Machine Learning Programming

Classification + K-NN

Lecture : Prof. Aude Billard (aude.billard@epfl.ch)

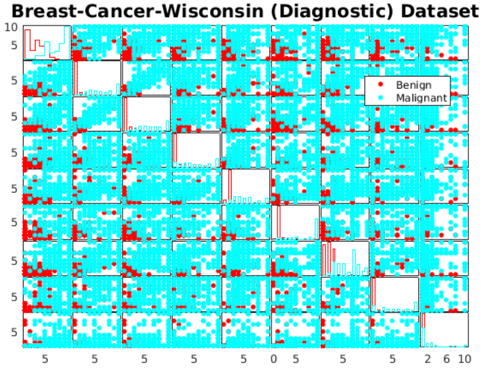
Teaching Assistants :
Nadia Figueroa, Laila El Hamamsy, Hala Khodr and Lukas Huber

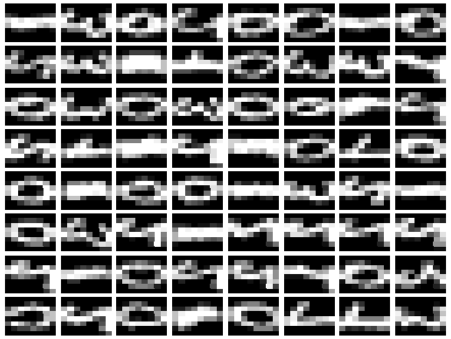
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

What have we covered so far?

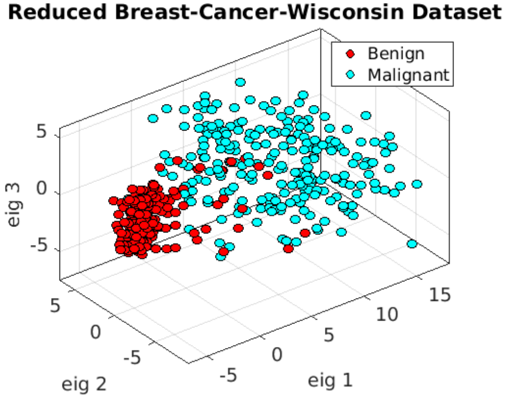
TP1 – Principal Component Analysis

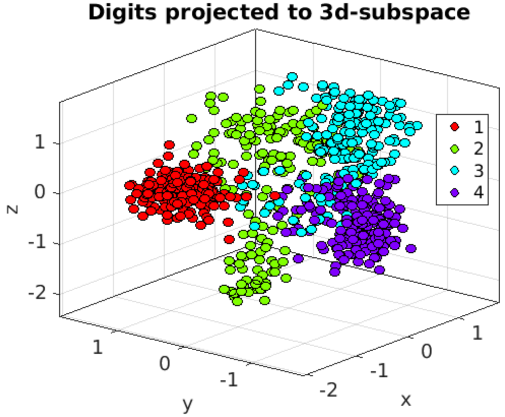
- Dimensionality Reduction
- Data De-correlation/Feature Extraction
- Dataset: $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^M\}$ where $\boldsymbol{x}^i \in R^N$





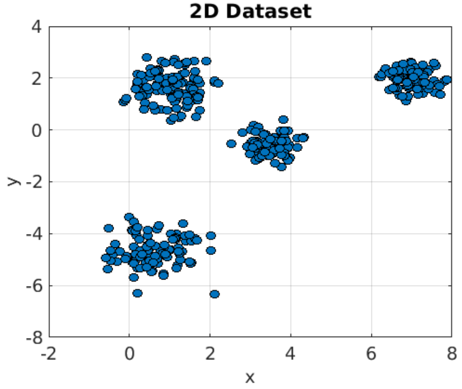
- Output: $\boldsymbol{y} = A\boldsymbol{x}$ where $A \in R^{p\times N}$ for $p \ll N$

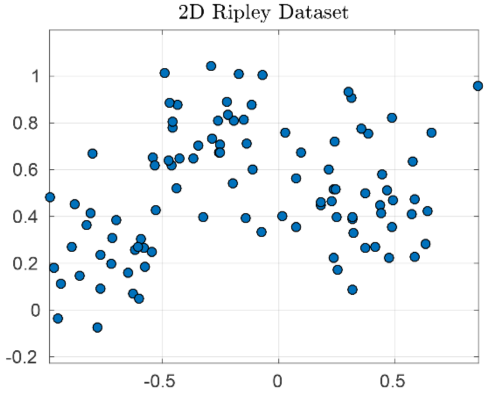




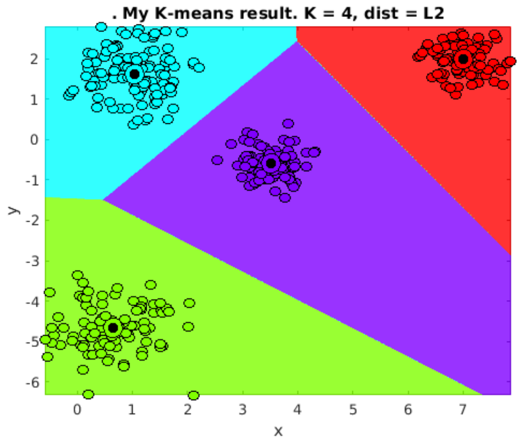
TP2 – K-Means Algorithm

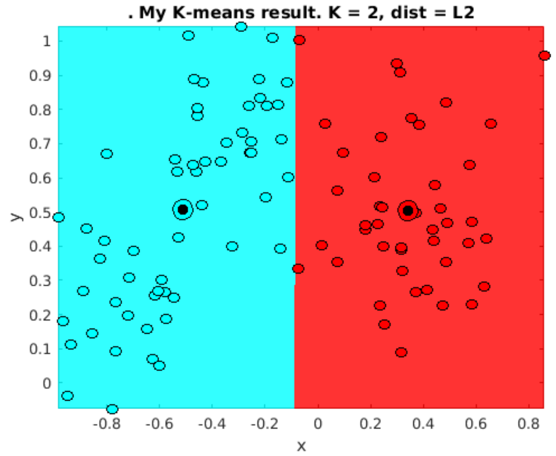
- Unsupervised Learning
- Finding sub-groups in data (clustering)
- Dataset: $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^M\}$ where $\boldsymbol{x}^i \in R^N$





- Output: Find K clusters that best describe $\boldsymbol{X}$.



