Classification … supervisée

Olivier Gascuel

LIRMM-CNRS, Montpellier, gascuel@lirmm.fr

Introduction

- Classification supervisée/le reste
- Les données
- La règle de Bayes
- Les grandes approches-tendances

Panorama des méthodes

- Bayes naïf
- Fonction discriminante de Fisher
- K plus proches voisins et noyaux de Parzen
- Arbres de décision
- Réseaux de neurones
- Machines à vecteurs supports

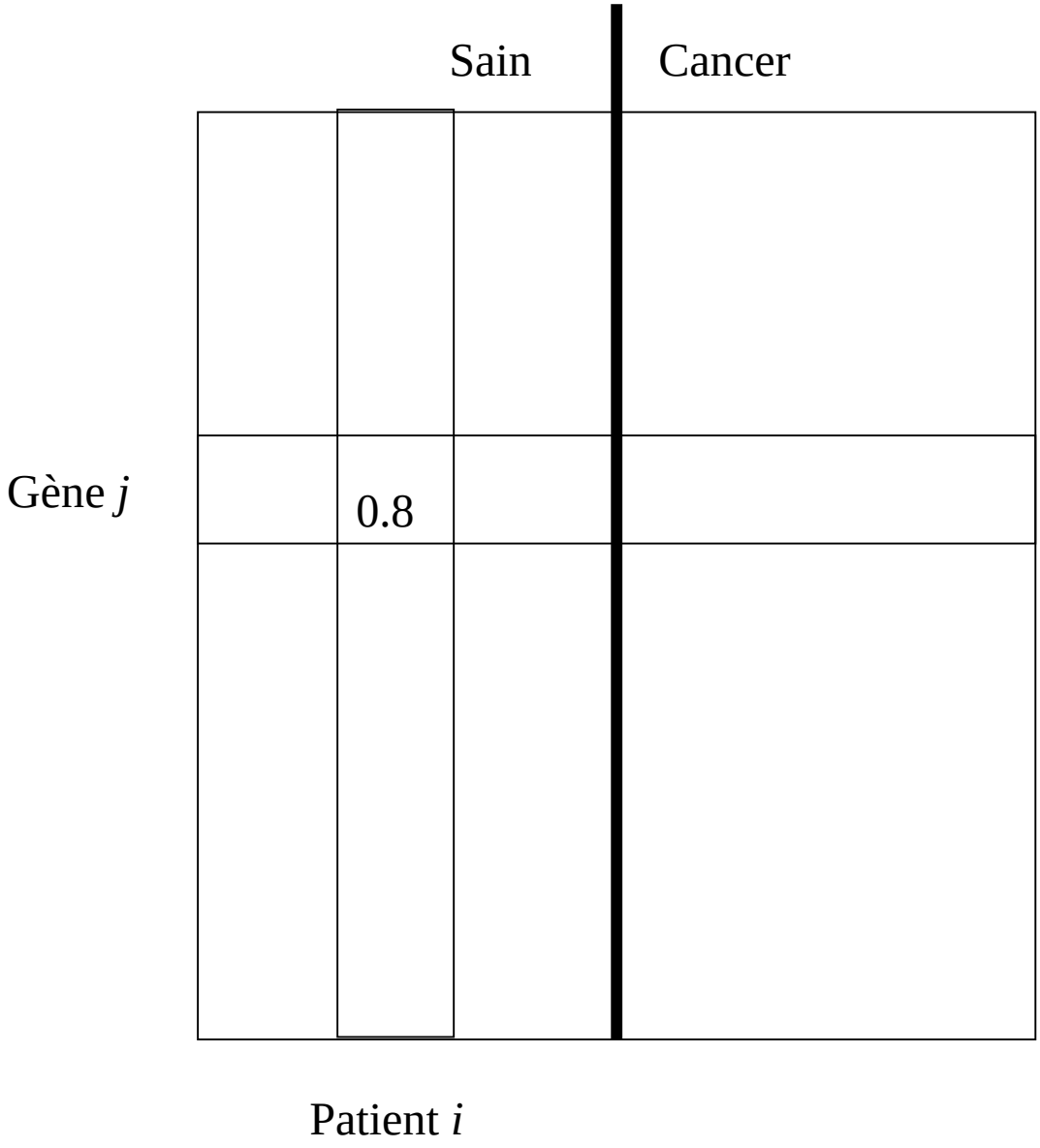
Validité, dimension de Vapnik-Chervonenkis

