

Techniques d'apprentissage
IFT 603 / IFT 712

Combinaison de modèles

Par
Pierre-Marc Jodoin
/
Hugo Larochelle

1

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 appelé **ensemble** ou **comité**

2

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

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

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

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**

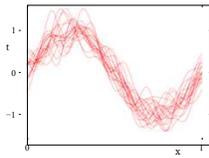
6

6

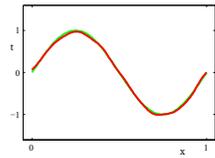
Bootstrap : réduction de la variance

- Régression polynomiale de degré 25

100 modèles entraînés sur
100 ensembles d'entraînement différents



Ensemble des 100 modèles vs. vrai modèle



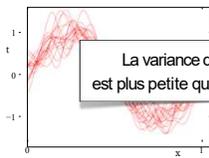
7

7

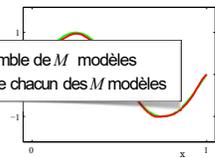
Bootstrap : réduction de la variance

- 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

8

Bootstrap

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

Solution : **Bootstrapping**.

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

Pour N itérations

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

retourner $D_{bootstrap}$

9

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

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. **Combiner** les m modèles

Régression: $y_{com}(\tilde{x}) = \frac{1}{m} \sum_{i=1}^m y_{w_i}(\tilde{x})$

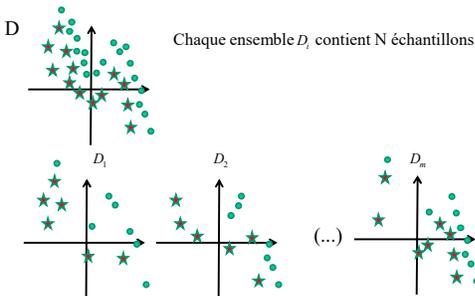
Classification: $y_{com}(\tilde{x}) = \text{sign} \left(\sum_{i=1}^m y_{w_i}(\tilde{x}) \right)$ (2 classes)
 $y_{com}(\tilde{x}) = \arg \max_c \left(\sum_{i=1}^m y_{w_i}(\tilde{x}) \right)$ (K classes)

11

11

Illustration: classification 2Classes

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



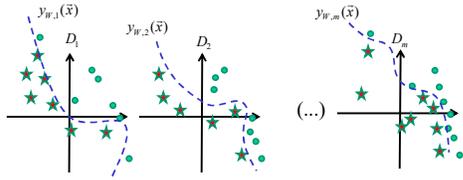
Note: un échantillon \tilde{x}_j peut apparaître plusieurs fois dans un même ensemble d'entraînement D_i

12

Illustration: classification 2Classes

2- Entraînement

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



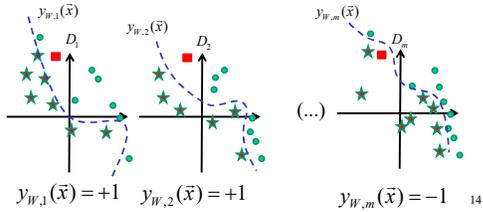
13

13

Illustration: classification 2Classes

3- Vote majoritaire

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



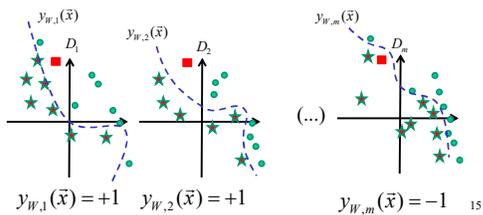
14

14

Illustration: classification 2Classes

3- Vote majoritaire

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



15

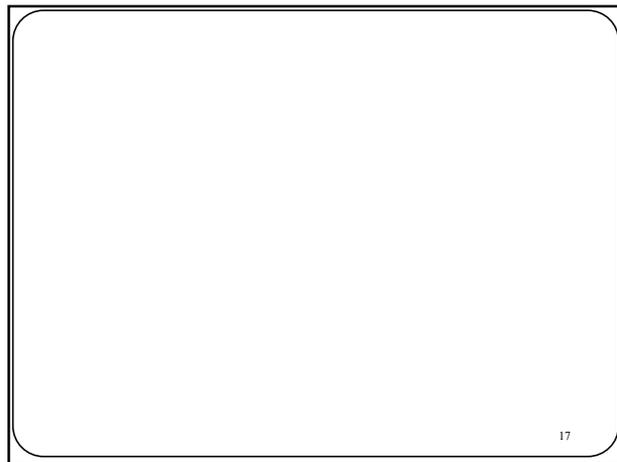
15

Bagging

Analyse théorique de l'erreur : au tableau

16

16



17

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

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. Implémente 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 $y_{w,i+1}(\vec{x})$

