

Apprentissage supervisé

Master parcours SSD - UE Apprentissage Statistique I

Pierre Mahé - bioMérieux & Université de Grenoble-Alpes

1. Introduction - formalisation
2. Validation croisée
3. Critères de performance de classification
4. Courbe ROC
5. Algorithme des k plus proches voisins (k -PPV)

Introduction

Apprentissage supervisé - principe

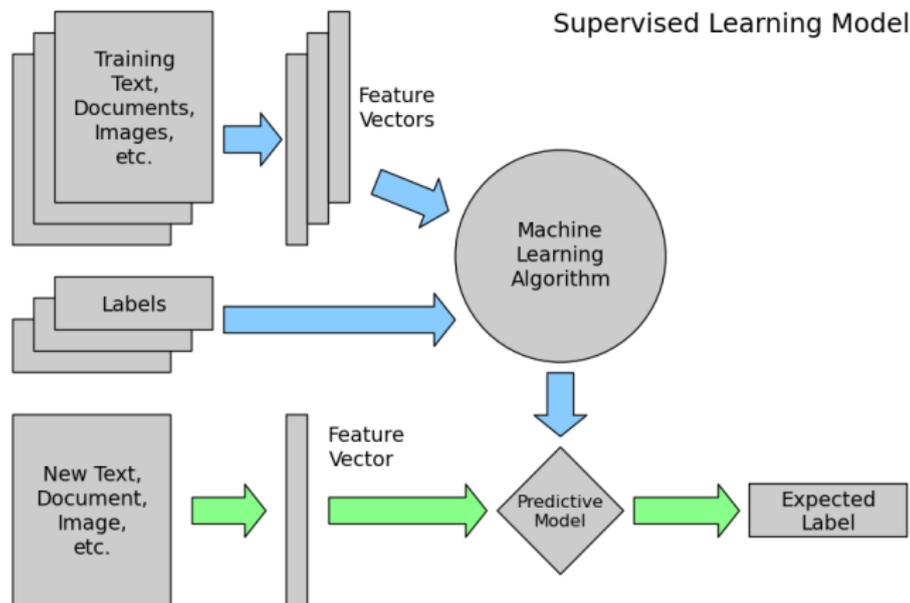


Figure: Image tirée de http://www.astroml.org/sklearn_tutorial/general_concepts.html

Apprentissage supervisé - exemples

► Catégorisation d'images



(a) Positive examples of 'whale'

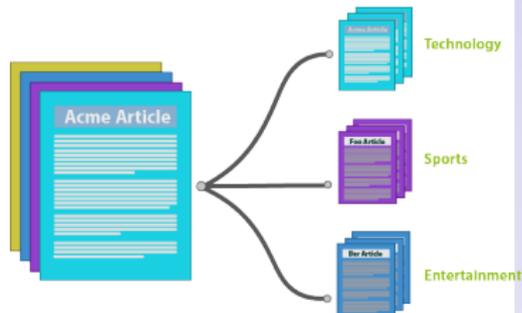


(b) Negative examples of 'whale' obtained by random sampling



(c) Relevant negatives of 'whale' (this paper)

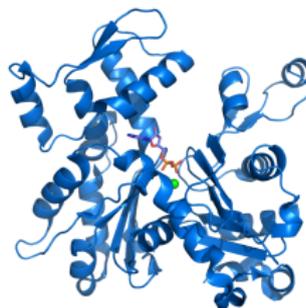
► Catégorisation de textes



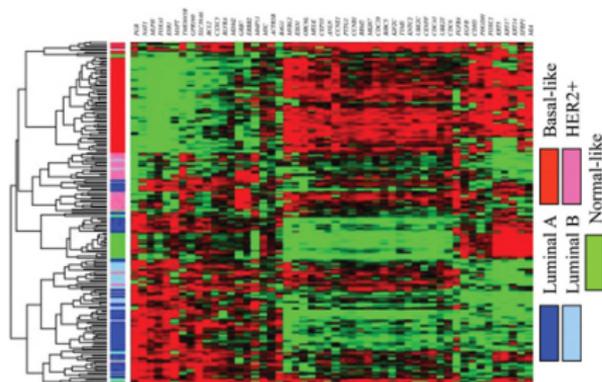
Apprentissage supervisé - exemples

- ▶ Prédire la fonction d'une protéine à partir de sa structure

MKLT LKNLSMAIMMSIVMGS
SAMAADSNEK...GAS
GYLPEHTLF...
ADYLEQD...
LHDHYLD...
DRARKDG...
DEIKSLKF...
QTYPGRFPMG...
HTFEEIEFVQGLNHSTGR
NIGIYPEIKAPWPHQEGKDI
AAKTLEVLKKYGYTGKDDKV



- ▶ Diagnostic/prognostic à partir de puces à ADN



On dispose d'un échantillon $\{(x_i, y_i)\}$, $i = 1, \dots, n$, :

- ▶ des **observations** $x_i \in \mathcal{X}$,
- ▶ des **réponses** associées $y_i \in \mathcal{Y}$.

On dispose d'un échantillon $\{(x_i, y_i)\}$, $i = 1, \dots, n$, :

- ▶ des **observations** $x_i \in \mathcal{X}$,
- ▶ des **réponses** associées $y_i \in \mathcal{Y}$.

Typiquement :

- ▶ $\mathcal{X} = \mathbb{R}^p$: on parle de vecteurs de descripteurs (**features**)
- ▶ Si $\mathcal{Y} = \mathbb{R}$, on parle de **régression**.
- ▶ Si $\mathcal{Y} = \{1, \dots, K\}$, on parle de **classification**
- ▶ Si $\mathcal{Y} = \{-1, +1\}$, on parle de **classification binaire**
 - ▶ on note parfois également $\mathcal{Y} = \{0, 1\}$

Apprentissage supervisé - formalisation

Données d'entrée : échantillon $\{(x_i, y_i)\}_{i=1, \dots, n} \in \mathcal{X} \times \mathcal{Y}$.

Objectif : apprendre une fonction $f : \mathcal{X} \rightarrow \mathcal{Y}$ permettant de prédire la réponse associée à une nouvelle observation.

Apprentissage supervisé - formalisation

Données d'entrée : échantillon $\{(x_i, y_i)\}_{i=1, \dots, n} \in \mathcal{X} \times \mathcal{Y}$.

Objectif : apprendre une fonction $f : \mathcal{X} \rightarrow \mathcal{Y}$ permettant de prédire la réponse associée à une nouvelle observation.

Critère : une fonction de perte L (pour "loss") mesurant l'erreur entre y et $f(x)$.

Apprentissage supervisé - formalisation

Données d'entrée : échantillon $\{(x_i, y_i)\}_{i=1, \dots, n} \in \mathcal{X} \times \mathcal{Y}$.

Objectif : apprendre une **fonction** $f : \mathcal{X} \rightarrow \mathcal{Y}$ permettant de **prédire** la réponse associée à une **nouvelle observation**.

Critère : une **fonction de perte** L (pour "loss") mesurant l'erreur entre y et $f(x)$.

Typiquement :

- ▶ l'**erreur quadratique** pour la **régression** :

$$L(y, f(x)) = (y - f(x))^2$$

- ▶ le **coût 0/1** pour la **classification** :

$$L(y, f(x)) = \mathbb{1}(y \neq f(x))$$

Apprentissage supervisé - formalisation

Cadre probabiliste : on considère que nos observations (x_i, y_i) sont des variables aléatoires régies par une **loi jointe** $P(X, Y)$.

Cadre probabiliste : on considère que nos observations (x_i, y_i) sont des variables aléatoires régies par une **loi jointe** $P(X, Y)$.

⇒ L'objectif de l'apprentissage supervisé est donc de trouver la fonction f minimisant **l'espérance de la fonction de perte** :

$$R(f) = E_{X,Y}[L(Y, f(X))],$$

à partir d'un échantillon $\{(x_i, y_i)\}, i = 1, \dots, n$.

Cadre probabiliste : on considère que nos observations (x_i, y_i) sont des variables aléatoires régies par une **loi jointe** $P(X, Y)$.

⇒ L'objectif de l'apprentissage supervisé est donc de trouver la fonction f minimisant **l'espérance de la fonction de perte** :

$$R(f) = E_{X,Y} [L(Y, f(X))],$$

à partir d'un échantillon $\{(x_i, y_i)\}, i = 1, \dots, n$.

$R(f)$ est appelée le **risque** (ou la **perte**) de la fonction f .

Apprentissage supervisé - risque empirique

A minima, un "bon" prédicteur devrait bien se comporter sur les données d'apprentissage.

Apprentissage supervisé - risque empirique

A minima, un "bon" prédicteur devrait bien se comporter sur les données d'apprentissage.

On s'intéresse donc en premier lieu au **risque empirique** :

$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)).$$

Apprentissage supervisé - risque empirique

A minima, un "bon" prédicteur devrait bien se comporter sur les données d'apprentissage.

On s'intéresse donc en premier lieu au **risque empirique** :

$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)).$$

Mais minimiser le risque empirique n'est pas suffisant, il faut également contrôler la **complexité du modèle** pour éviter le **sur-apprentissage**.