Validation croisée, bootstrap

Un exemple en analyse des données d'expression

1.1 Classification supervisée ou discrimination/le reste

Un ensemble *D* d'observations-exemples $(\mathbf{x}_i, c_i)$ décrits dans un espace *X* et classés dans *C*. Par exemple :



Analyse différentielle, classification non-supervisée, cinétique …

1.1 Classification supervisée/le reste

Objectifs

- Prédire la classe d'un nouvelle observation
- Expliquer la partition en classes

Applications

- Reconnaissance des formes ……
- …. sans limites

"Avantages"

- Très étudiées
- Méthodes solides de validation

1.2 Les données

- Quantitatives
- Qualitatives (nominales, ordinales, …)
- Mixtes
- Séquences …

Il existe des passerelles entre ces représentations.

La plupart des méhodes sont dédiées à un type de données.

1.3 La règle de Bayes

- Une règle de classement f : $X \xrightarrow{f} C$

- Si l'objectif est de minimiser la probabilité d'erreur :

$$P_{error}(f) = \Pr(f(\mathbf{x}) \neq c_{\mathbf{x}})$$

- Alors la règle optimale est :

$$f_{Bayes}(\mathbf{x}) = ArgMax_{c \in C}(\Pr(c/\mathbf{x}))$$

On choisit la classe dont la probabilité est la plus grande

Remarque : On est très rarement dans le cas déterministe où :

$$P_{error}(f_{Bayes}) = 0,$$

et l'erreur Bayesienne constitue une borne incompressible, généralement inconnue.

1.4 Les grandes approches-tendances

La plupart des méthodes se positionnent par rapport à la règle de Bayes, ou par rapport à la probabilité d'erreur. On distingue :

- Les approches paramétriques : on pose un modèle dont on estime les paramètres ;
- Les approches non-paramétriques : on estime "localement" la probabilité conditionnelle $\Pr(c/\mathbf{x})$;
- Les méthodes d'optimisation : on sélectionne *f* au sein d'une famille *F* de fonctions par minimisation de l'erreur (risque) empirique.

---

2. Panorama

- Bayes naïf
- Fonction discriminante de Fisher
- K plus proches voisins et noyaux de Parzen
- Arbres de décision
- Réseaux de neurones
- Machines à vecteurs supports

2.1 Bayes naïf

- Une observation **x** est représentée par un vecteur $\left(x_j\right)$ d'attributs nominaux, obtenus, par exemple, par discrétisation des valeurs d'expression des gènes.

- Par application de la formule de Bayes :

$$\Pr\left(c/\left(x_j\right)\right) = \frac{\Pr\left(\left(x_j\right)/c\right) \times \Pr(c)}{\Pr\left(\left(x_j\right)\right)}$$

$$\propto \Pr\left(\left(x_j\right)/c\right) \times \Pr(c)$$

- En supposant l'indépendance conditionnelle des attributs :

$$\Pr\left(c/\left(x_j\right)\right) \propto \left[\prod_j \Pr\left(x_j/c\right)\right] \times \Pr(c),$$

- Les probabilités $\Pr\left(x_j/c\right)$ et $\Pr(c)$ sont estimées sur l'échantillon d'apprentissage.

Propriétés :

- Simple mais bonnes performances.
- Nécessite un « lifting » pour les attributs rares.
- Sensible à un grand nombre d'attributs non-discriminants.

2.2 Fonction discriminante de Fisher

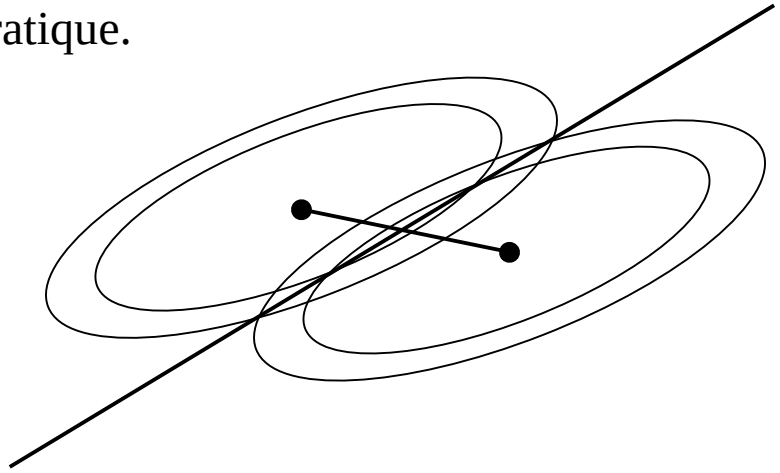
- Une observation **x** est représentée par un vecteur d'attributs quantitatifs, et on fait l'hypothèse qu'au sein de chaque classe c la distribution des observations est Gaussienne de paramètres (μ_c, Σ_c).

