

Techniques d'apprentissage  
IFT 603 / IFT 712

Combinaison de modèles

Par  
Pierre-Marc Jodoin  
/  
Hugo Larochelle

1

---

---

---

---

---

---

---

---

## Pourquoi utiliser un seul modèle?

- Pourquoi utiliser un seul modèle?
  - un système combinant une multitude de modèles différents ne serait-il pas meilleur?
- En pratique, la réponse est presque toujours oui !
  - le résultat de la combinaison de plusieurs modèles est appelée **ensemble** ou **comité**

2

---

---

---

---

---

---

---

---

## Pourquoi utiliser un seul modèle?

- La façon la plus simple d'obtenir  $M$  modèles est d'utiliser  $M$  algorithmes d'apprentissage différents :
  - POUR  $i = 1$  à  $m$ 
    - Entraîner un modèle  $y_{w,i}(\vec{x})$  à l'aide du  $i$ -ième algo d'entraînement
- Retourner le **modèle ensemble** (ou comité)
  - Pour la **régression**
    - $y_{COM}(\vec{x}) = \frac{1}{m} \sum_{i=1}^m y_{w,i}(\vec{x})$
  - Pour la **classification**
    - **vote majoritaire**

3

---

---

---

---

---

---

---

---

## Pourquoi utiliser un seul modèle?

- Les  $M$  algorithmes pourraient être le même algorithme avec avec  $M$  sélections d'hyperparamètres différents.

- POUR  $i = 1$  à  $m$ 
  - Entraîner un modèle  $y_{w,i}(\vec{x})$  à l'aide du  $i$ -ième algo d'entraînement

- Retourner le **modèle ensemble** (ou comité)

- Pour la **régression**
  - $y_{COM}(\vec{x}) = \frac{1}{m} \sum_{i=1}^m y_{w,i}(\vec{x})$

- Pour la **classification**
  - **vote majoritaire**

4

---

---

---

---

---

---

---

---

## Pourquoi utiliser un seul modèle?

- Même avec **un seul algorithme et les mêmes hyperparamètres**, on peut améliorer sa performance à l'aide d'un ensemble.

- **Bagging** : approprié pour combiner des modèles avec une **forte capacité**
- **Boosting** : approprié pour combiner des modèles avec une **faible capacité**

5

---

---

---

---

---

---

---

---

## Pourquoi utiliser un seul modèle?

- Même avec **un seul algorithme et les mêmes hyperparamètres**, on peut améliorer sa performance à l'aide d'un ensemble.

- **Bagging** : approprié pour combiner des modèles avec une **forte capacité**
- **Boosting** : approprié pour combiner des modèles avec une **faible capacité**

6

---

---

---

---

---

---

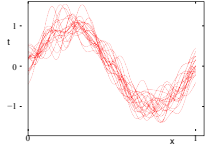
---

---

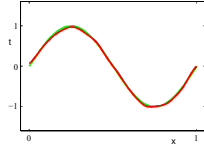
## Bootstrap : réduction de la variance

- Régression polynomiale de degré 25

100 modèles entraînés sur  
100 ensembles d'entraînement différents



Ensemble des 100 modèles vs. vrai modèle



7

---

---

---

---

---

---

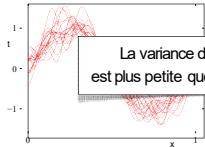
---

---

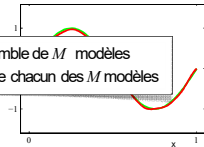
## Bootstrap : réduction de la variance

- Régression polynomiale de degré 25

100 modèles entraînés sur  
100 ensembles d'entraînement différents



Ensemble des 100 modèles vs. vrai modèle



La variance d'un ensemble de  $M$  modèles  
est plus petite que celle de chacun des  $M$  modèles

8

---

---

---

---

---

---

---

---

## Bootstrap

À part pour des données synthétiques, on ne peut pas générer des données sur demande.

Solution : **Bootstrapping**.

$$D_{bootstrap} = \{ \}$$

Pour  $N$  itérations

- Choisir aléatoirement un entier parmi  $\{1, \dots, N\}$
- $D_{bootstrap} = D_{bootstrap} \cup \{(\tilde{x}_n, t_n)\}$

retourner  $D_{bootstrap}$

9

---

---

---

---

---

---

---

---

## Bootstrap

À part pour des données synthétiques, on ne peut pas générer des données sur demande.