Risque empirique et sur-apprentissage

Illustration de **sous-apprentissage** et **sur-apprentissage** sur un problème (jouet) de régression :

- ▶ le risque empirique décroît de gauche à droite

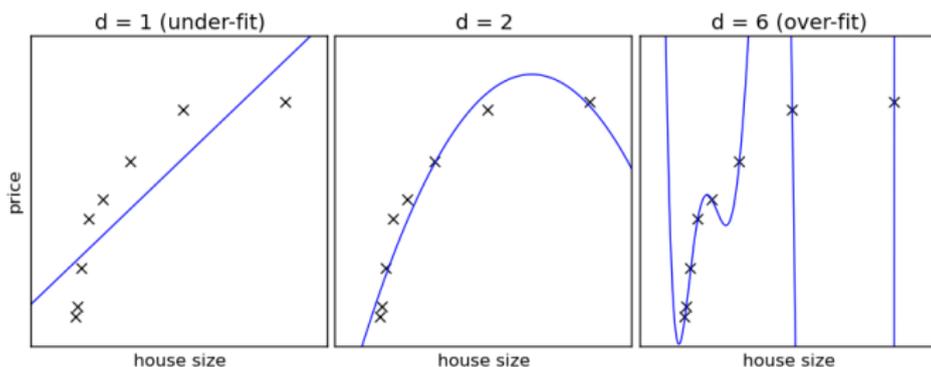


Figure: Image tirée de http://www.astroml.org/sklearn_tutorial/practical.html

Généralisation vs complexité du modèle

Une question clé : trouver le bon niveau de **complexité du modèle** pour éviter le **sous-** et le **sur-apprentissage**.

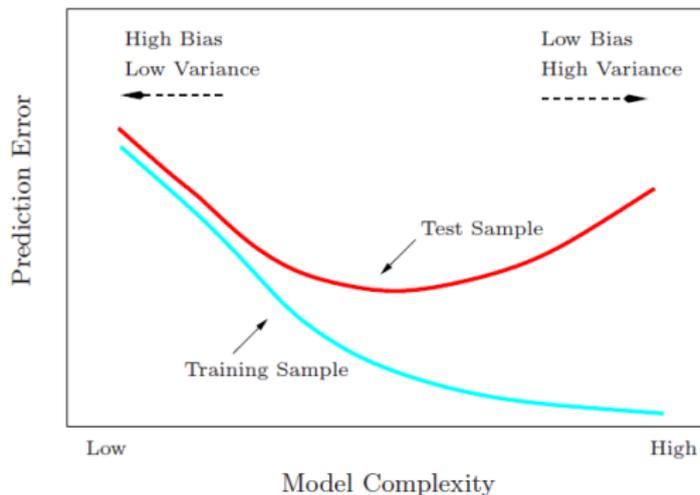


Figure: Image tirée de Hastie et al. (2001) (Fig.2.11)

Complexité d'un modèle ?

Les algorithmes d'apprentissage supervisé mettent en jeu un (voire des) **paramètre(s)** permettant de construire des modèles plus ou moins "complexes".

Complexité d'un modèle ?

Les algorithmes d'apprentissage supervisé mettent en jeu un (voire des) **paramètre(s)** permettant de construire des modèles plus ou moins "complexes".

Par exemple :

Complexité d'un modèle ?

Les algorithmes d'apprentissage supervisé mettent en jeu un (voire des) **paramètre(s)** permettant de construire des modèles plus ou moins "complexes".

Par exemple :

- ▶ **régression polynomiale** $f(x) = \beta_0 + \sum_{i=1}^d \beta_i x^i$: degré maximal d du polynôme.

Complexité d'un modèle ?

Les algorithmes d'apprentissage supervisé mettent en jeu un (voire des) **paramètre(s)** permettant de construire des modèles plus ou moins "complexes".

Par exemple :

- ▶ **régression polynomiale** $f(x) = \beta_0 + \sum_{i=1}^d \beta_i x^i$: degré maximal d du polynôme.
- ▶ **modèles linéaires multivariés** $f(x) = \sum_{i=1}^p w_i x_i$, où $x \in \mathbb{R}^p$: nombres de variables à inclure dans le modèle

Complexité d'un modèle ?

Les algorithmes d'apprentissage supervisé mettent en jeu un (voire des) **paramètre(s)** permettant de construire des modèles plus ou moins "complexes".

Par exemple :

- ▶ **régression polynomiale** $f(x) = \beta_0 + \sum_{i=1}^d \beta_i x^i$: degré maximal d du polynôme.
- ▶ **modèles linéaires multivariés** $f(x) = \sum_{i=1}^p w_i x_i$, où $x \in \mathbb{R}^p$: nombres de variables à inclure dans le modèle
- ▶ **algorithme des k plus proches voisins** : valeur de k

[Introduction](#)[Validation
Croisée](#)[Critères de
performance](#)[Courbe ROC](#)[k-PPV](#)[Conclusion](#)[Références](#)

Complexité d'un modèle ?

Les algorithmes d'apprentissage supervisé mettent en jeu un (voire des) **paramètre(s)** permettant de construire des modèles plus ou moins "complexes".

Par exemple :

- ▶ **régression polynomiale** $f(x) = \beta_0 + \sum_{i=1}^d \beta_i x^i$: degré maximal d du polynôme.
- ▶ **modèles linéaires multivariés** $f(x) = \sum_{i=1}^p w_i x_i$, où $x \in \mathbb{R}^p$: nombres de variables à inclure dans le modèle
- ▶ **algorithme des k plus proches voisins** : valeur de k
- ▶ **forêts aléatoires** : nombre d'arbres mis en jeu

En pratique, il est très **difficile de savoir a priori** quelles valeurs utiliser pour ces paramètres.

[Introduction](#)[Validation
Croisée](#)[Critères de
performance](#)[Courbe ROC](#)[k-PPV](#)[Conclusion](#)[Références](#)

Estimation de performance et validation croisée

Le paradigme train/test

A partir du jeu de données on doit résoudre **deux problèmes** :

1. trouver le **bon niveau de complexité** du modèle
 - ▶ risque empirique et sous/sur-apprentissage
2. estimer ses **performances de généralisation**
 - ▶ performances sur de nouvelles données

Le paradigme train/test

A partir du jeu de données on doit résoudre **deux problèmes** :

1. trouver le **bon niveau de complexité** du modèle
 - ▶ risque empirique et sous/sur-apprentissage
2. estimer ses **performances de généralisation**
 - ▶ performances sur de nouvelles données

Paradigme de l'apprentissage supervisé :

- ▶ **données d'apprentissage** pour construire le modèle
- ▶ **données de test** pour évaluer les performances

Le paradigme train/test

A partir du jeu de données on doit résoudre **deux problèmes** :

1. trouver le **bon niveau de complexité** du modèle
 - ▶ risque empirique et sous/sur-apprentissage
2. estimer ses **performances de généralisation**
 - ▶ performances sur de nouvelles données

Paradigme de l'apprentissage supervisé :

- ▶ **données d'apprentissage** pour construire le modèle
- ▶ **données de test** pour évaluer les performances

Attention : données de test uniquement utilisées à la toute fin pour évaluer les performances du modèle final

- ▶ n'interviennent **jamais** dans la construction du modèle

Pour optimiser la complexité du modèle :

- ▶ besoin d'estimer les performances de généralisation
- ▶ mais sans faire appel aux données de test

Pourquoi ?

- ▶ les données de test ne permettent que d'**estimer** l'erreur de généralisation
 - ▶ indicateurs de performance + intervalles de confiance
- ▶ optimiser le modèle pour maximiser les performances sur **CE jeu de test** serait une forme de sur-apprentissage !
 - ▶ et serait donc **optimiste**

Contrôler la complexité du modèle

Première solution : découpage train / validation / test :



1. **train** : pour apprendre les différents modèles
2. **validation** : pour les évaluer et retenir le meilleur
3. **test** : pour estimer ses performances

⇒ situation optimale - "data rich"

Deuxième solution : validation-croisée

Validation croisée

Si **peu de données** : délicat de découper train/validation

- ▶ forte incertitude sur l'estimation des performances

Principe de la **validation croisée** :

- ▶ **découper** le jeu d'apprentissage en K parties - les **folders**
 - ▶ les données de test sont toujours de côté
- ▶ pour $k = 1, \dots, K$:
 - ▶ fold k = données de validation
 - ▶ autres folders = données d'apprentissage

	train	
Fold 1	validation1	train1
Fold 2	train2	validation2
Fold 3	train3	validation3
Fold 4	train4	validation4
Fold 5	train5	validation5

⇒ si on prend $K = n$ on parle de **leave one out**

Pseudo-code :

1. Définir les K folds de validation croisée
 - ▶ en pratique : un vecteur de longueur n avec des valeurs entre 1 et K affectant les n observations aux K folds
2. Pour $k = 1$ à K :
 - 2.1 mettre de côté la k -ième fold
 - 2.2 apprendre le modèle sur les $(K - 1)$ folds restantes
 - 2.3 appliquer le modèle sur les données de la k -ième fold
3. Evaluer les performances du modèle en comparant les valeurs réelles et prédites.
 - ▶ estimation globale ou par fold

Validation croisée et sélection de modèle

La validation croisée est notamment utile pour choisir le **meilleur modèle** entre plusieurs modèles candidats.

- ▶ e.g., des modèles + ou - complexes

Pseudo-code :

1. Définir un ensemble de modèles candidats
 - ▶ régression polynomiale : différents degrés de polynôme
 - ▶ k -PPV : différentes valeurs de k
 - ▶ ...
2. Pour chaque modèle :
 - 2.1 Appliquer la procédure de validation croisée
 - 2.2 Enregistrer les performances de prédiction
3. Choisir le meilleur modèle.

En pratique : choisir le nombre de folds

Impact du **nombre de folds** :

- ▶ **K élevé** = beaucoup de points pour l'apprentissage
 - ▶ construction de meilleurs modèles
- ▶ **K faible** = beaucoup de données pour le test
 - ▶ meilleure évaluation des performances

En pratique : choisir le nombre de folds

Impact du **nombre de folds** :

- ▶ **K élevé** = beaucoup de points pour l'apprentissage
 - ▶ construction de meilleurs modèles
- ▶ **K faible** = beaucoup de données pour le test
 - ▶ meilleure évaluation des performances

⇒ la "bonne" valeur de K dépend de la complexité du problème, qu'on ne connaît pas !