- La règle de Bayes revient alors à minimiser :

$$\Delta^2_{\Sigma_c}(\mathbf{x}, \mu_c) - 2\ln(\Pr(c)) + \ln(\det(\Sigma_c)),$$

  $\Delta^2_{\Sigma_c}(\mathbf{x}, \mu_c)$ est la distance de Mahalanobis entre **x** et μ_c.

- Lorsque les Σ_c sont identiques, la séparation est linéaire, sinon quadratique.



Propriétés :

- Estimation-apprentissage simple et rapide
- Souvent efficace et robuste
- Mais pas toujours, faiblesse du modèle linéaire Gaussien

2.3 *k* plus proches voisins et noyaux de Parzen

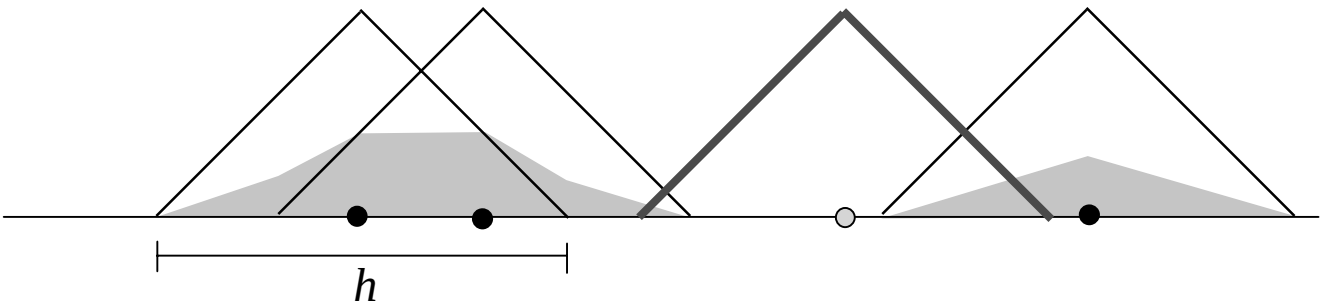
- $\Pr(c/\mathbf{x})$ est estimé dans un "voisinage" de **x**.

- ***k*-PP (*k*-NN) :** $\hat{\Pr}(c/\mathbf{x}) = n_c^{\mathbf{x}} / k$, où $n_c^{\mathbf{x}}$ est le nombre de k plus proches voisins de **x** dans c.

- **Noyaux de Parzen :** $\hat{\Pr}(\mathbf{x} / c) = \frac{1}{n_c V_h} \sum_{\mathbf{x}_i \in c} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$