Solution : **Bootstrapping**.

$$D_{bootstrap} = \{ \}$$

Pour N itérations

- Choisir aléatoirement un entier parmi  $\{1, \dots, N\}$
- $D_{bootstrap} = D_{bootstrap} \cup \{(\tilde{x}_n, t_n)\}$

retourner  $D_{bootstrap}$

échantillonne  $N$  exemples  
avec remplacement

10

## Bagging (Bootstrap AGGREGatING)

1. Générer  $m$  ensembles d'entraînement avec **Bootstrap**  $\{D_1, D_2, \dots, D_m\}$

2. **Entraîner**  $m$  modèles  $y_{w,j}(\tilde{x})$  (un pour chaque ensemble)

3. **Combinaison** des  $m$  modèles

**Régression:**  $y_{com}(\tilde{x}) = \frac{1}{m} \sum_{j=1}^m y_{w,j}(\tilde{x})$

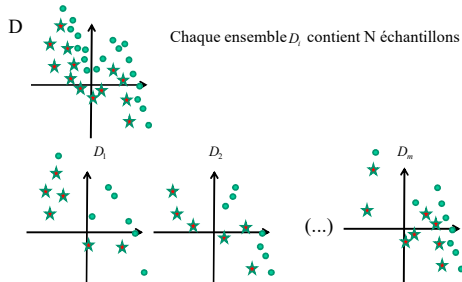
**Classification:**  $y_{com}(\tilde{x}) = \text{sign} \left( \sum_{j=1}^m y_{w,j}(\tilde{x}) \right)$  (2 classes)

$y_{com}(\tilde{x}) = \arg \max_c \left( \sum_{j=1}^m y_{w,j}(\tilde{x}) \right)$  (K classes)

11

## Illustration: classification 2Classes

1- **échantillonnage** avec remplacement (*Bootstrap*)

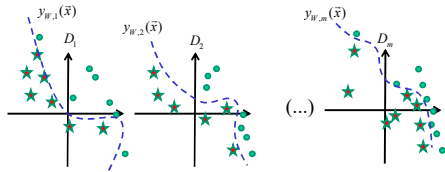


Note: un échantillon  $\tilde{x}_j$  peut apparaître plusieurs fois dans un même ensemble d'entraînement  $D_k$

## Illustration: classification 2Classes

### 2- Entraînement

Entraîner un modèle  $y_{w,j}(\vec{x})$  (peut être perceptron, SVM, noyaux, etc.) sur chaque ensemble d'entraînement



13

---

---

---

---

---

---

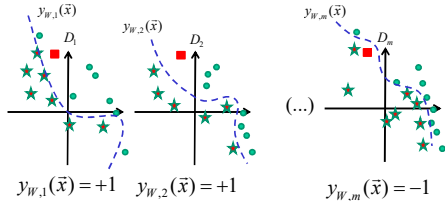
---

---

## Illustration: classification 2Classes

### 3- Vote majoritaire

$$y_{com}(\vec{x}) = \text{sign}\left(\sum_{i=1}^m y_{w,i}(\vec{x})\right)$$



14

---

---

---

---

---

---

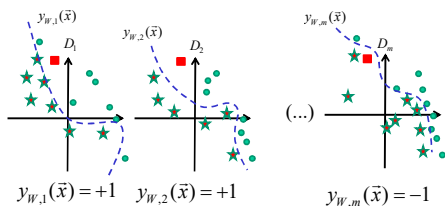
---

---

## Illustration: classification 2Classes

### 3- Vote majoritaire

$$y_{com}(\vec{x}) = \text{sign}(1+1-1+1-1-1+\dots+1) = +1$$



15

---

---

---

---

---

---

---

---

## Bagging

### Analyse théorique de l'erreur : au tableau

16

---

---

---

---

---

---

---

17

---

---

---

---

---

---

---

## Pourquoi utiliser un seul modèle?

- Même avec un seul algorithme sans hyper-paramètres, on peut améliorer sa performance à l'aide d'un ensemble.
  - *Bagging* : approprié pour combiner des modèles avec une forte capacité
  - *Boosting* : approprié pour combiner des modèles avec une faible capacité

18

---

---

---

---

---

---

---

## AdaBoost

La méthode du boosting a pour objectif de **combiner plusieurs modèles faibles** (*weak learners*) afin d'obtenir un classifieur avec une plus grande capacité.