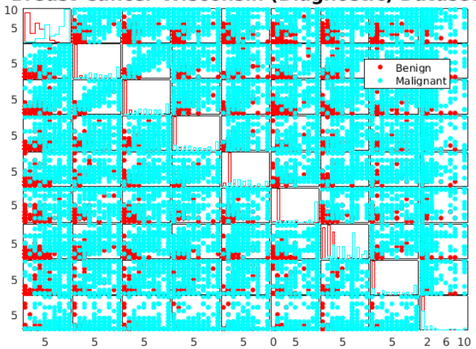


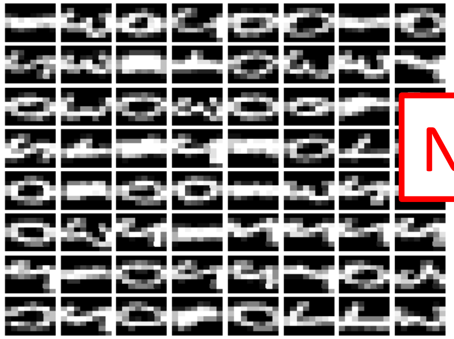
What have we covered so far?

TP1 – Principal Component Analysis

- Dimensionality Reduction
- Data De-correlation/Feature Extraction
- Dataset: $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^M\}$ where $\boldsymbol{x}^i \in R^N$

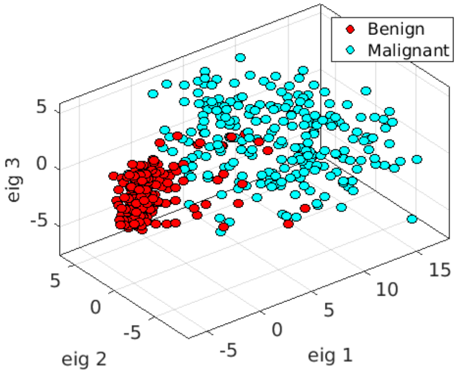
Breast-Cancer-Wisconsin (Diagnostic) Dataset



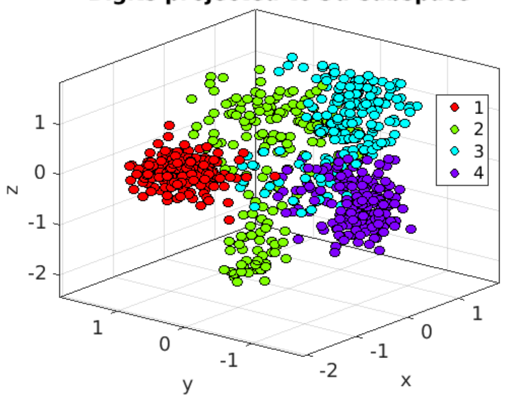


- Output: $\boldsymbol{y} = A\boldsymbol{x}$ where $A \in R^{p \times N}$ for $p \ll N$

Reduced Breast-Cancer-Wisconsin Dataset

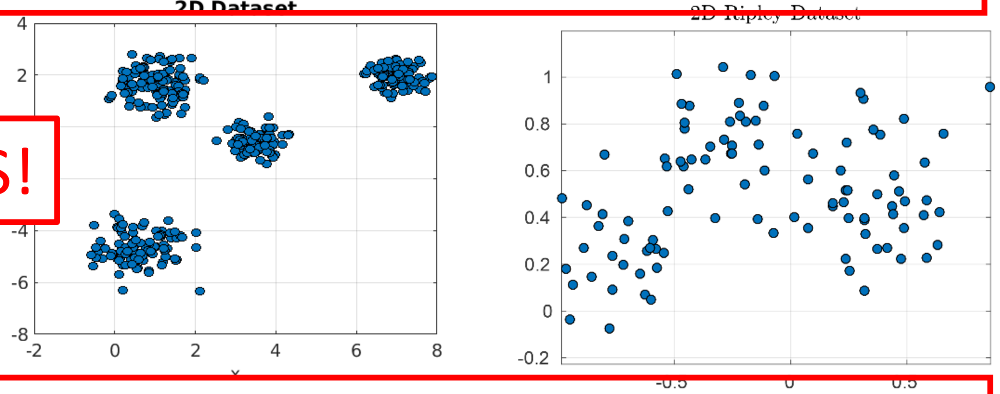


Digits projected to 3d-subspace



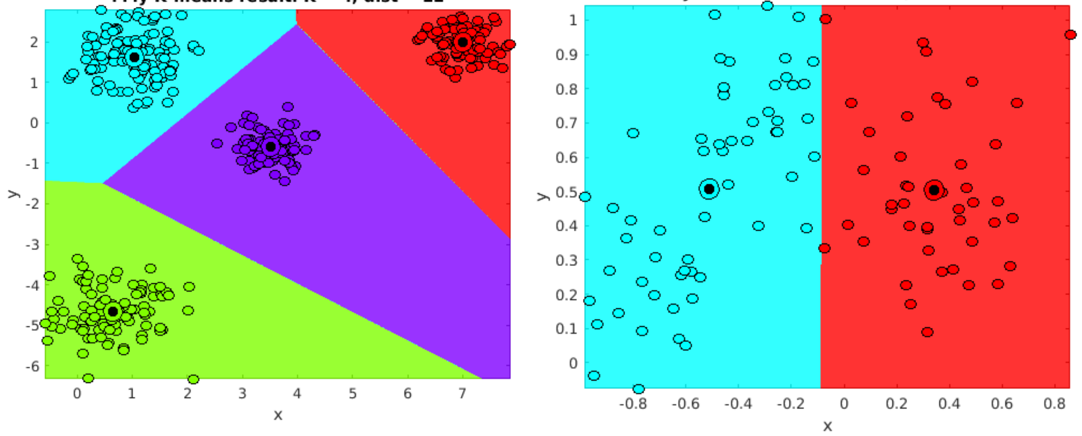
TP2 – K-Means Algorithm

- Unsupervised Learning
- Finding sub-groups in data (clustering)
- Dataset: $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^M\}$ where $\boldsymbol{x}^i \in R^N$



NO LABELS!