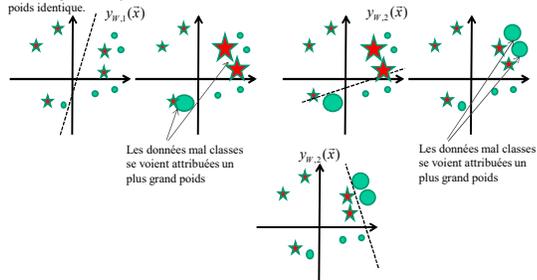
R. Schapire and Y. Freund A decision theoretic generalization of on-line learning and an application to Boosting Journal of Computer and System Sciences, 1997, 55: 119-139. 19

19

AdaBoost

(illustration 2 classes)

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



20

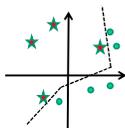
20

AdaBoost

(illustration 2 classes)

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

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



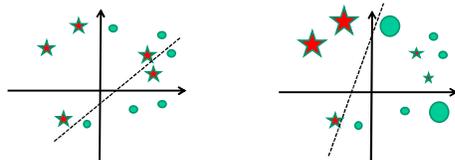
NOTE: plus un classifieur a une **exactitude élevée**, plus son α_i sera élevé. 21

21

AdaBoost

Idée fondamentale: chaque donnée \vec{x}_i a un poids β_i

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



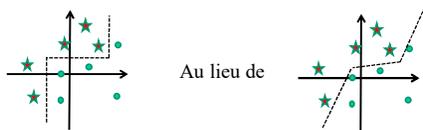
Poids égaux Vs poids non égaux

22

22

AdaBoost

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



Au lieu de

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

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

Les caractéristiques utiles sont celles pour lesquelles α_i est élevé

23

Adaboost

1- initialiser les 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: ϵ_i

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

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

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

$$\beta_i = \beta_i \exp\{\alpha_i\}$$

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

$$\beta_i = \frac{\beta_i}{\sum_j \beta_j}$$

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

24

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: ϵ_i

$$\epsilon_i = \frac{\sum_k \beta_k}{\sum_k \beta_k} \quad \text{où } \Psi \text{ l'ensemble de données}$$
- 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 sommeent à 1

$$\beta_i = \frac{\beta_i}{\sum_j \beta_j}$$

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

Plus un classifieur a une erreur faible, plus son poids α_i sera élevé

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

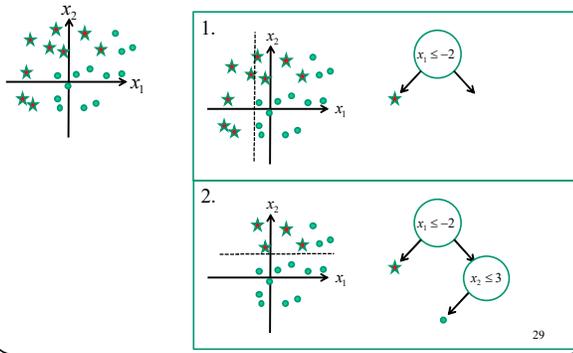
Arbres de décision

28

28

Arbres de décision

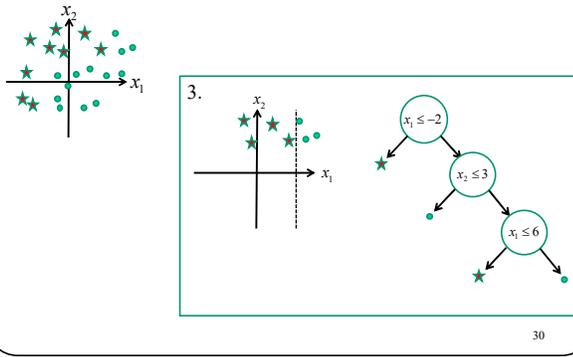
Exemple simple avec des classifieurs *stump*



