M2 CHPS

[Data & Apprentissage]

Introduction à la science des données et à l'apprentissage

Nicolas Vayatis

Méthodes non linéaires : SVM, méthodes locales, ensembles

Rappel
Modèle probabiliste des *données* de classification

Modèle probabiliste pour la classification supervisée

- (X, Y) - couple de variables aléatoires de loi de probabilité inconnue P
- $X \in \mathcal{X}$ - observation sur un espace mesurable (e.g. $\mathbb{R}^d$)
- $Y \in \{-1, +1\}$ - label/classe binaire (par simplicité)

$\longrightarrow$ Description de la loi jointe $P = \mathcal{L}(X, Y)$ du couple aléatoire (X, Y) à partir des lois conditionnelles ?

Description de la loi jointe

1. Approche générative : $\mathcal{L}(X, Y) = \mathcal{L}(Y) \otimes \mathcal{L}(X \mid Y)$
   - Modèle de type mélange de paramètre : $p = \mathbb{P}\{Y = +1\} \in [0, 1]$
   - Lois conditionnelles sur $\mathbb{R}^d$:

$$P_+ = \mathcal{L}(X \mid Y = +1) \qquad \text{et} \qquad P_- = \mathcal{L}(X \mid Y = -1)$$

2. Approche discriminative $\mathcal{L}(X, Y) = \mathcal{L}(X) \otimes \mathcal{L}(Y \mid X)$
   - Loi marginale sur $\mathbb{R}^d$: $P_X = \mathcal{L}(X)$
   - Meilleure prévision :

$$\eta(x) = \mathbb{P}\{Y = +1 \mid X = x\}\ , \quad \forall x \in \mathbb{R}^d$$

Lien entre les deux descriptions

- Loi marginale (les "dP" désignent les densités des lois) :

$$dP_X = p dP_+ + (1-p) dP_-$$

- Probabilité a posteriori (fonction de régression) :

$$\forall x \in \mathcal{X} \ , \qquad \eta(x) = \frac{p \ dP_+}{p \ dP_+ + (1-p) \ dP_-}(x)$$

- Une remarque :

$$\eta(x) > \frac{1}{2} \quad \Leftrightarrow \quad p \ dP_+(x) > (1-p) \ dP_-(x)$$

Formalisation du *problème* de classification

Le problème de classification binaire

- **Données disponibles :** $(x_1, y_1), \ldots, (x_n, y_n)$, $x_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$
- **Problème :** prédiction du label y connaissant x
- **On cherche :** un classifieur $g : \mathbb{R}^d \to \{-1, +1\}$
- **Question :** trouver un classifieur g qui "généralise" bien.
- **Idée :** on choisit g qui "interprète" bien, mais pas trop !
- **Concrètement :** souvent on cherche une fonction de décision $f : \mathbb{R}^d \to \mathbb{R}$ et on y associe le classifieur $g = \mathrm{sgn}(f)$

Critère d'évaluation : erreur d'un classifieur

- Etant donnée une observation x, un classifieur g réalise une prédiction $g(x)$ à comparer à la classe y.

Erreur du classifieur = Taux d'observations mal classées

$$\hat{L}_n(g) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{[g(x_i) \neq y_i]} = \frac{\#\{i \ : \ g(x_i) \neq y_i\}}{n}$$

Cette erreur s'appelle aussi erreur d'apprentissage.

- Un classifieur g "interprète" convenablement les données si son erreur d'apprentissage est faible.

En pratique : Stratégie du holdout

- On sépare les données disponibles en deux sous-groupes :
  - Base d'apprentissage : $(X_1, Y_1), \ldots, (X_n, Y_n)$
  - Base de test : $(X_{n+1}, Y_{n+1}), \ldots, (X_{n+m}, Y_{n+m})$
- L'erreur de test $\hat{L}'_m(g) = \frac{1}{m} \sum_{j=1}^{m} \mathbb{I}_{[g(X_{n+j}) \neq Y_{n+j}]}$ est une estimation de $L(g)$ pour tout classifieur g (on peut conditionner par rapport à la base d'apprentissage si g résulte d'un apprentissage).
- Meilleure pratique : la *validation croisée* pour rendre plus robuste l'estimateur de $L(g)$

Limites des méthodes génératives

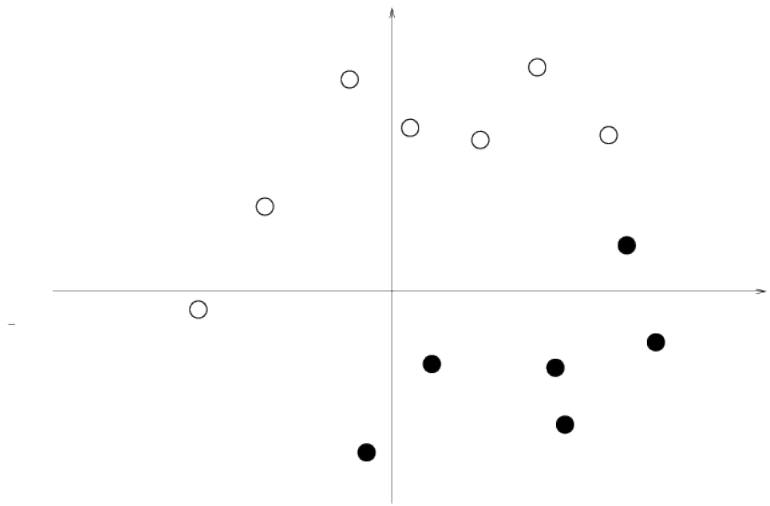
- Evoquées lors du dernier cours : analyse discriminante, régression logistique
- Modèles statistiques paramétriques : "All models are wrong"...
- Lourd a priori de modélisation : "... some are useful"
  - cadre gaussien
  - modèle linéaire
- Curse of dimensionality (cf. Bellmann)

Algorithmes de discrimination linéaire *non-paramétrique*

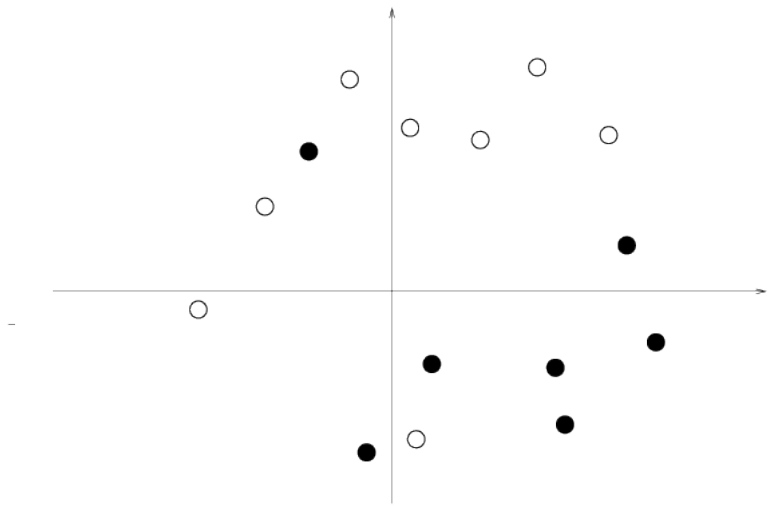
Les trois scénarios

1. les populations sont linéairement séparables
2. les populations sont *presque* linéairement séparables
3. les populations ne sont pas linéairement séparables

Séparabilité linéaire



Scénario 1



Scénario 2

Scénario 1 - Séparateurs linéaires

- **Forme des fonctions de décision :**

$$f(x) = b + < \beta, x >$$

où $b \in \mathbb{R}$, $\beta \in \mathbb{R}^d$.

- L'équation $f(x) = 0$ définit un **hyperplan séparateur** H dans $\mathbb{R}^d$

- **Classifieur associé :**

$$\forall x \in \mathbb{R}^d \quad g_f(x) = \begin{cases} +1 & \text{si } f(x) > 0 \\ \\ -1 & \text{si } f(x) \leq 0 \end{cases}$$

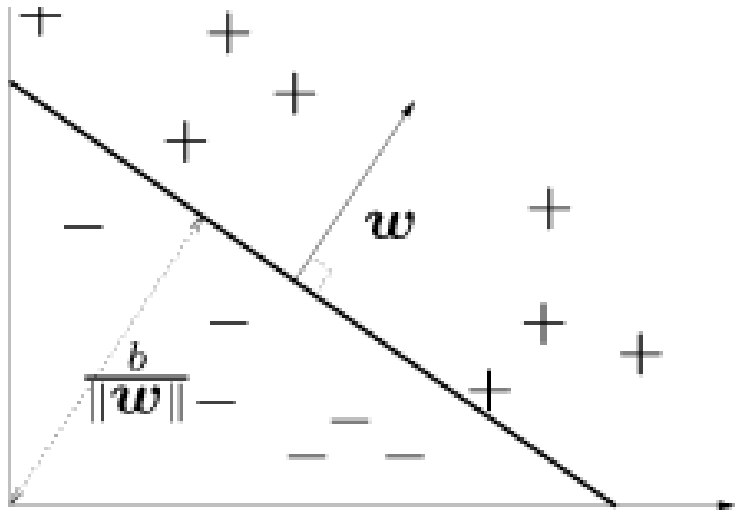
Quelques propriétés

1. $\beta^* = \dfrac{\beta}{\|\beta\|}$ est le vecteur normal à H
2. $\forall x_0 \in H, \quad < \beta, x_0 >= -b$
3. la distance signée (éventuellement négative !) d'un point $x \in \mathbb{R}^d$ à H est donnée par

$$d(x, H) =< \beta^*, x - x_0 >= \frac{1}{\|\beta\|}\left(b + < \beta, x >\right)$$

où $x_0 \in H$

Scénario 1 - Une figure



A separating hyperplane $(\boldsymbol{w}, b) \in \mathbb{R}^n \times \mathbb{R}$ for a 2D training set.

Attention ! ici $w = \beta$...