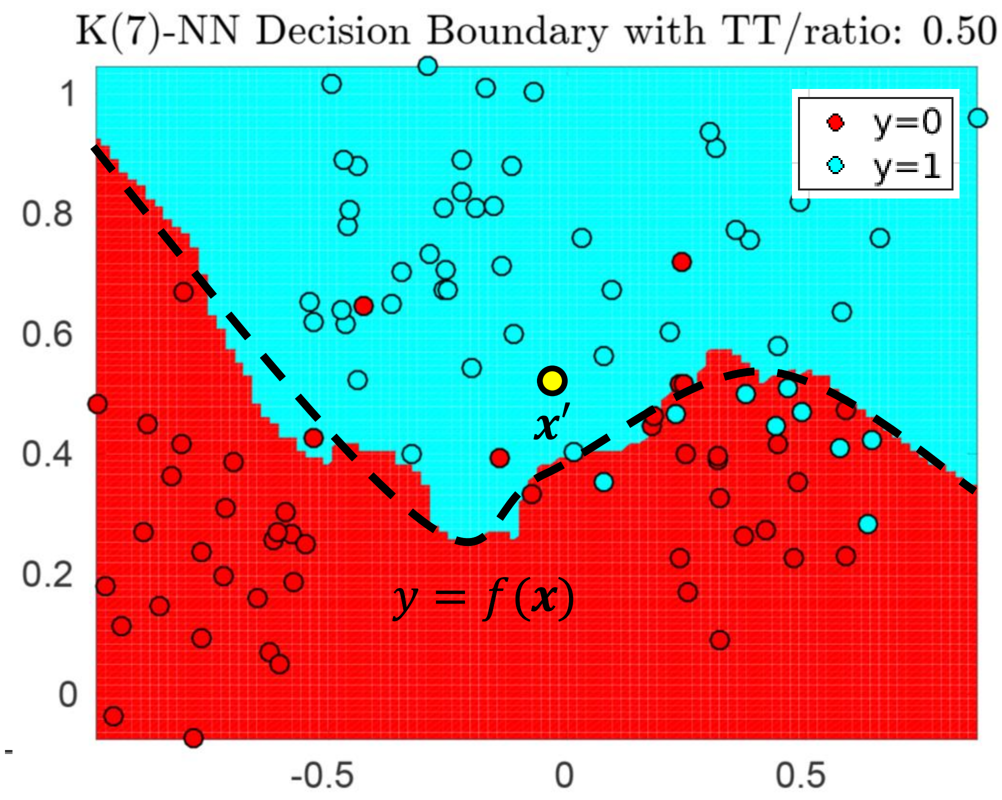
- Output: Find K clusters that best describe $\boldsymbol{X}$.



Classification = LABELS!

- Supervised Learning:
  The algorithm learns a function $y = f(x)$ that maps a set of inputs x to a set of outputs y.

- Classification: Inputs are a set of data-points or feature vectors $\boldsymbol{x}^i \in R^N$ and our outputs are a set of pre-defined classes denoted by a categorical label y, for binary classification problem $y \in \{0,1\}$

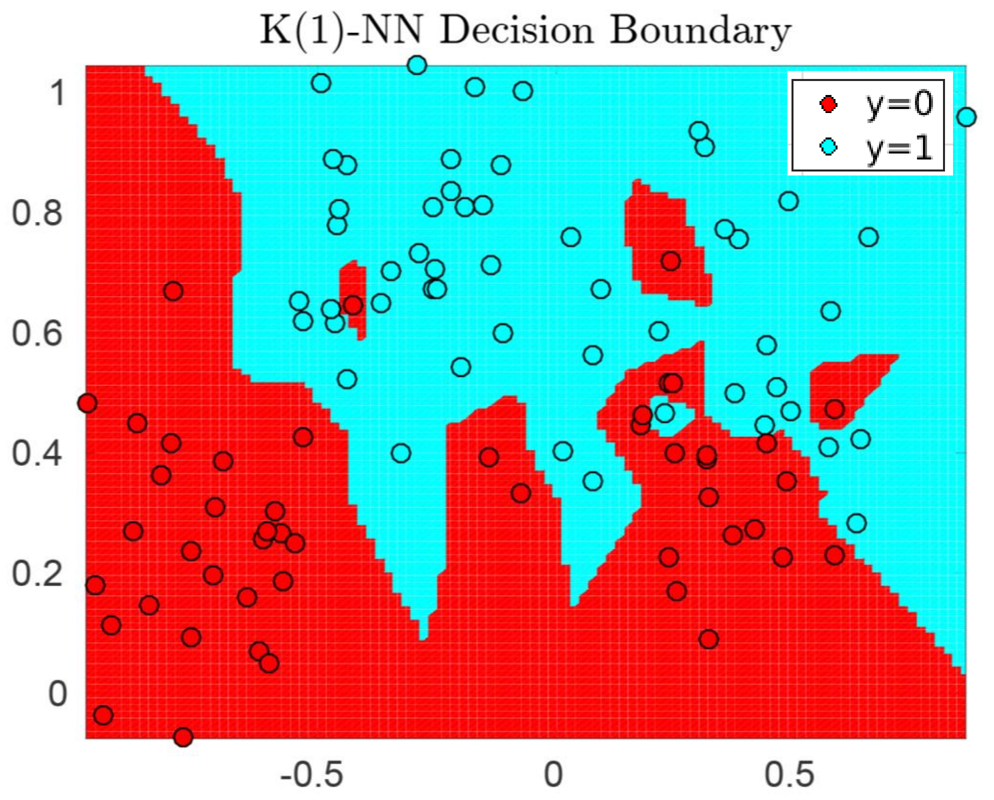


- Dataset: $\boldsymbol{D} = \{(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^1, y^2), \ldots, (\boldsymbol{x}^1, y^M)\}$ where $\boldsymbol{x}^i \in R^N$ and $y^i \in \{0,1\}$