29

Arbres de décision

Exemple simple avec des classifieurs *stump*



30

Arbres de décision

Exemple simple avec des classifieurs *stump*

Au final

31

31

Arbres de décision

Exemple simple avec des classifieurs *stump*

Au final

32

32

Arbres de décision

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

Exemple avec deux données aberrantes:

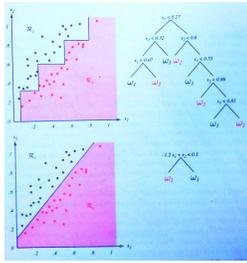
Grosse question : soit le noeud d'un arbre dont l'erreur d'entraînement n'est pas nulle **devons-nous le subdiviser ou non?**

33

33

Arbres de décision

Les arbres de décision peuvent également inclure des classificateurs linéaires plus généraux

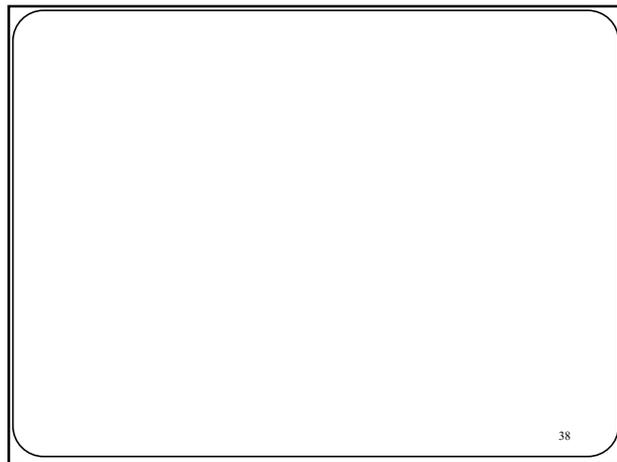


Credit, Duda Hart.

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

37

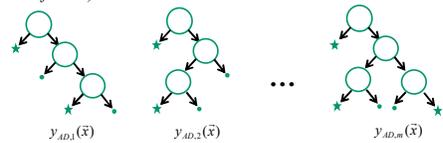


38

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

39

39

Entraînement d'une FAD

FOR $i=1$ to m

→ Sélectionner au hasard N données $D_i = \{(\bar{x}_1, t_1), (\bar{x}_2, t_2), \dots, (\bar{x}_N, t_N)\}$, $\bar{x}_i \in \mathbb{R}^d$

*Sélectionner au hasard P dimensions $\Rightarrow \hat{D}_i = \{(\bar{x}_1, t_1), (\bar{x}_2, t_2), \dots, (\bar{x}_N, t_N)\}$, $\bar{x}_i \in \mathbb{R}^p, p < d$

Entraîner un arbre de décisions $y_{AD,i}(\bar{x})$ avec \hat{D}_i

Bootstrapping

Généralisation

Soit une nouvelle donnée $\bar{x} \in \mathbb{R}^d$

FOR $i=1$ to m

$C_i = y_{AD,i}(\bar{x})$

$class = VoteMajoritaire(C_1, C_2, \dots, C_m)$

* Afin que les **erreurs** produites par les AD soient les **moins corrélées** possible.

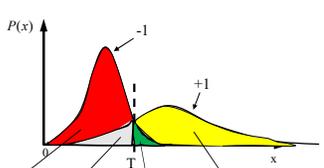
40

Métriques d'évaluation

41

41

Évaluation



Taux de vrais négatifs $\int_{-\infty}^T P(x|-1)dx$

Taux de faux négatifs $\int_{-\infty}^T P(x|+1)dx$

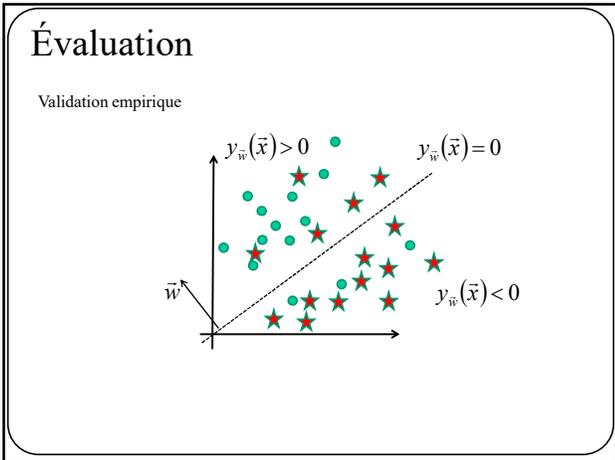
Taux de faux positifs $\int_T^{\infty} P(x|-1)dx$