Algorithme du perceptron (Rosenblatt, 1958)

Perceptron - version simplifiée $b = 0$

Génère une suite $\beta_0, \ldots, \beta_n$ de valeurs pour β

1. **Initialisation** - $\beta_0 = 0$
2. **Etape i** - on considère le couple (x_i, y_i) et on regarde s'il est correctement classé ou non

$$\beta_i = \begin{cases} \beta_{i-1} & \text{si } y_i \cdot < \beta_{i-1}, x_i > > 0 \\ \beta_{i-1} + y_i x_i & \text{si } y_i \cdot < \beta_{i-1}, x_i > \leq 0 \end{cases}$$

Algorithme du perceptron général

Perceptron - version générale

- **Paramètres :**
  - taux d'apprentissage η
  - rayon des observations $R = \max_{1 \leq i \leq n} \|x_i\|$

- **Algorithme :**
  1. **Initialisation** - $\beta_0 = 0,\ b_0 = 0$
  2. **Etape i** - si (x_i, y_i) est mal classé par l'hyperplan (b_{i-1}, β_{i-1}), alors :

$$\begin{aligned} \beta_i &= \beta_{i-1} + \eta y_i x_i \\ b_i &= b_{i-1} + \eta y_i^2 R^2 \end{aligned}$$

  sinon $\beta_i = \beta_{i-1},\ b_i = b_{i-1}$

Propriétés du perceptron

Théorème de Novikoff

Si les populations sont linéairement séparables alors l'algorithme du perceptron converge en un nombre fini $T \leq n$ d'étapes où :

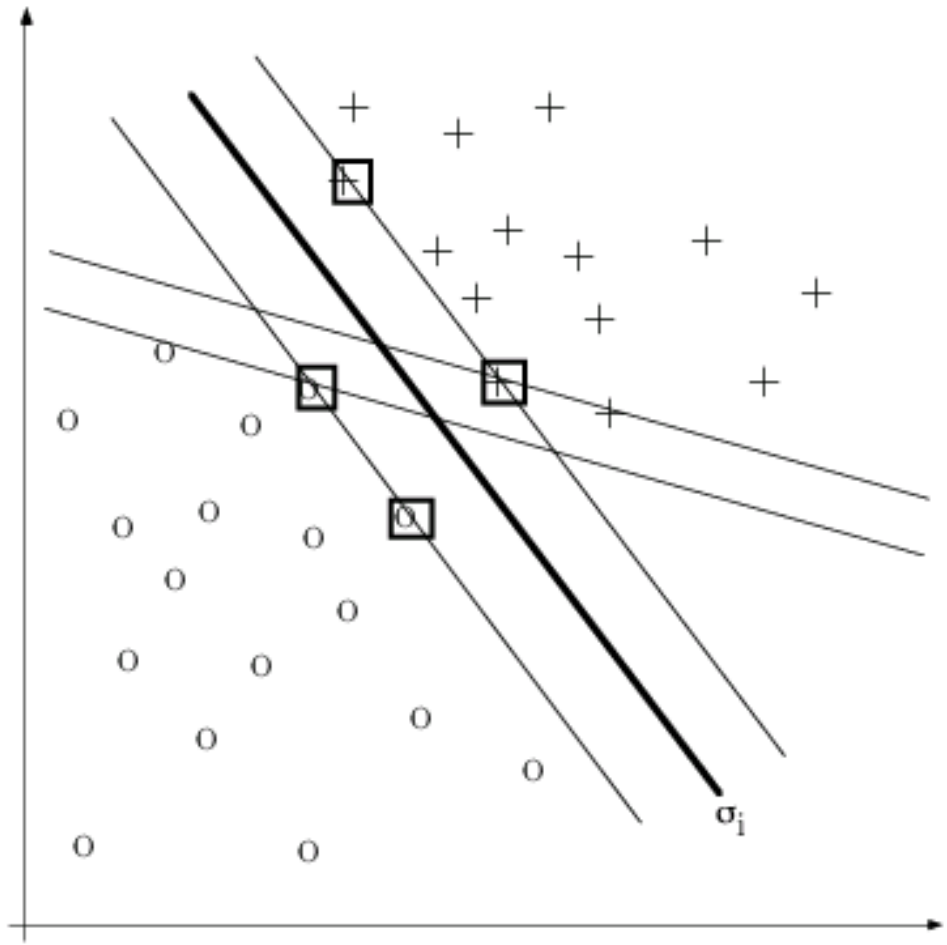
$$T \leq \frac{2R^2}{M^2}$$

avec $M = \min_{1 \leq i \leq n} \{y_i d(x_i, H^*)\}$ pour un certain séparateur H^*.

- **Défaut du perceptron :** mauvaise généralisation
- **Vertu du perceptron :** algorithme séquentiel (online)

Scénario 1 - hyperplan à bonne généralisation

Question : hyperplan se trouvant à distance maximale de chaque population ?



Hyperplan à marges optimales (Vapnik-Chervonenkis, 1964)

Problème d'optimisation

$$\max_{\beta \in \mathbb{R}^d,\ b \in \mathbb{R}} M$$

sous les contraintes :

$$\forall i = 1, \ldots, n\ , \quad y_i \cdot d(x_i, H) \geq M$$

On rappelle :

$$d(x_i, H) = \frac{1}{\|\beta\|}\left(b + < \beta, x_i >\right)$$

Problème quadratique

Contraintes :

$$\forall i = 1, \ldots, n \ , \quad y_i \cdot \frac{1}{\|\beta\|}\big(b + < \beta, x_i >\big) \geq M$$

On peut très bien poser : $M = 1/\|\beta\|$

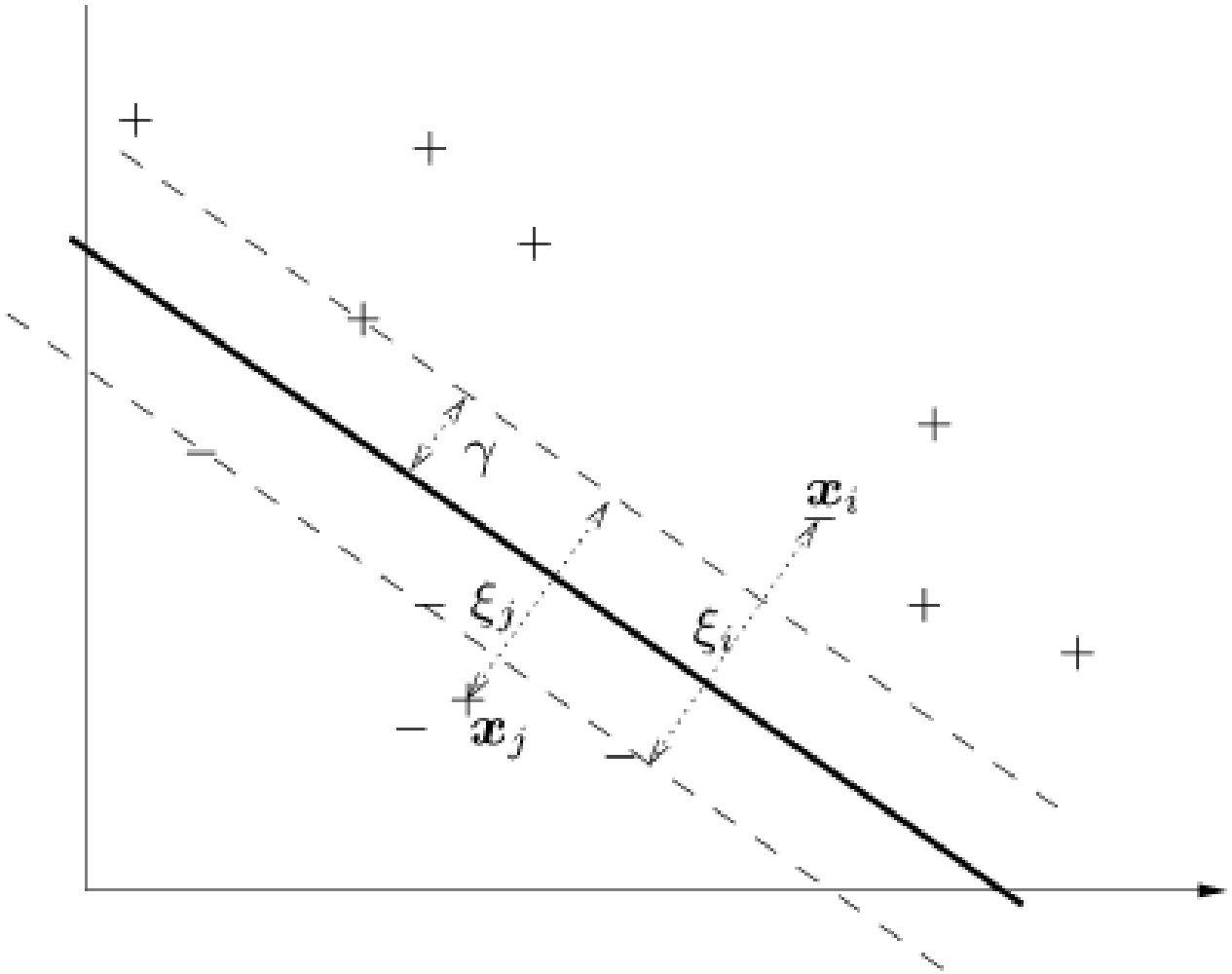
Formulation équivalente

$$\min_{\beta, b} \frac{1}{2}\|\beta\|^2$$

sous les contraintes :

$$\forall i = 1, \ldots, n \ , \quad y_i \cdot \big(b + < \beta, x_i >\big) \geq 1$$

Scénario 2 - Variables "ressorts"



Scénario 2 - Variables "ressorts" (suite)

On introduit n variables supplémentaires ("slacks" ou "ressorts") : $\xi = (\xi_1, \ldots, \xi_n)$ avec $\xi_i \geq 0, \forall i$