- Output: $y = f(\boldsymbol{x})$ such that:
  Given a query point $\boldsymbol{x}' \in R^N$ we can predict which class it belongs to $\hat{y} = f(\boldsymbol{x}')$

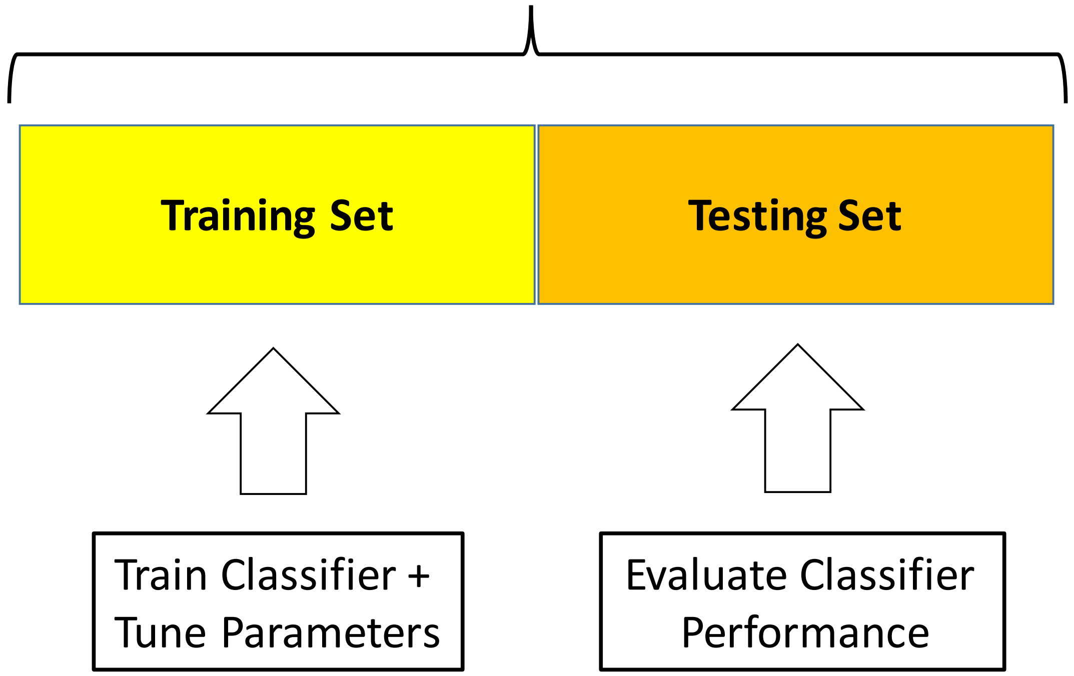
Training/Testing Sets for Classification

- **Over-fitted model:**
  - Fits noise + outliers
  - Does not generalize well
  - Example: KNN learned on 100% of data



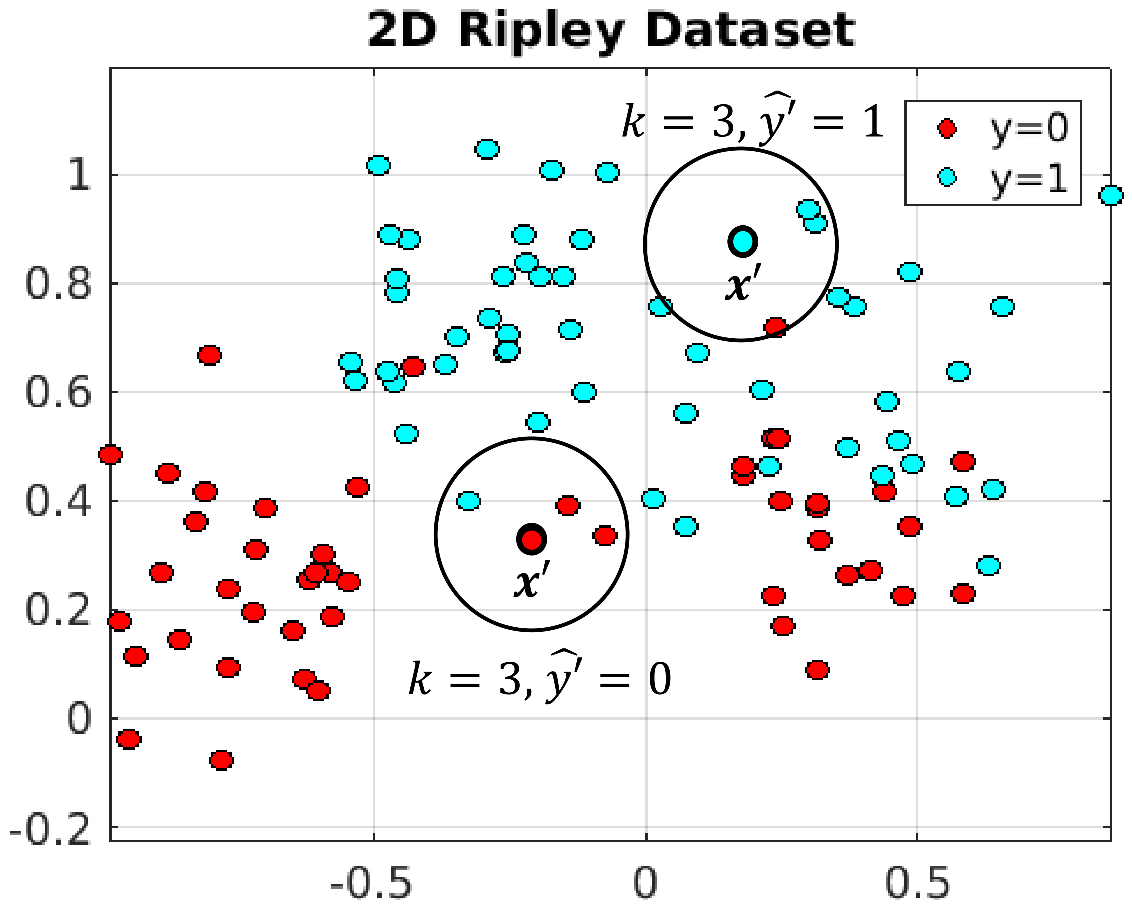
- To **avoid over-fitting** a classifier, one must partition the dataset into train/test sets.

