Apprentissage supervisé

Aurélien Garivier

aurelien.garivier@gmail.com

12 et 13 juin 2013

Roadmap

Qu'est-ce que l'apprentissage ?

Apprentissage (*machine learning*) = discipline visant à la construction de règles d'inférence et de décision pour le traitement automatique des données.

Variantes : machine learning, fouille de données (data-mining).

SAS Entreprise Miner vendu avec le slogan :

Data Mining

Comment trouver un diamant dans un tas de charbon sans se salir les mains

Trois grands types d'apprentissage

1. **Apprentissage supervisé :**
   A partir d'un échantillon d'apprentissage $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, inférer la relation entre x et y.
   Synonymes : discrimination, reconnaissance de formes
   Voc : x_i = caractéristique = feature = variable explicative

2. Apprentissage non supervisé :
   A partir d'un échantillon d'apprentissage $\mathcal{D}_n = \{x_1, \ldots, x_n\} \subset \mathcal{X}$, partionner $\mathcal{X}$ en classes pertinentes.
   Voc : ⚠ parfois appelé 'Classification' en français (jamais en anglais)

3. Apprentissage séquentiel :
   A chaque date n, prendre une décision à l'aide des données passées.

Différentes approches :

- Approche structurelle, logique et logique floue.
- **Apprentissage statistique :** modélisation probabiliste des données à des fins *instrumentales*.
- Apprentissage séq. robuste (théorie des jeux, optim. convexe séq.).

Dans tous les cas :

- science relativement *récente*
- à la frontière des *mathématiques* et de l'*informatique* (et intelligence artificielle)
- en *évolution rapide et constante* avec les technologies =
  - nouveaux moyens (de calcul)
  - nouveaux problèmes...

Problèmes d'apprentissage supervisé

- Détecteur de spam
- Risque de crédit
- Prédiction des pics d'ozone
- Aide au diagnostic médical (ex : Breast Cancer)
- Aide au pilotage
- Moteurs de recommandation, apprentissage sur les graphes
- etc. . .

Algorithme de d'apprentissage

Cadre classique (batch) :

Données : *échantillon d'apprentissage* $(x_k, y_k)_{1 \leq k \leq n}$ constitué d'observations que l'on suppose représentatives et sans lien entre elles.

Objectif : prédire les valeurs de y associées à chaque valeur possible de $x \in \mathcal{X}$.

Classification : $\mathcal{Y}$ discret (typiquement, binaire) pour chaque valeur de $x \in \mathcal{X}$, il faut prédire la classe la plus souvent associée.

Régression : $\mathcal{Y}$ continu, voire plus (fonctionnel).

Règle de classification : à partir de l'échantillon d'apprentissage, construire $f_n : \mathcal{X} \to \mathcal{Y}$ associant, à chaque entrée possible x, la classe y prédite pour elle.

Apprentissage vs. Statistiques ?

- Les données sont des réalisations de v.a. iid de même loi que

$$(X, Y) \sim P_{(X,Y)} \in \mathcal{M}$$

- Modèle *instrumental* : on ne pense pas que ça soit vrai, ni que $P \in \mathcal{M}$
- Pas de "vrai modèle", de "vraies valeurs du paramètre", etc.
- Consistance statistique sans intérêt
- Souvent ($\neq$ étude statistique) données disponibles avant intervention du statisticien (malheureusement)
- Tous les coups sont permis, seul critère = efficacité prédictive
- Classification :

$$R_n = \mathbb{E}_{(X_1,Y_1),\ldots,(X_n,Y_n)}\left[P_{(X,Y)}(f_n(X) \neq Y)\right] .$$

- Régression : typiquement

$$R_n = \mathbb{E}_{(X_1,Y_1),\ldots,(X_n,Y_n)}\left[E_{(X,Y)}\left((Y - f_n(X))^2\right)\right] .$$

Paramétrique vs. Non-paramétrique !

- Théoriquement, quand un modèle est vrai il est optimal de l'utiliser :
  - Théorème de Gauss-Markov : parmi les estimateurs sans biais, celui des moindres carrés est de variance minimale
  - MAIS on peut avoir intérêt à sacriier du biais contre de la variance !

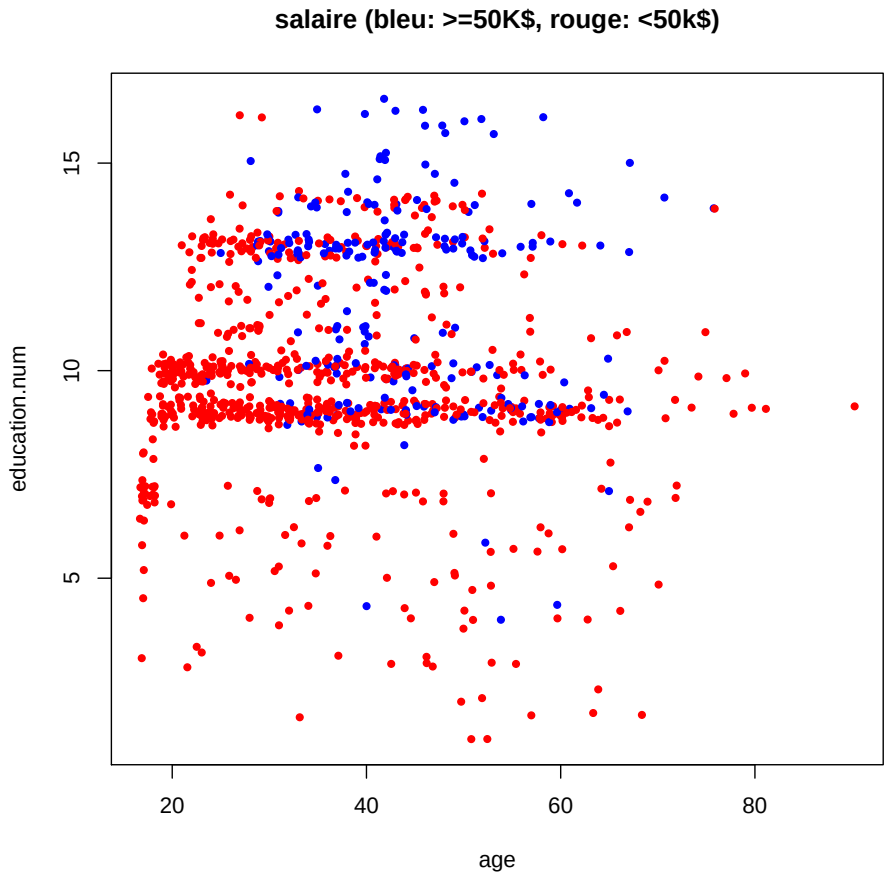
$\Longrightarrow$ Même quand il y en a un 'vrai' modèle, on n'a pas forcément intérêt à l'utiliser

- Des approches non-paramétriques peuvent avoir une efficacité proche :
  - cf Test de Student vs Mann-Whitney
  - exemple : k-NN versus régression polynomiale
- ... et ils sont beaucoup plus robustes !

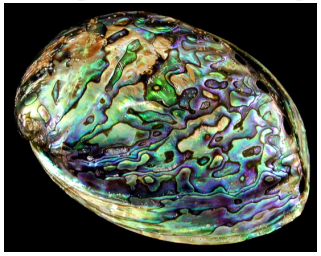
Exemple de problème de classification

United States™ Census Bureau

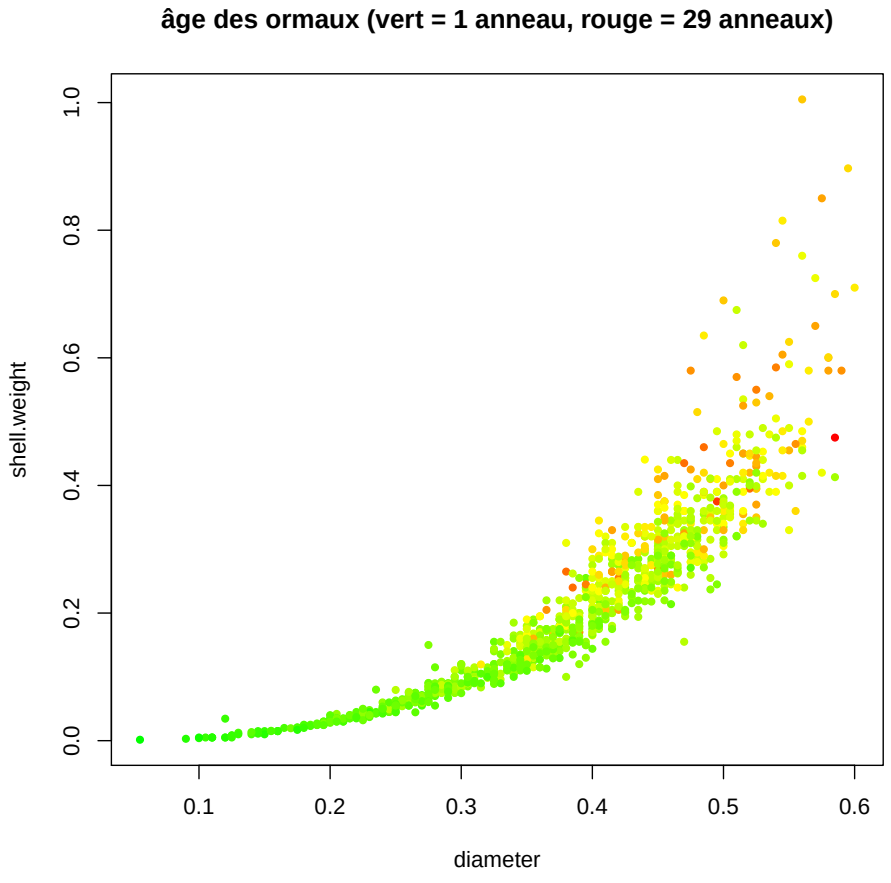
Objectif : prédire qui gagne plus de 50k$ à partir de données de recensement.



Exemple de problème de régression



Prédire l'âge d'un ormeau (abalone) à partir de sa taille, son poids, etc.



Nous disposons d'un ensemble d'observations. Les caractéristiques ou variables $\mathbf{X} = (\mathbf{X}^1, \ldots, \mathbf{X}^p)$ dites explicatives ont été observées sur un ensemble de n objets, individus ou unités statistiques.

- Premier travail : mener une exploration statistique des données.
  - allure des distributions,
  - présence de données atypiques,
  - corrélations et cohérence,
  - transformations éventuelles des données,
  - description multidimensionnelle,
  - classification.
- Deuxième travail : modélisation statistique ou encore d'apprentissage pour la prédiction d'un variable cible Y par les variables explicatives $(\mathbf{X}^1, \ldots, \mathbf{X}^p)$.

L'enchaînement de ces étapes (exploration puis apprentissage) constitue le fondement de la fouille de données.

But : Déterminer la stratégie à mettre en oeuvre pour aboutir au bon apprentissage ou au bon modèle prédictif à partir des données observées.

Contrairement à une démarche statistique traditionnelle dans laquelle l'observation des données est intégrée à la méthodologie (plannification expérimentale), les données sont ici préalable à l'analyse.

Néanmoins, il est clair que les préoccupations liées à leur analyse et à son objectif doivent intervenir le plus en amont possible pour s'assurer quelques chances de succès.

Étapes de la fouille de données

1. Extraction des données avec ou sans apprentissage : techniques de sondage appliquées ou applicables à des bases de données.

2. Exploration des données
   - pour la détection de valeurs aberrantes ou seulement atypiques, d'incohérences,
   - pour l'étude des distributions, des structures de corrélation, recherche de typologies,
   - pour des transformations de données.

3. Partition aléatoire de l'échantillon (apprentissage, validation, test) en fonction de sa taille et des techniques qui seront utilisées pour estimer une erreur de prédiction en vue des choix de modèles, choix et certification de méthode.

Étapes de la fouille de données (suite)

4. Pour chacune des méthodes considérées : modèle linéaire général (gaussien, binomial ou poissonien), discrimination paramétrique (linéaire ou quadratique) ou non-paramétrique, *k* plus proches voisins, arbre, réseau de neurones (perceptron), support vecteur machine, combinaison de modèles (bagging, boosting)
   - estimer le modèle pour une valeur donnée d'un paramètre de complexité : nombre de variables, de voisins, de feuilles, de neurones, durée d'apprentissage, largeur de fenêtre...
   - optimiser ce paramètre (sauf pour les combinaisons de modèles affranchies des problèmes de sur-apprentissage) en fonction de la technique d'estimation de l'erreur retenue : échantillon de validation, validation croisée, approximation par pénalisation de l'erreur d'ajustement.

Étapes de la fouille de données (suite et fin)

5. Comparaison des modèles optimaux obtenus (un par méthode) par estimation de l'erreur de prédiction sur l'échantillon test ou, si la présence d'un échantillon test est impossible, sur le critère de pénalisation de l'erreur (Akaike par exemple) s'il en existe une version pour chacune des méthodes considérées.