Nouveau problème d'optimisation

$$\min_{\beta, b, \xi} \frac{1}{2}\|\beta\|^2$$

sous les contraintes :

$$\begin{aligned} \forall i = 1, \ldots, n\ , \quad y_i \cdot \left(b + < \beta, x_i >\right) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \\ \sum_{i=1}^{n} \xi_i &\leq \Xi \end{aligned}$$

Formulation lagrangienne I

$$\min_{\beta,b,\xi} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \xi_i$$

sous les contraintes :

$$\begin{aligned} \forall i = 1,\ldots,n\ , \quad \xi_i &\geq 0 \\ \xi_i &\geq 1 - [y_i \cdot \left(b + < \beta, x_i >\right)] \end{aligned}$$

Formulation lagrangienne II

Multiplicateurs de Lagrange : $\alpha = (\alpha_1, \ldots, \alpha_n)$, $\mu = (\mu_1, \ldots, \mu_n)$

Formulation lagrangienne II

$$\min_{\beta,b,\xi} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\big(y_i\cdot\big(b+<\beta,x_i>\big)-(1-\xi_i)\big)+\sum_{i=1}^{n}\mu_i\xi_i$$

Conditions du premier ordre (gradient nul)

$$\begin{aligned} \beta &= \sum_{i=1}^{n}\alpha_i y_i x_i \\ \sum_{i=1}^{n}\alpha_i y_i &= 0 \\ \forall i = 1,\ldots,n\ , \qquad \alpha_i &= C + \mu_i \end{aligned}$$

Formulation duale

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j < x_i, x_j >$$

sous les contraintes :

$$\begin{aligned} \forall i = 1, \ldots, n \ , \qquad 0 &\leq \alpha_i \leq C \\ \sum_{i=1}^{n} \alpha_i y_i &= 0 \end{aligned}$$

On note $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)$ la solution de ce problème.

Conditions de Karush-Kuhn-Tucker

$$\forall i = 1, \ldots, n \ , \qquad \begin{aligned} \alpha_i \big(y_i \cdot f(x_i) - (1 - \xi_i)\big) &= 0 \\ y_i \cdot f(x_i) - (1 - \xi_i) &\geq 0 \\ \alpha_i + \mu_i &= C \\ \mu_i \xi_i &= 0 \\ \beta &= \sum_{i=1}^{n} \alpha_i y_i x_i \\ \sum_{i=1}^{n} \alpha_i y_i &= 0 \end{aligned}$$

Interprétation

Lien entre les coefficients et la position des observations

- si $\hat{\alpha}_i = 0$ alors $y_i \cdot f(x_i) \geq 1 \Rightarrow$ le point x_i est bien classé
  car $\mu_i = C > 0$ et on a $\xi_i = 0$
- si $0 < \hat{\alpha}_i < C$ alors $y_i \cdot f(x_i) = 1 \Rightarrow$ le point x_i est sur la frontière de la marge
  car $\mu_i > 0$ et $\xi_i = 0$
- si $\hat{\alpha}_i = C$ alors $y_i \cdot f(x_i) \leq 1 \Rightarrow$ le point x_i dépasse la frontière de la marge
  car $\mu_i = 0$ et donc $\xi_i \geq 0$

Phénomène remarquable !

En pratique, beaucoup de $\hat{\alpha}_i$ sont nuls !

Solution du problème

Définition

Les $\hat{\alpha}_i \neq 0$ correspondent aux vecteurs de support. On note I l'ensemble des indices parmi $\{1, \ldots, n\}$ correspondants.

Représentation de la solution

Fonction de décision :

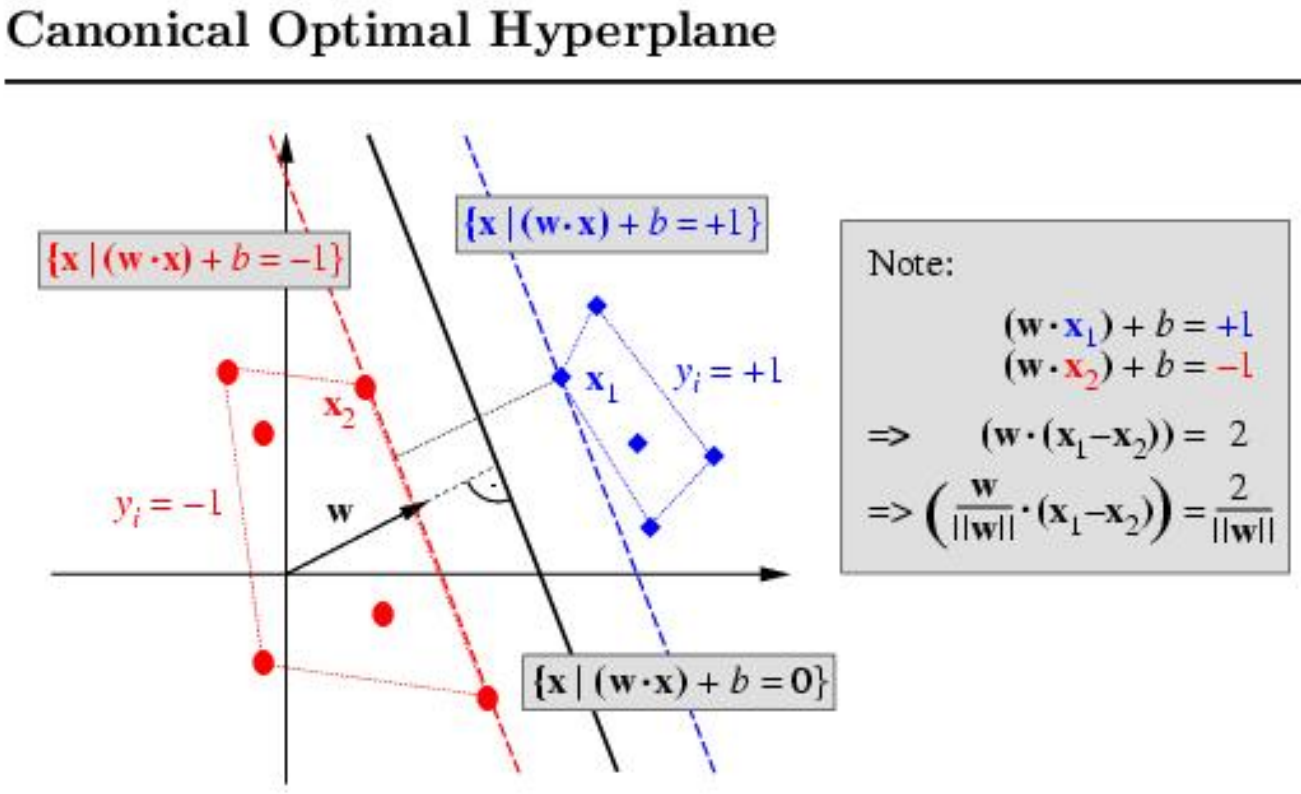
$$\hat{f}(x) = \hat{b} + \sum_{i \in I} \hat{\alpha}_i y_i < x_i, x >$$

où :

$$\hat{\beta} = \sum_{i \in I} \hat{\alpha}_i y_i x_i \ , \quad I = \{i \ : \ \hat{\alpha}_i \neq 0\}$$

$$\hat{b} = y_j - \sum_{i \in I} \hat{\alpha}_i y_i < x_i, x_j > \ , \quad \text{pour un certain } j \in I$$

Vecteurs de support



$\Rightarrow$ Représentation parcimonieuse ("sparse") des SVM

Scénario 3 - remarques préliminaires

- La plupart des problèmes de classification relèvent de séparations non-linéaires
- L'algorithme de construction de l'hyperplan à marges optimales ne fait intervenir les observations que sous la forme des produits scalaires $< x_i, x_j >$ pour tout i, j
- La fonction de décision dépend du produit scalaire entre le nouveau point x et les vecteurs de support x_i

L'astuce du noyau

- **Astuce du noyau ("kernel trick") :** l'algorithme de construction de l'hyperplan à marges optimales ne dépend des observations qu'au travers des coefficients de la matrice de Gram

$$K = (< x_i, x_j >)_{1 \leq i,j \leq n}$$

- **Méthodes à noyaux :** on remplace le produit scalaire canonique par un noyau positif

$$k \ : \ \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

et alors K devient la matrice des $k(x_i, x_j)$.

Formulation duale

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

sous les contraintes :

$$\begin{aligned} \forall i = 1, \ldots, n\ , \qquad 0 &\leq \alpha_i \leq C \\ \sum_{i=1}^{n} \alpha_i y_i &= 0 \end{aligned}$$

Fonction de décision :

$$\hat{f}(x) = \hat{b} + \sum_{i \in I} \hat{\alpha}_i k(x_i, x)$$

avec I indices des vecteurs de support.