- ▶ ("bon" = permet d'estimer au mieux les performances)

En pratique : choisir le nombre de folds

Impact du **nombre de folds** :

- ▶ **K élevé** = beaucoup de points pour l'apprentissage
 - ▶ construction de meilleurs modèles
- ▶ **K faible** = beaucoup de données pour le test
 - ▶ meilleure évaluation des performances

⇒ la "bonne" valeur de K dépend de la complexité du problème, qu'on ne connaît pas !

- ▶ ("bon" = permet d'estimer au mieux les performances)

En général $K = 10$ ou $K = 5$, selon le jeu de données.

- ▶ jeu petit : K élevé pour maintenir un nombre d'observations suffisant pour l'apprentissage
- ▶ jeu conséquent : on peut être tenté de diminuer K

Il est en général recommandé :

- ▶ de construire les folds de manière **stratifiée**, i.e., de respecter les proportions relatives des différentes classes au sein des folds.
- ▶ d'effectuer **plusieurs répétitions** de la procédure de validation croisée, pour être robuste aux aléas de la définition des folds.
- ▶ de considérer **plusieurs indicateurs** de performance (pour la classification notamment).

Deux remarques importantes :

- ▶ procédure valide si les données d'apprentissage et de test sont **indépendantes et identiquement distribuées** (iid)
 - ▶ attention aux modifications / dérives "cachées"

Deux remarques importantes :

- ▶ procédure valide si les données d'apprentissage et de test sont **indépendantes et identiquement distribuées** (iid)
 - ▶ attention aux modifications / dérives "cachées"
- ▶ le jeu de test ne permet que d'**estimer** l'erreur de généralisation
 - ▶ indicateurs de performance + intervalles de confiances

Critères de performance de prédiction

Critères de performance de prédiction

Mesure de performance les plus simples (et classiques) :

- ▶ **régression** : erreur quadratique moyenne (MSE) :

$$\text{MSE}(f) = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

(à minimiser)

- ▶ **classification** : taux de bonne classification (accuracy) :

$$\text{Acc}(f) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}(y_i = f(x_i))$$

(à maximiser)

⇒ liées aux deux fonctions de perte évoquées précédemment.

Critères de performance de prédiction

En pratique, la notion de meilleur modèle peut être dictée par l'application

- ▶ classification \Rightarrow éviter certains types d'erreur
- ▶ e.g., contexte médical, ne pas déclarer un malade sain

Critères de performance de prédiction

En pratique, la notion de meilleur modèle peut être dictée par l'application

- ▶ classification \Rightarrow éviter certains types d'erreur
- ▶ e.g., contexte médical, ne pas déclarer un malade sain

Pour la classification binaire, on travaille souvent à partir de la matrice de confusion

- ▶ table de contingence valeurs réelles / valeurs prédites

		Prédiction	
		+	-
Réalité	+	TP	FN
	-	FP	TN

- ▶ TP = True Positive
- ▶ TN = True Negative
- ▶ FP = False Positive
- ▶ FN = False Negative

Critères de performance de prédiction

Matrice de confusion et critères de performance :

		Prédiction	
		+	-
Réalité	+	TP	FN
	-	FP	TN

- ▶ TP = True Positive
- ▶ TN = True Negative
- ▶ FP = False Positive
- ▶ FN = False Negative

- ▶ **accuracy** = $(TP + TN) / n$
 - ▶ taux de bonne classification global
- ▶ **sensibilité** (sensitivity) = $TP / (TP + FN)$
 - ▶ taux de bonne classification des instances positives
- ▶ **spécificité** (specificity) = $TN / (TN + FP)$
 - ▶ taux de bonne classification des instances négatives

Critères de performance de prédiction

Matrice de confusion et critères de performance :

		Prédiction	
		+	-
Réalité	+	TP	FN
	-	FP	TN

- ▶ TP = True Positive
- ▶ TN = True Negative
- ▶ FP = False Positive
- ▶ FN = False Negative

- ▶ **valeur prédictive positive (VPP)** $= TP / (TP + FP)$
 - ▶ taux d'instances positives dans les prédictions positives
- ▶ **valeur prédictive négative (VPN)** $= TN / (TN + FN)$
 - ▶ taux d'instances négatives dans les prédictions négatives

Critères de performance de prédiction

Indicateurs utilisés en **recherche d'information** :

		Prédiction	
		+	-
Réalité	+	TP	FN
	-	FP	TN

▶ TP = True Positive

▶ TN = True Negative

▶ FP = False Positive

▶ FN = False Negative

- ▶ **précision** = $TP / (TP + FP)$
 - ▶ proportion de "documents" au sein des résultats
 - ▶ taux de vraies positives au sein des prédictions positives.

⇒ équivalent à la **Valeur Prédictive Positive**

- ▶ **rappel** (recall) = $TP / (TP + FN)$
 - ▶ proportion de "documents" retrouvés
 - ▶ taux de bonne classification des instances positives

⇒ équivalent à la **sensibilité**

Introduction

Validation
Croisée

Critères de
performance

Courbe ROC

k -PPV

Conclusion

Références

Analyse ROC

Taux de bonne classification et **classes déséquilibrées** :

		Prédiction	
		+	-
Réalité	+	0	1
	-	0	99

⇒ **modèle "nul"** : prédit la classe majoritaire

⇒ **accuracy** = 99%

Taux de bonne classification et **classes déséquilibrées** :

		Prédiction	
		+	-
Réalité	+	0	1
	-	0	99

⇒ **modèle "nul"** : prédit la classe majoritaire

⇒ **accuracy** = 99%

Dans cet exemple :

▶ **sensibilité** = 0%

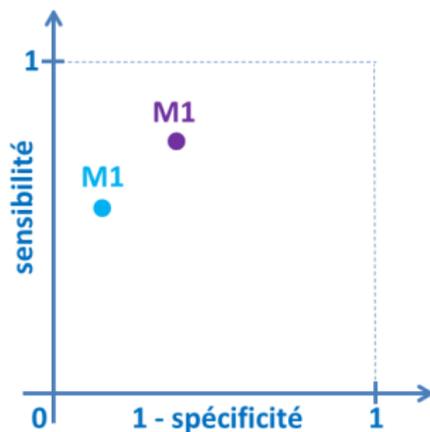
▶ **spécificité** = 100%

⇒ meilleur reflet des performances de prédiction...

...si prises **ensemble** (et pas séparément)

Espace ROC

- ▶ Cadre de la **classification binaire**
- ▶ Représentation **sensibilité = f(1 - spécificité)**
- ▶ **1 modèle = 1 point** dans l'espace ROC
 - ▶ modèle \Leftrightarrow matrice de confusion



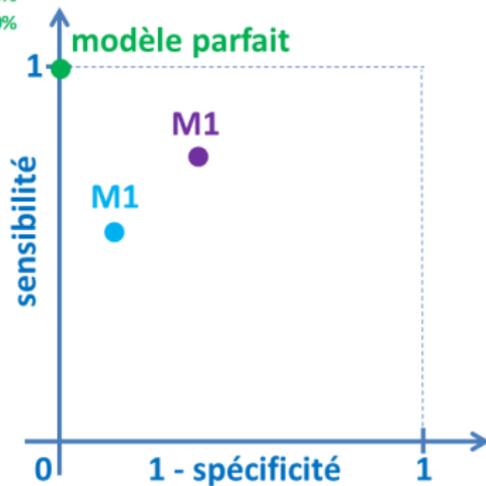
\Rightarrow taux de **vrai positifs** en fonction du taux de **faux positifs**