6. Itération éventuelle de la démarche précédente (validation croisée), si l'échantillon test est trop réduit, depuis l'étape 3. Partitions aléatoires successives de l'échantillon pour moyenner sur plusieurs cas l'estimation finale de l'erreur de prédiction et s'assurer de la robustesse du modèle obtenu.

7. Choix de la méthode retenue en fonction de ses capacités de prédiction, de sa robustesse mais aussi, éventuellement, de l'interprétabilité du modèle obtenu.

Explicatives L'ensemble des p variables explicatives ou prédictives est noté X, il est constitué de variables

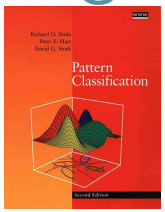
- $X_{\mathbb{R}}$ toutes quantitatives (rq : variable explicative qualitative à 2 modalités (0,1) peut être considérée comme quantitative)
- X_E toutes qualitatives,
- $X_{\mathbb{R}\cup E}$ un mélange de qualitatives et quantitatives.

À expliquer La variable à expliquer ou à prédire ou *cible* (target) peut être

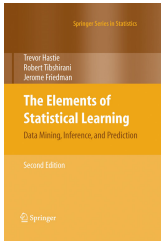
- Y quantitative,
- Z qualitative à 2 modalités,
- T qualitative.

1. Modèle linéaire généralisé
   - `RLM` $X_{\mathbb{R}}$ et Y
   - `ANOVA` X_E et Y
   - `ACOVA` $X_{\mathbb{R}\cup E}$ et Y
   - `Rlogi` $X_{\mathbb{R}\cup E}$ et Z
   - `Lglin` X_T et T
2. Analyse discriminante
   - `ADpar/nopar` $X_{\mathbb{R}}$ et T
3. Classification and regression Tree
   - `ArbReg` $X_{\mathbb{R}\cup E}$ et Y
   - `ArbCla` $X_{\mathbb{R}\cup E}$ et T
4. Réseaux neuronaux
   - `percep` $X_{\mathbb{R}\cup E}$ et Y ou T
5. Agrégation de modèles
   - Bagging $X_{\mathbb{R}\cup E}$ et Y ou T
   - RandFor $X_{\mathbb{R}\cup E}$ et Y ou T
   - Boosting $X_{\mathbb{R}\cup E}$ et Y ou T
6. Support Vector Machine
   - SVM-R $X_{\mathbb{R}\cup E}$ et Y
   - SVM-C $X_{\mathbb{R}\cup E}$ et T

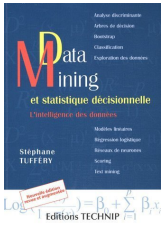
Bibliographie - Ressources



Pattern Classification (2001) - Wiley Interscience, *R. Duda, P. Hart, D. Stork*



The Elements of Statistical Learning (2001) - Springer, *T. Hastie, R. Tibshirani, J. Friedman*
Disponible en ligne : `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`



Data Mining - Technip, *S. Tufféry*

Cours en ligne de Andrew Ng (Stanford) :
`https://www.coursera.org/course/ml`

WikiStat `http://wikistat.fr/`

UCI Machine Learning Repository
Base de données de benchmarking :
`http://archive.ics.uci.edu/ml/`

Logiciels

Référence :

-  `http://cran.r-project.org/`
- The R Project for Statistical Computing
- Avantages : libre, ouvert, bien documenté, à la pointe de la recherche
- Inconvénients : pas 'presse-bouton', syntaxe, pas très rapide (MAIS extensions en C possibles !)
- *Aide en ligne* + google indispensables : logiciel vivant !

Alternatives :

- Tous les Data Managers s'y mettent (SAS, Oracle, IBM Dataminer...)
- Quelques outils dédiés faciles à utiliser, par exemple See5/C5.0 `http://www.rulequest.com/see5-info.html`

Roadmap

Classification multi-classe

- Pour certaines règles de classification, extension possible directement (exemple : kNN, régression logistique).
- Sinon, deux possibilités de se ramener à la classification binaire :

  OvA (One vs. All) Pour chaque classe, construire un classifieur pour discriminer entre cette classe et tout le reste.
  Défaut : classes souvent très déséquilibrées (à prendre en compte)

  AvA (All vs. All) Pour chaque paire de classes (C_1, C_2), construire un classifieur pour discriminer entre c_1 et C_2.
  A priori, $\approx n^2/2$ classifieurs MAIS classifs entre paires plus rapides ET classes équilibrées $\Longrightarrow$ souvent plus rapide.

  Puis Error-Correcting Output Codes, typiquement : vote majoritaire

Règles de classification binaires aléatoires

- La règle randomisée : $f_n(x) = 1$ avec probabilité $1/2$ donne la bonne réponse une fois sur 2 !
- La règle constante classifiant toujours dans la classe qui a la probabilité p la plus grande se trompe avec probabilité $1 - p < 1/2$.

Règle optimale : Erreur de Bayes

Théorème :

La meilleure règle possible est la *règle de Bayes* :

- en classification : avec $\mathcal{Y} = \{0, 1\}$ et $\eta(x) = P(Y = 1|X = x)$:

$$f^*(x) = \mathbb{1}\{\eta(x) > 1/2\} \, .$$

- en régression : avec $\mathcal{Y} = \mathbb{R}$ et la perte quadratique :

$$f^*(x) = E[Y|X = x] \, .$$

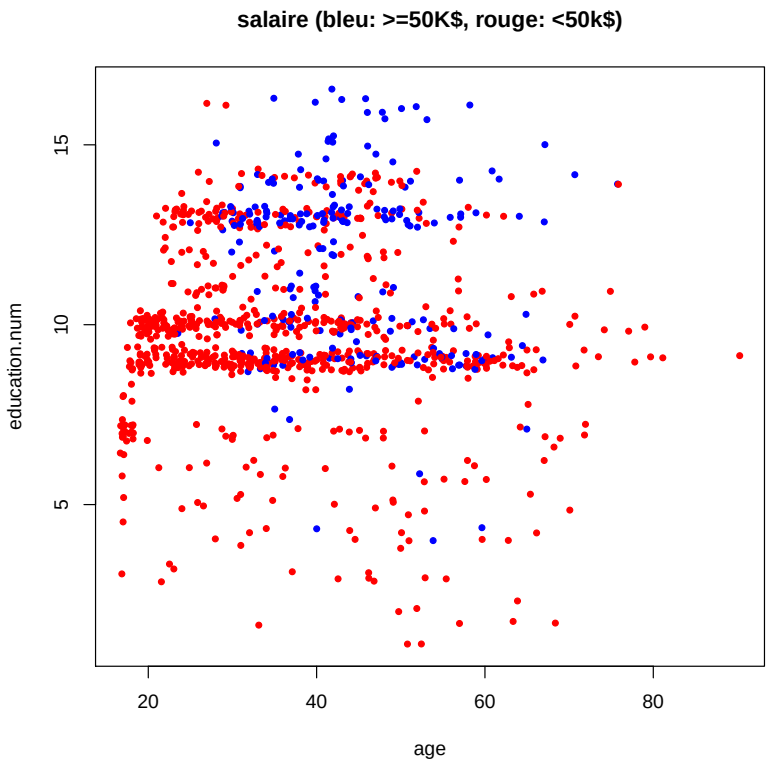
Problème : on ne connaît *jamais* P, donc on ne peut pas la calculer.

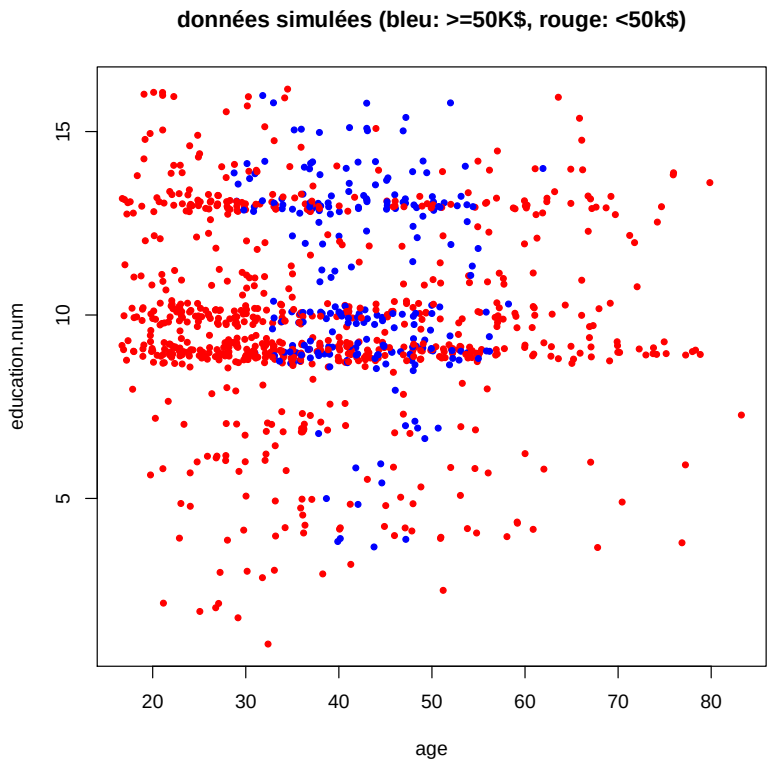
Attention : le risque de la règle de Bayes n'est pas nul !

Données simulées

- Pour contrôler la qualité d'un algorithme, on est donc amené à utiliser des *modèles génératifs* et à manipuler des *données simulées*.
- Sert à comprendre ce qu'on peut attendre (ou pas) des algorithmes d'apprentissage.
- Exemple : Sachant $X = (age, etudes)$, Y suit une loi de Bernoulli de paramètre $\max(\frac{etudes}{20} - \frac{(age-45)^2}{500}, 0)$.
- Avantage : on peut calculer la règle de Bayes (et visualiser la *frontière de Bayes*).

Exemple : salaires





Il n'y a pas de meilleure méthode !

- Chacune est plus ou moins adaptée au problème considéré, à la nature des données, aux propriétés de la relation entre descripteurs et variable expliquée...
- Il faut apprendre les qualités et les défauts de chaque méthode
- Il faut apprendre à expérimenter pour trouver la plus pertinentes
- L'estimation de la qualité des méthodes est donc centrale (mais pas toujours évidente)

Qu'est-ce qu'un bon algorithme d'apprentissage ?

Interprétabilité : la règle de classification est 'compréhensible'

Critique : fournit un score en classification, un intervalle en régression

Consistance : convergence vers l'erreur bayésienne : quand n tend vers l'infini, f_n tend vers la règle de Bayes

Minimax : cette convergence est la plus rapide possible

Non-asymptotique : garanties de performance pour n donné

Parameter-free : Paramétrage automatique

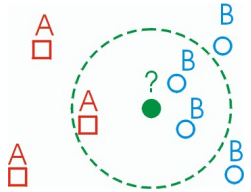
Vitesse : complexité linéaire, possibilité de parallèliser

Online : mise à jour séquentielle

Roadmap

Définition

Règle des k plus proches voisins : pour tout $x \in \mathcal{X}$, trouver ses plus proches voisins $x_{(1)}, \ldots, x_{(k)}$