Exemples de noyaux

- noyaux polynomiaux

$$\begin{aligned} k_r(x, x') &= (< x, x' >)^r \\ k_{r,c}(x, x') &= (< x, x' > + c)^r \end{aligned}$$

- noyau à fonctions de base radiales gaussiennes (RBF)

$$k_\sigma(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

- noyau sigmoïde (réseau de neurones)

$$k_{\kappa,\theta}(x, x') = \tanh\left(\kappa < x, x' > + \theta\right)$$

Problèmes pratiques

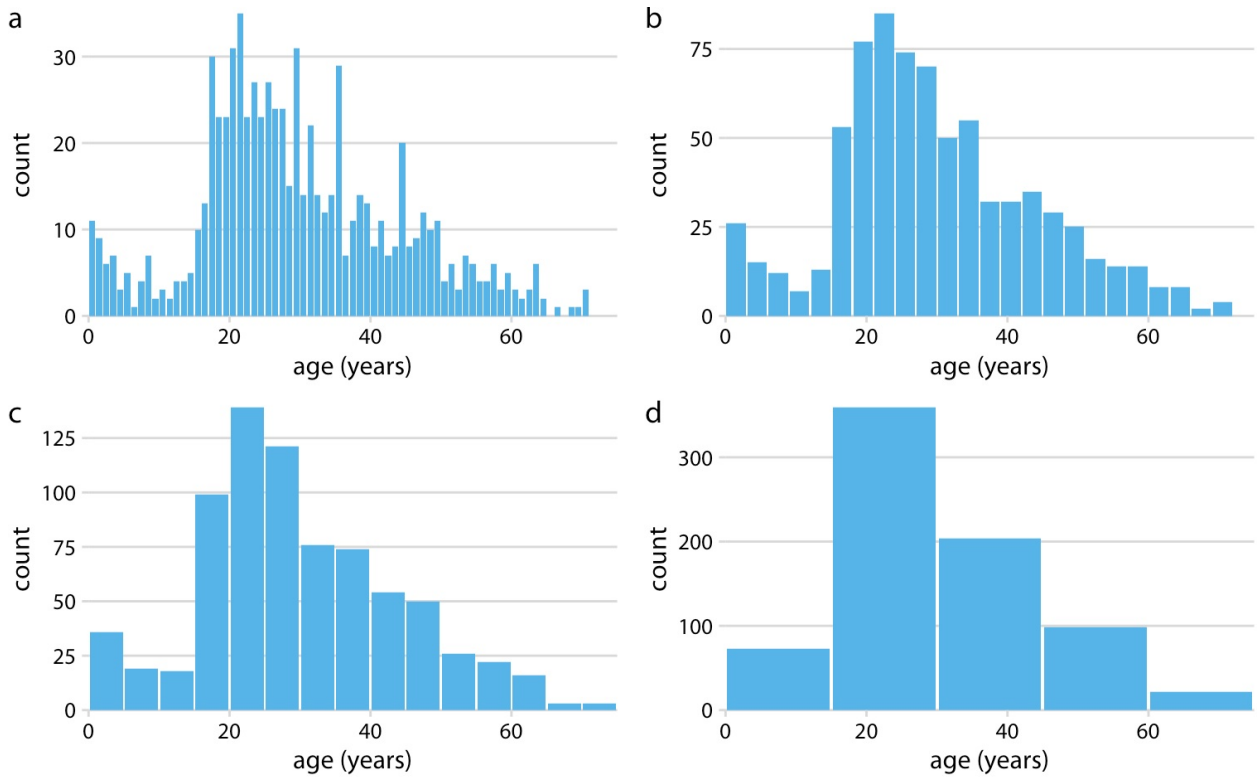
- Sélection d'un noyau
- Réglage des paramètres : utilisation d'une base de validation
- Mesures de performances : erreur sur la base de test
- Comparaison à d'autres méthodes
- Extensions :
  - problème de classification à plus de deux classes
  - cas de populations très disproportionnées

Modèles non linéaires basés sur la localité

Other forms of regularization

- General idea : Regularized function estimation without global optimization
- Two directions :
  - Local methods : nearest-neighbors and decision trees
  - Ensemble methods : bagging, boosting, random forests

Regularization without optimization
The case of histograms



Distribution of the age of the passengers of the Titanic with bins varying from 1 year to 15 years

Ingredients for that type of regularization

- Histograms use two general ideas of locality (bins) and averaging (piecewise constant function)
  - define local : which training data can be considered to be close to the point where a prediction has to be made ?
  - averaging (or voting if discrete outcome) : take the average of the values over each bin
- Regularization through hyperparameter selection : find the optimal bin size amounts to finding the right hypothesis class

From histograms to Machine Learning

- In the previous example, the objective was to estimate a density function from a sample drawn from this distribution (problem known in the literature as *nonparametric density estimation* or *kernel density estimation*)
- Density estimation can be seen as an *unsupervised learning problem*
- In the supervised setting, we establish the values of the function on every bin either by averaging (regression setup) or by voting (classification setup). The general terminology for averaging/voting is aggregating/combining.

Two popular types of local methods

- Nearest neighbors : local are the closest points
- Partition-based rules (also called *decision trees*) : local are the points within a cell from a partition of the input space only

Works for classification, regression and other problems... but here we will focus on classification

Problem considered
(Multiclass) Classification

- Given :
  - Consider a sample of classification data

$$(X_1, Y_1)...(X_n, Y_n)$$

where $X_i \in \mathbb{R}^d$ vector of independent variables,
$Y_i \in \{1, \ldots, C\}$ the label

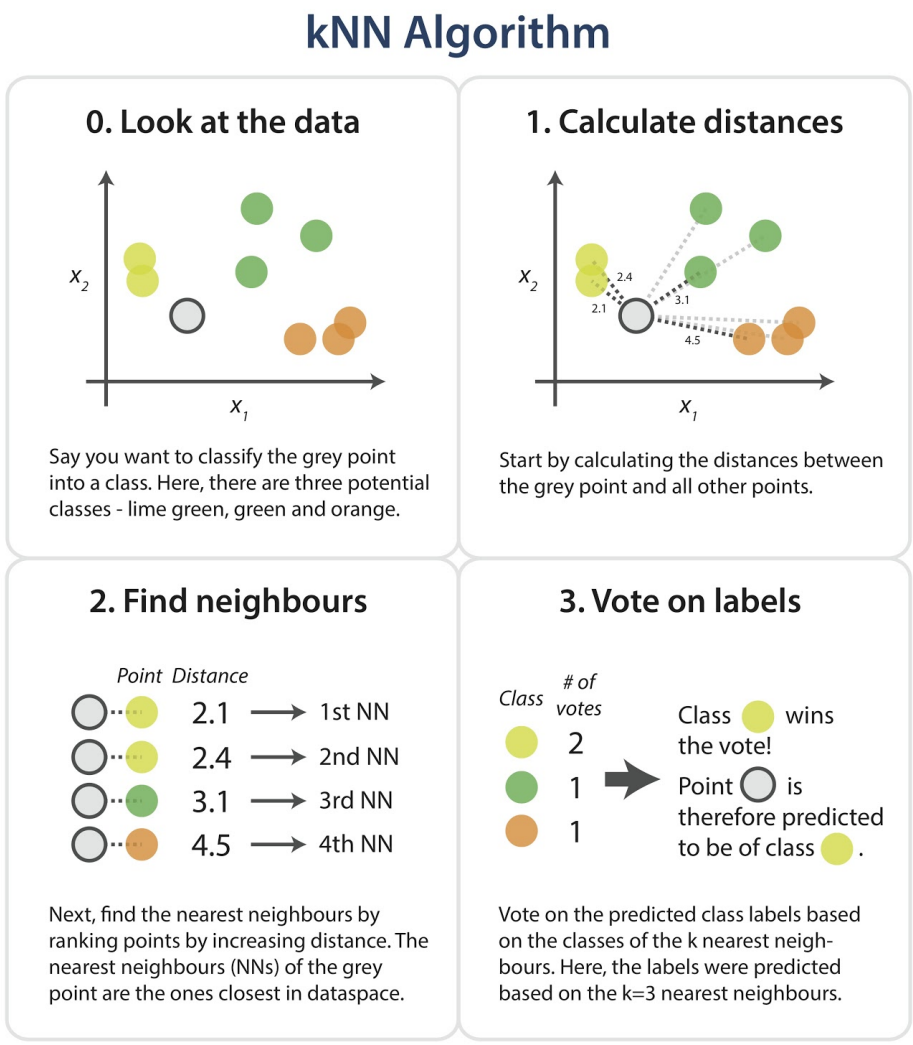
- Want :
  - to predict the label y at any position x

Local methods #1 : k-Nearest neighbors (k-NN)

k-Nearest Neighbor (1/4)
Principle of the k-NN algorithm

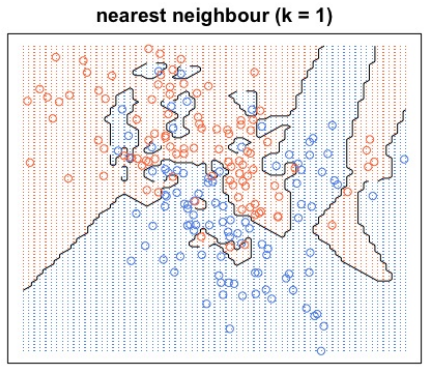
1. Compute distances
   - Compute pairwise distances $d(x, X_i)$ for all $i = 1, \ldots, n$
2. Sort training data
   - Sort the data points from the closest $X_{(1)}$ to the farthest $X_{(n)}$ (i.e. $d(x, X_{(1)}) \leq \ldots \leq d(x, X_{(n)})$
3. Prediction $\hat{h}(x, k)$ = Majority vote of the k-NN
   - Consider the labels $Y_{(1)}, \ldots, Y_{(k)}$ of the k closest points to x and take the majority vote
     $\hat{h}(x, k) = \arg\max_c \{\sum_{l=1}^{k} \mathbb{I}\{Y_{(l)} = c\}\}$

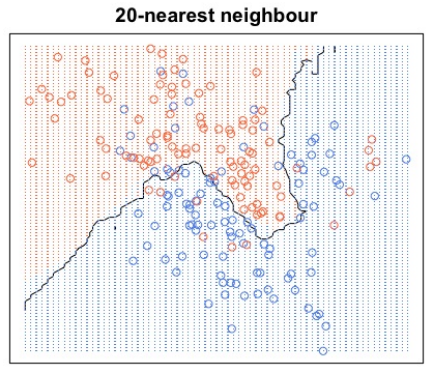
k-Nearest Neighbor (2/4)
Principle of the k-NN algorithm