Espace ROC - points remarquables

	1	-1
+1	100	0
-1	0	100

VPP = 100%
VPN = 100%

sensi = 100%
speci = 100%

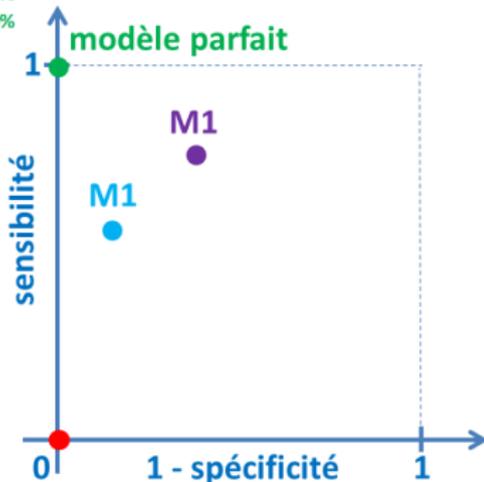


Espace ROC - points remarquables

	1	-1
+1	100	0
-1	0	100

sensi = 100%
speci = 100%

VPP = 100%
VPN = 100%



**Prédictions toujours
négatives**

	1	-1
+1	0	100
-1	0	100

sensi = 0%
speci = 100%

VPP = 0%
VPN = 50%

Espace ROC - points remarquables

	1	-1
+1	100	0
-1	0	100

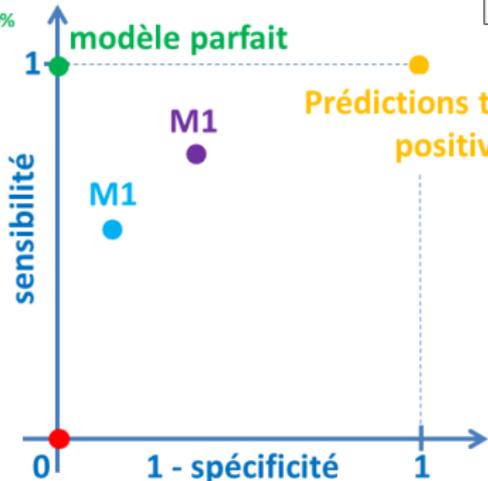
sensi = 100%
speci = 100%

VPP = 100%
VPN = 100%

	1	-1
+1	100	0
-1	100	0

sensi = 100%
speci = 0%

VPP = 50%
VPN = 0%



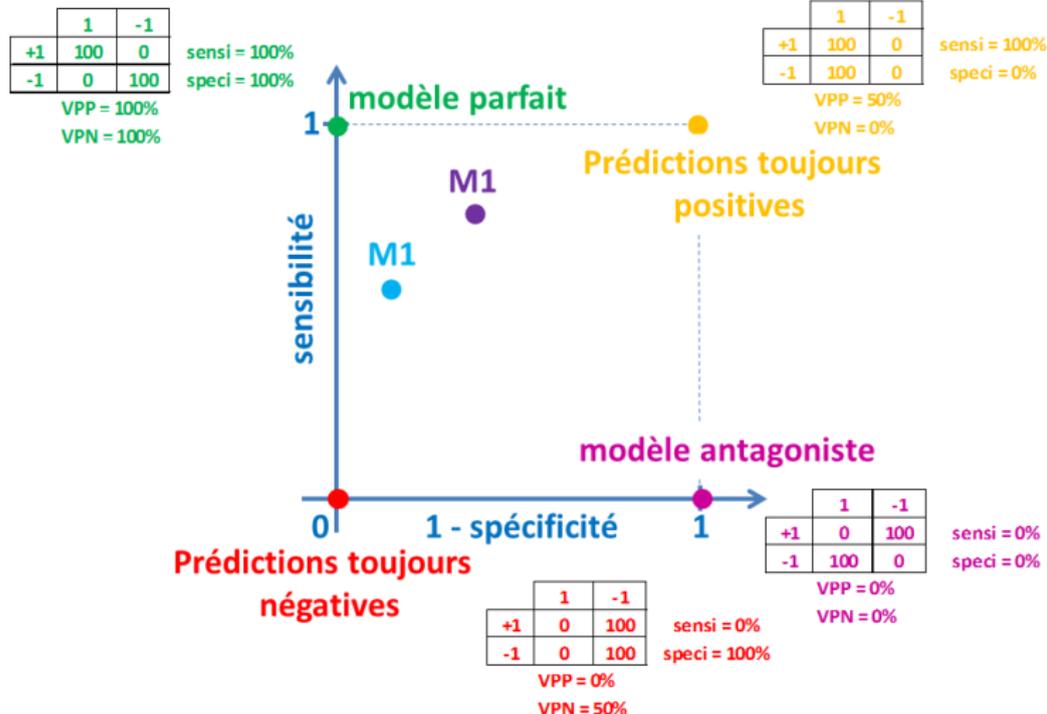
Prédictions toujours
négatives

	1	-1
+1	0	100
-1	0	100

sensi = 0%
speci = 100%

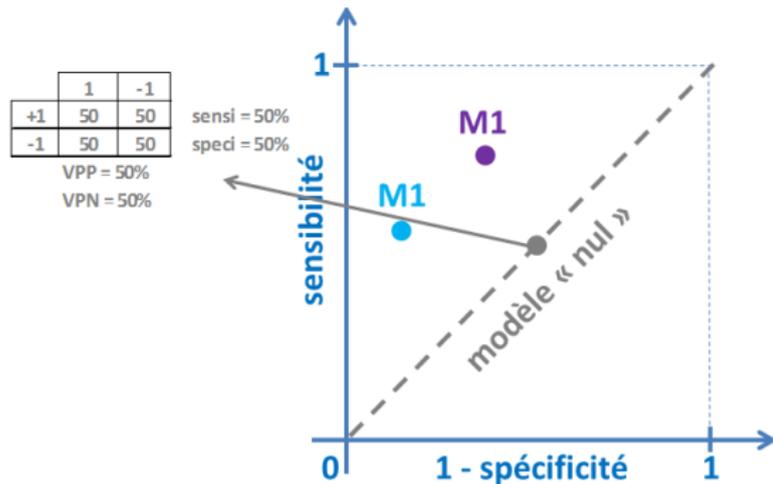
VPP = 0%
VPN = 50%

Espace ROC - points remarquables



Espace ROC - modèle aléatoire

Diagonale de l'espace ROC = modèle aléatoire

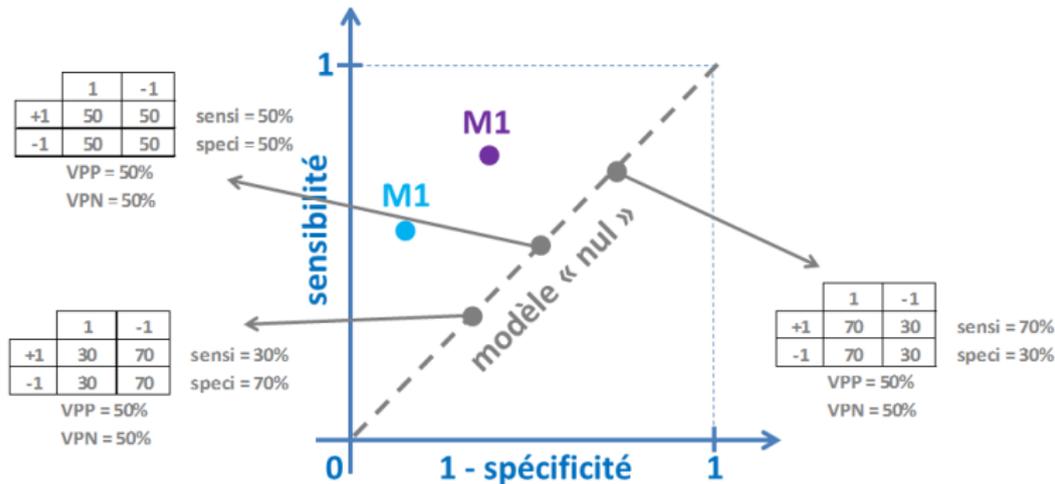


⇒ si $\text{sensi} = 1 - \text{speci}$ le modèle n'a pas de valeur prédictive

► i.e., si $\text{sensi} + \text{speci} = 1$

Espace ROC - modèle aléatoire

Diagonale de l'espace ROC = modèle aléatoire



⇒ si $\text{sensi} = 1 - \text{speci}$ le modèle n'a pas de valeur prédictive

► i.e., si $\text{sensi} + \text{speci} = 1$

Espace ROC et comparaison de modèles

Comment choisir le meilleur modèle ?

Outline

Apprentissage
Statistique I

Introduction

Validation
Croisée

Critères de
performance

Courbe ROC

k -PPV

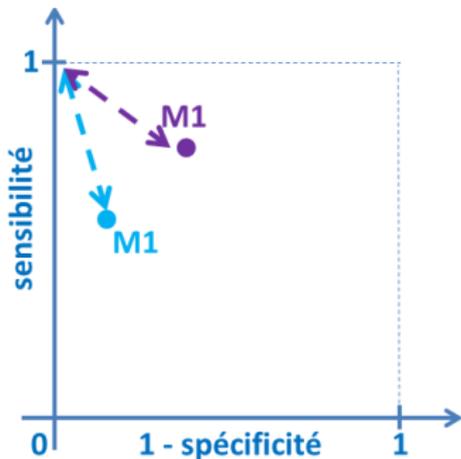
Conclusion

Références

Espace ROC et comparaison de modèles

Comment choisir le meilleur modèle ?

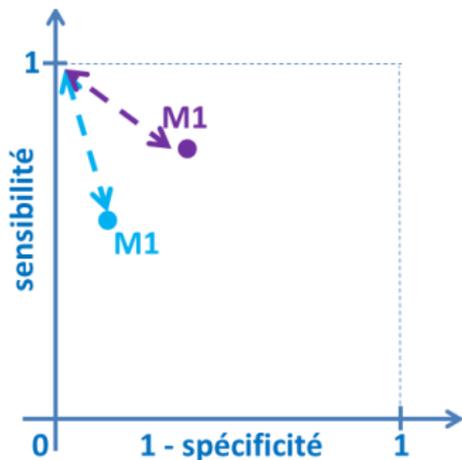
Première solution : le plus proche du modèle idéal



Espace ROC et comparaison de modèles

Comment choisir le meilleur modèle ?

Première solution : le plus proche du modèle idéal



⇒ pas toujours le meilleur choix vis à vis de l'**application**

- ▶ sensi/speci & risques de 1ère/2ème espèces

Espace ROC et comparaison de modèles

Deuxième solution : considérer le compromis sensi/speci réalisable par le modèle

- ▶ l'importance de la sensi et la speci peut être différente
- ▶ (e.g., dans le domaine médical)

Espace ROC et comparaison de modèles

Deuxième solution : considérer le compromis sensi/speci réalisable par le modèle

- ▶ l'importance de la sensi et la speci peut être différente
- ▶ (e.g., dans le domaine médical)

Première approche : perte asymétrique pour l'apprentissage

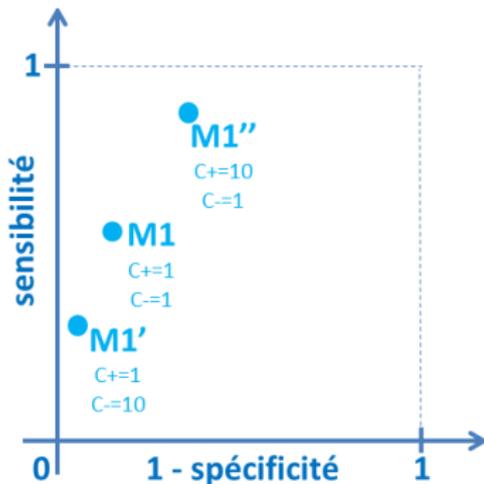
$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i))$$

$$R_{emp}^{(w)}(f) = \frac{1}{n_+} \sum_{i:y_i=+1} C_+ L(y_i, f(x_i)) + \frac{1}{n_-} \sum_{i:y_i=-1} C_- L(y_i, f(x_i))$$

- ▶ C_+ / C_- : poids donné aux instances positives/négatives
- ▶ facteurs multiplicatifs à la fonction de perte $L(y, f(x))$
- ▶ (e.g., pour perte 0/1, une erreur $+1 = \frac{C_+}{C_-}$ erreur -1)