**Trois (3) différences majeures** avec le Bagging.

1. Implémenter une **combinaison pondérée** de modèles. Ex. 2 classes

$$y_{com}(\vec{x}) = \text{sign} \left( \sum_{i=1}^m \alpha_i y_{w,i}(\vec{x}) \right)$$

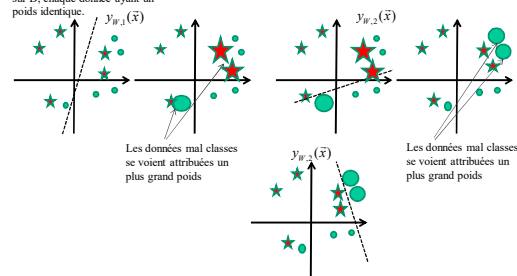
2. **Pas de bootstrap**: chaque donnée  $\vec{x}_n$  est utilisée pour entraîner les modèles
3. Les **données mal classées** par un modèle  $y_{w,i}(\vec{x})$  auront **plus de poids** lors de l'entraînement du prochain modèle  $y_{w,i+1}(\vec{x})$

R. Schapire and Y. Freund **A decision theoretic generalization of on-line learning and an application to Boosting** Journal of Computer and System Sciences, 1997, 55: 119-139. 19

## AdaBoost

(illustration 2 classes)

Le premier modèle est entraîné sur D, chaque donnée ayant un poids identique.



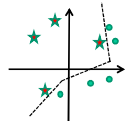
20

## AdaBoost

(illustration 2 classes)

Le modèle combiné implémente une **combinaison pondérée** des 3 classifieurs faibles

$$y_{com}(\vec{x}) = \text{sign}(\alpha_1 y_{w,1}(\vec{x}) + \alpha_2 y_{w,2}(\vec{x}) + \alpha_3 y_{w,3}(\vec{x}))$$

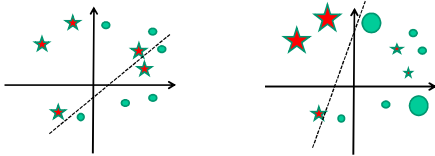


NOTE: plus un classifieur a une **exactitude élevée**, plus son  $\alpha_i$  sera élevé. 21

## AdaBoost

**Idée fondamentale:** chaque donnée  $\vec{x}_i$  a un **poids**  $\beta_i$

Lorsque les données ont toutes un **poids égale**, alors le **modèle faible** devient un classifieur linéaire comme un **perceptron** ou une regression logistique.

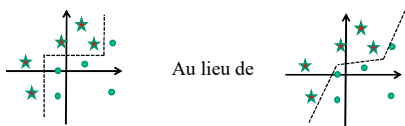


Poids égaux Vs poids non égaux

22

## AdaBoost

Les modèles de type **stumps** sont des classifieurs perpendiculaires à un axe  
La combinaison de modèles **stumps** mène à des frontières de décision "crénelées"



Au lieu de

**2 avantages:** très **rapide** et permet d'identifier les **caractéristiques utiles**:

$$y_{com}(\vec{x}) = \text{sign} \left( \sum_{i=1}^m \alpha_i y_{w,i}(\vec{x}) \right)$$

Les caractéristiques utiles sont celles pour lesquelles  $\alpha_i$  est élevé

### Adaboost

1- initialiser le poids des N données d'apprentissage:  $\beta_i = \frac{1}{N} \quad \forall i$

POUR  $i=1$  à  $m$

2- Entraîner le modèle  $y_{w,i}(\vec{x})$  avec les données  $D$  et les poids  $\{\beta_1, \dots, \beta_N\}$

3- Calculer l'erreur d'entraînement:  $\epsilon_i$

$$\epsilon_i = \frac{\sum_{\vec{x} \in \Psi} \beta_{\vec{x}}}{\sum_{\vec{x} \in \Psi} \beta_{\vec{x}}} \quad \text{où } \Psi \text{ l'ensemble des données mal classées}$$

4- calculer  $\alpha_i = \ln \left( \frac{1 - \epsilon_i}{\epsilon_i} \right)$

5- mise à jour du poids des données mal classées par  $y_{w,i}(\vec{x})$

$$\beta_{\vec{x}} = \beta_{\vec{x}} \exp(\alpha_i)$$

6- Normaliser les poids afin qu'ils somment à 1

$$\beta_{\vec{x}} = \frac{\beta_{\vec{x}}}{\sum_{\vec{x}} \beta_{\vec{x}}}$$

Le classifieur combiné:  $y_{com}(\vec{x}) = \text{sign} \left( \sum_{i=1}^m \alpha_i y_{w,i}(\vec{x}) \right)$