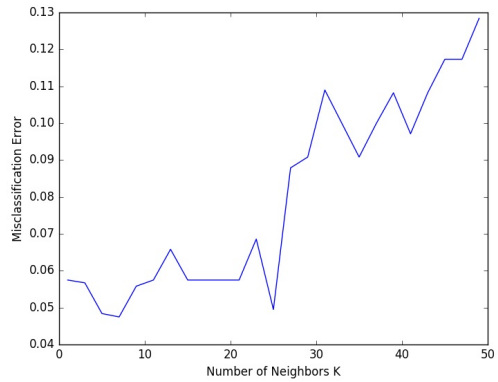


Nearest Neighbors (3/4)
Hyperparameters

- Choice of a distance d between points of $\mathbb{R}^d$
- Number k of Nearest Neighbors, estimated by cross-validation :







k-Nearest Neighbor (4/4)
Theory

- Recall : classification error $L(h) = \mathbb{P}(Y \neq h(X))$ and $L^* = \inf L$
- Consistency result :

$$\mathbb{E}L(\hat{h}(\cdot, k_n)) \to L^*$$

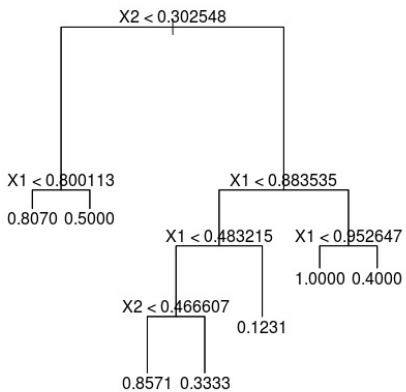
  under the condition : $k_n \to \infty$ and $k_n/n \to 0$ when $n \to \infty$
- No closed-form solution for optimal k_n (in practice, we use cross-validation)
- No theoretical clue on the choice of the distance (related to data representation and the physics of the problem)

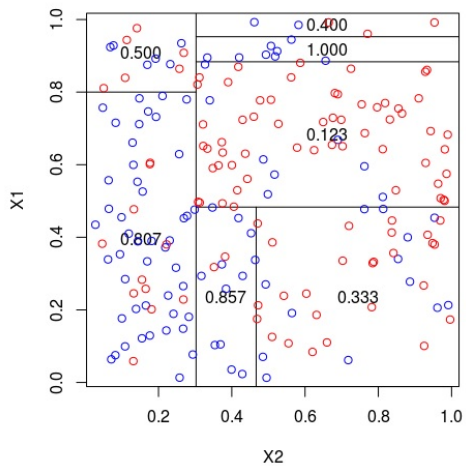
Local methods #2 : Partition-based (decision trees)

Partition-based classifier (1/4)
Computing the prediction for fixed partition

Denote the partition by $c = \bigcup_j \gamma_j$ with cells γ_j

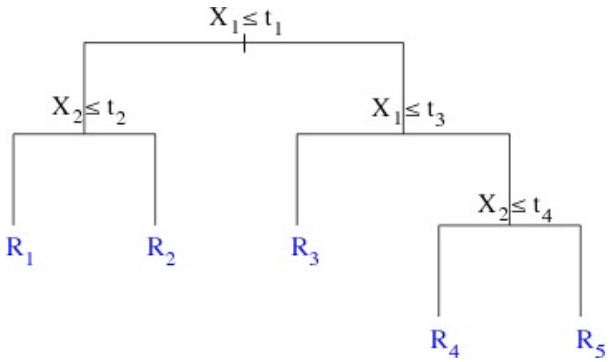
1. Find the cell $\gamma(x)$ where x falls
2. Consider the training data in the cell $\gamma(x)$
3. Prediction $\hat{h}(x, c)$ = Majority vote over the training data in cell $\gamma(x)$



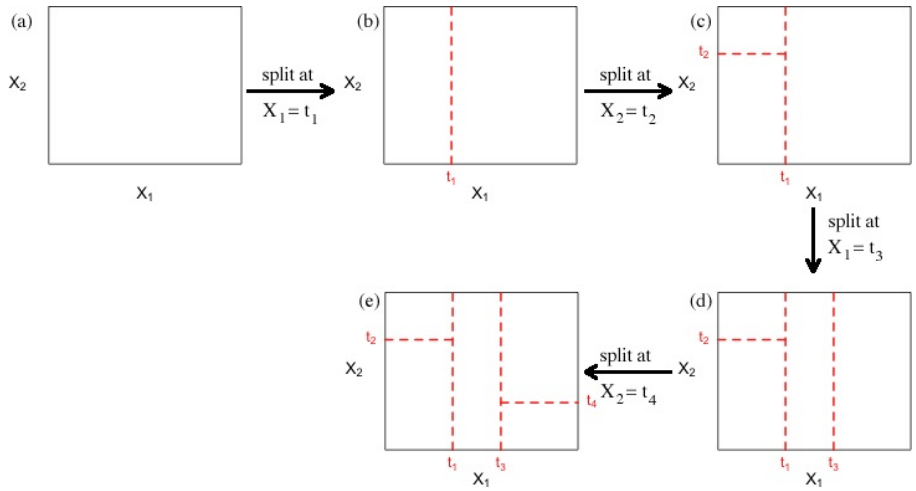


Partition-based classifier (2/4)
Building data-driven partitions

- Start with all the training data and find a (simple) classifier which minimizes some cost function
- Repeat the process with the subset of training data on each side of the frontier of the classifier $\longrightarrow$ this is called *recursive partitioning*



tree representation



recursive partitioning of the X-domain

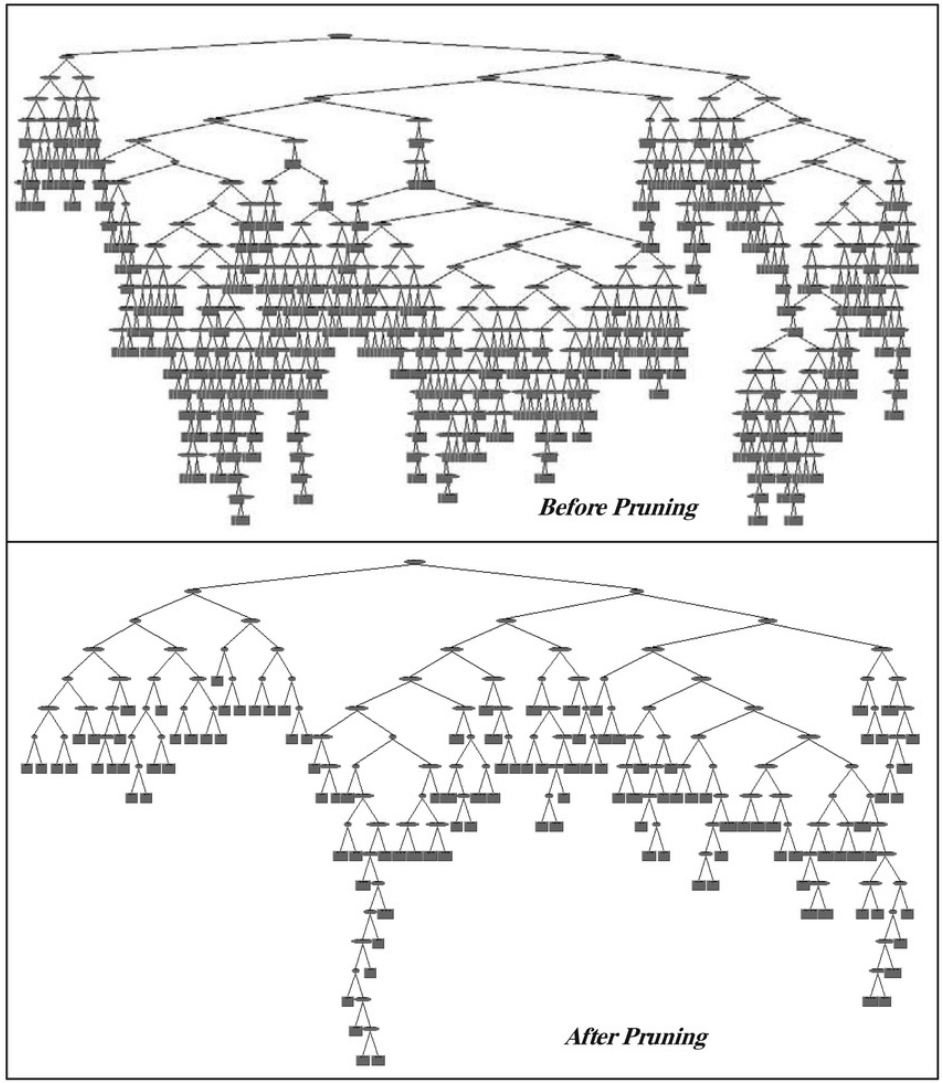
Partition-based classifier (3/4)
Hyperparameters

- Cost function optimized locally (at the cell level for the data within the cell)
- Number of minimal points in a cell
- Maximal depth of the tree or total number of cells estimated by pruning the tree - pruning amounts to explore the class of all subpartitions (subtrees) and optimize a penalized criterion of the form

$$\arg\min_{c} \hat{L}_n(h_c) + \lambda|c|$$

where $c \subset \hat{c}$ is the collection of subpartitions obtained from the learned partition by pruning from bottom to top

Pruning example



Partition-based classifier (4/4)
Theory

- Case of regular partitions with cells which are hypercubes of $\mathbb{R}^d$ with edges of length δ_n :

$$\mathbb{E}L(\hat{h}(\cdot, \delta_n)) \to L^*$$

under the condition : $n\delta_n^d \to \infty$ and $\delta_n \to 0$ when $n \to \infty$ (need enough data points in every cell and cell diameter go to zero as sample size grows)

- Case of data-driven partitions : VC and Rademacher theory applies

Take-home message on local methods

Major limitations :

- The k-Nearest Neighbor method requires to store all the training data in order to predict the label of new entries.
- Decision trees are extremely unstable.
- Both display prediction performance below state-of-the-art methods

Virtue of decision trees :

- Can handle missing/categorical data, scale change
- Can be expressed in terms of logical rule $\longrightarrow$ explainable machine learning

What can be saved from decision trees ?