  avec : $_h = \int K(\mathbf{x}/h) d\mathbf{x}$

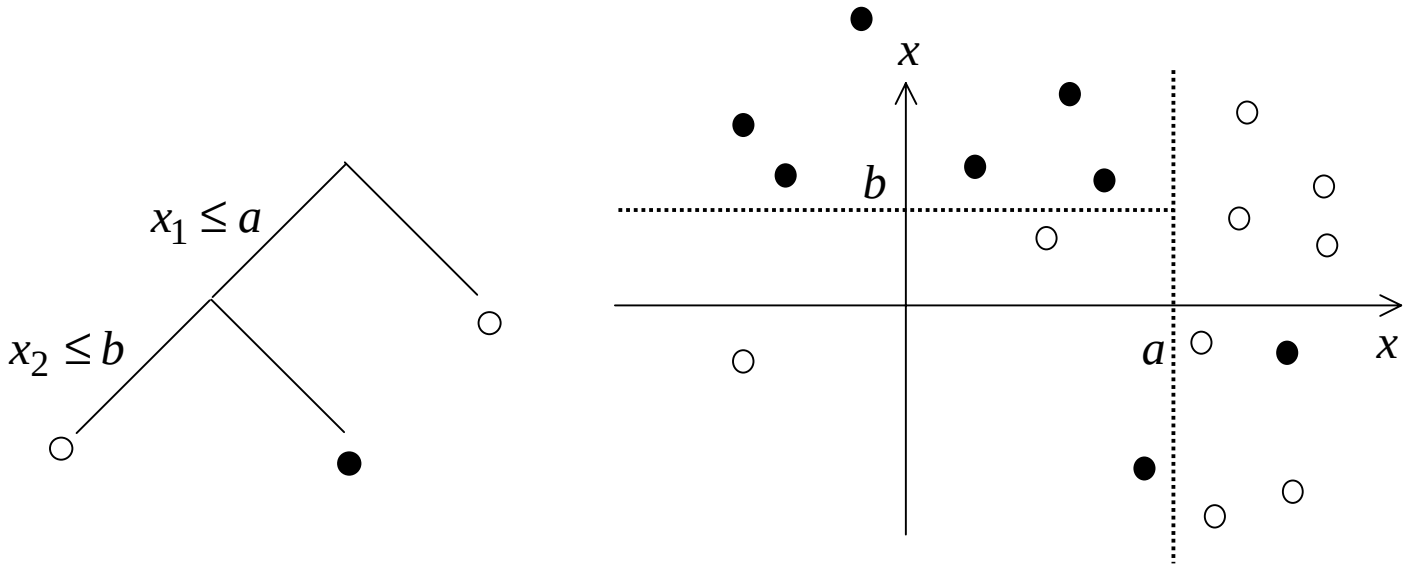
  E.g., noyau Gaussien : $K(\mathbf{x}) = (2\pi)^{-p/2} \exp\left(-\frac{1}{2}\mathbf{x'x}\right)$



Propriétés :

- Non-paramétrique, presque toujours performant
- Du moins si attributs continus pas trop nombreux
- Pas d'apprentissage, long à prédire

2.4 Arbres de décision



- L'arbre est construit par segmentations successives.

- A chaque étape on maximise un critère de pureté, e.g. :

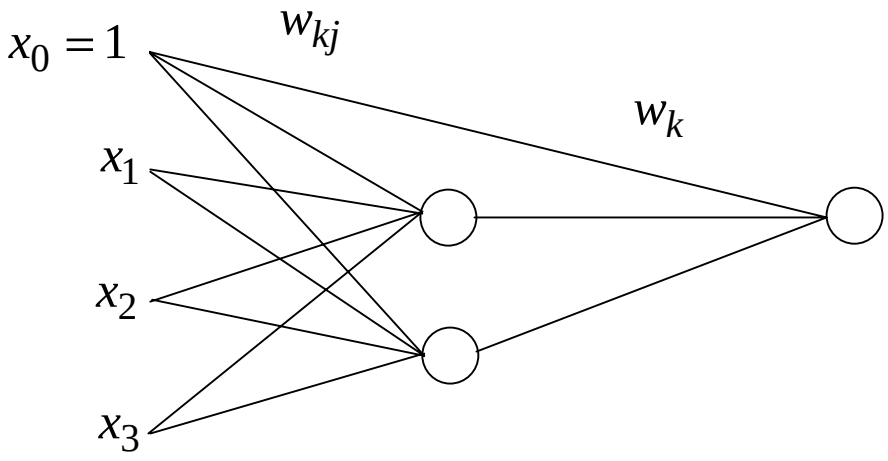
$$p^g\left[\sum_c -p_c^g \log\left(p_c^g\right)\right] + p^d\left[\sum_c -p_c^d \log\left(p_c^d\right)\right]$$

p_c^g (p_c^d) proportion d'exemples de gauche(droite) dans c

Propriétés

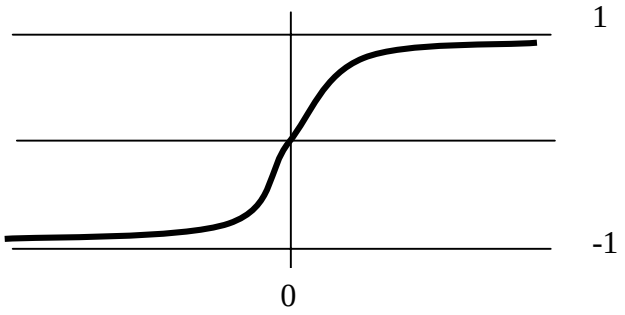
- Non-paramétrique
- Explicatif, mais des performances moyennes
- A même d'intégrer des attributs de différents types
- Peu sensible à la présence d'attributs non discriminants