Espace ROC et comparaison de modèles

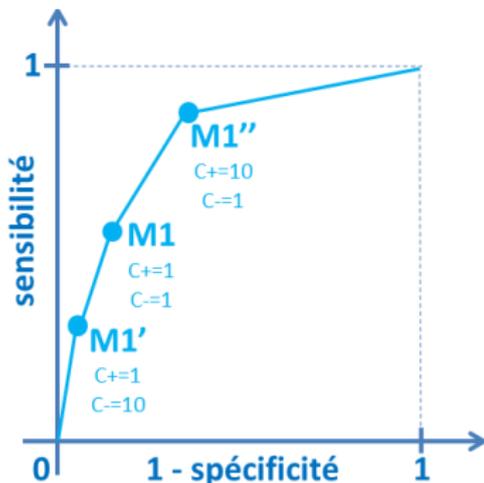
Illustration :



- ▶ on construit plusieurs modèles avec différents coûts
 - ▶ M' : $C_- > C_+ \Rightarrow$ gain en spéci, perte en sensi
 - ▶ M'' : $C_+ > C_- \Rightarrow$ gain en sensi, perte en spéci
 - ▶ (NB : même modèle, différentes paramétrisations)

Espace ROC et comparaison de modèles

L'**enveloppe convexe** permet de balayer toute la gamme :



- ▶ 1 point = 1 combinaison de deux classifieurs
- ▶ pas forcément atteignable pour un choix de C_+/C_-

Espace ROC et comparaison de modèles

Approche par **coûts asymétriques** :

- ▶ + : applicable pour tout classifieur
 - ▶ (si coûts asymétriques pour l'apprentissage)
- ▶ - : besoin de construire plusieurs modèles
- ▶ - : on discrétise le compromis sensi/speci
 - ▶ (enveloppe convexe)

Approche par **coûts asymétriques** :

- ▶ + : applicable pour tout classifieur
 - ▶ (si coûts asymétriques pour l'apprentissage)
- ▶ - : besoin de construire plusieurs modèles
- ▶ - : on discrétise le compromis sensi/speci
 - ▶ (enveloppe convexe)

Approche alternative = **courbe ROC**

- ▶ + : basé sur 1 seul modèle
- ▶ + : balaye l'ensemble des compromis sensi/speci
- ▶ - : nécessite un **classifieur fournissant un score**
 - ▶ e.g., une probabilité $P(y = +1|x)$ – voir cours suivants

Courbe ROC - principe

On dispose d'un **classifieur** fournissant un score $f(x)$

▶ e.g., $f(x) = P(y = +1|x)$

⇒ **convention** : score élevé = classe positive

Courbe ROC - principe

On dispose d'un **classifieur** fournissant un score $f(x)$

- ▶ e.g., $f(x) = P(y = +1|x)$

⇒ **convention** : score élevé = classe positive

Critère de décision = **seuil** sur le score

- ▶ e.g., $\hat{y}(x) = +1$ si $P(y = +1|x) > \text{seuil}$

Courbe ROC - principe

On dispose d'un **classifieur fournissant un score $f(x)$**

- ▶ e.g., $f(x) = P(y = +1|x)$

⇒ **convention** : score élevé = classe positive

Critère de décision = **seuil** sur le score

- ▶ e.g., $\hat{y}(x) = +1$ si $P(y = +1|x) > \text{seuil}$

Sensi/speci "nominales" basées sur un **seuil par défaut**

- ▶ e.g., $\hat{y}(x) = +1$ si $P(y = +1|x) > 0.5$

Courbe ROC - principe

Principe de la courbe ROC : faire varier le seuil par défaut pour obtenir différents compromis sensi/speci

Outline

Apprentissage
Statistique I

Introduction

Validation
Croisée

Critères de
performance

Courbe ROC

k -PPV

Conclusion

Références

Courbe ROC - principe

Principe de la courbe ROC : faire varier le seuil par défaut pour obtenir différents compromis sensi/speci

Exemple : réduction du seuil :

$$\hat{y}(x) = +1 \text{ si } P(y = +1|x) > 0.3$$

⇒ + de prédictions positives : ne peut qu'améliorer la sensi

Courbe ROC - principe

Principe de la courbe ROC : faire varier le seuil par défaut pour obtenir différents compromis sensi/speci

Exemple : réduction du seuil :

$$\hat{y}(x) = +1 \text{ si } P(y = +1|x) > 0.3$$

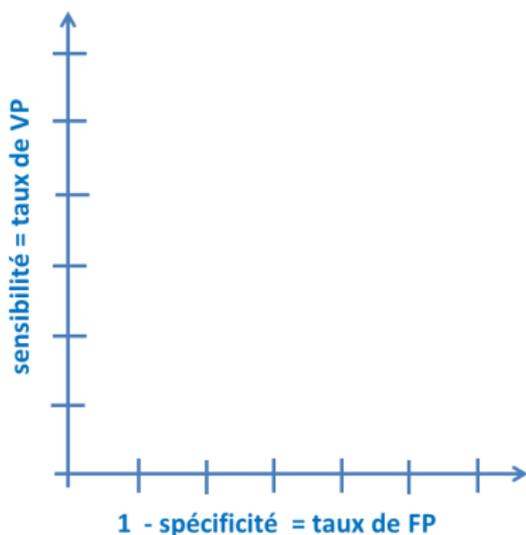
⇒ + de prédictions positives : ne peut qu'améliorer la sensi

Procédure :

1. on part de la valeur maximum du score
 - ▶ toutes prédictions négatives : sensi = 0 / speci = 1
2. on réduit graduellement le score
3. on termine à sa valeur minimum
 - ▶ toutes prédictions positives : sensi = 1 / speci = 0

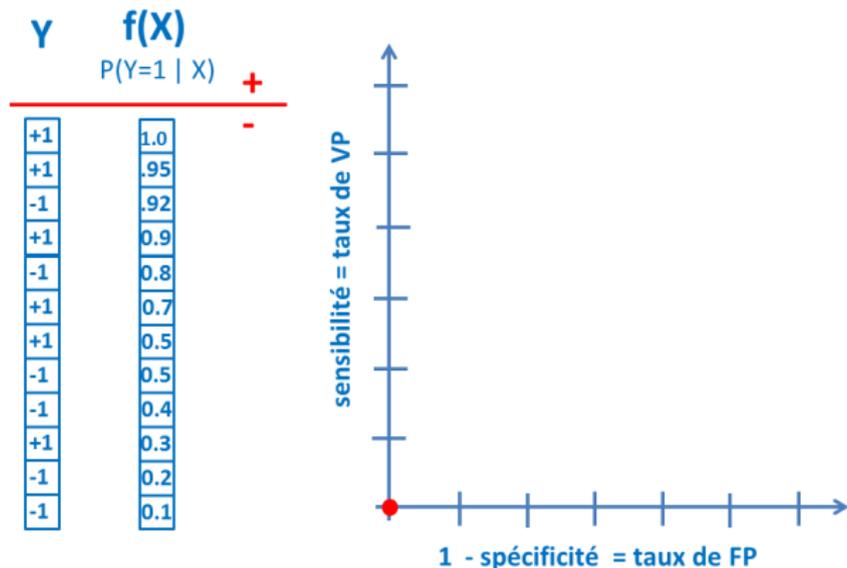
Courbe ROC - Illustration

Y	f(X) P(Y=1 X)
+1	1.0
+1	.95
-1	.92
+1	0.9
-1	0.8
+1	0.7
+1	0.5
-1	0.5
-1	0.4
+1	0.3
-1	0.2
-1	0.1



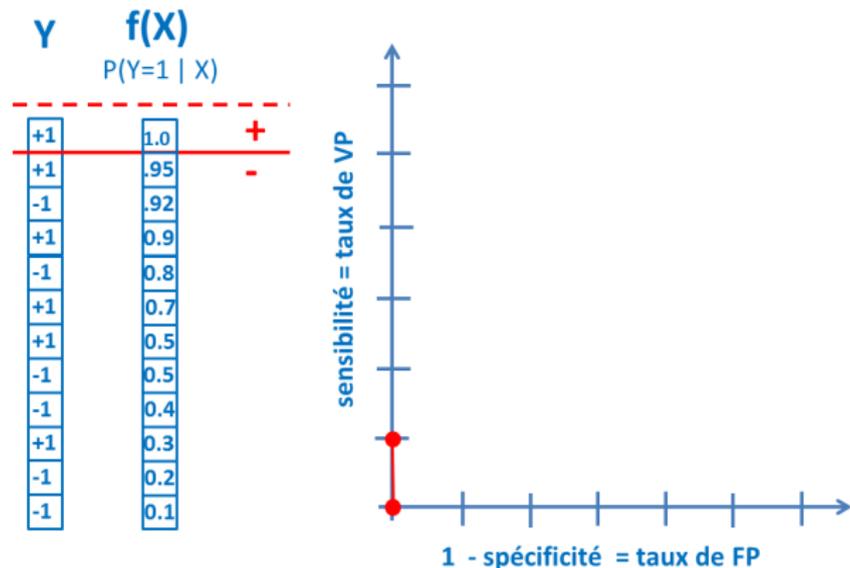
- ▶ t_0 : on trie les labels y_i en fonction du score $f(x_i)$