Shallow and efficient Machine Learning algorithms : Ensemble methods

1. Bagging and Random Forests
2. Boosting

Motivation for ensembles
Pointers to other fields

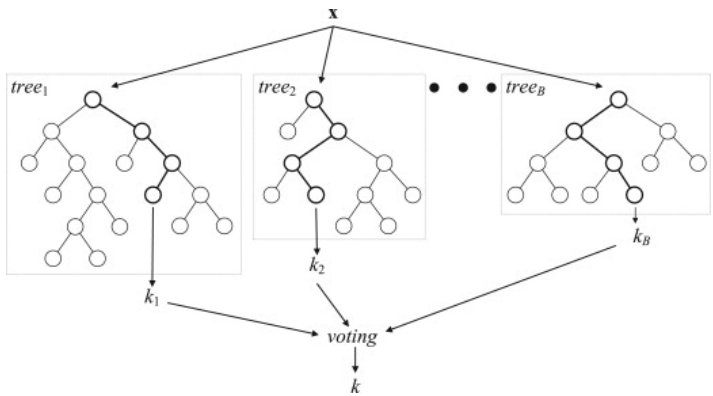
- Technology : the champions in data science competitions combine several methods to boost performance (e.g. BelKor team, winner of the Netflix challenge)
- Decision theory : Social choice theory
- Probability : Ergodic theorem
- Nonparametric statistics : aggregation estimators

Ensemble methods
Starting point

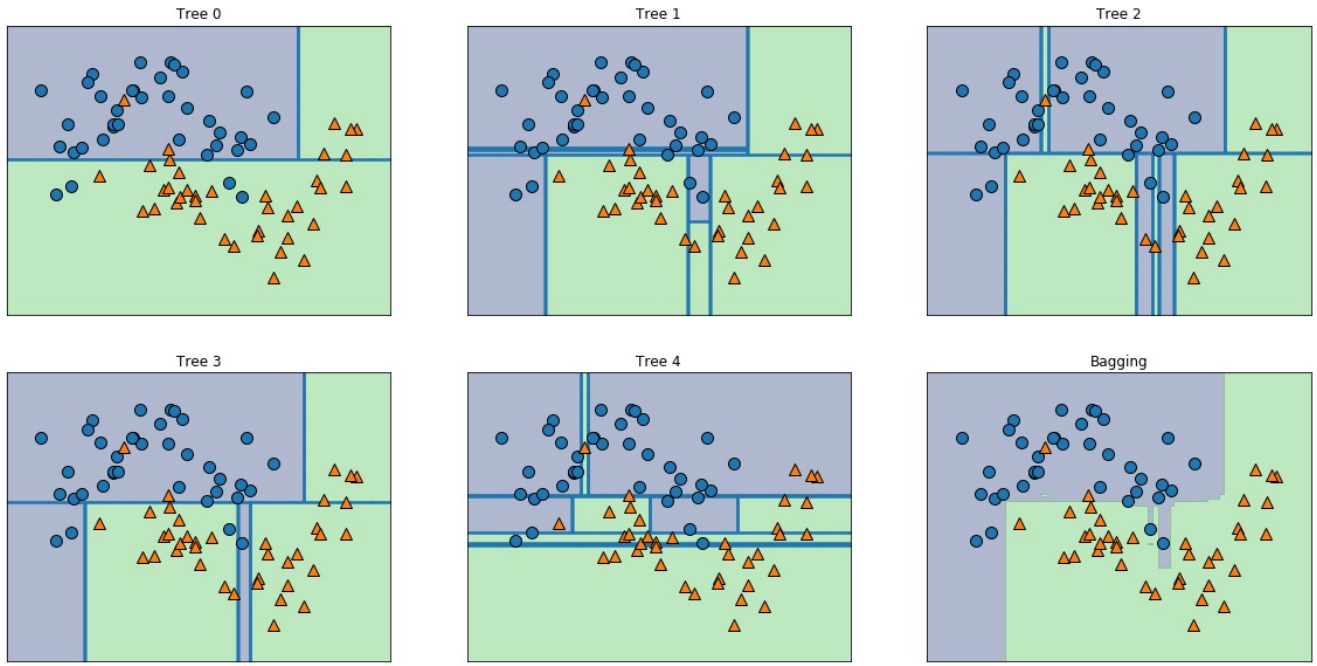
- Consider we already have a machine learning algorithm with reasonable performance that we want to improve, e.g. decision tree, k-NN, SVM, ...
- The idea of the ensemble is to generate different functions from the same training data and the same hypothesis space
- In the illustration coming next and most of the discussion, the basic hypothesis space is the one with decision trees obtained with orthogonal splits (such splits are called decision stumps).

Ensembles of decision trees
General principle

- Generate a collection of *weak* predictors (ensemble) obtained with a basic Machine Learning algorithm (e.g. decision tree)
- For every point x, compute their individual predictions
- Take an average or a majority vote of the individual predictions to determine the prediction of the ensemble



Ensembles of decision trees
Resulting classifier



Ensembles of decision trees
Three popular methods

- Bagging (Breiman, 1996)
- Random forests (Amit-Geman, 1997 ; Breiman, 2000)
- Boosting (Freund-Schapire, 1996)

Ensemble methods #1 : Bagging and Random Forests

Bagging and Random Forests
What is their hypothesis space ?

- Denote by $\mathcal{H}$ the base hypothesis space (for the not so brilliant algorithm we already have, e.g. decision trees)
- Denote by D_n the training data and assume that we can sample functions $\hat{h}_1, \ldots, \hat{h}_t$ (the ensemble) from $\mathcal{H}$ conditionally to D_n
- With an ensemble of T functions, the output of bagging/random forests is the average of those "random" (generated based on the data) functions :

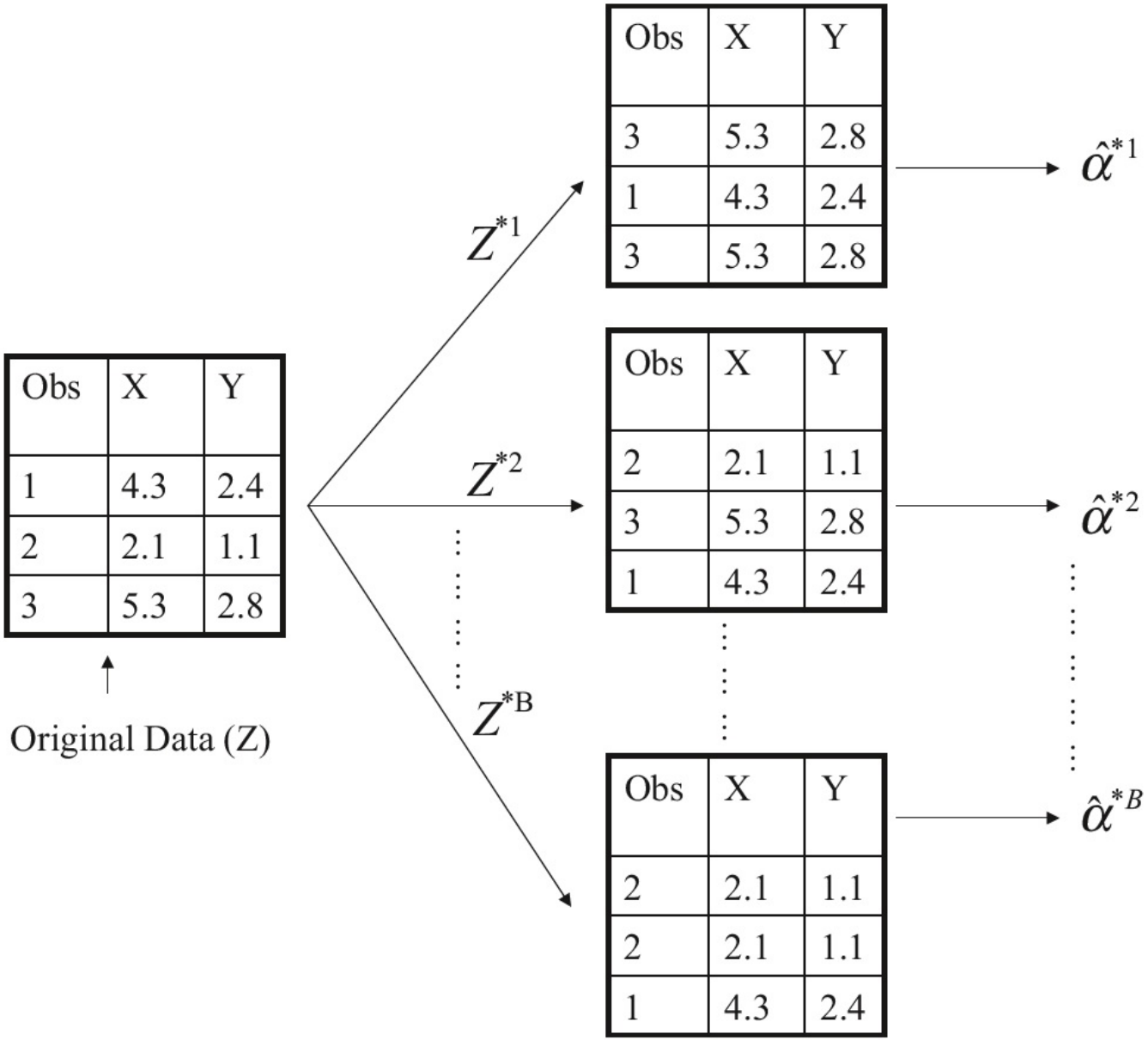
$$\hat{f}_T = \frac{1}{T} \sum_{t=1}^{T} \hat{h}_t$$

- The hypothesis space for those methods is the linear span of the base hypothesis space $\mathcal{H}$. This can be a huge space !

How to generate the ensemble ?
Bootstrap and aggregation

- Bagging and random forests rely on bootstrap samples of the training data
- They differ by some different specifications of the recursive partitioning procedure to build each tree (no pruning involved)

What is bootstrap in general ?



