

# Machine/Statistical Learning

## Model based and model free examples

### k-Nearest Neighbors

SICOM, M2 Sigma

2022-23

## Model based vs Model free methods

Prediction rule :

$$Y = \hat{f}(X)$$

### Model based approach

Learning of the joint or conditional distribution of the data  $X, Y$ , then deduce  $\hat{f}$

- 👉 Linear models (regression), Discriminant analysis, Naïve Bayes,...
- + interpretability : give more insights (e.g. relations between input and output variables, ...) than just  $\hat{f}$
- model may be sometimes too restrictive, badly adapted to the data

### Model free approach




Direct learning of  $\hat{f}$

- 👉 k nearest neighbors (k-NN), SVM, Random Forests, (deep) Neural Nets...
- + may be more flexible than a model for real-world data
- interpretability

## Nearest Neighbors algorithm

- ▶ algorithm which is conceptually among the simplest of all machine learning algorithms
- ▶ **model-free** method which does not rely on a fixed model : the prediction function  $\hat{Y}(x) \equiv \hat{f}(X)$  is directly estimated from the training set : does not require any parametric assumption (e.g., Gaussian distribution) on the distribution of  $X$  and  $Y$

### Supplementary materials

-  Coursera MOOC short (5mn) and simple video <https://fr.coursera.org/lecture/big-data-machine-learning/k-nearest-neighbors-MamQ9>
-  Wikipedia page (more involved)  
[https://en.wikipedia.org/wiki/K-nearest\\_neighbor\\_algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)
-  Scikit-learn documentation (with examples)  
<https://scikit-learn.org/stable/modules/neighbors.html#neighbors>

## $k$ Nearest Neighbors ( $k$ -NN) for regression

Regression problem :  $Y \in \mathbb{R}$

- ▶ The prediction value  $\hat{Y}(x)$  is directly defined for an input  $X = x \in \mathbb{R}^p$  as the **average of the values of its  $k$  nearest neighbors** in the training set.
- ▶ More formally,

$$\hat{Y}(x) = \frac{1}{k} \sum_{X_i \in N_k(x)} Y_i,$$

where  $N_k(x)$  is the neighborhood of  $x$  defined by the  $k$  closest inputs  $X_i$  in the training set  $\{(X_i, Y_i)\}_{i=1 \dots n}$ .

- 👉 the number of neighbors  $k$  is an *hyperparameter* of the algorithm : its value must be fixed (positive integer) prior to apply the algorithm. But of course, the best choice of  $k$  depends upon the data (see next slides).

## $k$ Nearest Neighbors ( $k$ -NN) for classification

### Binary classification problem

For a binary classification problem  $Y \in \{-1, +1\}$ , the classification rule can be derived, for  $X = x$ , as

$$f(x) = \begin{cases} +1 & \text{if } \hat{Y}(x) > 0, \\ -1 & \text{otherwise} \end{cases}$$

where  $\hat{Y}(x) = \frac{1}{k} \sum_{X_i \in N_k(x)} Y_i$  is the average of the binary labels of the  $k$  nearest neighbors of the testing point  $X = x$ .

### Multiclass problem with $k$ -NN

The binary classification problem can be directly extended for an arbitrary number of class  $K$  :

$f(x) \equiv$  **majority vote** among the  $k$  closest neighbors of the testing point  $x$ ,  
 $\equiv$  assignment to the **most common class** among the  $k$  nearest neighbors

## Limitations of $k$ -Nearest-Neighbors ( $k$ -NN) algorithm

### Curse of dimensionality

$$\hat{Y}(x) = \text{Average} \{Y_i | X_i \in N_k(x)\} \approx E[Y | X = x]$$

which is the optimal prediction rule for the mean squared error  $E[(Y - \hat{Y}(x))^2]$ .

But two approximations problematic in high dimension, i.e. when the number  $p$  of variables is large :

- ▶ Expectation  $\approx$  Average,
- ▶ Conditioning at a point  $\approx$  conditioning on a (large) neighborhood

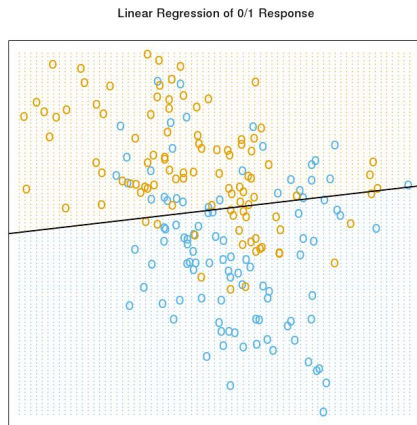
### High computational cost for huge data

$k$ -NN is as **Lazy algorithm** : nothing is done during the training step (no computation). All computation is deferred until prediction function evaluation (require to compute the distances from the test sample to the training samples to find nearest neighbors).

- ▶ very expensive to find the nearest neighbors for large datasets (large number  $n$  of samples) with high dimensional features (e.g.  $p \geq 10$ ).