24



### Adaboost

- 1- initialiser le poids des N données d'apprentissage:  $\beta_i = \frac{1}{N} \quad \forall i$
- POUR  $i=1$  à  $m$ 
  - 2- Entraîner le modèle  $y_{w,i}(\vec{x})$  avec les données  $D$  et les poids  $\{\beta_1, \dots, \beta_{i-1}\}$
  - 3- Calculer l'erreur d'entraînement:  $\epsilon_i$ 

$$\epsilon_i = \frac{\sum_{\vec{x} \in W} \beta_i}{\sum_{\vec{x} \in W} \beta_i}$$

où  $\Psi$  l'ensemble des
  - 4- calculer  $\alpha_i = \ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$
  - 5- mise à jour du poids des données mal classées par  $y_{w,i}(\vec{x})$ 

$$\beta_i = \beta_i \exp\{\alpha_i\}$$
  - 6- Normaliser les poids afin qu'ils somment à 1
 
$$\beta_i = \frac{\beta_i}{\sum_j \beta_j}$$

Plus un classifieur a une erreur faible, plus son poids  $\alpha_i$  sera élevé

Le classifieur combiné:  $y_{com}(\vec{x}) = \text{sign}\left(\sum_{i=1}^m \alpha_i y_{w,i}(\vec{x})\right)$

25

---

---

---

---

---

---

---

---

## Sklearn

```

from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
(...)

bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),
                        algorithm="SAMME",
                        n_estimators=200)

bdt.fit(X, y)

```

Classifieur « Stump »

Algo de la page précédente

200 modèles

26

---

---

---

---

---

---

---

---

27

---

---

---

---

---

---

---

---

9

## Arbres de décision

28

---

---

---

---

---

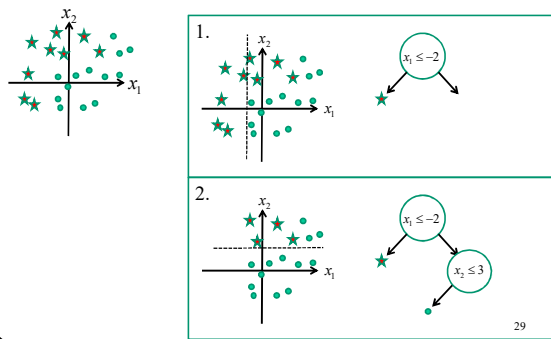
---

---

---

## Arbres de décision

Exemple simple avec des classificateurs *stump*




---

---

---

---

---

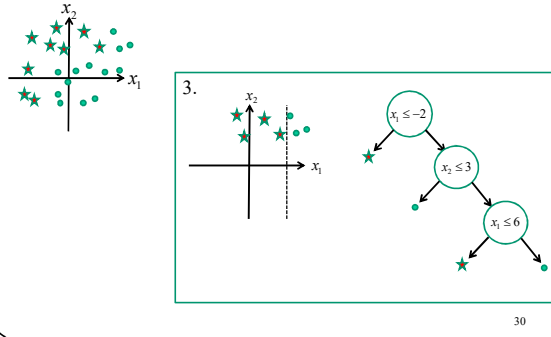
---

---

---

## Arbres de décision

Exemple simple avec des classificateurs *stump*




---

---

---

---

---

---

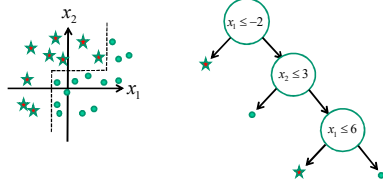
---

---

## Arbres de décision

Exemple simple avec des classifieurs *stump*

Au final



31

---

---

---

---

---

---

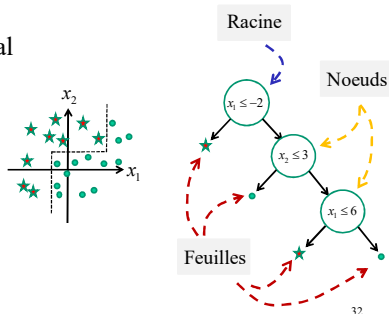
---

---

## Arbres de décision

Exemple simple avec des classifieurs *stump*

Au final



32

---

---

---

---

---

---

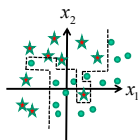
---

---

## Arbres de décision

Le plus gros problème des arbres de décision est qu'ils ont tendance à **sur-apprendre**