Bagging Theory

- Consistency result for some idealized version of bagging
- Most important ! Bagging can render inconsistent rules consistent !
  - Biau, Devroye and Lugosi (2008) have considered bagging applied to 1-NN, given that 1-NN is inconsistent in general classification scenarios (except zero-noise or pure random labels)
  - Bagging applied to 1-NN classifier is consistent under some reasonable conditions on the sampling process

Ensemble methods #2 : Boosting

Historical perspective on Boosting

- Original paper : Freund, Y. and Schapire, R. E. (ICML, 1996).
- Interpretation of the optimization problem solved as stochastic gradient descent : Friedman, J. H. (CSDA, 2002).
- Wald Memorial lecture (IMS, 2000) : Leo Breiman declares that *"understanding Boosting is the most important problem in Machine Learning"*
- Proof of boosting consistency : Lugosi, G. and Vayatis, N. (Special issue with discussion of the Annals of Statistics, 2004).
- Xgboost, a scalable implementation : Chen, T. and Guestrin, C. (ACM SIGKDD, 2016).

Boosting (1/7)
Principle

- **Input**
  - Data sample $D_n = \{(X_i, Y_i) \ : \ i = 1, \ldots, n\}$ with classification data $\{-1, +1\}$
  - Base hypothesis class $\mathcal{H}$ of *weak* classifiers such as decision trees (assumed to be symmetric, i.e. $h \in \mathcal{H}$ iff $-h \in \mathcal{H}$)
- **Iterations** $t = 1, \ldots, T$.
  - Compute weights $w_t > 0$ and weak classifiers $\widehat{h}_t \in \mathcal{H}$
- **Output.**
  - The Boosting classifier takes the sign of the following linear combination of weak classifiers : $\widehat{f}_n(x) = \sum_{t=1}^{T} w_t \widehat{h}_t(x)$

Boosting (2/7)
Notations

- Boosting distributions on the data : sequence of discrete probability distributions over $\{1, \ldots, n\}$ denoted by Π_t, $t \geq 1$
- Weighted training error : for any weak classifier $h \in \mathcal{H}$ and for $t \geq 1$

$$\widehat{\varepsilon}_t(h) = \sum_{i=1}^{n} \Pi_t(i) \mathbb{I}\{h(X_i) \neq Y_i\}$$

Boosting (3/7)
Original Algorithm : AdaBoost

1. **Initialization.** Π_1 is the uniform distribution on $\{1, \ldots, n\}$
2. **Boosting iterations.** For $t = 1, \ldots, T$, find the weak classifier such that :

$$\widehat{h}_t = \arg\min_{h \in \mathcal{H}} \widehat{\varepsilon}_t(h)$$

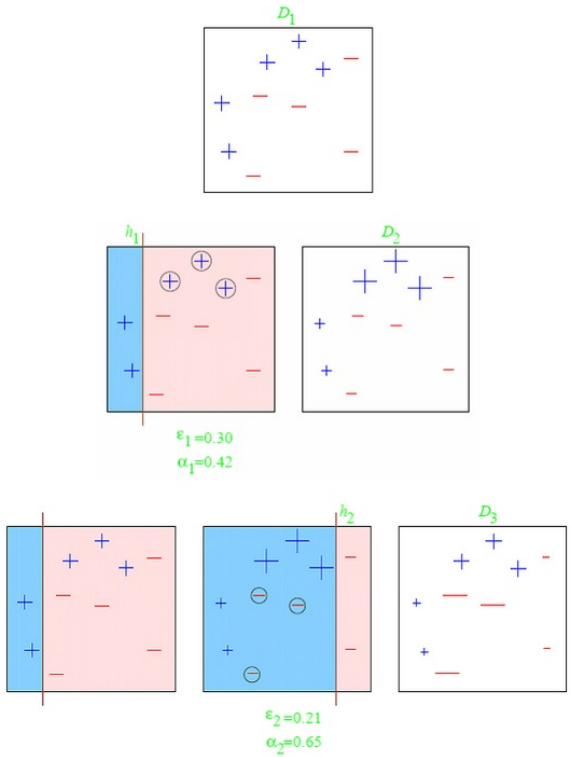
then set $e_t = \widehat{\varepsilon}_t(\widehat{h}_t)$ and take the weight to be

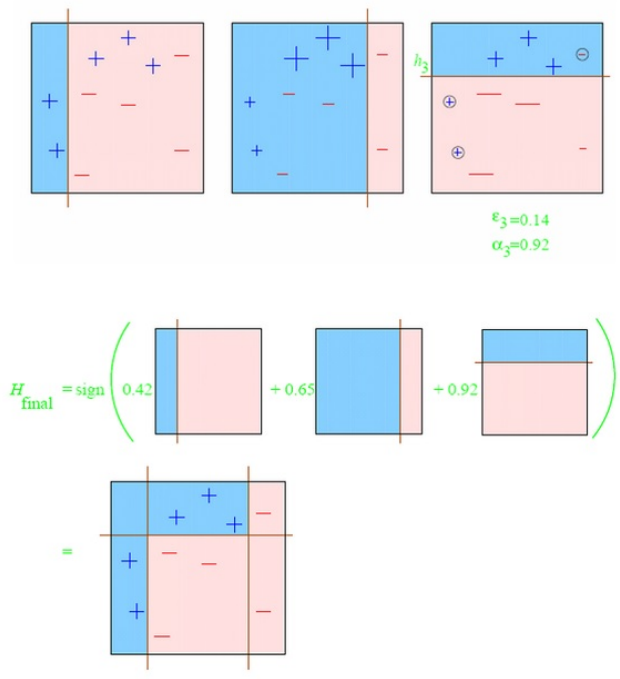
$$w_t = \frac{1}{2} \log\left(\frac{1 - e_t}{e_t}\right)$$

3. **Boosting distribution update.** For any $i = 1, \ldots, n$,

$$\Pi_{t+1}(i) \propto \Pi_t(i) \exp\left(-w_t Y_i \cdot \widehat{h}_t(X_i)\right)$$

Boosting (4/7)
Example





Boosting (5/7)

- Boosting can be interpreted as a functional gradient descent on the following functional :

$$\hat{A}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \exp\left(-Y_i f(X_i)\right)$$

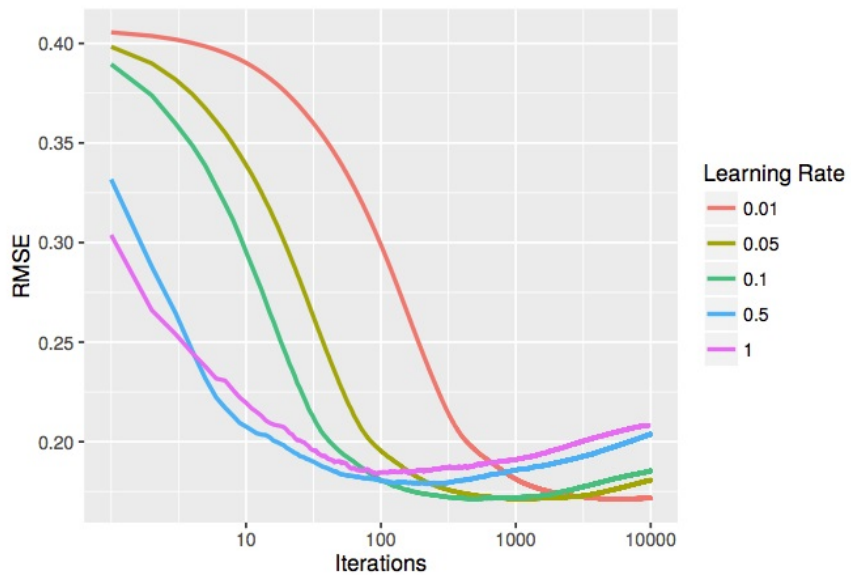
  where f is taken in a hypothesis space which is the linear span of 'simple' set $\mathcal{H}$ of classifiers.

- Exercise : why ?

  Refer to : J. Friedman, ?Greedy Function Approximation : A Gradient Boosting Machine ?, The Annals of Statistics, Vol. 29, No. 5, 2001.

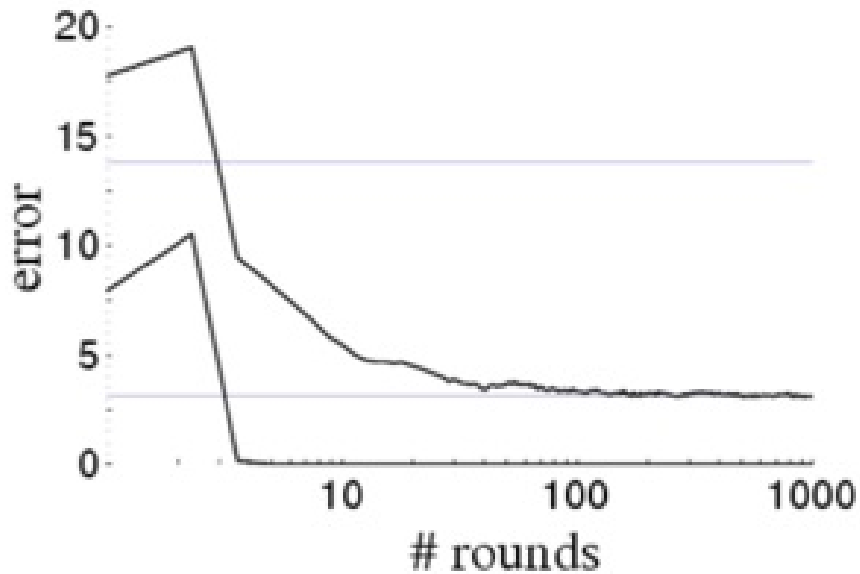
Boosting (6/7)
Hyperparameters for Gradient Boosting

- The number T of iterations : the bigger, the higher the chance of overfitting.
- The stepsize η is fixed : decreasing learning rate tends to improve generalization performance.