Courbe ROC - Illustration



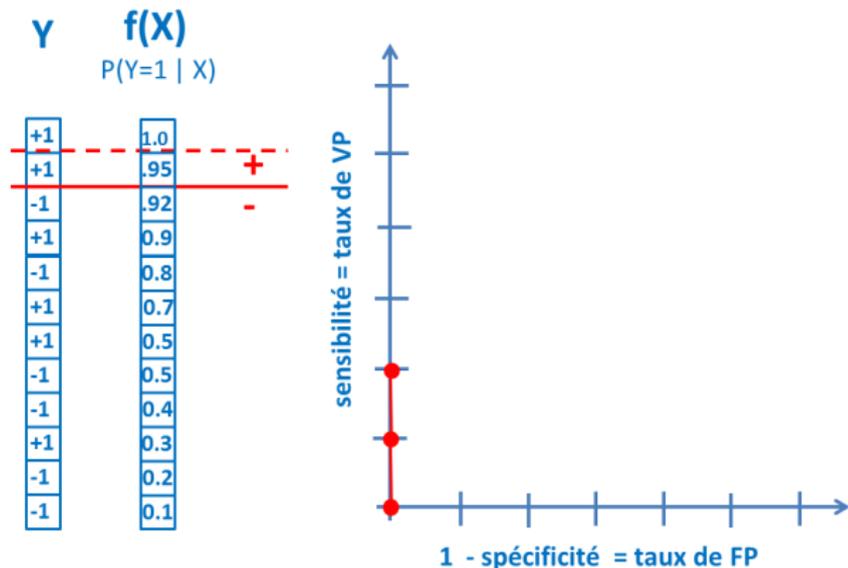
- ▶ t_0 : on choisit un seuil supérieur à $\max_i f(x_i)$
 - ▶ tout le monde est classifié comme négatif
 - ▶ $\text{sensi} = 0$; $\text{speci} = 1$

Courbe ROC - Illustration



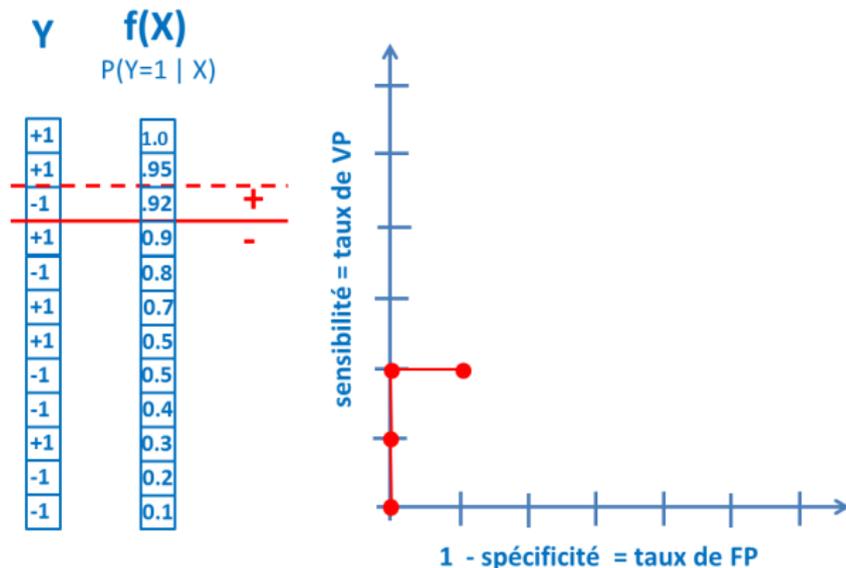
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



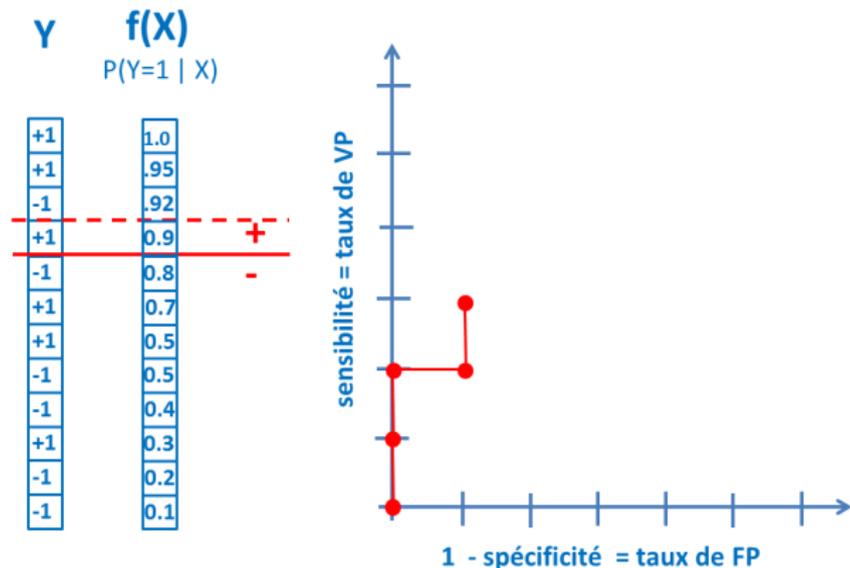
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



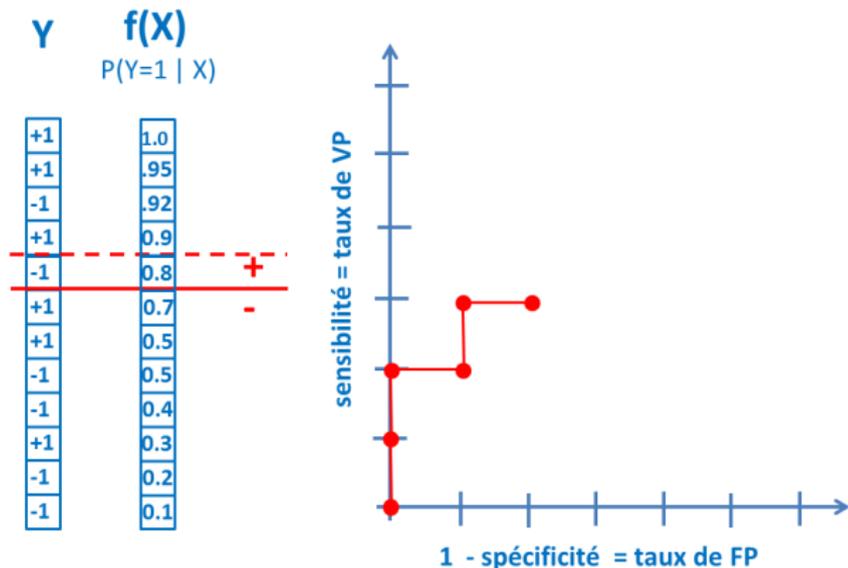
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



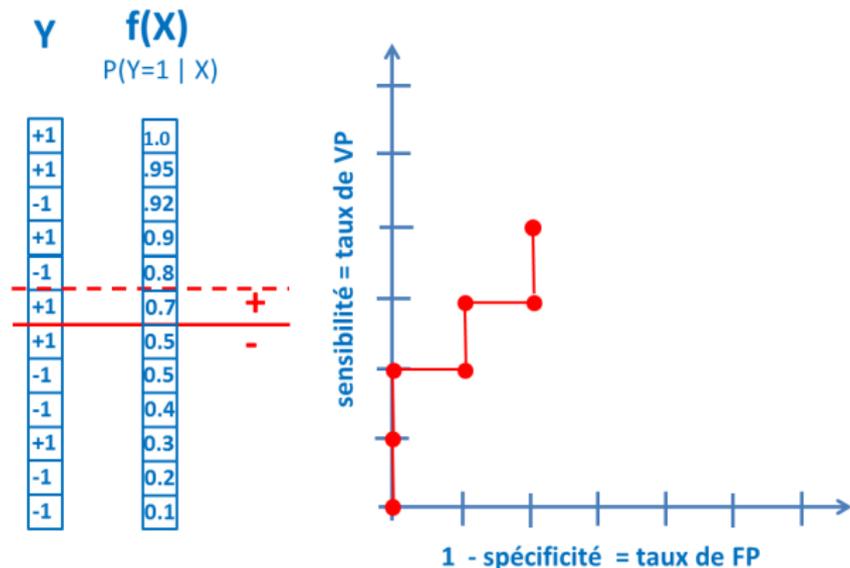
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



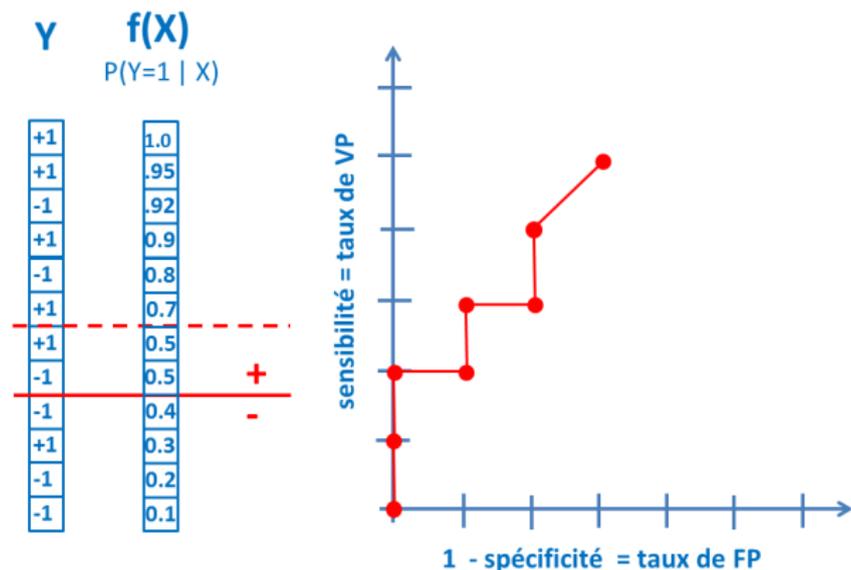
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