Dataset: $\boldsymbol{D} = \{(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^1, y^2), \dots, (\boldsymbol{x}^1, y^M)\}$



TP3: K-Nearest Neighbors Algorithm

"The query point $\boldsymbol{x}' \in R^N$ is assigned the label $\widehat{y}' = f(\boldsymbol{x}')$ most frequently represented among its k nearest neighbors." - Duda



K-NN Algorithm:

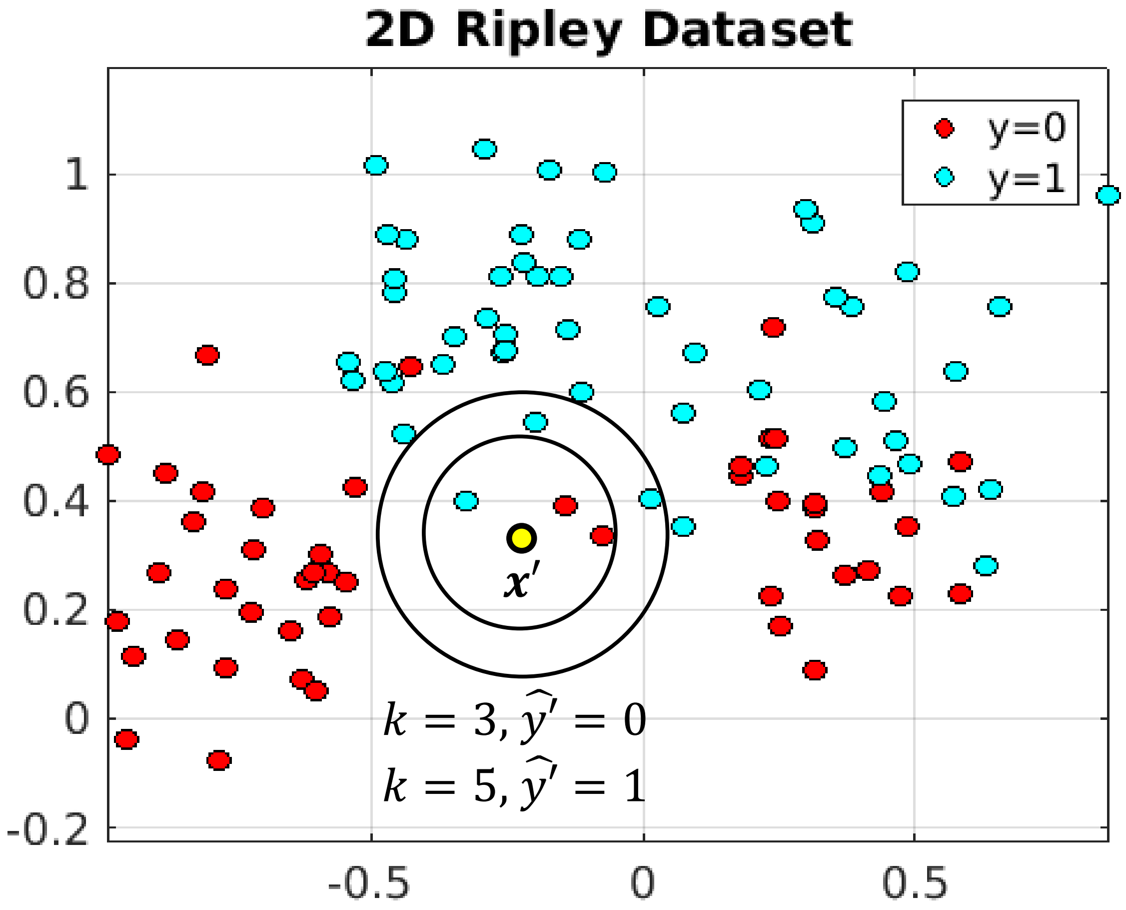
Step 1: *Compute distances between $\boldsymbol{x}'$ and all $\boldsymbol{x}$ in $\boldsymbol{D_{train}}$*

Step 2: *Extract the k-Nearest Neighbors*

Step 3: *Majority vote from k-NN labels.*

TP3: K-Nearest Neighbors Algorithm

"The query point $\boldsymbol{x'} \in R^N$ is assigned the label $\widehat{y}' = f(\boldsymbol{x'})$ most frequently represented among its k nearest neighbors." - Duda



K-NN Algorithm:

Step 1: *Compute distances between $\boldsymbol{x'}$ and all $\boldsymbol{x}$ in $\boldsymbol{D_{train}}$*