Taux de vrais positifs $\int_T^{\infty} P(x|+1)dx$

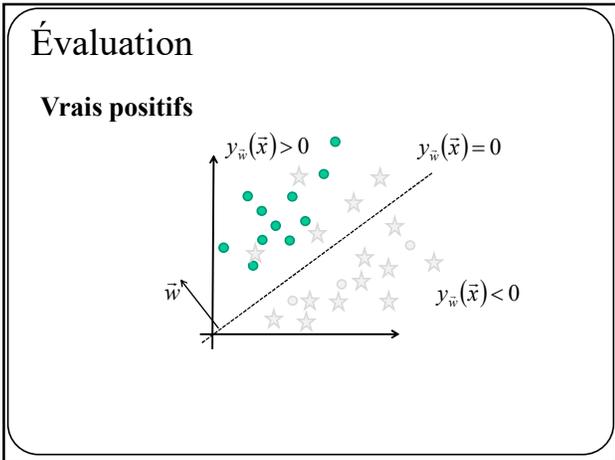
Matrice de confusion

		Vérité terrain		TVN + TFP = 1 TVP + TFN = 1
		Positifs	Négatifs	
Sortie du modèle	Positifs	TVP	TFP	TVN + TFP = prop des données dans la classe -1 TVP + TFN = prop des données dans la classe 1
	Négatifs	TFN	TVN	