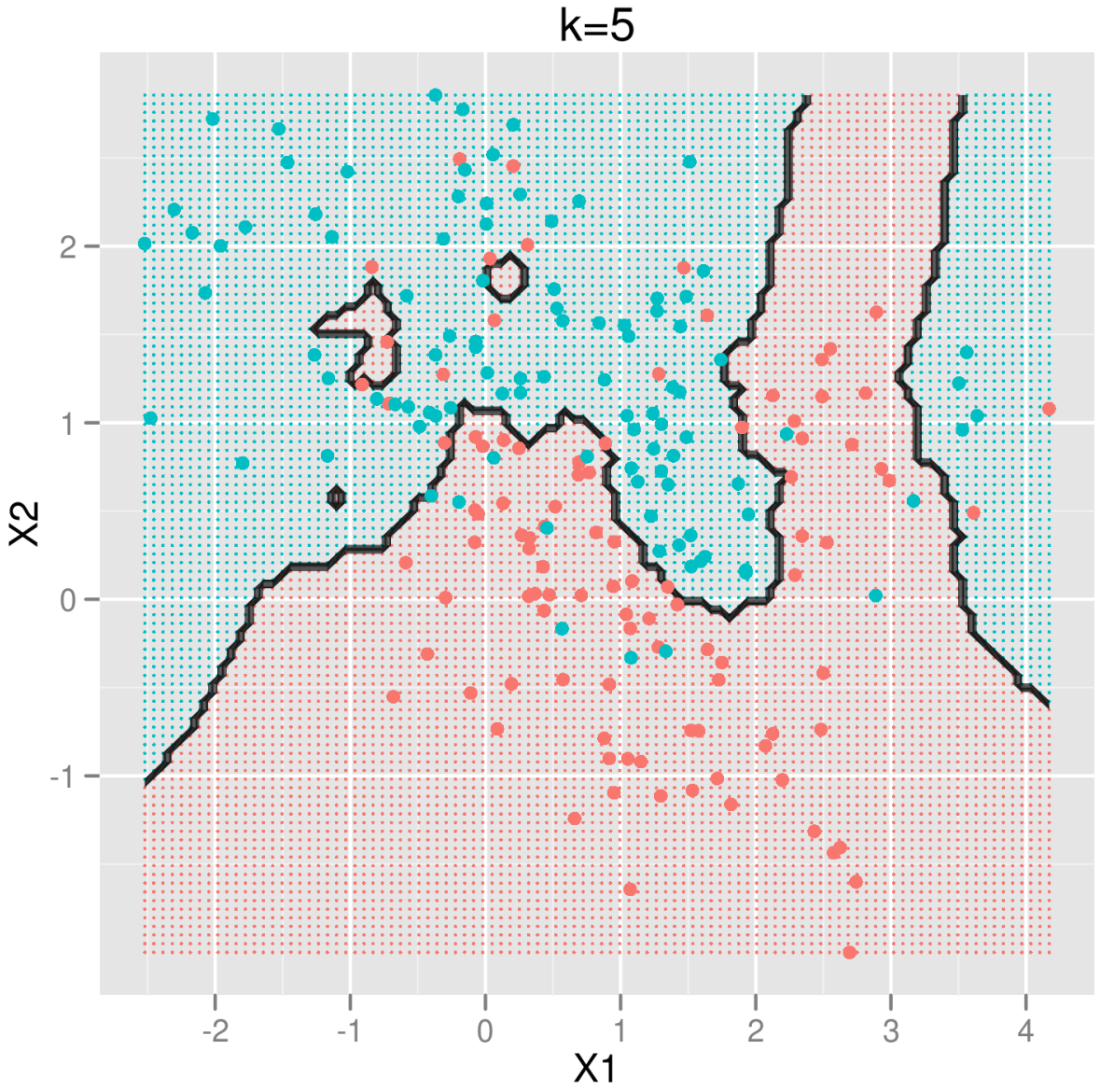
- classification :

$$f_n(x) = \arg\max_{y \in \mathcal{Y}} \sum_{j=1}^{k} \mathbb{1}\{y_{(j)} = y\}$$

- régression :

$$f_n(x) = \frac{1}{k} \sum_{j=1}^{k} y_{(j)}$$

Visualisation d'une règle k-NN



Propriétés de k-NN

Qualités :

- simplicité
- interprétabilité (?)
- pas de consistance (quel que soit k)
- MAIS asymptotiquement erreur au plus 2x supérieure à la règle de Bayes.
- possibilité de faire croître k avec n, consistance (théorique) par exemple pour $k = \log(n)$

Paramétrage :

- quelle métrique sur $\mathcal{X}$?

$\Longrightarrow$ au minimum, *normaliser* les variables (pb pour les qualitatives)

- Quelle valeur de k choisir ?

Qualités de kNN

Interprétabilité : OUI et NON

Critique : OUI mais pas très fiable

Consistance : NON mais possible si $k = \log(n)$ (par exemple)

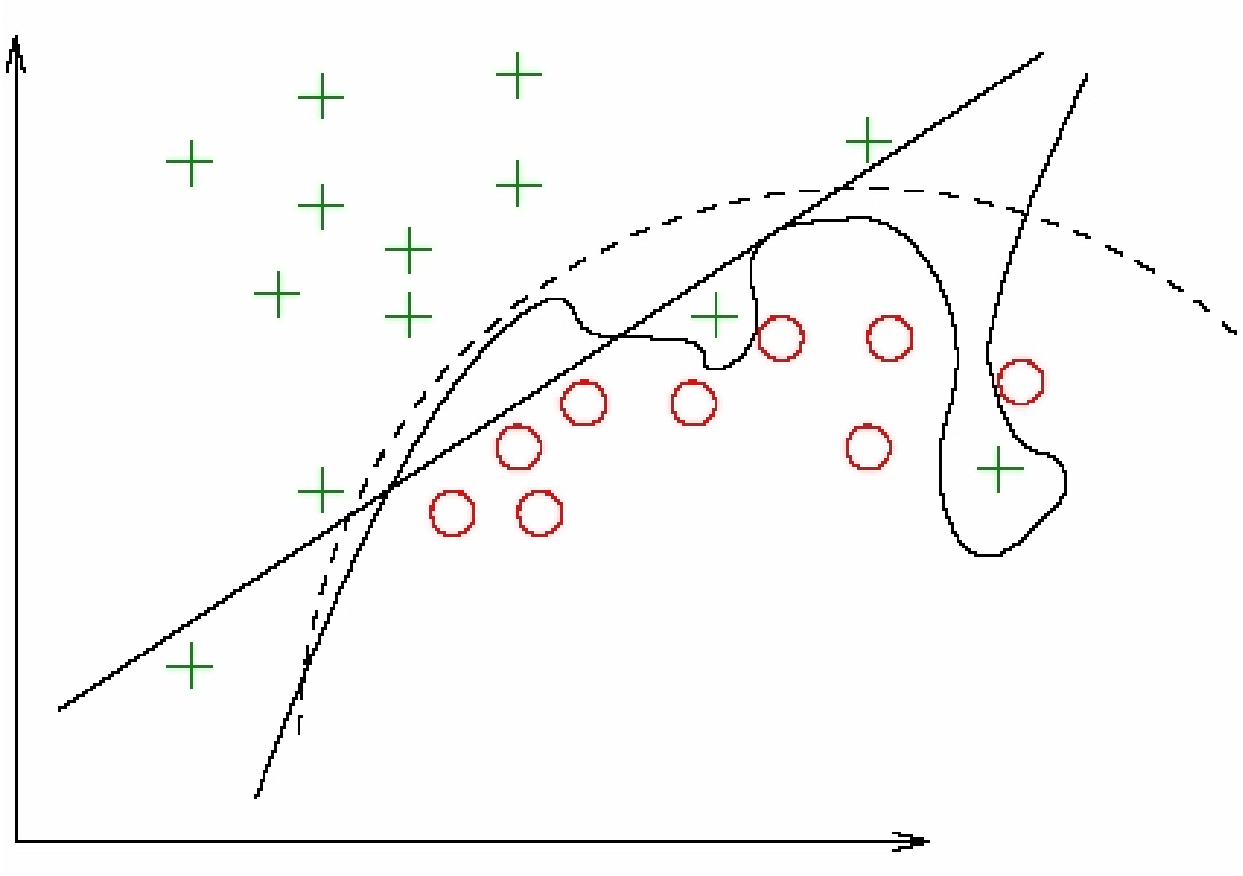
Minimax : NON

Parameter-free : NON

Vitesse : OUI et NON, implémentation possible en $O(n \log n)$

Online : OUI

Sur-apprentissage



Sur-apprentissage

- On dit qu'une règle de classification est en sur-apprentissage si elle "colle trop" aux données d'entraînement.
- Le sur-apprentissage apparaît facilement dans les modèles complexes, quand on cherche à apprendre trop de choses par rapport à la richesse des données disponibles.
- L'inverse du sur-apprentissage est le "sous-apprentissage" (pas une notion aussi claire) : il consiste à utiliser un modèle trop grossier.
- Le sur-apprentissage est plus trompeur, car l'ajustement aux données d'entraînement paraît très bon !
  Erreur d'apprentissage faible, ex : R^2 en ajustement linéaire.
- Beaucoup de méthodes d'apprentissage font intervenir un ou plusieurs paramètres. Suivant la valeur de celui-ci, on peut tomber dans le sur-apprentissage : il faut ajuster les paramètres pour l'éviter.

Equilibre biais-variance

On peut souvent décomposer le risque d'une méthode comme somme de deux termes :

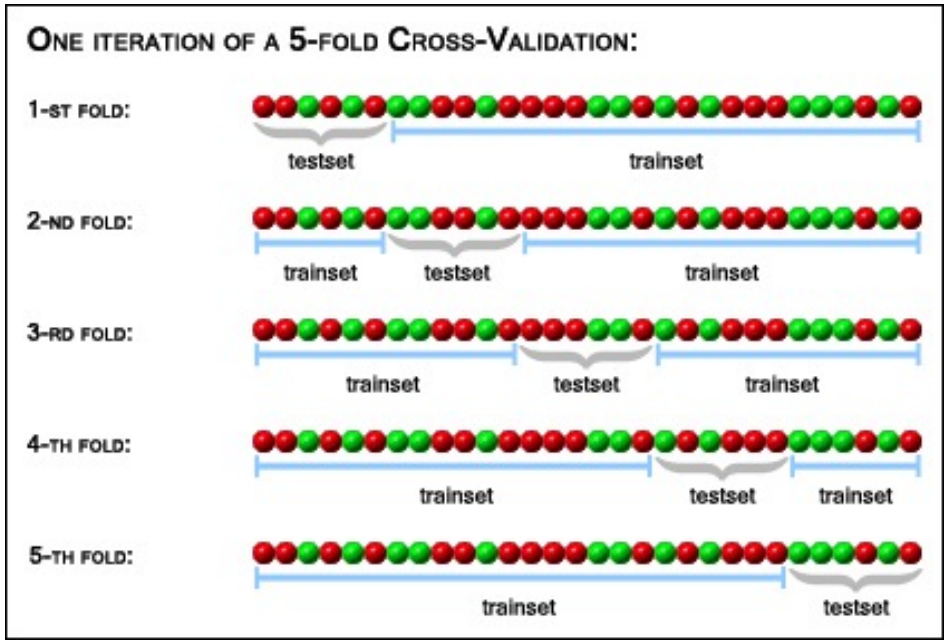
Décomposition biais-variance

risque $=$ (variabilité $+$) approximation $+$ estimation

- Les deux types d'erreur varient en sens contraire avec les paramètres.
- L'objectif est de trouver un bon *équilibre* entre les deux types d'erreur afin de minimiser leur somme.
- De manière générale il faut préférer les modèles parcimonieux.

Validation croisée

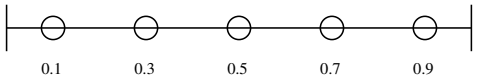
- Pour ajuster les paramètres, il existe
  - des méthodes structurelles (ex : critères AIC, BIC, etc.) ;
  - des méthodes de validation sur les données (ex : échantillon de test, validation croisée).
- Méthode 0 : découpage train/test
  Inconvénient : on apprend sur peu de données
- Solution très souvent adoptée : *validation croisée*



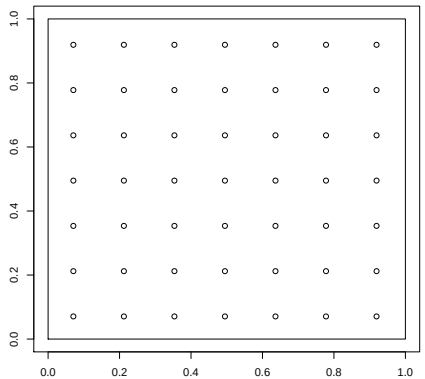
Curse of dimensionality (Bellman)

Dans $[0,1]^d$, combien de points doit contenir une grille régulière pour que tout point ait un voisin de la grille au plus à distance 0.1 ?

$d = 1$: 5 points



$d = 2$: 49 points



$d = 10$: 976562500000 points

$d = 20$: 97656250000000000000000000 points

Modèle plus riche = meilleur ?

- Exemple : régression polynomiale

$$Y_i = f\left(\frac{i}{n}\right) + \epsilon_i \ , \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- f fonction régulière (par exemple polynomiale)
- estimation par régression polynomiale

$\Longrightarrow$ le meilleur modèle n'est pas le plus gros

$\Longrightarrow$ si f est polynomiale, son degré ne donne pas non plus la solution

$\Longrightarrow$ plus le nombre d'observation est grand, plus on peut considérer des modèles riches

- des méthodes non basées sur des modèles donnent aussi d'excellents résultats (ex : k-NN)

PLAN

- Introduction
- Odds
- Modélisation d'une variable qualitative à 2 modalités
- Modèle binomial
- Choix de modèle

Objectif : Explication d'observations constituées d'effectifs.

Les lois concernées sont discrètes et associées à des dénombrements :

- loi de Poisson
- loi binomiale
- loi multinomiale.

Tous ces modèles appartiennent à la famille du *modèle linéaire général* et partagent à ce titre beaucoup de concepts : famille exponentielle, estimation par maximum de vraisemblance, tests, diagnostics, résidus.

Soit Y une variable qualitative à m modalités. On désigne la chance ou l'odds de voir se réaliser la $j^{\text{ème}}$ modalité plutôt que la $k^{\text{ème}}$ par le rapport

$$\Omega_{jk} = \frac{\pi_j}{\pi_k},$$

où π_j est la probabilité d'apparition de la $j^{\text{ème}}$ modalité.

Cette quantité est estimée par $\frac{n_j}{n_k}$ des effectifs observés sur un échantillon.