Step 2: *Extract the k-Nearest Neighbors*

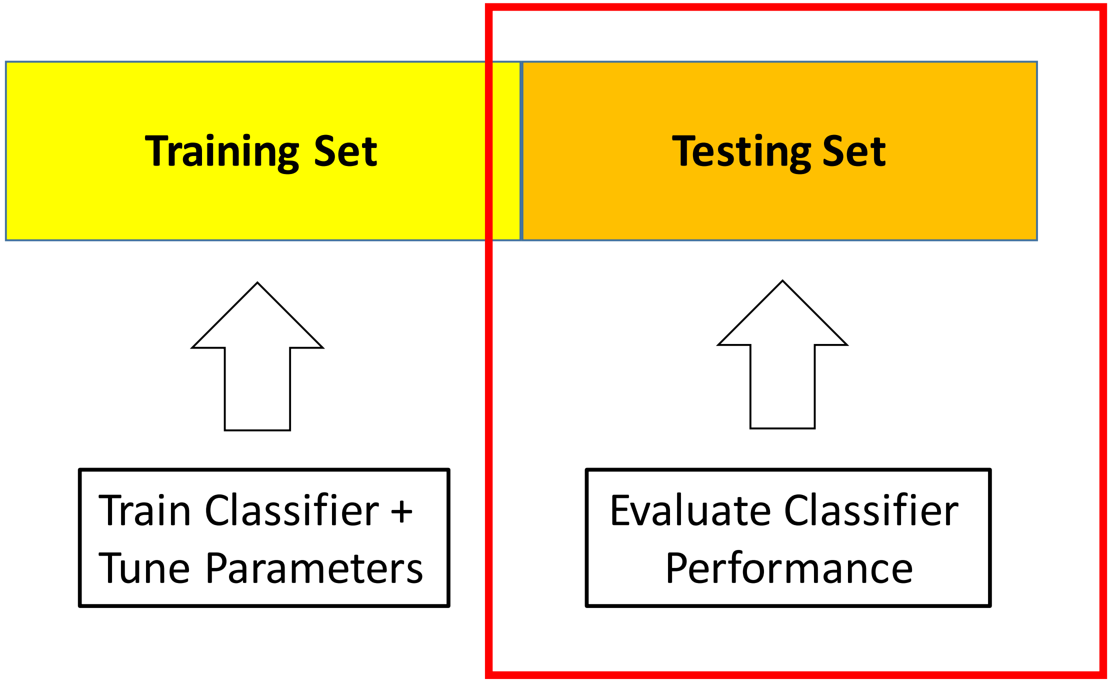
Step 3: *Majority vote from k-NN labels.*

➢ ***K-NN is very sensitive to the choice of K!***

How can we find the best K?

