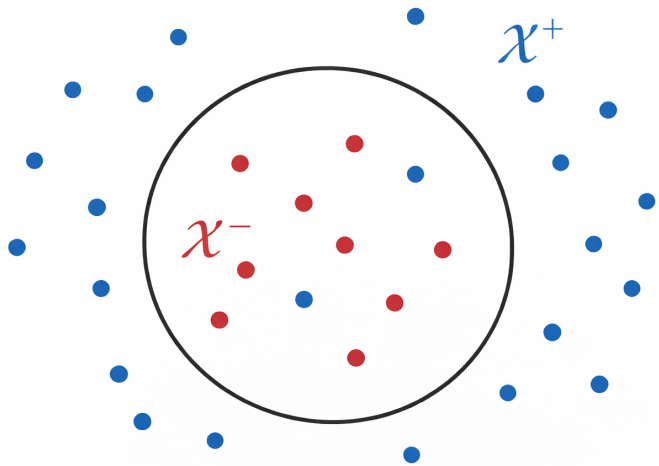
Two convex sets A and B are said to be separated by a third set K if

$$[a, b] \cap K \neq \emptyset \quad \forall a \in A, b \in B$$



Spherical separation

- Let $\mathcal{X}^+$ and $\mathcal{X}^-$ be the two classes we want to separate.
- We want to find a sphere that contains the class $\mathcal{X}^-$ and possibly contains as few points of $\mathcal{X}^+$ as possible.



- We focus only on $\mathcal{X}^-$ and we try to find the smallest sphere that contains its points

Spherical separation

- First step: determine which class is staying inside the sphere:
  - choose randomly;
  - choose the class A whose elements have minimum average norm
- Compute the center and the radius of the sphere:
  - The center is given by the barycenter of the points in the inner class

$$c = \frac{1}{|A|} \sum_{i=1}^{|A|} a_i$$

  - the radius is given by the maximum distance from the center to a point in A:

$$\rho = \max_{a_i} \|a_i - c\|_2$$

- Given any point $z \in \mathbb{R}^n$ we assign a label to it according to the function

$$\hat{y} = \begin{cases} -1 & \text{if } \|z - c\|_2 \leq \rho \\ +1 & \text{if } \|z - c\|_2 \geq \rho \end{cases}$$

k-nearest neighbor classifier

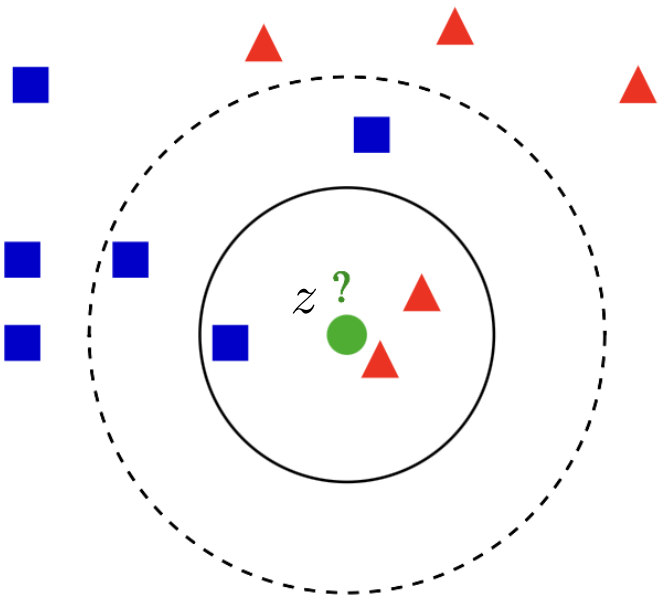
- Another simple binary classifier we can consider is based on the k-nearest neighbors of a given point:

Definition

Fix $k \in \mathbb{N}$, given a set $\mathcal{X} \subset \mathbb{R}^n$, we denote with $\mathcal{U}_k(x) \subset \mathcal{X}$ the set of points in $\mathcal{X}$ that contains the k points closest to $x \in \mathcal{X}$ according to a specified distance.

- Let $\mathcal{X} \times \mathcal{Y}$ be a set of labeled points and $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ a distance function
- In the training phase we save the positions of the elements in $\mathcal{X}$
- In the prediction phase, given a point $z \in \mathcal{X}$ (or $\mathbb{R}^n$) we consider the k elements in $\mathcal{X}$ closest to z and we assign it the label that is more represented by those k elements

k-nearest neighbor classifier: example



- if $k = 3$ we assign to z the red label
- if $k = 5$ we assign to z the blue label

Metrics to evaluate a classifier

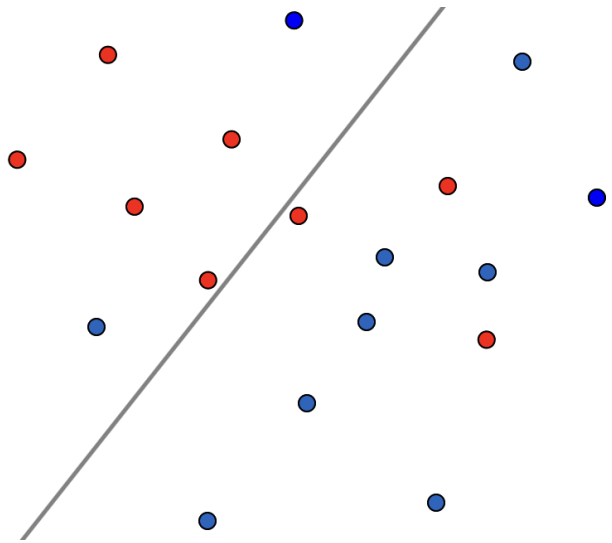
- There exist several metrics (or scores) that can help us to evaluate the quality of the solution obtained by a classifier during the prediction phase
- Consider the following quantities:

$$
\begin{aligned}
TP &= \text{ elements with positive class labeled positively by the classifier} \\
TN &= \text{ elements with negative class labeled negatively by the classifier} \\
FP &= \text{ elements with negative class labeled positively by the classifier} \\
FN &= \text{ elements with positive class labeled negatively by the classifier}
\end{aligned}
$$