Si Y est une variable de Bernoulli de paramètre π, alors l'odds est égal à $\frac{\pi}{1-\pi}$, la cote ou chance de gain.

Soit Y une variable qualitative à 2 modalités : 1 ou 0, succès ou échec, présence ou absence de verglas...

Les modèles de régression linéaire adaptés à l'explication d'une variable quantitative ne s'appliquent plus directement car le régresseur linéaire usuel $\mathbf{X}\beta$ ne prend pas des valeurs simplement binaires.

Objectif : Adapter la modélisation linéaire à cette situation en cherchant à expliquer les probabilités

$$\pi = \mathbb{P}(Y = 1) \text{ ou } 1 - \pi = \mathbb{P}(Y = 0)$$

ou plutôt une transformation de celles-ci, par l'observation conjointe des variables explicatives.

Idée : Faire intervenir une fonction réelle monotone g opérant de $[0; 1]$ dans $\mathbb{R}$ et chercher un modèle linéaire de la forme

$$g(\pi) = \mathbf{x}'\beta.$$

Plusieurs possibilités :

- **probit** : g est la fonction inverse de la fonction de répartition d'une loi normale, mais son expression n'est pas explicite.
- **log-log** : $g(\pi) = \ln[-\ln(1-\pi)]$, mais cette fonction est dissymétrique.
- **logit** : $g(\pi) = logit(\pi) = \ln\left(\frac{\pi}{1-\pi}\right)$ avec $g^{-1}(x) = \frac{e^x}{1+e^x}$.

La régression logistique s'interprète comme la recherche d'une modélisation linéaire du "log odds".

On va chercher β tel que

$$\frac{\pi_X}{1-\pi_X} \simeq e^{X'\beta}.$$

- La régression logistique s'exprime généralement sous la forme

$$\frac{\pi_{x_1,\cdots,x_p}}{1-\pi_{x_1,\cdots,x_p}} = e^{\alpha_0+\alpha_1 x_1+\cdots+\alpha_p x_p}.$$

- Si $\alpha_j \simeq 0$, alors l'odds ratio ne dépend pas de x_j.
- Un incrément de 1 dans x_j correspond à un incrément de e^{α_j} dans l'odds ratio.
- Si $\alpha_j > 0$, alors un incrément de 1 dans x_j entraîne une augmentation de e^{α_j} du risque
(ou de la chance) que Y prenne la valeur 1.

$$\frac{\frac{\pi_{x_1,\cdots,x_j+1,\cdots,x_p}}{1-\pi_{x_1,\cdots,x_j+1,\cdots,x_p}}}{\frac{\pi_{x_1,\cdots,x_p}}{1-\pi_{x_1,\cdots,x_p}}} = e^{\alpha_j}.$$

- On teste l'effet de chaque variable en testant l'hypothèse

$$H_0 : \beta_i = 0.$$

- On rejette l'hypothèse H_0 dès que la p-valeur de H_0 est trop faible.
  - En effet

$$P(\text{Observations}|H_0) = \text{p-valeur}.$$

    Si la p-valeur est trop faible, alors aux vues de ces observations, l'hypothèse H_0 est peu vraisemblable et par conséquent le coefficient β_i est significativement non nul.
- Les logiciels scientifiques donnent cette p-valeur.
- On peut également déterminer des régions de confiance pour la régression logistique sur π_X grâce à g^{-1}(région de confiance classique).

PLAN D'EXPÉRIENCE

- Pour $i \in \{1, \cdots, I\}$, on effectue n_i mesures de la variable Y.
- y_i désigne le nombre de réalisations $Y = 1$.
- On suppose que π_i est la probabilité de succès de Y sachant que les $x^1, \cdots, x^p$ appartiennent au goupe i

$$\pi_i = P(Y = 1 | X \in \text{Groupe } i).$$

- On suppose que les groupes sont homogènes dans leurs réalisations de Y : la probabilité pour Y d'être égal à 1 au sein d'un même groupe est indépendante de la valeur de X dans ce groupe.
- Proposition : En effectuant n_i mesures dans le groupe i et en supposant tous les échantillons indépendants, si Y_i désigne le nombre de valeurs $Y = 1$, alors

$$P(Y_i = y_i) = C_{n_i}^{y_i} \pi_i^{y_i} (1 - \pi_i)^{n_i - y_i}.$$

ESTIMATION DES COEFFICIENTS

- On suppose que le vecteur des fonctions *logit* des probabilités π_i appartient au sous-espace $Vect\{X^1, \ldots, X^p\}$

$$logit(\pi_i) = \mathbf{x}_i'\beta \quad \text{ou} \quad \pi_i = \frac{e^{\mathbf{x}_i'\beta}}{1 + e^{\mathbf{x}_i'\beta}},\ i = 1, \ldots, I.$$

- La log-vraisemblance s'écrit

$$L(\beta) = \Pi_{i=1}^{I} C_n^{y_i} g^{-1}(\mathbf{x}_i'\beta)^{y_i}(1 - g^{-1}(\mathbf{x}_i'\beta))^{n_i - y_i}.$$

- β est estimé par maximisation de la log-vraisemblance. Il n'y a pas de solution analytique, celle-ci est obtenue par des méthodes numériques itératives (algorithme de Newton Raphson).
- L'optimisation fournit une estimation $\mathbf{b}$ de β. On peut alors en déduire les estimations ou prévisions des probabilités π_i et celles des effectifs

$$\widehat{\pi}_i = \frac{e^{\mathbf{x}_i'\mathbf{b}}}{1 + e^{\mathbf{x}_i'\mathbf{b}}} \quad \text{et} \quad \widehat{y}_i = n_i\widehat{\pi}_i.$$

REMARQUES

- La matrice $\mathbf{X}$ issue de la planification expérimentale est construite avec les mêmes règles que celles utilisées dans le cadre de la covariance mixant variables explicatives quantitatives et qualitatives.

- La situation décrite précédemment correspond à l'observation de données groupées. Dans de nombreuses situations concrètes et souvent dès qu'il y a des variables explicatives quantitatives, les observations $\mathbf{x}_i$ sont toutes distinctes. Ceci revient donc à fixer $n_i = 1, i = 1, \ldots, I$ dans les expressions précédentes et la loi de Bernoulli remplace la loi binomiale.

Le test du rapport de vraisemblance et le test de Wald permettent de comparer un modèle avec un sous-modèle et d'évaleur l'intérêt de la présence des termes complémentaires. Stratégie descendante à partir du modèle complet.

Idée : Supprimer, un terme à la fois, la composante d'intéraction ou l'effet principal qui apparaît comme le moins significatif au sens du rapport de vraisemblance ou du test de Wald.

ATTENTION du fait de l'utilisation d'une transformation non linéaire (*logit*), même si des facteurs sont orthogonaux, aucune propriété d'orthogonalité ne peut être prise en compte pour l'étude des hypothèses.

- Élimination des termes un par un et ré-estimation du modèle.
- Un terme prinicipal ne peut être supprimé que s'il n'intervient plus dans les termes d'interaction.

Les logiciels calculent en plus l'AIC pour finaliser le choix pour une meilleure qualité prédictive.

Qualités du classifieur de régression logistique

Interprétabilité : NON

Critique : OUI ! (très utilisé pour le scoring)

Consistance : NON (sauf si le modèle est exact)

Minimax : NON

Parameter-free : OUI

Vitesse : OUI

Online : possible

Analyse discriminante décisionnelle

- Idée : chaque caractéristique est associé à la classe dont le barycentre des caractéristiques dans l'échantillon d'apprentissage est le plus proche

$\Longleftrightarrow$ recherche directe d'une frontière linéaire

- Problème : hétérogénéité des variables
- Nombreuses variantes :

LDA = AFD (analyse factorielle discriminante)

interprétation bayésienne

k-NN est parfois vu comme une variante non-paramétrique

noyau : idem pour la méthode du noyau

PLAN

- Introduction
- Construction d'un arbre binaire
  - Principe
  - Critère de division
  - Règle d'arrêt
  - Affectation
- Critères d'homogénéité
  - Y quantitative
  - Y qualitative
- Élagage
  - Construction de la séquence d'arbres
  - Recherche de l'arbre optimal

La segmentation par arbre est une approche non-paramétrique de l'analyse discriminante.

But : expliquer une variable réponse (qualitative ou quantitative) à l'aide d'autres variables.

Principe : construire un arbre à l'aide de divisions successives des individus d'un ensemble E en deux segments (appelés aussi noeuds) homogènes par rapport à une variable Y (binaire, nominale, ordinale ou quantitative) en utilisant l'information de p variables $X^1, \ldots, X^p$ (binaires, nominales, ordinales ou quantitatives).

L'arbre obtenu est sous forme d'un arbre inversé comportant à la racine l'échantillon total E à segmenter et les autres segments sont

- soit des segments intermédiaires (encore divisibles),
- soit des segments terminaux.

L'ensemble des segments terminaux constitue une partition de l'ensemble E en classes homogènes et distinctes, relativement à la variable Y.

Il s'agit d'arbre de classement si Y est qualitative et d'arbre de régression si Y est quantitative.

Avantages / Inconvénients

- La méthode CART (Classification And Regression Tree) fournit des solutions sous formes graphiques simples à interpréter.
- Elle est complémentaire des méthodes statistiques classiques, très calculatoire et efficace à condition d'avoir de grandes tailles d'échantillon.
- Elle est capable de gérer à la fois les variables quantitatives ET qualitatives simultanément.
- Peu d'hypothèses requises !
- Algorithme étant basé sur une stratégie pas à pas hiérarchisée, il peut passer à côté d'un optimum global.

Soient p variables quantitatives ou qualitatives explicatives X^j et une variable à expliquer Y qualitative à m modalités $\{\tau_l, l = 1, \ldots, m\}$ ou quantitative réelle, observée sur un échantillon de n individus.

La construction d'un arbre de discrimination binaire consiste à déterminer une séquence de noeuds.

- Un noeud est défini par le choix conjoint d'une variable parmi les explicatives et d'une division qui induit une partition en deux classes.
- Une division est elle-même définie par une valeur seuil de la variable quantitative sélectionnée ou un partage en deux groupes des modalités si la variable est qualitative.
- À la racine ou au noeud initial correspond l'ensemble de l'échantillon. La procédure est ensuite itérée sur chacun des sous-ensembles.

L'algorithme considéré nécessite

1. la définition d'un critère permettant de sélectionner la "meilleure" division parmi toutes celles admissibles pour les différentes variables ;
2. une règle permettant de décider qu'un noeud est terminal : il devient alors feuille ;
3. l'affectation de chaque feuille à l'une des classes ou à une valeur de la variable à expliquer.

Le point 2. correspond encore à la recherche d'un modèle parcimonieux. Un arbre trop détaillé, associé à une sur-paramétrisation, est instable et donc probablement plus défaillant pour la prévision d'autres observations.

Breiman et al. ont mise en place une stratégie de recherche de l'arbre optimal.

1. Construire l'arbre maximal A_{max}.
2. Ordonner les sous-arbres selon une séquence emboîtée suivant la décroissance d'un critère pénalisé de déviance ou de taux de mal-classés.
3. Sélectionner le sous-arbre optimal : c'est la procédure d'élagage.

CRITÈRE DE DIVISION

Une division est dite admissible si aucun des segments descendants n'est vide.

- Si la variable explicative est qualitative ordinale à m modalités, elle conduit à $m - 1$ divisions binaires admissibles.
- Si elle est nominale, le nombre de divisions devient égal à $2^{m-1} - 1$.
- Pour une variable quantitative à m valeurs disctinctes, on se ramène au cas ordinal.

Le critère de division repose sur la définition d'une fonction d'hétérogénéité ou de désordre.

Objectif : Partager les individus en deux groupes les plus homogènes au sens de la variable à expliquer.

CRITÈRE DE DIVISION

L'hétérogénéité d'un noeud se mesure par une fonction non négative qui doit être

1. nulle si et seulement si le segment est homogène : tous les individus appartiennent à la même modalité ou prennent la même valeur de Y,
2. maximale lorsque les valeurs de Y sont équiprobables ou très dispersées.

La division du noeud k crèe deux fils notés $(k+1)$ et $(k+2)$, mais une renumérotation sera nécessaire pour respecter la séquence de sous-arbres.

Parmi toutes les divisions admissibles du noeud k, l'algorithme retient celle qui rend la somme $D_{(k+1)} + D_{(k+2)}$ des désordres des noeuds fils minimale, c-à-d

$$\max_{\{\text{divisions de } X^j\,;\, j=1,\ldots,p\}} D_k - D_{(k+1)} - D_{(k+2)}.$$

RÈGLE D'ARRÊT

La croissance de l'arbre s'arrête à un noeud qui devient donc feuille

- lorsqu'il est homogène, c-à-d lorsqu'il n'existe plus de division admissible,
- si le nombre d'observations qu'il contient est inférieur à un seuil fixé par l'utilisateur d_{min}. En général, $1 \leq d_{min} \leq 5$.
- si le nombre de noeuds est supérieur à n_{max}, nombre fixé par l'utilisateur.