Exemple avec 2 données aberrantes:



Grosse question : soit le noeud d'un arbre dont l'erreur d'entraînement n'est pas nulle **devons-nous le subdiviser ou non?**

33

---

---

---

---

---

---

---

---

## Arbres de décision

La décision de subdiviser ou non un noeud dépend d'une notion d' "impureté" d'un noeud

Si l'impureté d'un noeud est **élevée** → alors **on subdivise**  
 Si l'impureté d'un noeud est **faible** → alors **on ne subdivise pas**

Deux mesures d'impureté fréquemment utilisées

### 1. L'entropie

$$i(\text{node}) = - \sum_{j=1,2} P(c_j) \log_2(P(c_j))$$

où  $P(c_j)$  est la proportion de données dans la classe  $c_j$

### 2. L'indice de Gini

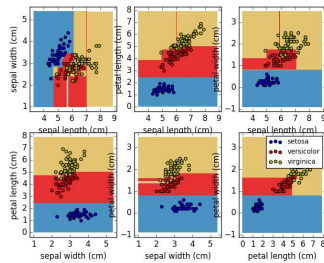
$$i(\text{node}) = 1 - \sum_{j=1,2} P^2(c_j)$$

34

## scikit-learn

```
from sklearn.tree import DecisionTreeClassifier
(....)
clf = DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=10,
                             min_impurity_split=0.0001)
clf.fit(X, Y)
```

Decision surface of a decision tree using paired features

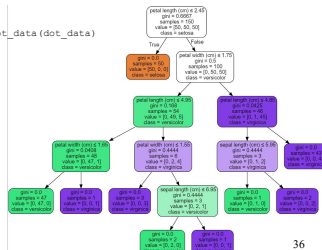


35

## scikit-learn

```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from IPython.display import Image
import pydotplus
(....)
clf = DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=10,
                             min_impurity_split=0.0001)
clf.fit(X, Y)
```