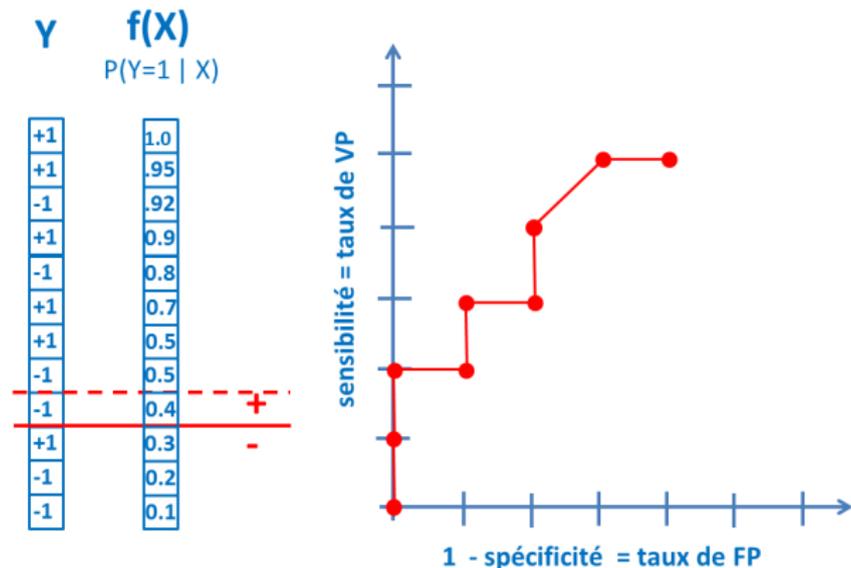
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



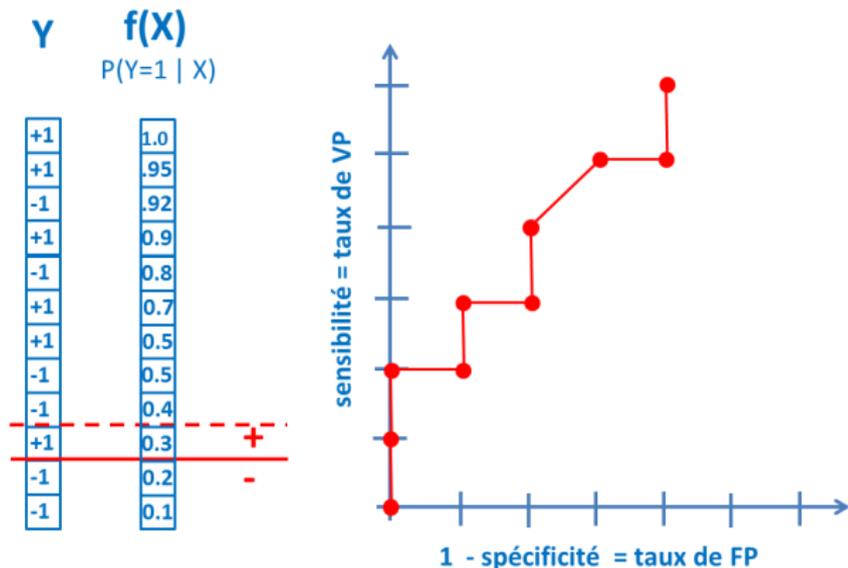
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



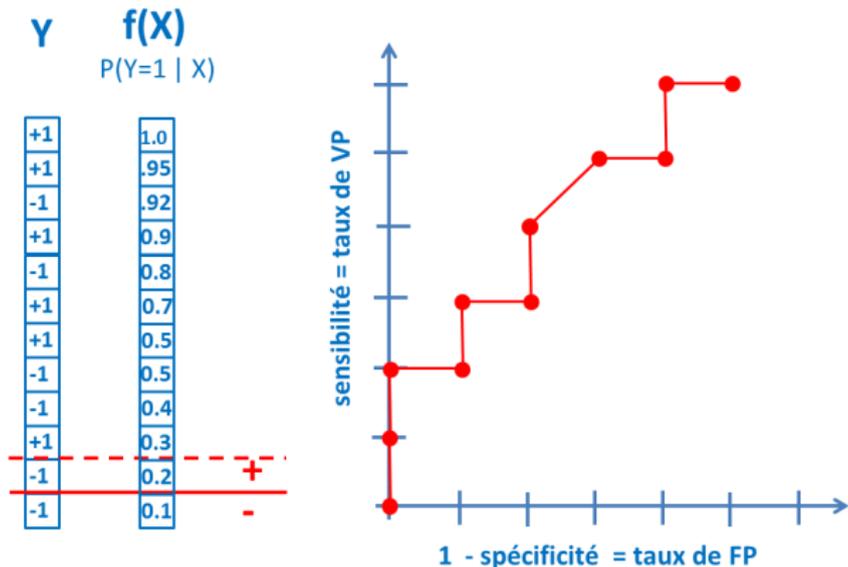
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



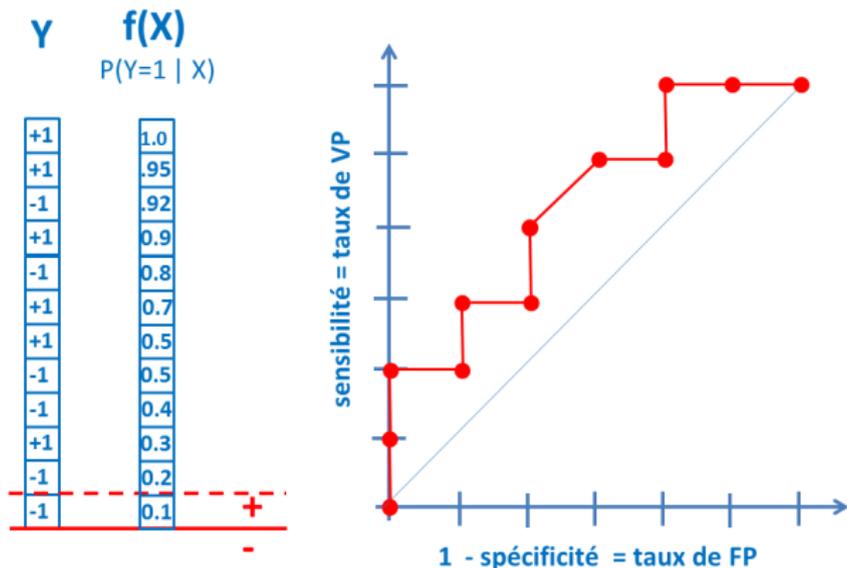
- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration



- ▶ on diminue le seuil case par case et on fait un pas :
 - ▶ vers le haut si $y = +1$: vrai positif
 - ▶ vers la droite si $y = -1$: faux positif

Courbe ROC - Illustration

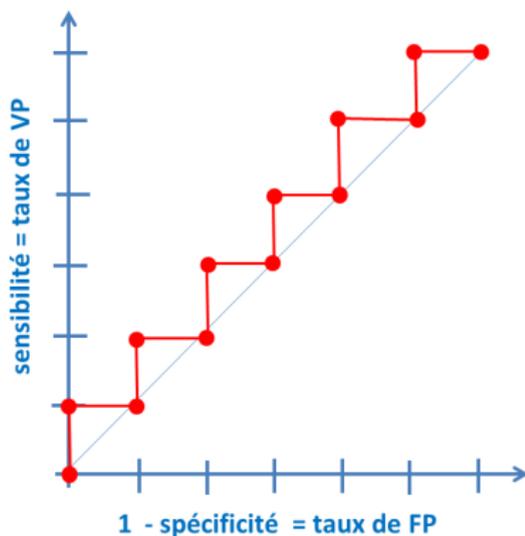


- ▶ on atteint $\min_i f(x_i)$
 - ▶ tout le monde est classifié comme positif
 - ▶ sensi = 1 ; speci = 0

Courbe ROC - remarque

Y f(X)
P(Y=1 | X)

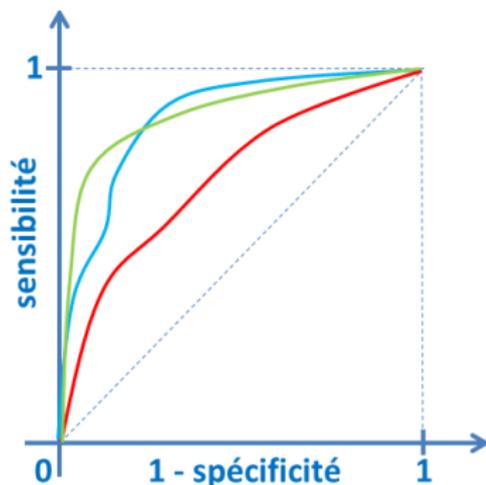
+1	1.0
-1	1.0
+1	1.0
-1	0.9
+1	0.8
-1	0.7
+1	0.5
-1	0.5
+1	0.4
-1	0.3
+1	0.2
-1	0.1



Modèle aléatoire = séquence de +1/-1 dans les labels triés

- ▶ pas de structure dans les scores
- ▶ on se déplace sur la diagonale

Courbe ROC & comparaison de modèles

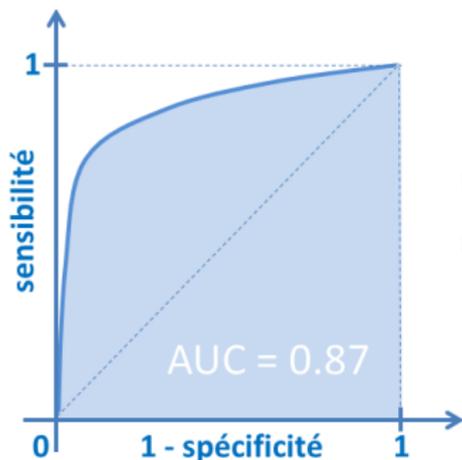


- ▶ comparaison de performances aux différents niveaux de sensi/speci : on s'affranchit du choix du seuil
- ▶ modèles bleu et vert : optimaux à différents niveaux
- ▶ modèle rouge : jamais optimal

Aire sous la courbe ROC (AUC)

Critère AUC : Area Under the (ROC) Curve

- ▶ une manière de résumer une courbe ROC



- ▶ modèle aléatoire : $AUC = 0.5$
- ▶ modèle parfait : $AUC = 1$

Interprétation :

$$AUC = P(f(x_1) > f(x_2) \mid y_1 = +1, y_2 = -1)$$

Courbe ROC - mise en oeuvre

Différentes implémentations en R : exemple du [package ROCR](#)