AFFECTATION

Une fois les critères d'arrêt atteints, il faut affecter une valeur à chaque feuille.

- Si Y est quantitative, attribution de la valeur moyenne aux observations de cette feuille.
- Si Y est qualitative, chaque feuille est affectée à une modalité τ_l de Y en considérant le mode conditionnel
  - celle la plus représentée dans la feuille, c-à-d celle ayant la proportion la plus élevée à l'intérieur de cette feuille. Il est alors facile de comparer le nombre de données mal classées.
  - la modalité la moins coûteuse si des coûts de mauvais classements sont donnés.
  - la classe *a posteriori* la plus probable au sens bayésien si des probabilités *a priori* sont connues.

Soit une partition de n individus en deux sous-populations E_1 et E_2 de tailles respectives n_1 et n_2. Soit μ_{ij} la valeur "théorique" de Y pour l'individu i du sous-ensemble E_j.

L'hétérogénéité du sous-ensemble E_j est mesurée par

$$D_j = \sum_{i=1}^{n_j} (\mu_{ij} - \mu_{.j})^2 \text{ avec } \mu_{.j} = \frac{1}{n_j} \sum_{i=1}^{n_j} \mu_{ij}.$$

Alors l'hétérogénéité de la partition est définie par

$$D = D_1 + D_2 = \sum_{j=1}^{2} \sum_{i=1}^{n_j} (\mu_{ij} - \mu_{.j})^2.$$

C'est l'inertie intragroupe qui vaut 0 si et seulement si $\forall i, j, \mu_{ij} = \mu_{.j}$.

La différence entre l'hétérogénéité de l'ensemble non partagé et celle de la partition est

$$\begin{aligned}\Delta &= \sum_{j=1}^{2}\sum_{i=1}^{n_j}(\mu_{ij}-\mu_{..})^2 - \sum_{j=1}^{2}\sum_{i=1}^{n_j}(\mu_{ij}-\mu_{.j})^2 \text{ où } \mu_{..} = \frac{1}{n}\sum_{i,j}\mu_{ij}.\\ &= \sum_{j=1}^{2} n_j(\mu_{..}-\mu_{.j})^2\\ &= \frac{n_1 n_2}{n}(\mu_{.1}-\mu_{.2})^2.\end{aligned}$$

Δ correspond au "désordre" des barycentres et est homogène à la variance intergroupe.
Objectif : À chaque étape, maximiser Δ, c-à-d trouver la variable explicative induisant une partition en deux sous-ensembles associée à une inertie intragroupe minimale ou encore qui rende l'inertie intergroupe maximale.

Les quantités sont estimées

$$D_j \text{ par } \widehat{D}_j = \sum_{i=1}^{n_j} (y_{ij} - y_{.j})^2 \quad \text{et} \quad D \text{ par } \widehat{D} = \sum_{j=1}^{2} \sum_{i=1}^{n_j} (y_{ij} - y_{.j})^2.$$

Soit Y une variable à expliquer à m modalités τ_l. L'arbre induit une partition pour laquelle n_{+k} désigne l'effectif du $k^{\text{ème}}$ noeud.

Soit $p_{lk} = \mathbb{P}[\tau_l|k]$ avec $\sum_{l=1}^{m} p_{lk} = 1$, la probabilité qu'un élément du $k^{\text{ème}}$ noeud appartienne à la $l^{\text{ème}}$ classe.

Le désordre du $k^{\text{ème}}$ noeud, défini à partir de l'entropie, s'écrit

$$D_k = -2 \sum_{l=1}^{m} n_{+k} p_{lk} \log(p_{lk}).$$

L'hétérogénéité de la partition est encore

$$D = \sum_{k=1}^{2} D_k = -2 \sum_{k=1}^{2} \sum_{l=1}^{m} n_{+k} p_{lk} \log(p_{lk}).$$

Cette quantité est positive et nulle si et seulement si les probabilités p_{lk} ne prennent que des valeurs 0 sauf une égale à 1 correspondant à l'absence de mélange.

ENTROPIE

Soit n_{lk} l'effectif observé de la $l^{\text{ème}}$ classe dans le $k^{\text{ème}}$ noeud :

$$n_{+k} = \sum_{l=1}^{m} n_{lk}.$$

Les quantités sont estimées

$$D_k \text{ par } \widehat{D}_k = -2 \sum_{l=1}^{m} n_{+k} \frac{n_{lk}}{n_{+k}} \log \left(\frac{n_{lk}}{n_{+k}} \right)$$

et

$$D \text{ par } \widehat{D} = -2 \sum_{k=1}^{2} \sum_{l=1}^{m} n_{+k} \frac{n_{lk}}{n_{+k}} \log \left(\frac{n_{lk}}{n_{+k}} \right).$$

Critère de Gini

Le critère de Gini du noeud k est défini par $D_k = \sum_{l \neq h} p_{lk} p_{hk}$ avec $l, h = 1, \ldots, m$ et est estimé par $\widehat{D}_k = \sum_{l \neq h} \frac{n_{lk}}{n_{+k}} \frac{n_{hk}}{n_{+k}}$.

- Le désordre D_k est maximal si $p_{lk} = \frac{1}{m}$; l'échantillon présente autant d'éléments de chaque modalité.
- D_k est nul si l'échantillon est pur : $p_{lk} = 1$ et $p_{hk} = 0$ si $h \neq l$.
- D_k représente la probabilité de mauvais classement pour un individu tiré au hasard parmi les individus du noeud k.

Critère de Gini

Le désordre de l'échantillon initial de taille n est estimé par $\widehat{D} = \sum_{l \neq h} \frac{n_l}{n} \frac{n_h}{n}$, où n_l représente l'effectif observé de la $l^{\text{ème}}$ modalité dans l'échantillon initial.

La réduction d'impureté correspond à une division binaire est alors estimée par

$$\widehat{\Delta} = \widehat{D} - \frac{n_{+1}}{n} \widehat{D}_1 - \frac{n_{+2}}{n} \widehat{D}_2.$$

Objectif : Rechercher le meilleur compromis entre

- un arbre très détaillé, fortement dépendant des observations qui ont permis son estimation, qui fournira un modèle de prévision très instable
- un arbre trop robuste mais grossier qui donne des prédictions trop approximatives.

Principe

- Construire une suite emboîtée de sous-arbres de l'arbre maximum par élagage successif.
- Choisir, parmi cette suite, l'arbre optimal au sens d'un critère.

La solution obtenue par algorithme pas à pas n'est pas nécessairement, globalement optimale mais l'efficacité et la fiabilité sont préférées à l'optimalité.

CONSTRUCTION DE LA SÉQUENCE D'ARBRES

Pour un arbre A donné, on note K le nombre de feuilles ou noeuds terminaux de A ; la valeur de K exprime la complexité de A.

La qualité de discrimination d'un arbre A se mesure par le critère $D(A) = \sum_{k=1}^{K} D_k(A)$ où $D_k(A)$ est le nombre de mal classés ou la déviance ou le coût de mauvais classement de la $k^{\text{ème}}$ feuille de l'arbre A.

CONSTRUCTION DE LA SÉQUENCE D'ARBRES

La construction de la séquence d'arbres emboîtés repose sur une pénalisation de la complexité de l'arbre

$$C(A) = D(A) + \gamma K.$$

- Pour $\gamma = 0$, $A_{max} = A_K$ minimise $C(A)$.
- En faisant croître γ, l'une des divisions de A_K, celle pour laquelle l'amélioration de D est la plus faible (inférieure à γ) apparaît comme superflue et les deux feuilles sont regroupées (élaguées) dans le noeud père qui devient terminal ; $A_K \supset A_{K-1}$.

Construction de la séquence d'arbres

Soit $\mathcal{N}$ un noeud. On appelle $A_{\mathcal{N}}$ le sous-arbre (ou la branche) de A extrait(e) à partir de $\mathcal{N}$, donc constitué des descendants de $\mathcal{N}$ et de la racine $\mathcal{N}$. On appelle A' le sous-arbre de A auquel on a enlevé la branche $A_{\mathcal{N}}$. On a alors

$$C(A') = C(A) + C(\mathcal{N}) - C(A_{\mathcal{N}}).$$

Par conséquent,

$$C(A') \geq C(A) \iff \gamma \leq \frac{D(\mathcal{N}) - D(A_{\mathcal{N}})}{|A_{\mathcal{N}}| - 1} = \alpha.$$

Ceci signifie que si la valeur de γ fixée est inférieure à α, le coût du sous-arbre élagué A' est supérieur à celui de A : on gardera donc l'arbre complet A.

CONSTRUCTION DE LA SÉQUENCE D'ARBRES

Le procédé est itéré pour la construction de la séquence emboîtée

$$A_{max} = A_K \supset A_{K-1} \supset \ldots \supset A_1$$

où A_1, le noeud racine, regroupe l'ensemble de l'échantillon.

Un graphe représente la décroissance ou l'éboulis de la déviance (ou du taux de mal classé) en fonction du nombre croissant de feuilles dans l'arbre ou en fonction de la valeur décroissante du coefficient de pénalisation γ (graphes équivalents).

Élagage lorsque l'augmentation de la complexité de l'arbre n'est plus compensée par la diminution de la déviance.

Recherche de l'arbre optimal

Les procédures d'élagage diffèrent par la façon d'estimer l'erreur de prédiction. Quand l'amélioration du critère est jugée trop petite ou négligeable, on élague l'arbre au nombre de feuilles obtenues.

- L'évaluation de la déviance ou du taux de mauvais classement estimée par resubsitution sur l'échantillon d'apprentissage est biaisée (trop optimiste).

RECHERCHE DE L'ARBRE OPTIMAL

- Une estimation sans biais est obtenue par l'utilisation d'un autre échantillon (validation) ou encore par validation croisée.

  La procédure de validation croisée a une particularité : la séquence d'arbres obtenue est différente pour chaque estimation sur l'un des sous-échantillons.

  - L'erreur moyenne n'est pas calculée pour chaque sous-arbre avec un nombre de feuilles donné mais pour chaque sous-arbre correspondant à une valeur fixée du coefficient de pénalisation γ.
  - À la valeur de γ minimisant l'estimation de l'erreur de prévision, correspond ensuite l'arbre jugé optimal dans la séquence estimée sur tout l'échantillon d'apprentissage.

RECHERCHE DE L'ARBRE OPTIMAL

Le principe de sélection d'un arbre optimal est donc décrit par l'algorithme suivant

- Construction de l'arbre maximal A_{max}.
- Construction de la séquence $A_K, \ldots, A_1$ d'arbres emboîtés.
- Estimation sans biais (échantillon de validation ou validation croisée) des déviances $D(A_K), \ldots, D(A_1)$.
- Représentation de $D(A_k)$ en fonction de k ou de γ.
- Choix de k rendant $D(A_k)$ minimum.

Qualités du classifieur CART

Interprétabilité : OUI !

Consistance : OUI (sous certaines réserves) MAIS instable !

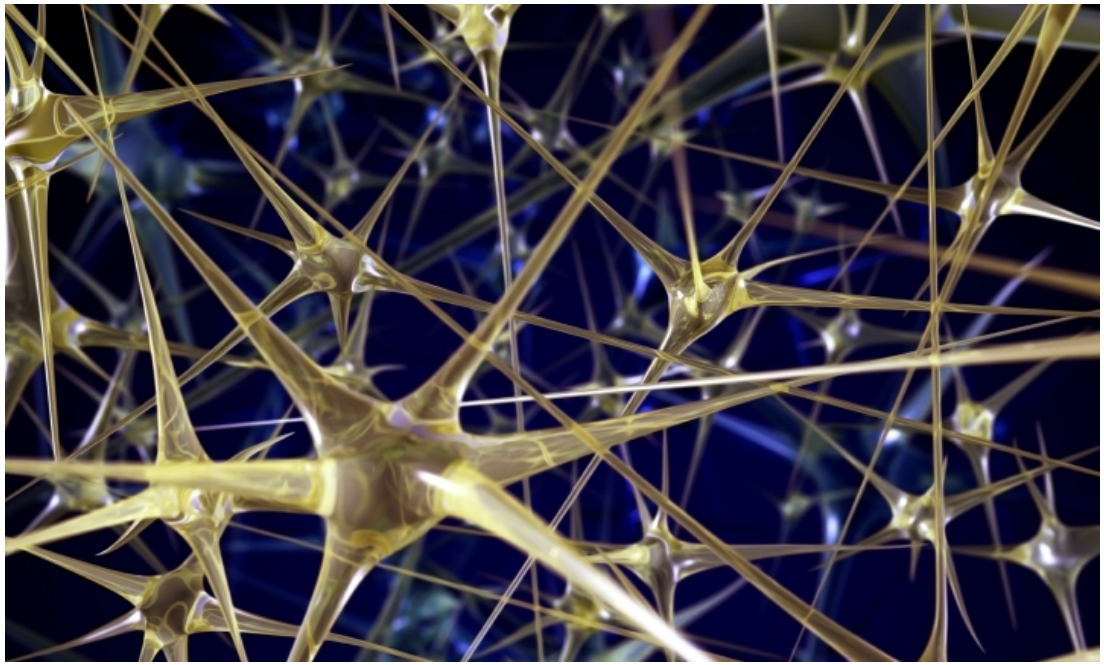
Minimax : NON !