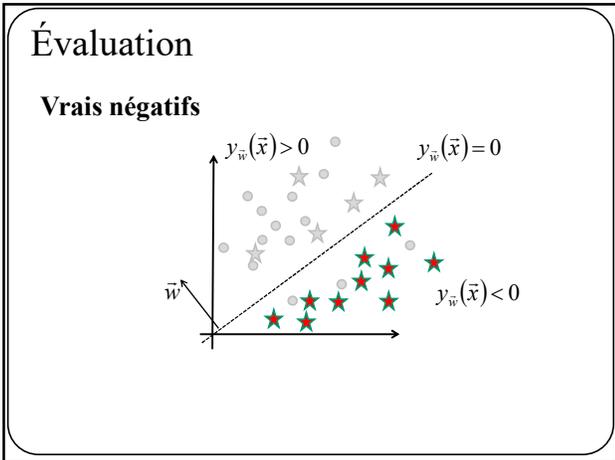
42



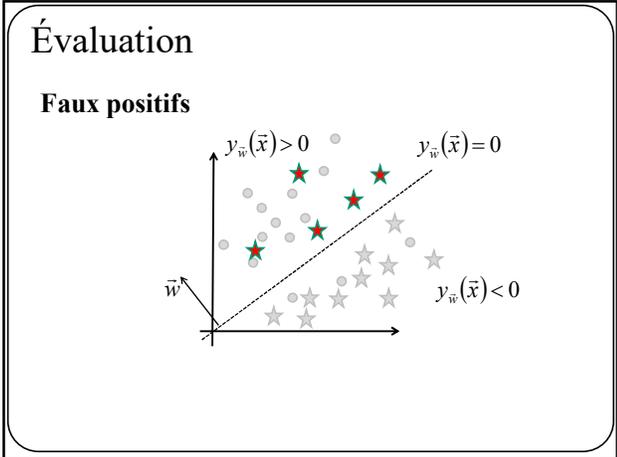
43



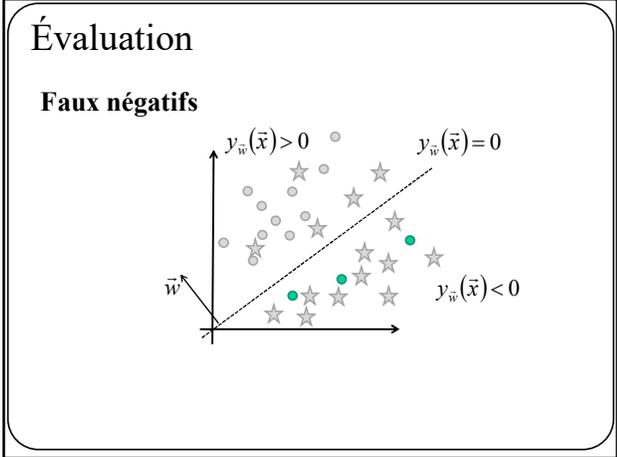
44



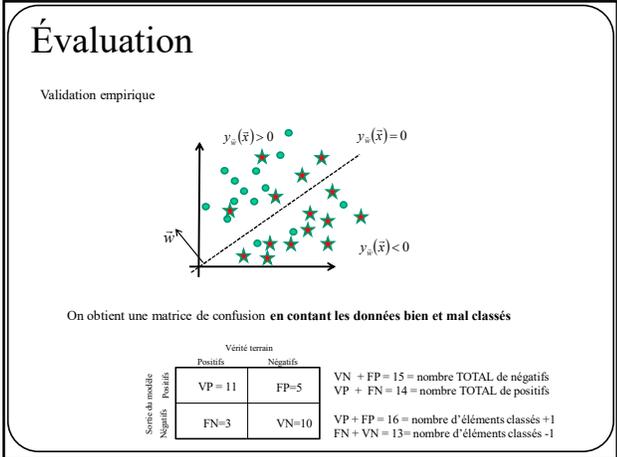
45



46



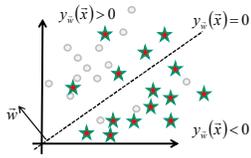
47



48

Évaluation

Validation empirique



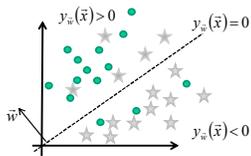
$$\text{Taux de faux positifs (TFP)} = \text{FP}/(\text{FP}+\text{VN}) = 5/15$$

49

49

Évaluation

Validation empirique



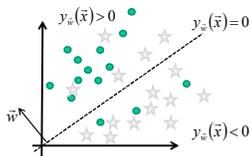
$$\text{Taux de faux négatifs (FNR)} = \text{FN}/(\text{FN}+\text{VP}) = 3/14$$

50

50

Évaluation

Validation empirique



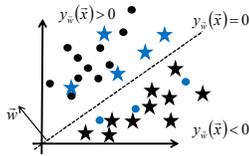
$$\text{Rappel (Recall - Re)} = \text{VP}/(\text{FN}+\text{VP}) = 1 - \text{TFN} = 11/14$$

51

51

Évaluation

Validation empirique



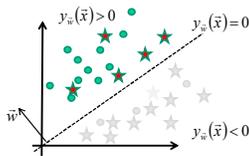
Justesse (« accuracy », **prop. de biens classés**) = $(TP+TN)/(FP+FN+TP+TN) = 21/29$

52

52

Évaluation

Validation empirique



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

53

53

Évaluation

		Vérité terrain	
		Positifs	Négatifs
Sortie du modèle Positif / Négatif	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**) = $VN/(FP+VN) = 10/15$
 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

54

Évaluation

Différents seuils = différents résultats

TFP: élevé
TFN: faible

TFP: méd
TFN: méd

TFP: faible
TFN: élevé

55

55

Évaluation

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

56

56

Validation

Courbes ROC: très bonne façon de comparer des méthodes

Quelle méthode est la meilleure?

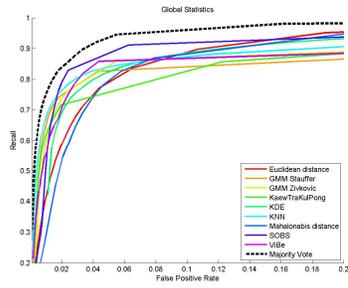
Objectif ultime
La meilleure méthode est celle avec TFP=TFN=0

57

57

Évaluation

Exemple : 10 méthodes de détection de mouvement

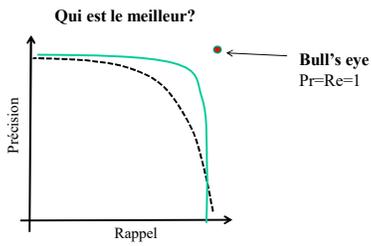


58

58

Évaluation

Courbe de précision-rappel : similaire à la courbe ROC



59

59