➢ ***Evaluate range of K on test set.***

Evaluate Classifier Performance

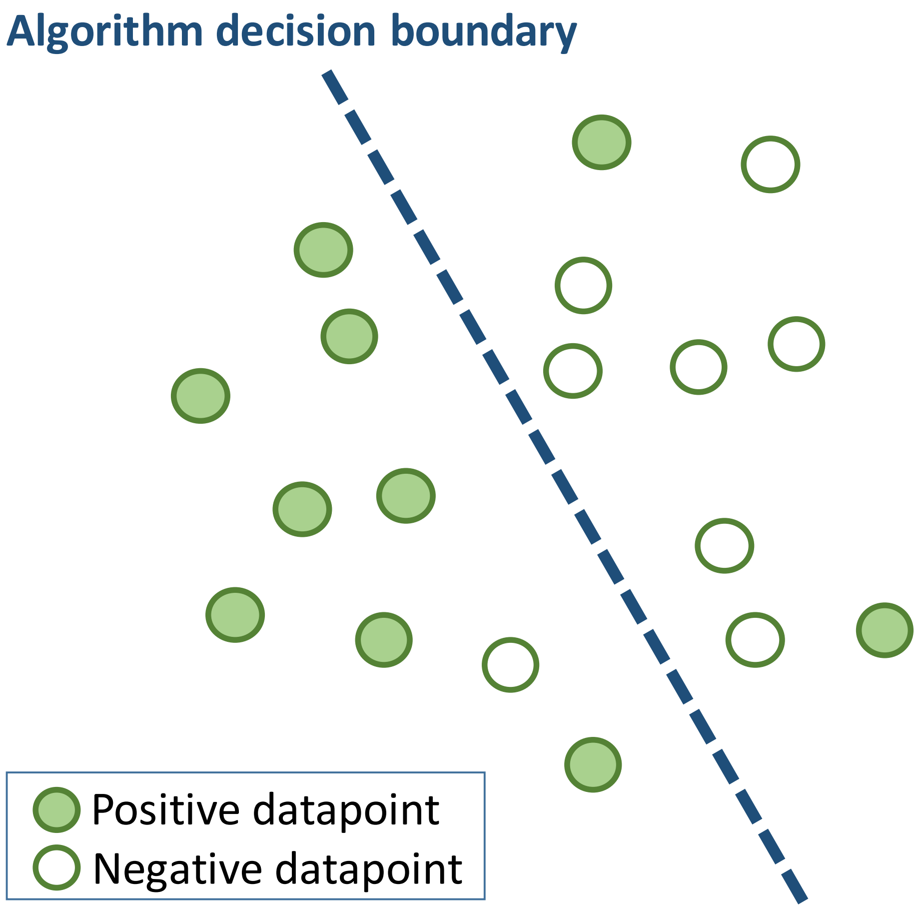


<u>Classification Evaluation:</u>

- Confusion Matrix
- ROC Curve
- Cross-Validation

Confusion Matrix

True/False Positives and True/False Negatives



Confusion matrix computation

		Estimated labels	
		Positive	Negative
Real labels	Positive		
	Negative		

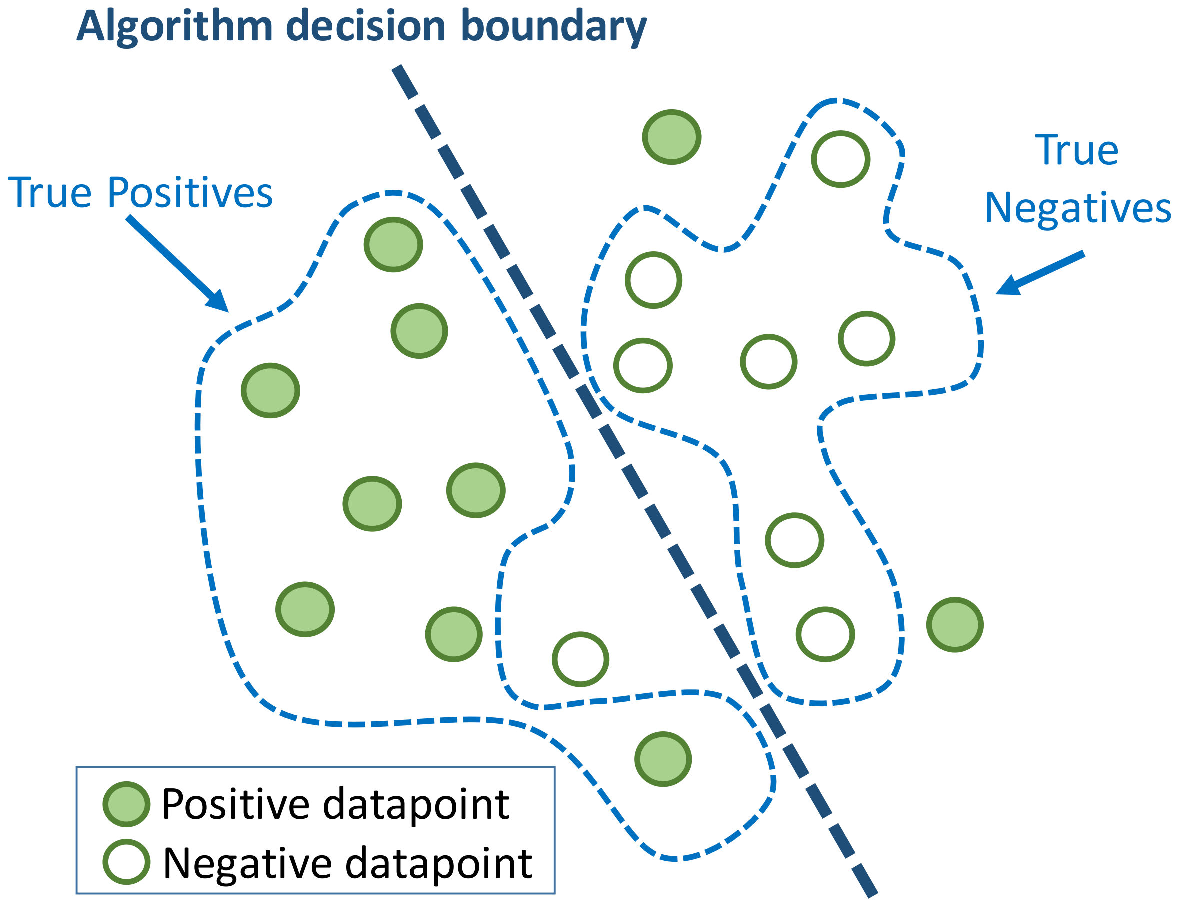
Rates formulation

True Positives Rate: (sensitivity or recall) $TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$

False Negative Rate: $FPR = \frac{FP}{FP+FN} = \frac{FP}{N}$

Confusion Matrix

True/False Positives and True/False Negatives



Confusion matrix computation

		Estimated labels	
		Positive	**Negative**
Real labels	**Positive**	True Positive (TP)	
	Negative		True Negatives (TN)