Parameter-free : NON

Vitesse : OUI

Online : NON

Réseau mono-couche



Src : `http://insanedev.co.uk/open-cranium/`

Réseau mono-couche

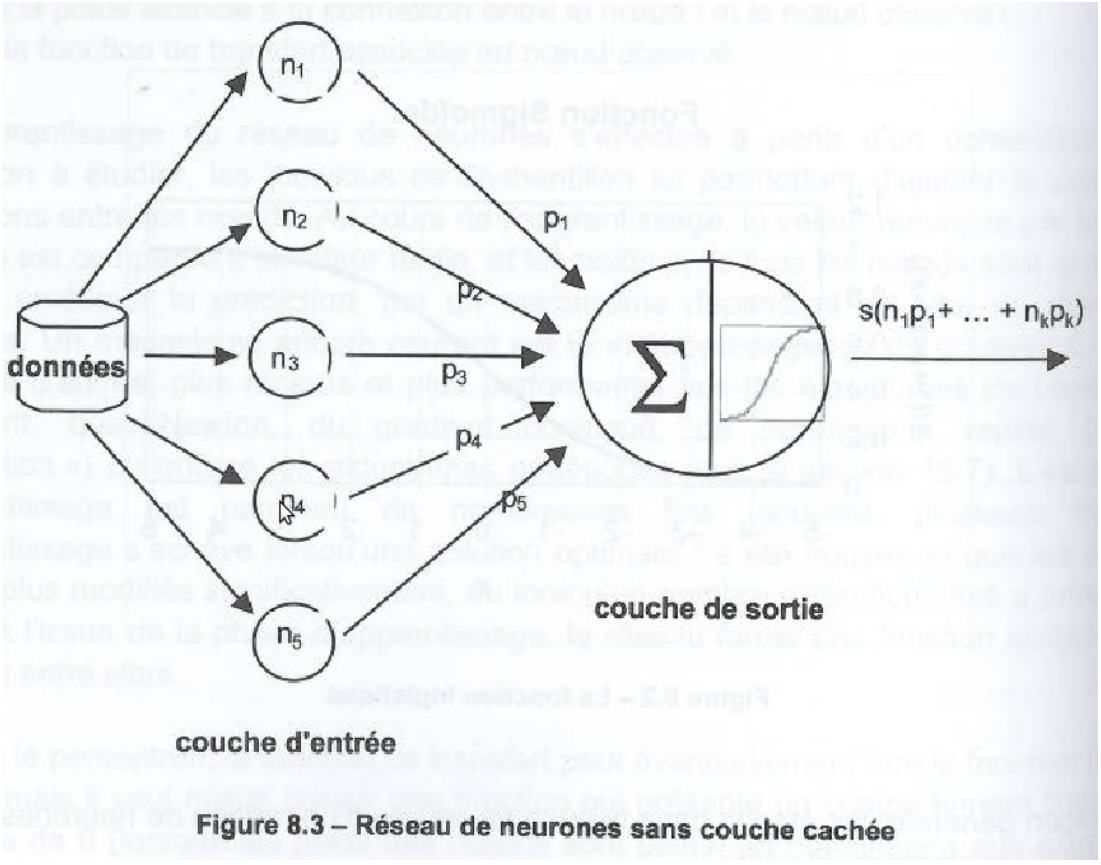
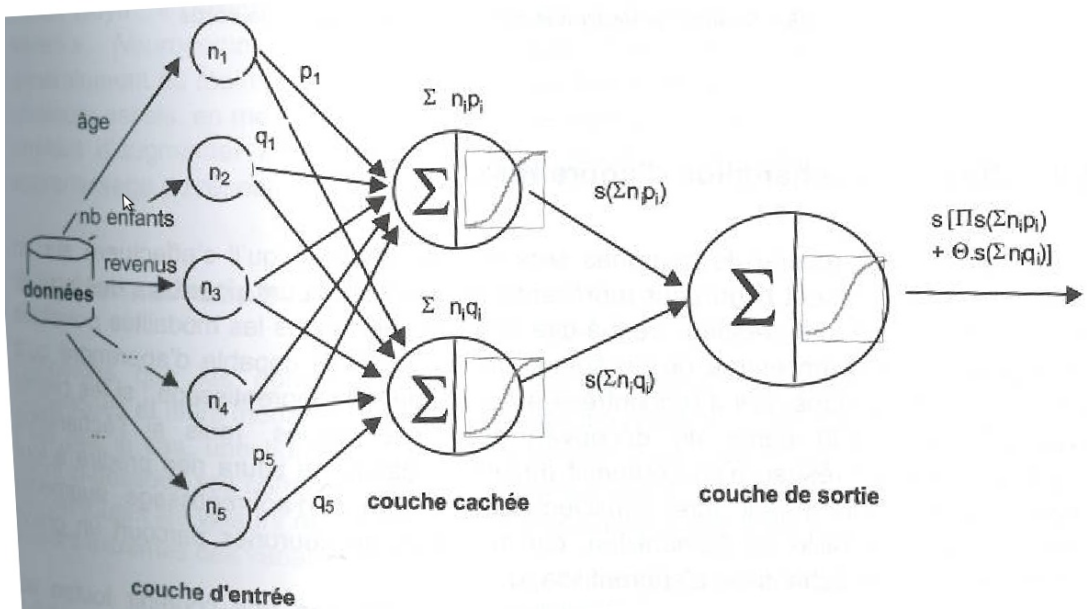


Figure 8.3 – Réseau de neurones sans couche cachée

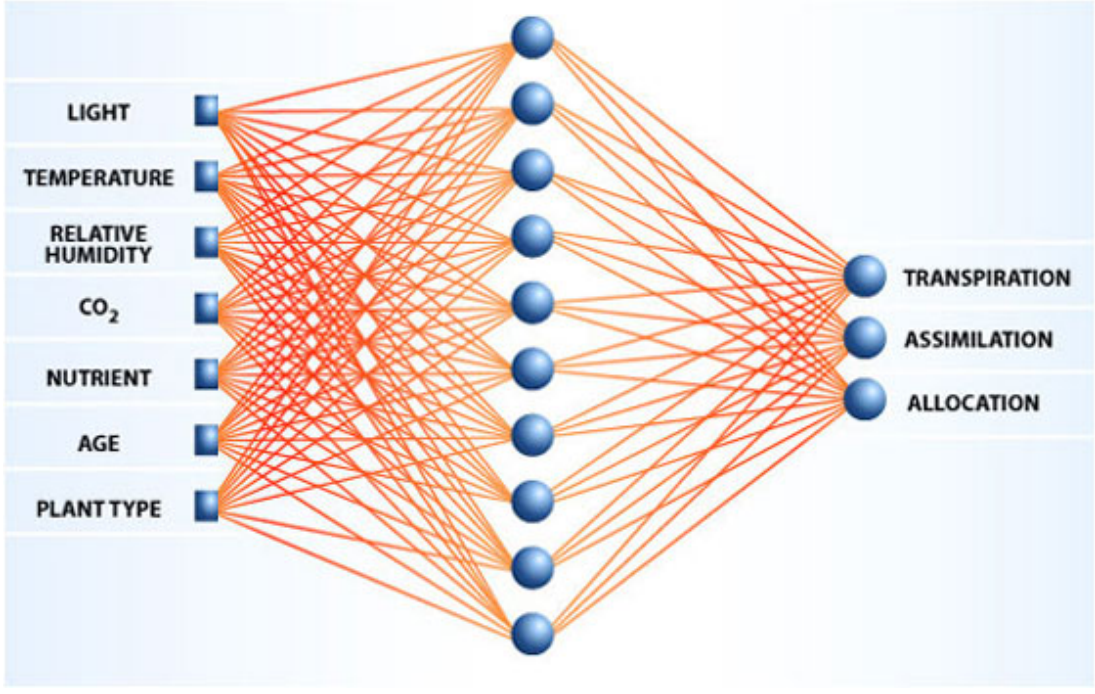
Source : [Tufféry, Data Mining et Informatique Décisionnelle]

Réseau avec couche intermédiaire



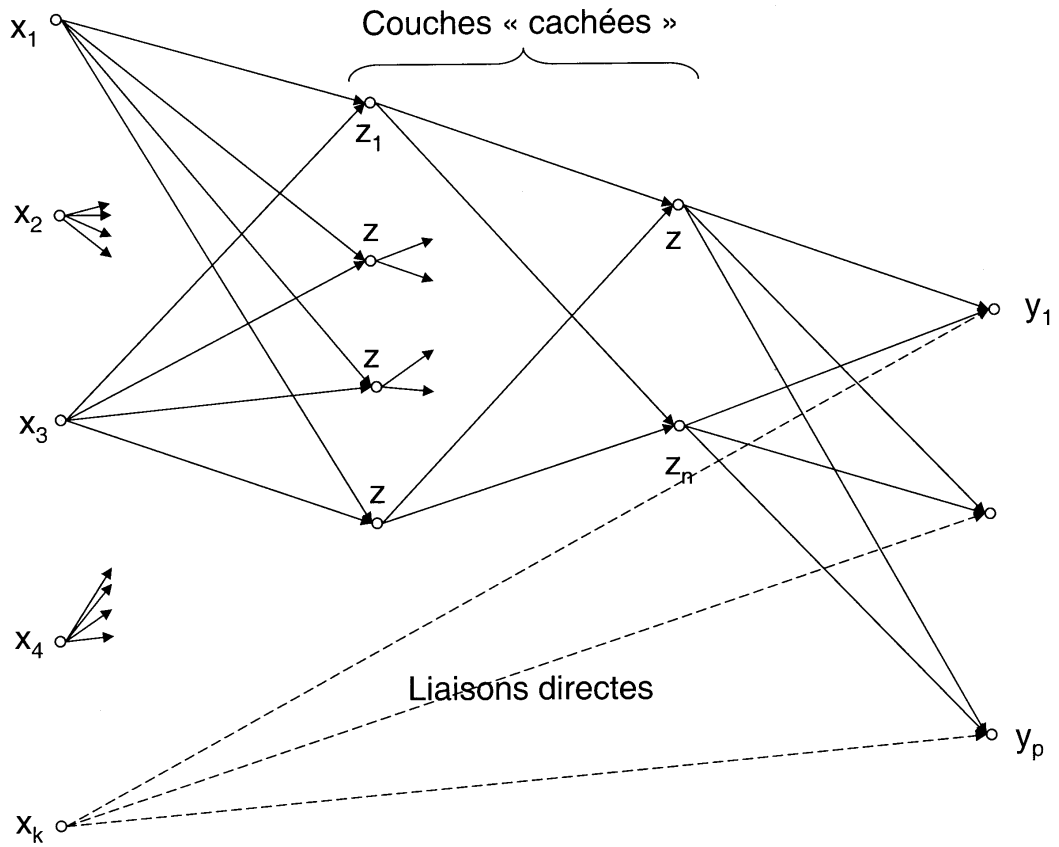
Source : [Tufféry, Data Mining et Informatique Décisionnelle]

Réseau mono-couche



Src : `http://www.makhfi.com`

- Un réseau de neurone est l'association d'objets élémentaires : les neurones formels. C'est en général l'organisation de ces neurones, le nombre de neurones, et leurs types, qui distinguent ces différents réseaux.



- Le neurone formel est un modèle qui se caractérise par un état interne $s \in \mathcal{S}$, des signaux d'entrée $x_1, \cdots, x_p$ et une fonction de transition d'état

$$s = f\left(\beta_0 + \sum_{j=1}^{p} \beta_j x^j\right).$$

- β_0 correspond au biais du neurone.

- Combinaison affine est déterminée par un vecteur de poids $[\beta_0, \cdots, \beta_p]$ associé à chaque neurone et dont les valeurs sont estimées dans la phase d'apprentissage ("mémoire" ou "connaissance répartie" du réseau).

- Il y a plusieurs fonctions de transition possibles :
  - $f(x) = x$ neurone linéaire
  - $f(x) = 1/(1 + e^x)$ neuron sigmoïde
  - $f(x) = \mathbf{1}_{[0;+\infty[}$ fonction de seuillage
  - $f(x) = 1$ avec la probabilité $1/(1 + e^{-H(x)})$ neurone stochastique

- La plupart des problèmes d'apprentissage utilisent les deux premières fonctions de transition.

- La fonction représentant vraisemblablement le mieux la réalité est la fonction de seuillage. Mais elle n'est pas différentiable et donc peu adaptée pour les statisticiens.