2.5 Réseaux de neurones (PMC)



- $c \in \{-1,+1\}, \hat{c} = f\left(w_0 + \sum_k w_k f\left(w_{k0} + \sum_j w_{kj} x_j \right) \right)$

- f est une fonction sigmoïde

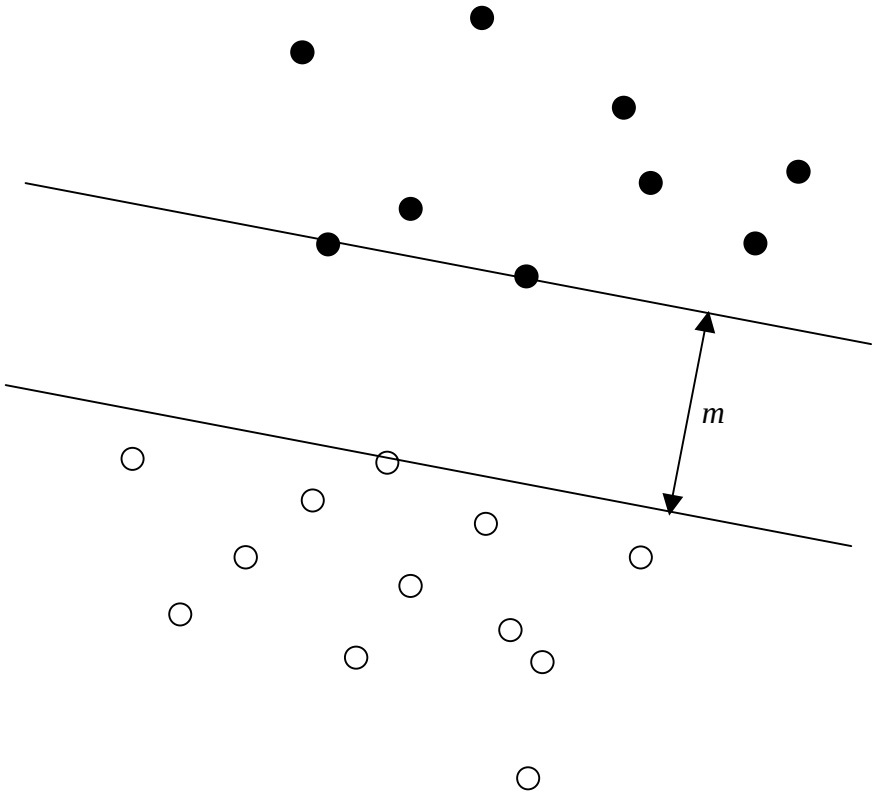


- Apprentissage par rétropropagation de l'erreur quadratique

Propriétés

- Tend à minimiser la fréquence d'erreur.
- Approximateur universel.
- Excellentes performances (e.g., structure des protéines).
- Difficultés de mise en œuvre.

2.6 Machines à vecteurs supports



- Tend à maximiser la marge
- Et minimiser le nombre d'erreurs de classement
- Réalise des séparations non-linéaires en utilisant en entrée des expressions (e.g. polynomiales) des attributs initiaux.

Propriétés

- Bonnes performances
- Relativement plus simple de mise en œuvre que les RN

3.1 Validité, compromis biais-variance

- ***k*-PPV**

  k trop petit → grande variance

  k trop grand → biais important

- **Noyaux de Parzen**

  h trop petit → grande variance

  h trop grand → biais important

- **Arbres de décision**

  grand arbre → grande variance

  petit arbre → biais important

…….