Boosting (7/7)
A mystery not fully explained yet...

The test error continues to drop along the iterations even though the training error is zero $\longrightarrow$ Regularization effect thanks to averaging ? ?



Packages

- Python : scikit-learn
- R :
  - `rpart` : recursive partitioning
  - `caret` : classification and regression training (SVM, random forest...)
  - `xgboost` : extreme gradient boosting

Les principes inférentiels

Notion de risque

- **Une mesure d'erreur :**

$$L(g) = \mathbb{P}\{Y \cdot g(X) < 0\} = \mathbb{E}\left(\mathbb{I}_{[Y \cdot g(X) < 0]}\right)$$

- **Classifieur et erreur de Bayes :**

$$\begin{aligned} g^* &= \arg\min_g L(g) = \operatorname{sgn}\left(\eta - \frac{1}{2}\right) \\ L^* &= L(g^*) \end{aligned}$$

- **Critère empirique :**

$$\hat{L}_n(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}_{[Y_i \cdot g(X_i) < 0]}$$

Minimisation du risque empirique (ERM)

La théorie de Vapnik ('95 $\rightarrow$) a permis de développer des stratégies consistantes pour la classification de données en grande dimension.

Inconvénients de l'ERM

- algorithmique : problème NP-difficile .
- contrôle de la complexité : propriété de Glivenko-Cantelli pour éviter le surapprentissage (overfitting).

Mais :

Les méthodes efficaces construisent les estimateurs dans des classes massives !

Algorithmes efficaces

1. **Support Vector Machines** - Vapnik (1995)
   - Kernel trick : envoyer les données dans un espace de Hilbert où les données soient (presque) linéairement séparables
   - Hyperplan à marge maximale : optimization convexe sous contraintes quadratiques
2. **Boosting** - Freund (1990) - Freund, Schapire (1996)
   - on commence avec une classe $\mathcal{H}$ de classifieurs simples
   - puis on construit itérativement une combinaison linéaire de classifieurs simples qui fait décroître l'erreur empirique

"Boosting is the best off-the-shelf classifier in the world" - Breiman, 1996.

Caractéristique commune

Si on fait abstraction :

1. des intuitions géometriques
2. de la dynamique particulière de chaque algorithme, alors :

Boosting et SVM peuvent s'envisager comme des :

procédures de minimisation d'un risque convexe pénalisé dans des espaces fonctionnels massifs.

Elles sont caractérisées par

- des classes d'estimateurs différentes,
- des risques différents,
- des pénalités différentes.

Classe $\mathcal{F}$ d'estimateurs

- **Support Vector Machines** - soit k un noyau positif

$$\mathcal{F} = \left\{ f = \sum_{i=1}^{+\infty} \alpha_i k(x_i, \cdot) \ : \ \alpha_i \in \mathbb{R}, x_i \in \mathcal{X} \right\}$$

- **Boosting** - soit $\mathcal{G}$ famille de classifieurs simples de VC dimension V finie

$$\mathcal{F} = \left\{ f = \sum_{i=1}^{+\infty} w_i g_i \ : \ w_i \in \mathbb{R}, g_i \in \mathcal{G} \right\}$$

Les deux algorithmes construisent des combinaisons linéaires de fonctions.

Cas des SVM - Retour sur la formulation lagrangienne I

On pose

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$$

$$\|f\|_{\mathcal{F}} = \sqrt{\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)}$$

Formulation lagrangienne I

$$\min_{f \in \mathcal{F}} \frac{1}{2} \|f\|_{\mathcal{F}}^2 + C \sum_{i=1}^{n} \xi_i$$

sous les contraintes :

$$\forall i = 1, \ldots, n \ , \quad \xi_i \geq (1 - y_i \cdot f(x_i))_+$$

Interprétation - "hinge loss"

On pose : $\varphi(x) = (1+x)_+$ "hinge loss"

Minimisation d'un risque pénalisé

$$\min_{f \in \mathcal{F}} \frac{1}{2}\|f\|_{\mathcal{F}}^2 + C\sum_{i=1}^{n} \varphi(-y_i f(x_i))$$

Commentaires :

- formulation importante pour la théorie statistique
- moins commode pour l'optimisation que la formulation duale
- $\|f\|_{\mathcal{F}}$ fournit une mesure de régularité de f

Cadre général

- **Fonction de décision :**

$$f \ : \ \mathcal{X} \to \mathbb{R}$$

- **Classifieur :**

$$g(x) = g_f(x) = \operatorname{sgn}(f(x)) \ \in \{-1, +1\}$$

- **Critère naturel :**

$$L(f) = \mathbb{P}\{Y \cdot f(X) < 0\} = \mathbb{E}\left\{\mathbb{I}_{[Y \cdot f(X) < 0]}\right\}$$

φ-risque

- **Fonction de perte :** φ convexe, positive, telle que $\varphi(x) \geq \mathbb{I}_{\mathbb{R}_+}(x)$
- **Critère pratique (φ-risque) :**

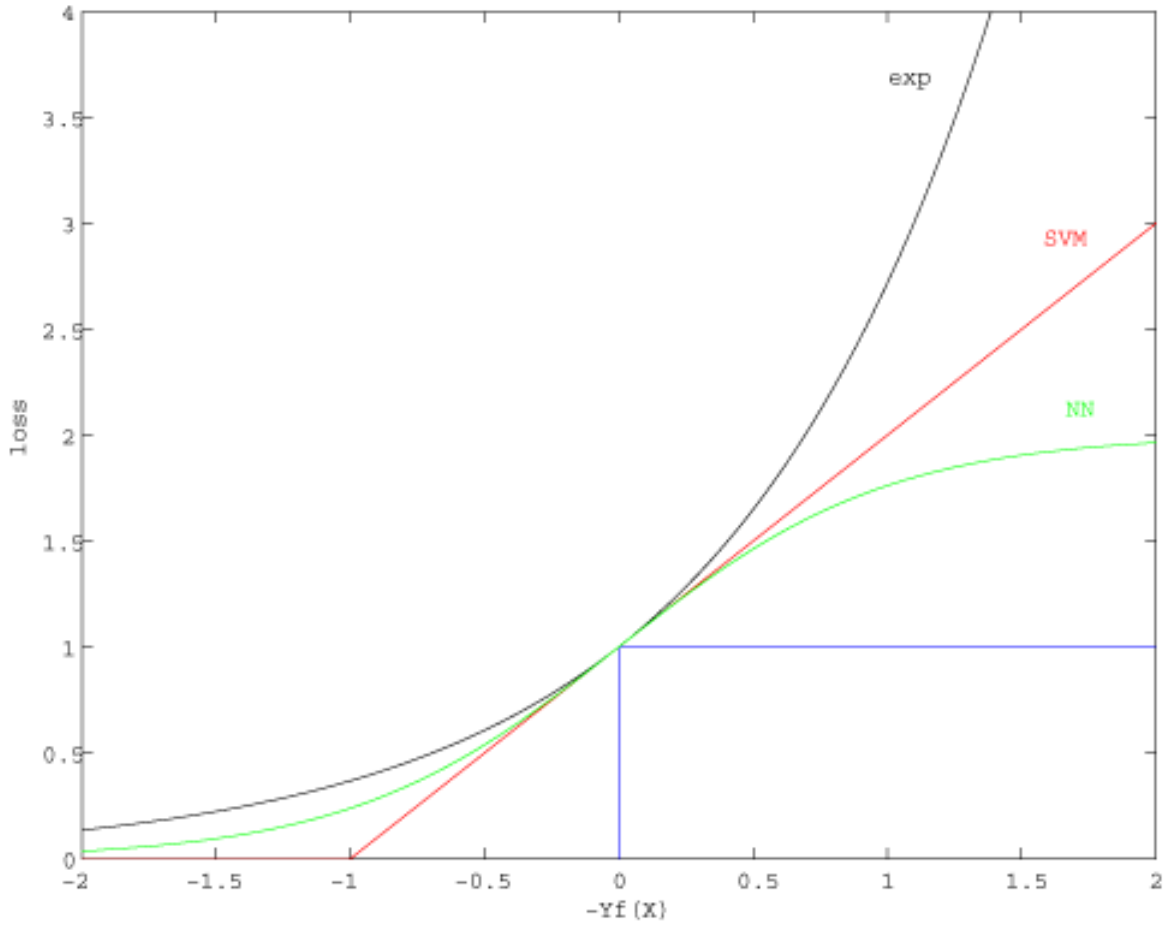
$$\begin{aligned} A(f) &= \mathbb{E}\varphi(-Yf(X)) \\ &= \mathbb{E}\left[\eta(X)\varphi(-f(X)) + (1-\eta(X))\varphi(f(X))\right] \end{aligned}$$

où $\eta(X) = \mathbb{P}\{Y = 1 | X = x\}$

Question : minimiser A revient-il à minimiser L ?

On a seulement que : $L(f) \leq A(f)$...

Fonctions de perte



Fonctions cibles

- Minimiseur de la fonctionnelle A noté f^*
- Minimum du φ-risque : $A^* = \min_f A(f) = A(f^*)$
- On peut montrer que $\operatorname{sgn}(f^*) = g^*$ le classifieur optimal
- On peut également montrer que l'excès de risque en erreur de classification $L(f) - L^*$ est contrôlé par $A(f) - A^*$
- Exemples :
  - perte exponentielle (boosting)

$$f^*(x) = \frac{1}{2} \log \left(\frac{\eta(x)}{1 - \eta(x)} \right)$$

  - "hinge loss" (SVM)

$$f^*(x) = \operatorname{sgn}(\eta(x) - 1/2) = g^*(x)$$

Further topics

- Explainability
- Reinforcement Learning
- Adapting these concepts to other problems :
  - either in terms of objectives : such as preference learning, scoring, ranking, anomaly detection, novelty detection...
  - or in terms of learning setups : online learning, unsupervised learning, transfer learning, multitask learning, budgeted learning, active learning...