- Réseau composé de couches successsives.
  - Pas de connexion entre des réseaux de la même couche.
  - Couche d'entrée $\mathcal{C}_1$: un neurone par variable x^j.
  - Couche de sortie fournit la prédiction y.
- Plusieurs couches cachées.
- Chaque neurone de la couche cachée $\mathcal{C}_k$ est connectée en entrée à chaque neurone de la couche précédente $\mathcal{C}_{k-1}$.
- y est donc calculé par le biais de

$$y = \phi(x^1, \cdots, x^p, \beta) \text{ où } \beta = (\beta_{j,k,l}),$$

$\beta_{j,k,l}$ représente le paramètre de la $j$ème entrée du $k$ème neurone de la $l$ème couche.

- On dispose de $(x_i^1, \cdots, x_i^p, y_i)$ n observations de varibales explicatives $X^1, \cdots, X^p$ et Y variable à prédire.
- On cherche $\hat{\beta}$ solution du problème

$$\hat{\beta} = \text{argmin}_{\mathbf{b}} \mathcal{Q}(\mathbf{b}) \quad \text{où} \quad \mathcal{Q}(\mathbf{b}) = \frac{1}{n} \sum_{i=1}^{n} \left[y_i - \phi\left(x_i^1, \cdots, x_i^p, \mathbf{b} \right) \right]^2 .$$

- C'est un problème hautement non linéaire, le plus souvent. On adopte une méthode de descente de gradient.

- En tout point **b**, le gradient $\mathcal{Q}$ pointe dans la direction de l'erreur croissante.

- Il suffit donc de se déplacer en sens contraire, l'algorithme est itératif modifiant les poids de chaque neurone selon

$$b_{jkl}(i) = b_{jkl}(i-1) + \Delta b_{jkl}(i).$$

- La correction $\Delta b_{jkl}(i)$ est proportionnelle au gradient et à l'erreur attribuée à l'entrée concernée $\epsilon_{jkl}(i)$ et incorpore un terme d'inertie $\alpha b_{jkl}(i-1)$ permettant d'amortir les oscillations du système

$$\Delta b_{jkl}(i) = -\tau \epsilon_{jkl}(i) \frac{\partial \mathcal{Q}}{\partial b_{jkl}}(\mathbf{b}) + \alpha b_{jkl}(i-1).$$

- Le coefficient de proportionnalité τ est appelé taux d'apprentissage. Il peut être fixe, à déterminer par l'utilisateur ou varier en cours d'exécution selon certaines règles paramétrées par l'utilisateur.
- Intuitivement : τ doit être grand au début pour aller plus vite puis décroître pour aboutir à un réglage plus fin au fur et à mesure que le système s'approche d'une solution.
- Une amélioration importante consiste à introduire un terme de pénalisation ou régularisation dans le critère à optimiser :

$$\hat{\beta} = \text{argmin}_{\mathbf{b}} \mathcal{Q}(\mathbf{b}) + \delta ||\mathbf{b}||^2.$$

Le paramètre δ (decay) doit être fixé par l'utilisateur. Plus il est important et moins les paramètres ou poids peuvent prendre des valeurs "chaotiques"' contribuant ainsi à limiter le risque de sur-apprentissage.

- Initialisation
  - Les poids $\beta_{j,k,l}$ par tirage aléatoire selon une loi uniforme sur $[0; 1]$.
  - Normalisation dans $[0; 1]$ des données d'apprentissage.
- Tant que $\mathcal{Q} >$ errmax ou niter $<$ itermax Faire
  - Ranger la base d'apprentissage dans unn nouvel ordre aléatoire.
  - Pour chaque élément $i = 1, \cdots, n$ de la base Faire
    - ⋆ Calculer $\epsilon(i) = y_i - \phi(x_i^1, \cdots, x_i^p, \mathbf{b}(i-1))$ en propageant les entrées vers l'avant.
    - ⋆ L'erreur est "rétropropagée" dans les différentes couches afin d'affecter à chaque entrée une responsabilité dans l'erreur globale.
    - ⋆ Mise à jour de chaque poids $b_{jkl}(i) = b_{jkl}(i-1) + \Delta b_{jkl}(i)$.
  - Fin du Pour
- Fin du Tant que

L'utilisateur doit déterminer

- les variables d'entrée et celle de sortie.
- L'architecture du réseau :
  - le nombre de couches cachées, en général 1 ou 2, qui correspond à une aptitude à triater des problèmes de non-linéarité.
  - le nombre de neurones par couche cachée.
  - Ces 2 choix conditionnent directement le nombre de paramètres (poids) à estimer. Ils participent à la recherche d'un bon compromis biais/variance, i.e. équilibre entre qualité d'apprentissage et qualité de prévision.
  - En pratique, on considère qu'il faut un échantillon d'apprentissage au moins 10 fois plus grand que le nombre de paramètes à estimer.
- Le nombre de maximum d'itérations pour la recherche de $\hat{\beta}$, l'erreur maximale tolérée et le terme de régularisation (decay). En renforçant ces critères, on améliore la qualité d'apprentissage au détriment éventuel de celle de la prévision.
- Le taux d'apprentissage et son éventuelle stratégie d'évolution.

En pratique, tous ces paramètres ne sont pas réglés simultanément par l'utilisateur. Celui-ci est confronté à des choix concernant principalement le contrôle du sur-apprentisage.

- choix du paramètre : limiter le nombre de neurones ou la durée d'apprentissage ou encore augmenter le coefficient de pénalisation de la norme des paramètres.
- choix du mode d'estimation de l'erreur : échantillon test, validation croisée ou bootstrap.

Ces choix sont souvent pris par défaut dans les logiciels. Il est important d'en connaître les implications.

Le nombre de couches reste restreint. En effet toute fonction continue dun compact de $\mathbb{R}^p$ dans $\mathbb{R}^q$ peut être approchée avec une précision arbitraire par un réseau de neurones à une couche cachée en adaptant le nombre de neurones.

Le contrôle de la complexité du modèle peut se faire à l'aide de plusieurs paramètres :

- le nombre de neurones,
- une pénalisation de la norme du vecteur des poids
- ou par la durée de l'apprentissage.

Ces paramètres sont optimisés en considérant un échantillon de validation. Le plus simple consiste à arrêter l'apprentissage lorsque

- l'erreur sur l'échantillon de validation commence à se dégrader
- tandis que celle sur l'échantillon sur l'échantillon d'apprentissage ne peut que continuer à décroître.

- Inconvénients :
  - Temps de calcul important.
  - Taille de l'échantillon.
  - Aspect boîte noire : impossible de connaître l'influence effective d'une entrée (variable) sur le système dès qu'une couche cachée intervient.
  - Minimum local de $\mathcal{Q}$.
- Avantages :
  - Applications aux situations non linéaires.
  - Pondération d'interactions possibles.

Qualités du classifieur par réseau de neurones

Interprétabilité : NON !

Critique : NON

Consistance : NON (mais bonne approximation)

Minimax : NON

Parameter-free : NON

Vitesse : NON

Online : OUI

Roadmap

Super-learning

- On cherche à prédire séquentiellement un phénomène (cours de bourse, charge d'électricité, météo)
- On n'utilise *aucun modèle* (probabiliste ou autre) sur le phénomène
- On s'appuie sur des *experts* plus ou moins fiables
- On cherche à faire (au moins) aussi bien que le meilleur expert

Cadre mathématique

Observations $y_1, y_2, \ldots \in \mathcal{Y}$ - on note $\mathbf{y}_t = (y_s)_{s \leq t}$

A l'instant t, l'expert $j \in \{1, \ldots, N\}$ fournit la *prédiction*

$$f_t^j = f_t^j(\mathbf{y}_{t-1}) \in \mathcal{X}$$

où $\mathcal{X}$ est un ensemble pouvant être distinct de $\mathcal{Y}$

On note $\mathbf{f}_t = (f_s^j)_{1 \leq j \leq t, 1 \leq s \leq t}$

La qualité d'une prédiction est quantifiée par la *fonction de perte* $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$

La *perte cumulée* d'une séquence de prédiction $\mathbf{x}_t = (x_1, \ldots, x_n)$ est

$$L_n(\mathbf{x}_t, \mathbf{y}_t) = \sum_{t=1}^{n} \ell(x_t, y_t)$$

Stratégie randomisée

Prédiction séquentielle $\hat{p}_1, \hat{p}_2, \cdots \in \mathcal{X}$ telles que :

$$\hat{p}_t = \hat{p}_t(\mathbf{y}_{t-1}, \mathbf{f}_{t-1})$$

Stratégie randomisée :

$$\hat{p}_t = f_t^j \text{ avec probabilité } p_t(j)$$

avec la *pondération* $p_t = (p_t(j))_{1 \leq j \leq N} = p_t(\mathbf{y}_{t-1}, \mathbf{f}_{t-1})$

On veut donc minimiser la perte cumulée du décideur

$$\hat{L}_n(\hat{p}, \mathbf{y}_n) = \sum_{t=1}^{n} \ell(\hat{p}_t, y_t)$$

Regret face au meilleur expert

On cherche à prédire au moins aussi bien que le meilleur expert

On définit le *regret* :

$$R_n(\hat{p}, \mathbf{y}_n) = \max_{1 \leq j \leq N} \hat{L}_n(\hat{p}, \mathbf{y}_n) - L_n(j, \mathbf{y}_n)$$

avec $L_n(j, \mathbf{y}_n) =$ regret cumulé de l'expert j.

Regret dans le pire des cas :

$$R_n(\hat{p}) = \sup_{\mathbf{y}_n \in \mathcal{Y}^n} R_n(\hat{p}, \mathbf{y}_n)$$

But : construire $\hat{p}$ de telle sorte que $\limsup R_n(\hat{p})/n \leq 0$

Théorie des Jeux

La théorie des jeux constitue une approche mathématique de problèmes de stratégie tels qu'on en trouve en recherche opérationnelle et en économie.

Elle étudie les situations où les choix de deux protagonistes — ou davantage — ont des conséquences pour l'un comme pour l'autre.

Elle étudie les comportements — prévus, réels, ou tels que justifiés a posteriori — d'individus face à des situations d'*antagonisme*, et cherche à mettre en évidence des *stratégies optimales*.

Formalisation en théorie des jeux

Cadre : ensemble prédictions $\mathcal{X}$, ensemble d'observations $\mathcal{Y}$, fonction de perte $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ bornée par M
Acteurs : un décideur, un environnement
Déroulement : pour chaque instant $t = 1, 2, \ldots$:

1. les experts publient leurs prédictions $(f_t^j)_{1 \leq j \leq N}$
2. le décideur choisit une pondération p_t
3. l'environnement choisit une observation y_t
4. le décideur accorde sa confiance à l'expert j avec probabilité $p_t(j)$
5. le décideur découvre l'observation y_t et enregistre les pertes

$$\Big(\ell(f_t^j, y_t)\Big)_{1 \leq j \leq N}$$

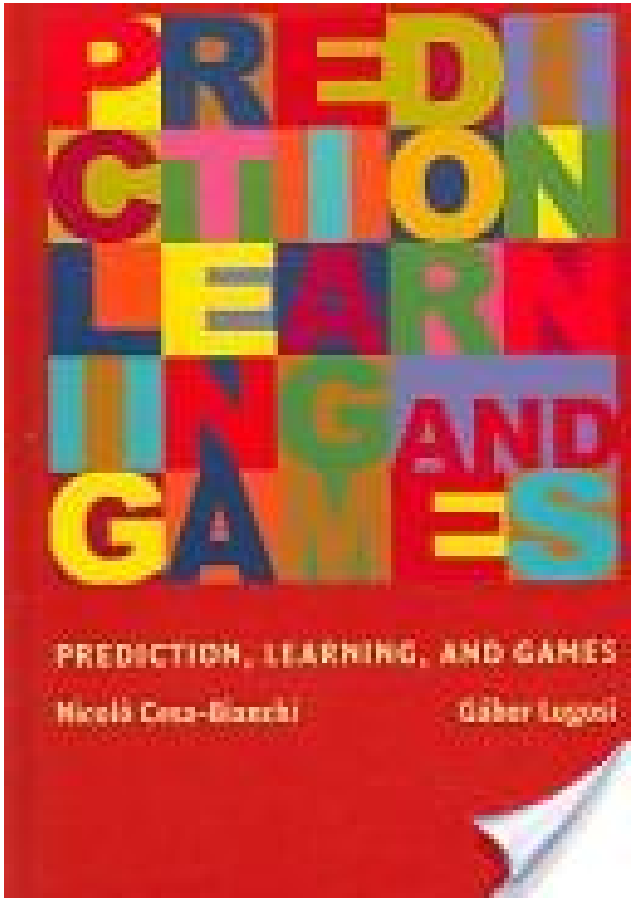
Références

[Cesa-Bianchi & Lugosi '06] Prediction, Learning, and Games

[Devroye & Györfi & Lugosi '96] A Probabilistic Theory of Pattern Recognition

[Györfi & Kohler & Krzyzak & Walk '02] A Distribution-Free Theory of Nonparametric Regression

cf. toute la littérature en théorie de l'information : les liens sont plus étroits qu'il n'y paraît



L'algorithme Exponential Weights (EW)

Stratégie randomisée avec comme choix de pondération :

$$\hat{p}_t(j) = \frac{\exp(-\beta L_{t-1}(j, \mathbf{y}_{t-1}))}{\sum_k \exp(-\beta L_{t-1}(k, \mathbf{y}_{t-1}))}$$

Parfois nommé Hedge, très lié à EXP3 (en feedback bandit).