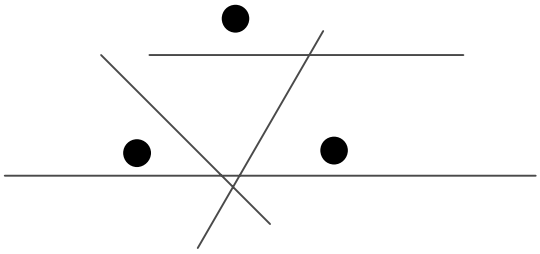
3.2 Validité, dimension de Vapnik-Chervonenkis (VC)

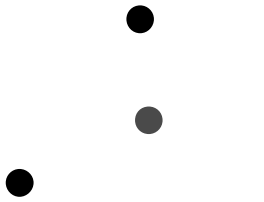
- F une famille de séparateurs (sous-ensembles) de X

- C_F : plus grand nombre de points de X pulvérisables par F

Exemple 1: F ={demi-plans du plan}, $C_F = 3$

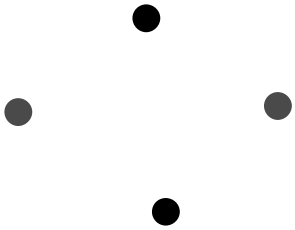
Ces 4 droites permettent de retrouver les 8 parties de cet ensemble de 3 points





On ne peut isoler le point central par une droite

On ne peut isoler les coins opposés par une droite



Exemple 2: F ={demi-espaces de R^p}, $C_F = p+1$

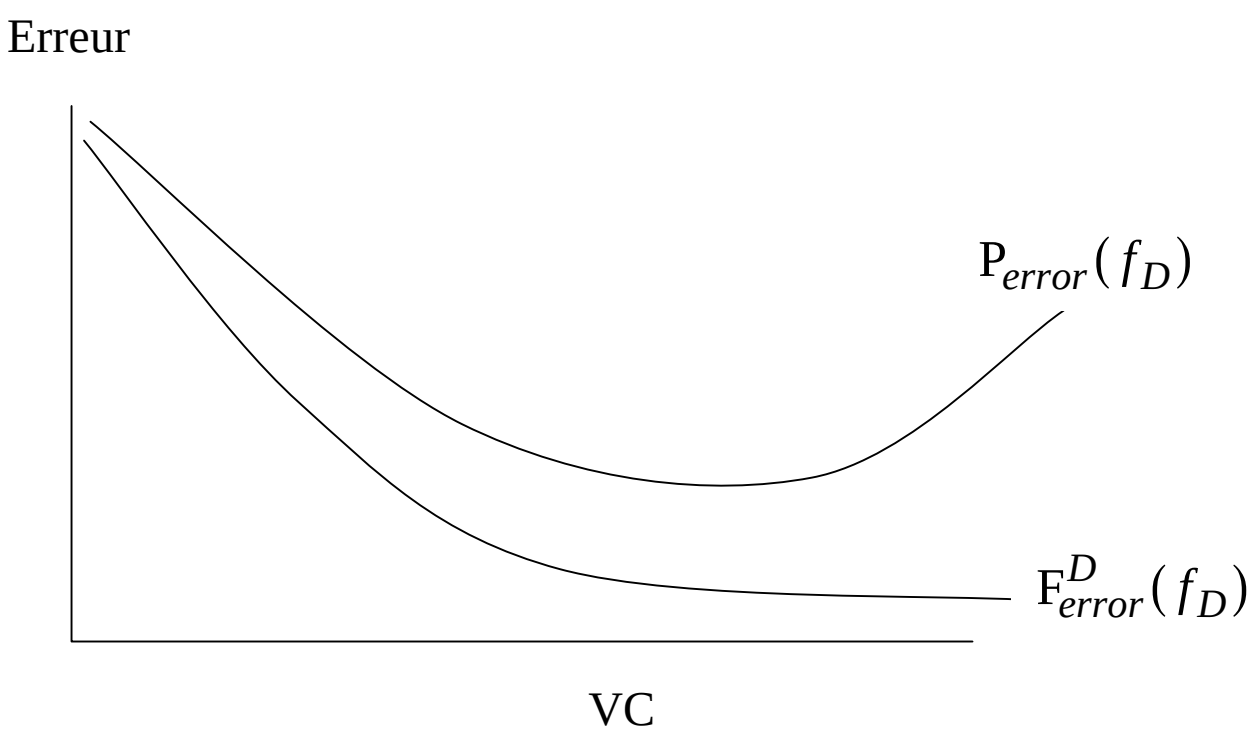
- Cela n'a aucun sens de chercher à apprendre un séparateur de F avec un nombre d'exemples égal à C_F.

- $\Pr\left(Max_{f \in F} \left| P_{error}(f) - F^n_{error}(f) \right| \geq \varepsilon \right) \leq n^{VC_F} e^{-n\varepsilon^2}$

  $F^n_{error}(f)$ est la fréquence d'erreur de f sur un n-échantillon.