## Academic example of binary classification

- ▶ Binary output variables :  $Y_i \in \{-1, +1\}$ ,
- ▶ Bivariate Input variables  $X_i \in \mathbb{R}^2$  (i.e.  $p = 2$ ), for  $i = 1, \dots, n$



Example of a binary classification problem in  $\mathbb{R}^2$ . The 2 classes are coded as a binary variable : **ORANGE**=1, **BLUE**=-1

## Model based classification : simple linear model

We seek a prediction model based on the linear regression of the outputs  $Y \in \{-1, +1\}$  :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon,$$

where  $\beta = (\beta_0, \beta_1, \beta_2)^T$  is a 3D unknown parameter vector

Learning problem  $\Leftrightarrow$  Estimation of  $\beta$

*Least Squares Estimator*  $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2)^T$  : minimize the training error rate (quadratic cost sense)

$$RSS(\beta) = \sum_{i=1}^N (Y_i - \beta_0 - \beta_1 X_{i,1} - \beta_2 X_{i,2})^2$$

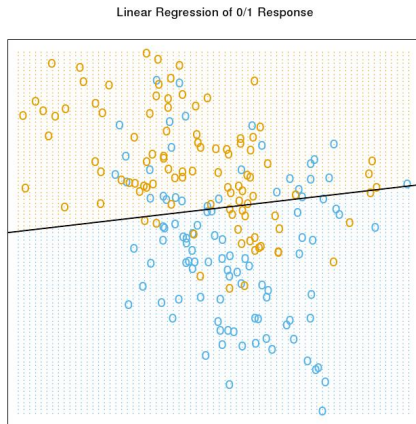
Classification rule based on least squares regression

$$\hat{f}(X) = \begin{cases} 1 & \text{if } \hat{Y} = \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 \geq 0, \\ -1 & \text{otherwise} \end{cases}$$



## Simple linear model classification

Black solid line is the decision boundary (least-squares classification)



*All the point above, resp. below, the black solid line are predicted as **ORANGE**=+1 class, resp. **BLUE**=-1 class*

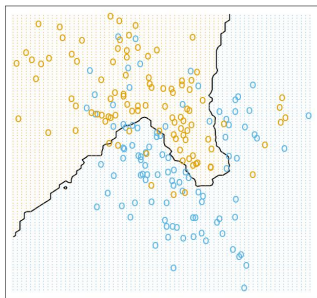
- Is there a room for improvement of the classification performance?

## K Nearest-Neighbors

$k$ -NN : complexity parameter  $k$

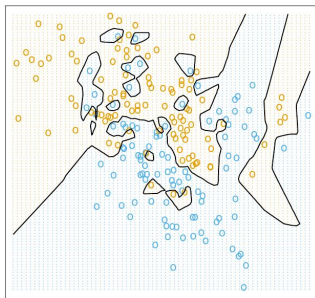
The effective number of parameters expresses as  $N_{\text{eff}} = \frac{n}{k}$ , where  $n$  is the size of the training sample (rationale : there is an order of  $n/k$  different neighborhoods and each is governed by a single parameter, the value of the majority label)

15-Nearest Neighbor Classifier



$k = 15, N_{\text{eff}} \approx 13$

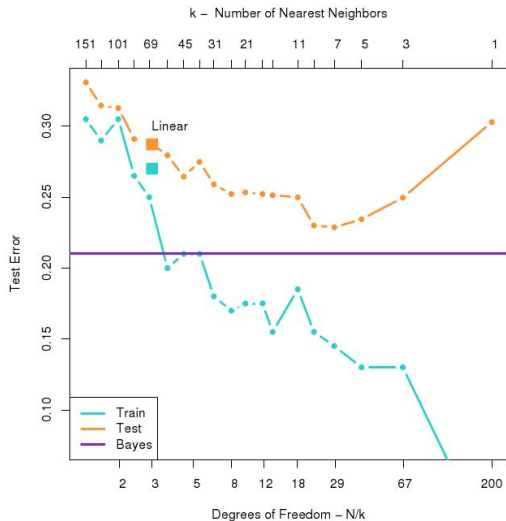
1-Nearest Neighbor Classifier



$k = 1, N_{\text{eff}} \approx 200$

- ▶  $k = 1 \rightarrow$  training error is always 0! Explain why.
- ▶ Question : in your opinion, which one gives the best classification rule ? (see next slide)

## Model Selection : under/over-fitting tradeoff



Train and test error (misclassification rate) vs the flexibility of the k-NN algorithm

- ▶ when  $k = 1$ , the model becomes too flexible  
← over-fitting
- ▶ when  $k$  is large, the model becomes too simple  
← under-fitting
- ▶  $k = 7$  is the optimal choice for this dataset
- ▶ square markers are the errors for the simple linear model (solid back line decision boundary)

📖 Notebooks [N1\\_blobs\\_knn.ipynb](#) and [N2\\_iris\\_knn.ipynb](#)

## Conclusions on Nearest Neighbors approach

### Recap on k-NN

- ▶ algorithm which is conceptually among the simplest of all machine learning algorithms
- ▶ model-free method which does not rely on a fixed model
- ▶ badly behaved procedure in high dimension (i.e. when the number  $p$  of features is large) : dimension reduction, e.g. *principal component analysis*, is usually performed prior to k-NN algorithm in order to avoid curse of dimensionality and
- ▶ highly computational for large data sets : data reduction is usually performed prior to k-NN algorithm to reduce computational complexity of the classification rule
- ▶ in general quite performant on small, or medium sized, real-world datasets