Outline

Apprentissage
Statistique I

Introduction

Validation
Croisée

Critères de
performance

Courbe ROC

k -PPV

Conclusion

Références

Courbe ROC - mise en oeuvre

Différentes implémentations en R : exemple du [package ROCR](#)

Pré-requis :

1. vecteur de labels :

▶ `y = c(1,1,0,1,0,1,1,0,0,1,0,0)`

2. vecteur de scores associés :

▶ `score = c(1,1,1,0.9,0.8,0.7,0.5,0.5,0.4,0.3,0.2,0.1)`

Courbe ROC - mise en oeuvre

Différentes implémentations en R : exemple du [package ROCR](#)

Pré-requis :

1. vecteur de labels :

▶ `y = c(1,1,0,1,0,1,1,0,0,1,0,0)`

2. vecteur de scores associés :

▶ `score = c(1,1,1,0.9,0.8,0.7,0.5,0.5,0.4,0.3,0.2,0.1)`

Procédure :

1. calculer un objet de type [prediction](#) :

▶ `pred = prediction(score, y, label.ordering = c(0,1))`

2. calculer un objet de type [performance](#) :

▶ `perf.roc = performance(pred, measure = "tpr", x.measure = "fpr")`

3. afficher la courbe ROC

▶ `plot(perf.roc, main = "ROC curve")`

Courbe ROC - mise en oeuvre

Outline

Apprentissage
Statistique I

Introduction

Validation
Croisée

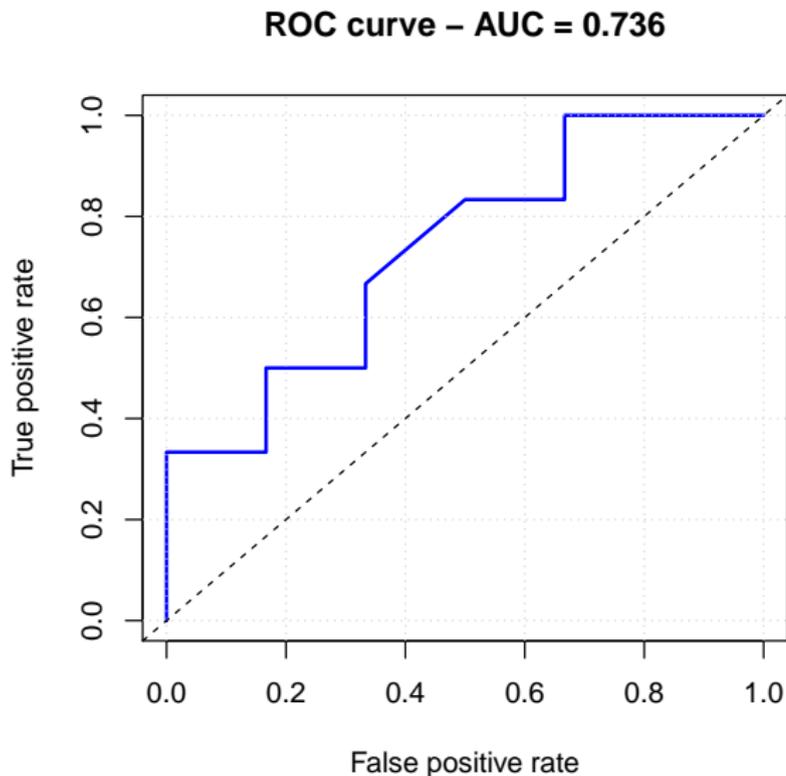
Critères de
performance

Courbe ROC

k-PPV

Conclusion

Références



Remarques :

1. fonction `prediction` :

- ▶ rappel : score élevé = classe positive
- ▶ `label.ordering` = identifiant des classes -1 et +1
 - ▶ facultatif

2. fonction `performance` :

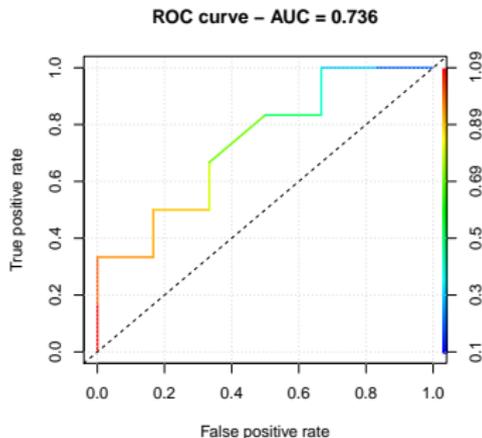
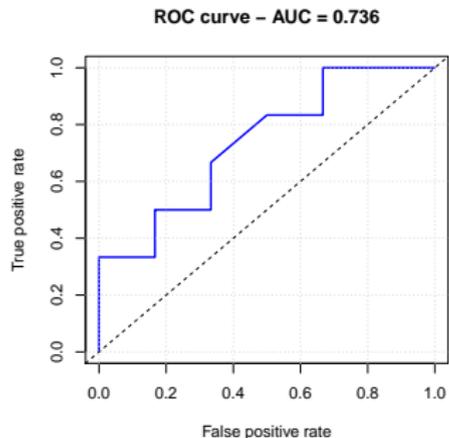
- ▶ permet de calculer de nombreux indicateurs
- ▶ courbe ROC :
 - ▶ $y = \text{sensi} = \text{taux de TP} \Rightarrow \text{measure} = \text{"tpr"}$
 - ▶ $x = 1 - \text{speci} = \text{taux de FP} \Rightarrow x.\text{measure} = \text{"fpr"}$
- ▶ AUC :
 - ▶ `perf.auc = performance(pred, measure = "auc")`
 - ▶ `auc = perf.auc@y.values[[1]]`

3. fonction `plot` :

- ▶ option `colorize = TRUE` : affiche les valeurs de seuils sur la courbe ROC

Courbe ROC - mise en oeuvre

Option `colorize = TRUE` :



Algorithme des k plus proches voisins (k -PPV)

Algorithme des k -PPV

Algorithme des k -plus proches voisins :

1. trouver les k observations x_i les plus proches de l'observation x' à classifier
2. définir $f(x')$ en fonction des réponses y_i des k -PPV
 - ▶ régression : valeur moyenne
 - ▶ classification : vote majoritaire

Algorithme des *k*-PPV

Algorithme des *k*-plus proches voisins :

1. trouver les *k* observations x_i les plus proches de l'observation x' à classifier
2. définir $f(x')$ en fonction des réponses y_i des *k*-PPV
 - ▶ régression : valeur moyenne
 - ▶ classification : vote majoritaire

Approche de mémorisation

- ▶ + : très simple à mettre en oeuvre
- ▶ - : passage à l'échelle

Algorithme des *k*-PPV

Algorithme des *k*-plus proches voisins :

1. trouver les *k* observations x_i les plus proches de l'observation x' à classifier
2. définir $f(x')$ en fonction des réponses y_i des *k*-PPV
 - ▶ régression : valeur moyenne
 - ▶ classification : vote majoritaire

Approche de mémorisation

- ▶ + : très simple à mettre en oeuvre
- ▶ - : passage à l'échelle

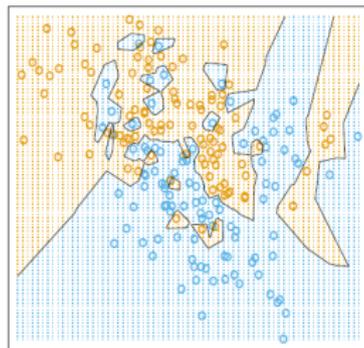
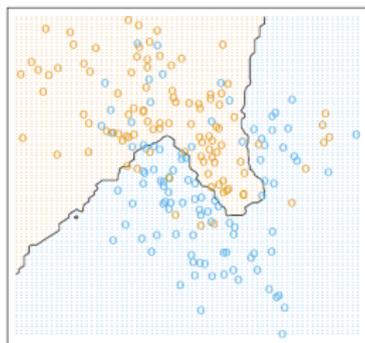
Questions ouvertes :

- ▶ choix du critère de distance
- ▶ choix de la valeur de *k*

Algorithme des k -PPV - complexité

Illustration tirée de Hastie et al. (2001) :

- ▶ à gauche : $k = 15$; à droite : $k = 1$.



Des petites valeurs de k conduisent à des modèles plus locaux et donc (en général) plus complexes.

Fonction `knn` du package `class` : `knn(X, X.test, y, k)`

- ▶ `X, X.test` : données d'apprentissage et de test
 - ▶ matrices de descripteurs
 - ▶ NB : uniquement pour la distance Euclidienne
- ▶ `y` : vecteur des catégories des données d'apprentissage
- ▶ `k` : nombre de voisins à considérer

⇒ renvoie un vecteur avec les catégories prédites

- ▶ si option `prob=TRUE` : proportion des votes de la classe prédite (\sim mesure de confiance dans la prédiction)

Introduction

Validation
Croisée

Critères de
performance

Courbe ROC

k -PPV

Conclusion

Références

Conclusion

Conclusion (1/2)

Apprentissage supervisé

- ▶ risque empirique, généralisation et complexité

Validation croisée

- ▶ estimer l'erreur de généralisation par ré-échantillonnage

Critères de performance de prédiction

- ▶ erreur quadratique vs matrice de confusion
- ▶ compromis sensibilité / spécificité

Conclusion (2/2)

Courbe ROC

- ▶ s'affranchir du seuil de décision
- ▶ critère AUC

Algorithme des k -PPV

- ▶ approche par mémorisation
- ▶ le BA-BA de la classification

TP : courbe ROC, k -PPV.

T. Hastie, R. Tibshirani, and J.. Friedman. *The Elements of Statistical Learning*. Springer, 2001.