Metrics to evaluate a classifier

- Accuracy: percentage of elements well classified

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

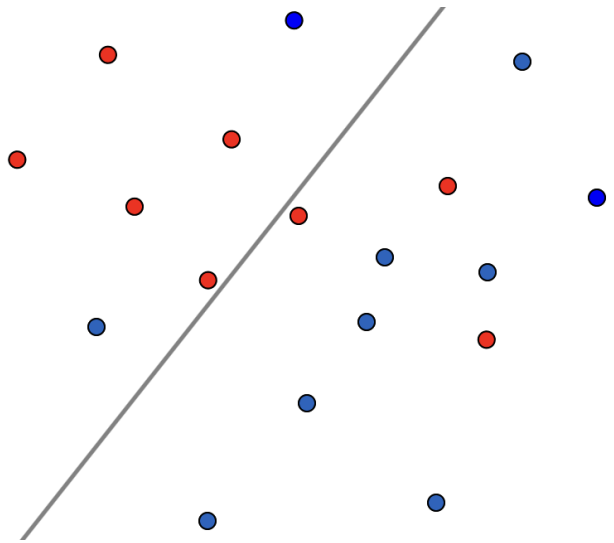


\# elements = 18
\# well-classified = 13
\# misclassified = 5
Accuracy = 13/18 =0.722

Metrics to evaluate a classifier

- Precision: determines the classifier's ability of not labeling as positive an element that is negative
- With positive element we now denote an element whose true label is the label of the class we are computing the score

$$\text{Precision} = \frac{TP}{TP + FP}$$

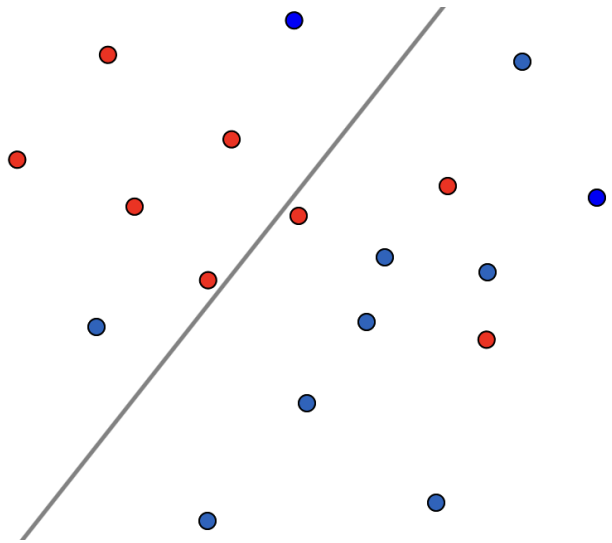


$\text{TP} = 5$
$\text{FP} = 2$
$\text{TP} = 8$
$\text{FP} = 3$
$\text{Precision(Red)} = \frac{5}{7}$
$\text{Precision(Blue)} = \frac{8}{11}$

Metrics to evaluate a classifier

- Recall: determines the classifier's ability to find all the positive elements

$$\text{Recall} = \frac{TP}{TP + FN}$$

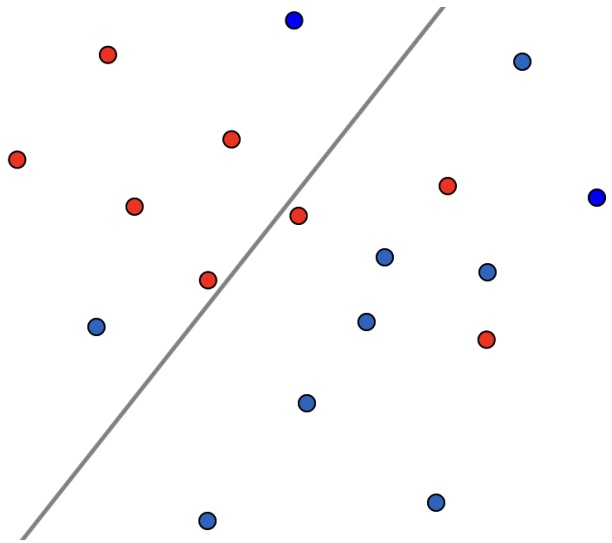


$\text{TP} = 5$
$\text{FN} = 3$
$\text{TP} = 8$
$\text{FN} = 2$
$\text{Recall(Red)} = \frac{5}{8}$
$\text{Recall(Blue)} = \frac{8}{10}$

Metrics to evaluate a classifier

- F_1-score: harmonic average of precision and recall

$$F_1\text{-score} = 2\frac{Precision \cdot Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$



$\text{Precision} = \frac{5}{7}$

$\text{Recall} = \frac{5}{8}$

$F_1\text{-score} = \frac{2}{3}$

$\text{Precision} = \frac{8}{11}$

$\text{Recall} = \frac{8}{10}$

$F_1\text{-score} = \frac{16}{21}$

Homework

- Implement the k-NN classifier using the sklearn template for custom classifiers
- Use as hyper-parameter the kind of distance to use to compute the closest points:
  - Euclidean norm: $\|x\|_2 = \sqrt{\sum_i x_i^2}$
  - Manhattan norm: $\|x\|_1 = \sum_i |x_i|$
  - Infinity norm: $\max_i |x_i|$
- Compare the results by fixing the norm to one of the three value
- Apply a k-fold cross validation gridsearch to determine the best norm