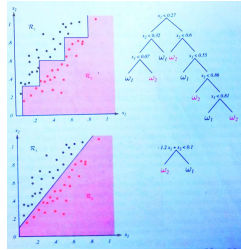
```
dot_data = export_graphviz(clf)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



36

## Arbres de décision

Les arbres de décision peuvent inclure des classificateurs linéaires plus généraux



Credit, Duda Hart.

J Shotton, A Fitzgibbon, M Cook, T Sharp, M Finocchio, R Moore, A Kipman, and A Blake **Real-Time Human Pose Recognition in Parts from a Single Depth Image**

37

---

---

---

---

---

---

---

---

38

---

---

---

---

---

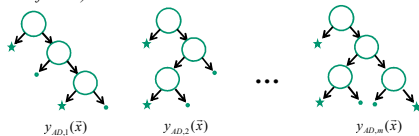
---

---

---

## Forêt d'arbres décisionnels (FAD)

(Random forests)



En gros, les **FAD** ont pour but de combiner  $m$  **arbres de décision** à l'aide d'un **vote majoritaire**

$$y_{FAD}(\vec{x}) = \text{sign} \left( \sum_{i=1}^m y_{AD,i}(\vec{x}) \right) \quad (\text{segmentation 2 classes})$$

Très bonne référence!

[https://www.stat.berkeley.edu/~breiman/RandomForests/cc\\_home.htm](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)

39

---

---

---

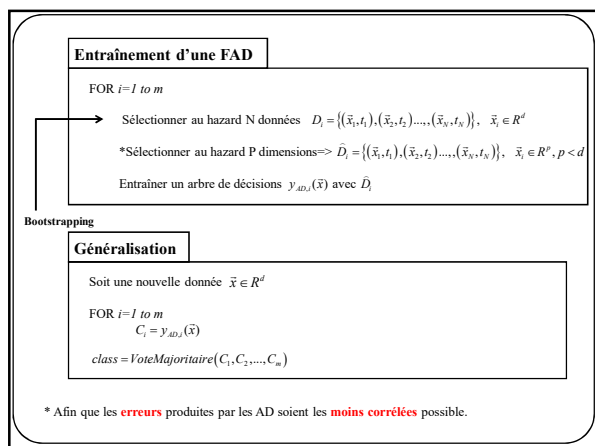
---

---

---

---

---




---

---

---

---

---

---

---

---

---

---

Métriques d'évaluation

41

---

---

---

---

---

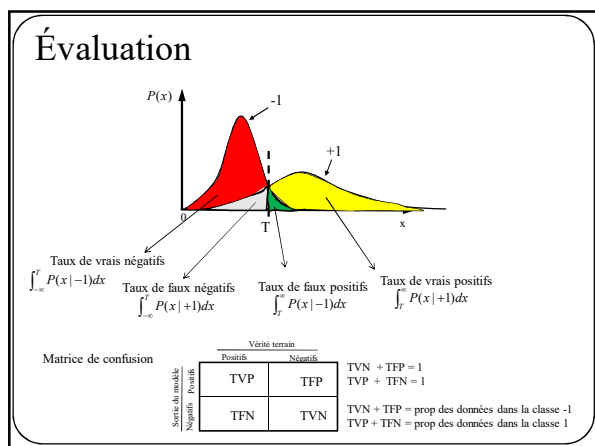
---

---

---

---

---




---

---

---

---

---

---

---

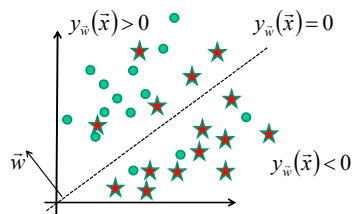
---

---

---

## Évaluation

Validation empirique



---

---

---

---

---

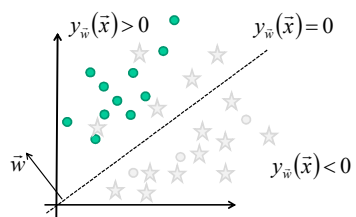
---

---

---

## Évaluation

Vrais positifs



---

---

---

---

---

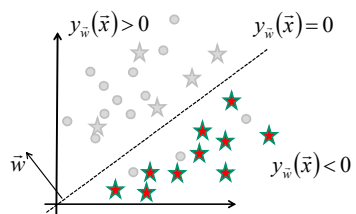
---

---

---

## Évaluation

Vrais négatifs



---

---

---

---

---

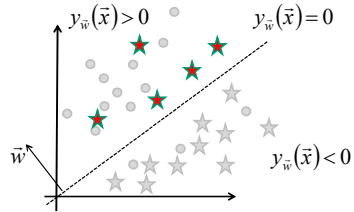
---

---

---

## Évaluation

### Faux positifs




---

---

---

---

---

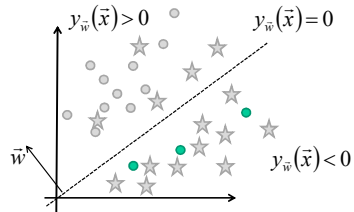
---

---

---

## Évaluation

### Faux négatifs




---

---

---

---

---

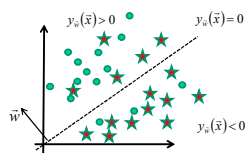
---

---

---

## Évaluation

### Validation empirique



On obtient une matrice de confusion en comptant les données bien et mal classés

		Vérité terrain		
		Positifs	Négatifs	
Sortie du modèle	Positifs	VP = 11	FP = 5	VN + FP = 15 = nombre TOTAL de négatifs VP + FN = 14 = nombre TOTAL de positifs
	Négatifs	FN = 3	VN = 10	VP + FP = 16 = nombre d'éléments classés +1 FN + VN = 13 = nombre d'éléments classés -1

---

---

---

---

---

---

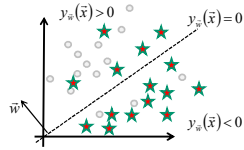
---

---



## Évaluation

Validation empirique



Taux de faux positifs (**TFP**) =  $FP/(FP+VN) = 5/15$

49

---

---

---

---

---