- La fréquence d'erreur mesurée sur *D* d'un séparateur *f* appris grâce à *D* donne une vision optimiste de la probabilité d'erreur de *f*.

- Dans tous les cas, se pose un problème de calibration : nombre d'attributs dans la description, nombre de paramètres dans le modèle, taille du voisinage, …



4. Validation croisée, bootstrap

Le problème de la calibration impose de disposer d'un ensemble test indépendant de l'ensemble d'apprentissage *D*. Lorsque l'on ne dispose pas de très nombreux exemples :

Validation croisée : T une fraction des éléments de D ; on apprend sur $D - T$ et on teste sur T ; on répéte l'expérience et on fait la moyenne.

Bootstrap : D^* un pseudo-échantillon issu de D par tirage avec remise ; on apprend sur D^* et on teste sur D ; on répéte l'expérience et on fait la moyenne.

On dispose alors d'une estimation peu biaisée de la probabilité d'erreur sur *D*. Celle-ci peut permettre d'ajuster les paramètres de réglage (nombre d'attributs dans la description, nombre de paramètres dans le modèle, taille du voisinage, …).

Grate, Bhattacharyya et al. (2002), Simultaneous Relevant Feature Identification and Classification, Workshop on Algorithms in Bioinformatics (WABI), Rome, Lecture Notes in Computer Science 2452, pp. 1-9.

- Jeu de données: $D = \{(\mathbf{x}_i, c_i)\}$, $i \in (1,2,...n)$
- Deux classes, $c_i \in \{+1,-1\}$
- $\mathbf{x}_i \in R^p$
- p nombre de gènes $\approx 10^3 - 10^4$
- n nombre d'observations $\approx 10^1 - 10^2$

Une approche de type SVM incluant la sélection d'attributs (plutôt que l'élargissement de la marge ou l'utilisation de séparateurs non-linéaires).

- L'objectif est de trouver un hyperplan séparateur :

$$c_i(\mathbf{w}'\mathbf{x}_i - b) \geq 0,\ \forall i \in (1,2,...n)$$

- ce qui est équivalent à : $c_i\mathbf{w}'\mathbf{x}_i \geq 1,\ \forall i \in (1,2,...n)$

  avec $\mathbf{w},\mathbf{x}_i \in R^{p+1}$

- D'un autre coté, on veut que la plupart des composantes de **w** soient nulles. On doit donc résoudre :

$$Min \|\mathbf{w}\|_0$$

avec les contraintes $c_i \mathbf{w}' \mathbf{x}_i \geq 1, \forall i \in (1,2,...n)$

- C'est un problème NP-difficile, on résout donc :

$$Min \|\mathbf{w}\|_1 = \sum_1^{p+1} |w_j|$$

avec les contraintes $c_i \mathbf{w}' \mathbf{x}_i \geq 1, \forall i \in (1,2,...n)$

- Ce qui s'écrit : $Min \sum_1^{p+1} \left(w_j^+ + w_j^- \right)$

avec les contraintes: $c_i \left(\mathbf{w}^+ - \mathbf{w}^- \right)' \mathbf{x}_i \geq 1, \forall i \in (1,2,...n)$

$w_j^+ \geq 0$ et $w_j^- \geq 0, \forall j \in (1,... p+1)$

- Finalement, pour résoudre les cas où les observations ne sont pas linéairement séparables, on introduit les variables ξ_i :

$$Min\left[\sum_{1}^{p+1}\left(w_j^+ + w_j^-\right) + C\sum_{1}^{n}\xi_i\right]$$

avec $$c_i\left(\mathbf{w}^+ - \mathbf{w}^-\right)' \mathbf{x}_i \geq 1 - \xi_i,\ \forall i \in (1,2,...n)$$

$$w_j^+ \geq 0 \text{ et } w_j^- \geq 0,\ \forall j \in (1,... p+1)$$

$$\xi_i \geq 0,\ \forall i \in (1,...n)$$

- C pondère le poids des erreurs : $C = 0$ autorise beaucoup d'erreurs, tandis que $C \to \infty$ revient à refuser toute erreur. Il est ajusté par validation croisée.

- Ce programme linéaire est traité en quelques secondes ou minutes, même pour de très grands jeux de données, en utilisant un logiciel standard.