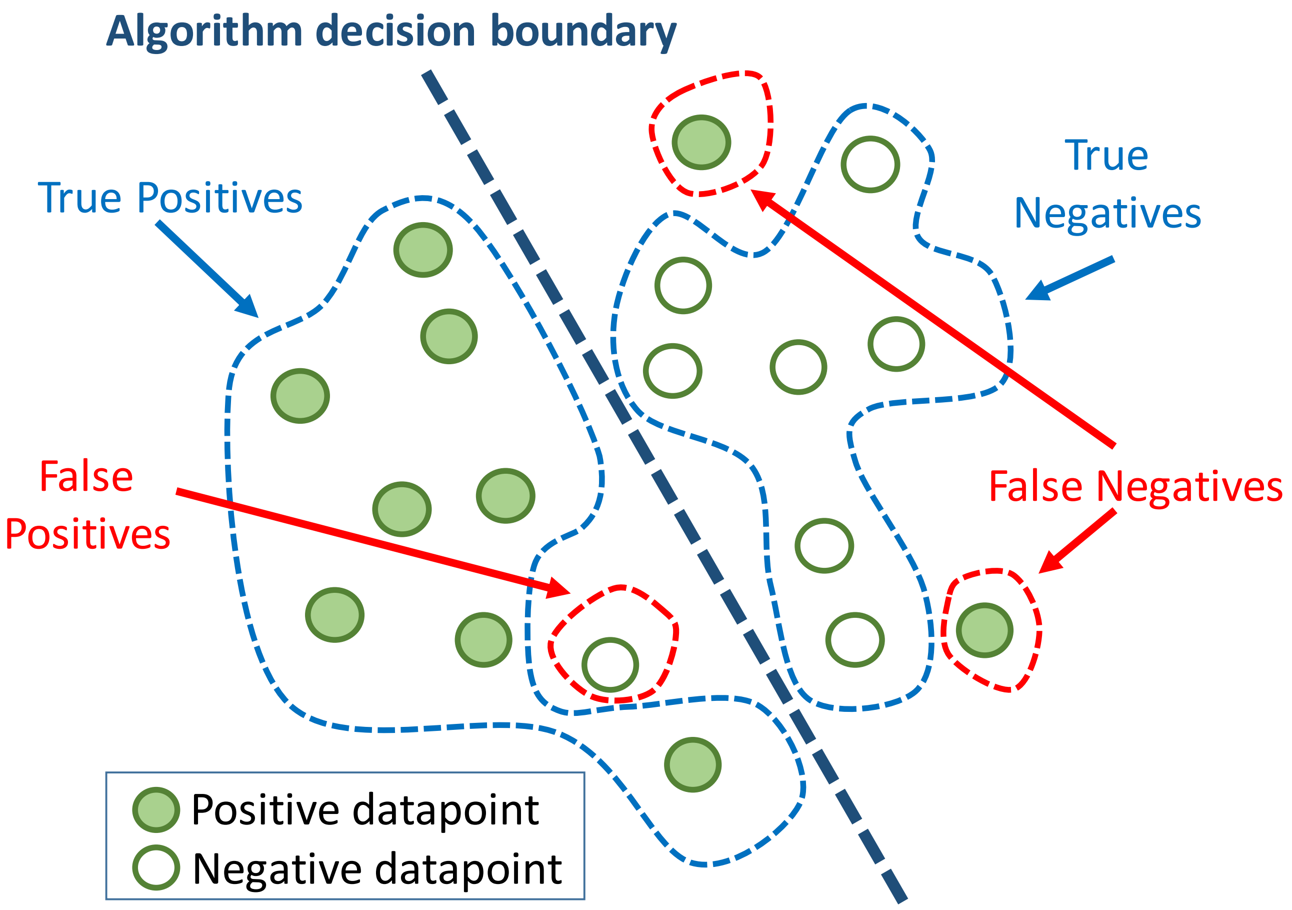
Rates formulation

True Positives Rate: (sensitivity or recall) $TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$

False Negative Rate: $FPR = \frac{FP}{FP+FN} = \frac{FP}{N}$

Confusion Matrix

True/False Positives and True/False Negatives



Confusion matrix computation

Estimated labels

Real labels		Positive	Negative
	Positive	True Positive (TP)	False Negatives (FN)
	Negative	False Positive (FP)	True Negatives (TN)

Rates formulation

True Positives Rate:
(sensitivity or recall)

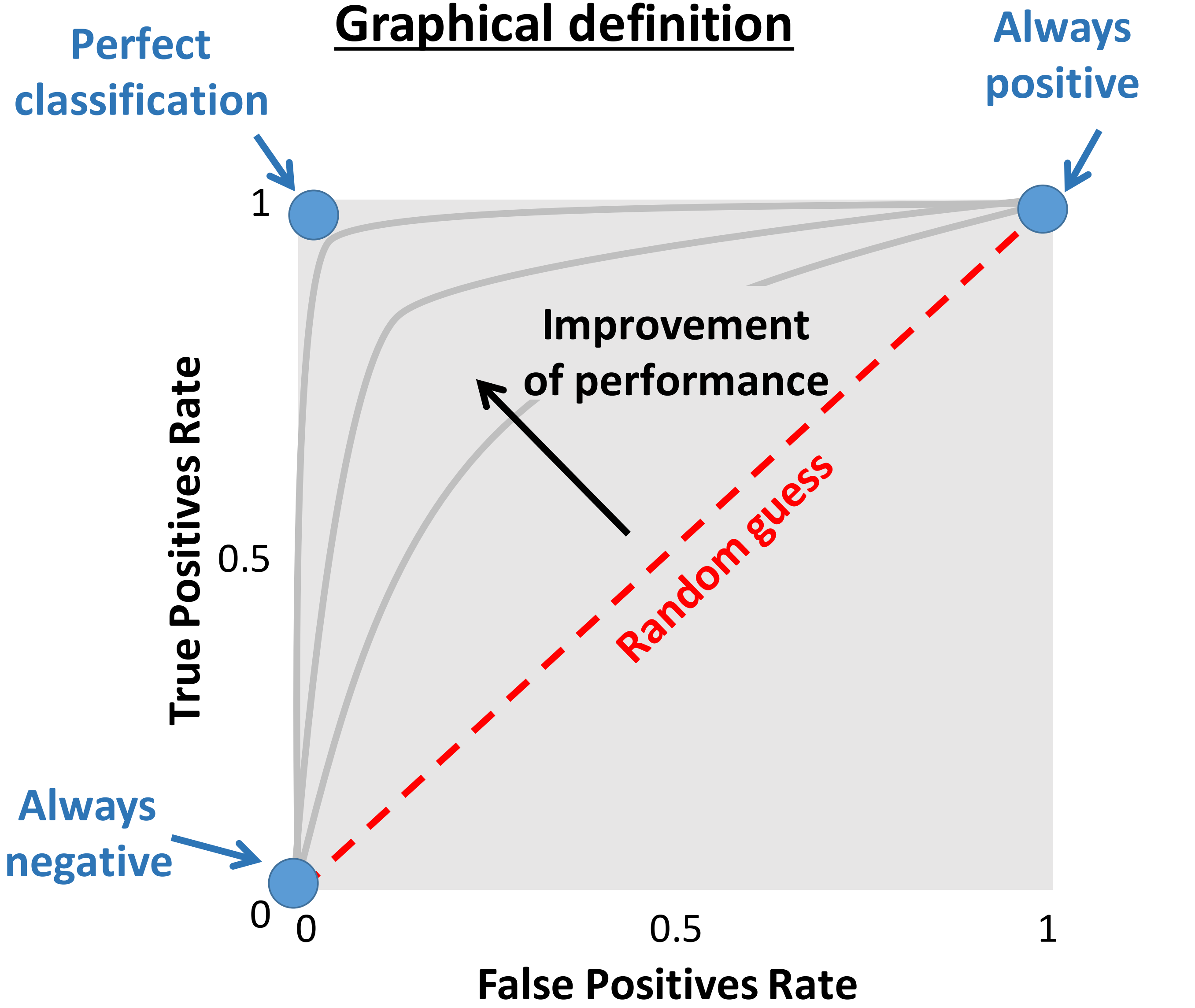
$$TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$$

False Negative Rate:

$$FPR = \frac{FP}{FP+FN} = \frac{FP}{N}$$

ROC Curve

Graphical definition



Why using ROC curves?

- Better assessment of performances
- Choice of a threshold of classification
- Comparison of different algorithms

Rates formulation

True Positives Rate: (sensitivity or recall) $TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$

False Negative Rate: $FPR = \frac{FP}{FP+FN} = \frac{FP}{N}$

Cross validation

Definition: "Cross validation is the practice of confirming an experimental finding by repeating the experiment using an independent assay technique"

F-fold cross validation

- Constant Train/Test ratio
- At each iteration:
  1) Random split of the data between Train and Test
  2) Repetition of classification
- Averaging of the result across folds