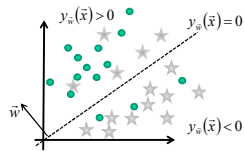
---

---

---

## Évaluation

Validation empirique



Taux de faux négatifs (**FNR**) =  $FN/(FN+VP) = 3/14$

50

---

---

---

---

---

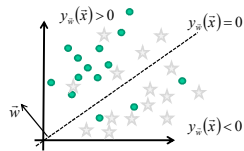
---

---

---

## Évaluation

Validation empirique



Rappel (*Recall* - **Re**) =  $VP/(FN+VP) = 1 - \text{TFN} = 11/14$

51

---

---

---

---

---

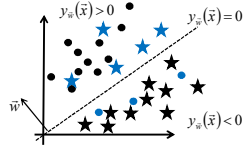
---

---

---

## Évaluation

Validation empirique



Justesse (« accuracy », **prop. de biens classés**) =  $(TP+TN)/(FP+FN+TP+TN) = 21/29$

52

---

---

---

---

---

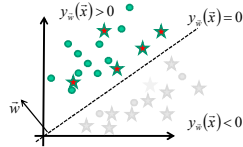
---

---

---

## Évaluation

Validation empirique



Précision (**Pr**) =  $VP/(VP+FP) = 11/16$

53

---

---

---

---

---

---

---

---

## Évaluation

		Vérité terrain	
		Positifs	Négatifs
Sortie du modèle	Positif	TP = 11	FP = 5
	Négatif	FN = 3	TN = 10

VN + FP = 15 = nombre de VRAIS négatifs  
VP + FN = 14 = nombre de VRAIS positifs

VP + FP = 16 = nombre d'éléments classés +1  
VP + FN = 13 = nombre d'éléments classés -1

Taux de faux positifs (**TFP**) =  $FP/(FP+VN) = 5/15$   
Taux de faux négatifs (**TFN**) =  $FN/(FN+VP) = 3/14$

Spécificité (**Sp**) =  $VN/(FP+VN) = 1-FPR = 10/15$   
Rappel (*recall*-**Re**) =  $VP/(VP+VN) = 11/14$   
Précision (**Pr**) =  $VP/(VP+FP) = 11/16$

Justesse (*Accuracy*, **prop. de biens classés**) =  $(VP+VN)/(FP+FN+VP+VN) = 21/29$   
**F-measure** =  $2 * (Re*Pr)/(Pr+Re) = 0.73$

54

---

---

---

---

---

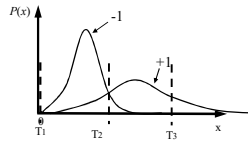
---

---

---

## Évaluation

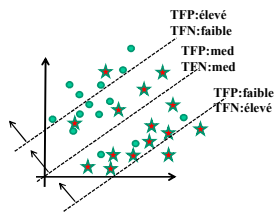
Différents seuils = différents résultats



TFP:élevé  
TFN:faible

TFP:med  
TFN:med

TFP:faible  
TFN:élevé



55

---

---

---

---

---

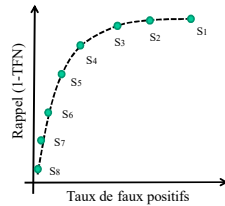
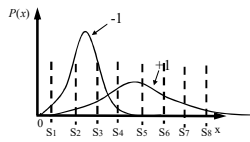
---

---

---

## Évaluation

Courbe ROC : enregistrer le TFP et le TVP pour différents seuils



56

---

---

---

---

---

---

---

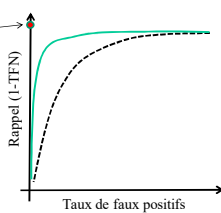
---

## Validation

Courbes ROC: très bonne façon de comparer des méthodes

Quelle méthode est la meilleure?

Objectif ultime  
La meilleure méthode  
est celle avec  
 $TFP=TFN=0$



57

---

---

---

---

---

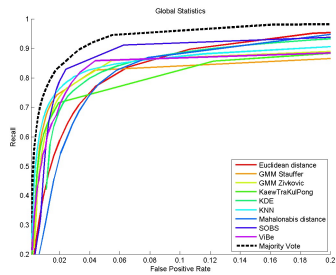
---

---

---

## Évaluation

Exemple : 10 méthodes de détection de mouvement



58

---

---

---

---

---

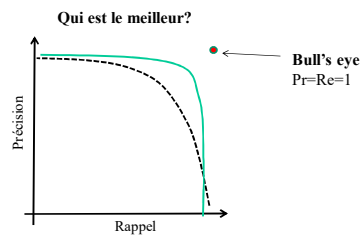
---

---

---

## Évaluation

Courbe de précision-rappel : similaire à la courbe ROC



59

---

---

---

---

---

---

---

---