Borne de regret pour EW

Théorème : Le regret de l'algorithme EW face à la meilleure stratégie constante vérifie :

$$\mathbb{E}\left[R_n(\hat{p})\right] \leq \frac{\log(N)}{\beta} + \frac{M^2\beta}{8}n$$

En particulier, pour $\beta = 1/M\sqrt{8\log(N)/n}$, on obtient :

$$\mathbb{E}\left[R_n(\hat{p})\right] \leq M\sqrt{\frac{n}{2}\log N}$$

Remarque : l'espérance $\mathbb{E}$ porte sur la seule randomisation des choix selon les pondérations

Extensions

- poursuite du meilleur expert
- observation partielle (problèmes de *bandits*)
- Minimisation du regret simple
- Faire aussi bien que la *meilleure combinaison d'expert fixée*

Remarques sur ce cadre

Cadre *méta-statistique* : chaque expert peut avoir son modèle...

L'*agrégation* fait souvent mieux que la *sélection*

Ici, on s'intéresse à de l'*agrégation séquentielle*

La formulation choisie requiert des stratégies *robustes*

Exemple : Prévision de charge

Cf thèse de Yannig Goude (EDF & Université Paris-Sud)
Une problématique cruciale de la production électrique est la *prévision de charge*

On dispose de *plusieurs modèles* plus ou moins évolués / robustes / expérimentés

Le but est d'agréger leurs résultats en les utilisant comme des emphboîtes noires

Prédiction de la qualité de l'air

Cf Gilles Stoltz (CNRS) et Vivient Mallet (INRIA), Journal of Geophysical Research

Objectif : prédire, jour après jour, les hauteurs des pics d'ozone du lendemain (ou les concentrations horaires, heure après heure)

Moyens : réseau de stations météorologiques à travers l'Europe (été 2001)

Experts : 48 prédicteurs fondamentaux, chacun construit à partir

- d'un modèle physico-chimique
- d'un schéma numérique de résolution approché des EDP en jeu
- d'un jeu de données

Tous les experts sont utiles

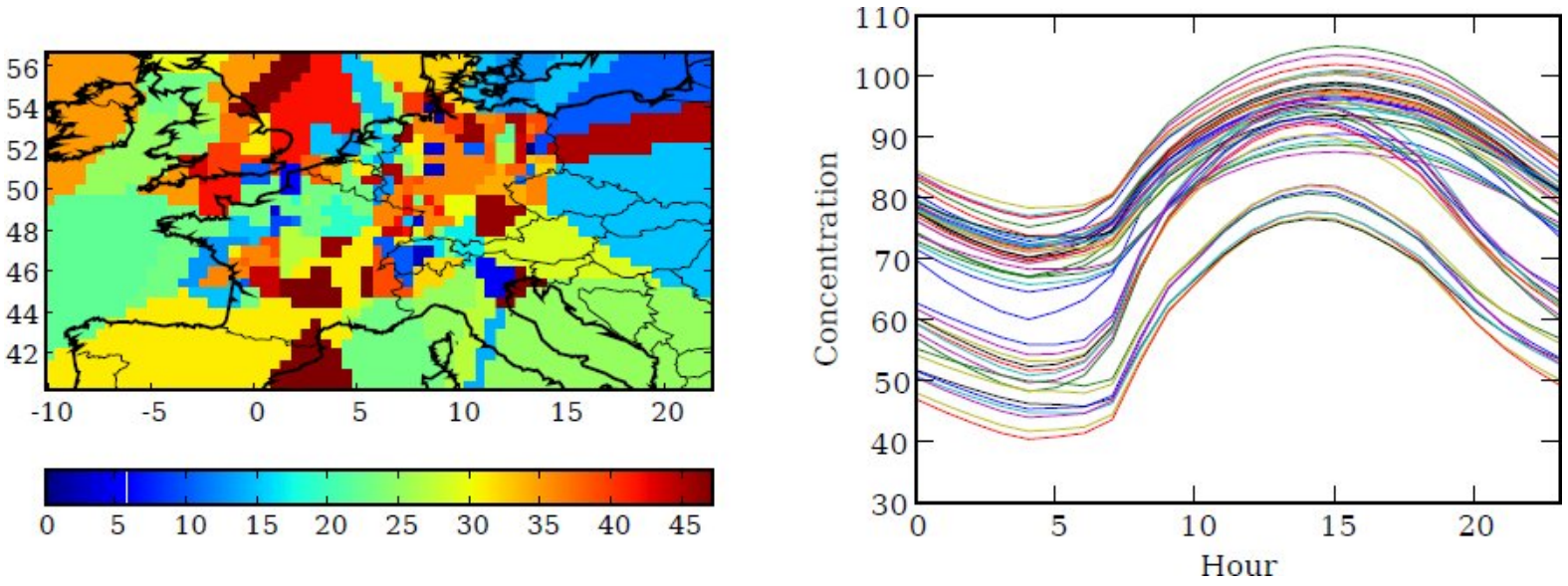


FIGURE: A gauche : Coloration de l'Europe en fonction de l'indice du meilleur expert local. A droite : Profils moyens de prédiction sur une journée (moyennes spatiales et temporelles, en $\mu g/m^3$)

Résultats

Moyenne	M. fondamental	M. convexe	M. linéaire	EW	
24.41	22.43	21.45	19.24	11.99	21.47

Ci-dessus, les erreurs quadratiques moyennes (en $\mu g/m^3$)

- de la moyenne des prédictions des 48 modèles
- du meilleur modèle fondamental parmi les 48
- de la meilleure combinaison convexe
- de la meilleure combinaison linéaire
- de l'algorithme EW

$\Longrightarrow$ la meilleure combinaison convexe constante est battue

on peut faire mieux (voir Stoltz & Mallet)

Gestion de portefeuilles

[Cover '91] Universal Portfolios
[Györfi & Urban & Vajda '07] Kernel-based semi-log-optimal portfolio selection strategies

N actifs $\{1, \ldots, N\}$, prix Z_t^j

Market Vector $x_t =$ vecteur d'évolution des prix $x_t^j = Z_t^j / Z_{t-1}^j$

Position $Q_t =$ vecteur des fractions d'investissement $Q_t^j =$ part du patrimoine dans l'actif j à l'instant t.

Wealth Factor = rendement du placement

$$S_n(Qn, \mathbf{x}_n) = \prod_{t=1}^{n} \left(\sum_{j=1}^{N} x_t^j Q_t^j \right)$$

Roadmap

Plan

- Introduction
- Familles de modèles aléatoires
  - Bagging
  - Forêts aléatoires
- Familles de modèles adaptatifs
  - Principes du Boosting
  - Algorithme de base
  - Pour la régression
  - Modèle additif pas à pas
  - Régression et boosting
  - Compléments

Présentation d'algorithmes basés sur des stratégies adaptatives (boosting) ou aléatoires (bagging) permettant d'améliorer l'ajustement par combinaison ou agrégation d'un grand nombre de modèles tout en évitant un sur-ajustement.

Deux types d'algorithmes sont décrits

- Ceux reposants sur une construction aléatoire d'une famille de modèle :
  - bagging pour bootstrap agregating
  - les forêts aléatoires (random forest) qui est une amélioration du bagging spécifique aux modèles définis par des arbres binaires.
- Ceux basés sur le boosting reposent sur une construction adaptative, déterministe ou aléatoire, d'une famille de modèles.

Les principes de bagging ou de boosting s'appliquent à toute méthode de modélisation (régression, CART, réseaux de neurones) mais n'ont d'intérêt et réduisent sensiblement l'erreur de prédiction, que dans le cas de modèles instables, donc plutôt non linéaires.

Soient Y une variable à expliquer quantitative ou qualitative, $X^1, \ldots, X^p$ les variables explicatives et $\phi(\mathbf{x})$ un modèle de fonction de $\mathbf{x} = \{x^1, \ldots, x^p\} \in \mathbb{R}^p$. On note n le nombre d'observations et $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ un échantillon de loi F.

$\phi = \mathbb{E}_F\left(\widehat{\phi}_{\mathbf{z}}\right)$ est un estimateur sans biais de variance nulle.

Considérons B échantillons indépendants notés $\left\{\mathbf{z}_{b=1,\ldots,B}\right\}$ et construisons une agrégation de modèles :

- Si Y est quantitative : $\widehat{\phi}_B(.) = \dfrac{1}{B}\displaystyle\sum_{i=1}^{B} \widehat{\phi}_{\mathbf{z}_b}(.)$.

  Moyenne des résultats obtenus pour les modèles associés à chaque échantillon.
- Si Y est qualitative : $\widehat{\phi}_B(.) = \underset{j}{\text{argmax}}\ \text{card}\left\{b\,;\, \widehat{\phi}_{\mathbf{z}_b}(.) = j\right\}$.

  Comité de modèles constituer pour *voter* et *élire* la réponse la plus probable.

Principe : Moyenner les prédictions de plusieurs modèles indépendants permet de réduire la variance et donc de réduire l'erreur de prédiction.

Cependant, il n'est pas réaliste de considérer B échantillons indépendants (nécessite trop de données). Ces échantillons sont donc remplacés par B réplications d'échantillons bootstrap obtenus chacun par n tirages avec remise selon la mesure empirique F_n.

Algorithme

- Soit $\mathbf{x}_0$ à prévoir.
- Soit $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ un échantillon.
- Pour $b = 1, \ldots, B$
  - Tirer un échantillon bootstrap $\mathbf{z}_b^*$.
  - Estimer $\widehat{\phi}_{\mathbf{z}_b}(\mathbf{x}_0)$ sur l'échantillon bootstrap.
- Calculer l'estimation $\widehat{\phi}_B(\mathbf{x}_0) = \dfrac{1}{B}\sum_{b=1}^{B} \widehat{\phi}_{\mathbf{z}_b}(\mathbf{x}_0)$ ou le résultat du vote.

UTILISATION

Il est naturel et techniquement facile d'accompagner ce calcul par une estimation bootstrap out-of-bag de l'estimation de prédiction.

- Elle mesure la qualité de généralisation du modèle.
- Elle permet de prévenir une éventuelle tendance au sur-ajustement.

La moyenne des erreurs de prédiction commises par chaque estimateur permet d'éviter le biais : chacune des erreurs étant estimée sur les observations n'ayant pas été sélectionnées par l'échantillon bootstrap correspondant.

UTILISATION

En pratique, CART est souvent la méthode de base pour construire une famille de modèles. Trois stratégies d'élagage sont alors possibles :

1. Laisser construire et garder un arbre complet pour chacun des échantillons.
2. Construire un arbre d'au plus d feuilles.
3. Construire à chaque fois l'arbre complet puis l'élaguer par validation croisée.

UTILISATION

La première stratégie semble en pratique un bon compromis entre volume des calculs et qualité de prédiction. Chaque arbre est alors affecté d'un faible biais et d'une grande variance mais la moyenne des arbres réduit avantageusement celle-ci.

En revanche, l'élagage par validation croisée pénalise lourdement les calculs sans gain substantiel de qualité.

UTILISATION

Avantages / Inconvénients :

- Cet algorithme simple s'adapte et se programme facilement quelque soit la méthode de modélisation mise en oeuvre.
- Temps de calculs important pour évaluer un nombre suffisant d'arbres jusqu'à ce que l'erreur de prédiction *out-of-bag* ou sur un échantillon de validation se stabilise et arrêt si elle tend à augmenter.
- Nécessité de stocker tous les modèles de la combinaison afin de pouvoir utiliser cet outil de prédiction sur d'autres données.
- L'amélioration de la qualité de prédiction se fait au détriment de l'interprétabilité. Le modèle finalement obtenu devient une *boîte noire* comme dans le cas du perceptron.

Dans les cas spécifiques des modèles CART, l'algorithme du bagging a été amélioré par l'ajout d'une randomisation.

Objectif : rendre plus *indépendants* les arbres de l'agrégation en ajoutant du hasard dans le choix des variables qui interviennent dans les modèles.

Intérêt : Cette approche semble plus particulièrement fructueuse dans des situations hautement multidimensionnelles, c-à-d lorsque le nombre de variables explicatives p est très important (discriminer des courbes, spectres, signaux, biopuces).

Algorithme

- Soient $\mathbf{x}_0$ à prévoir et $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ un échantillon.
- Pour $b = 1, \ldots, B$
  - Tirer un échantillon bootstrap $\mathbf{z}_b^*$.
  - Estimer un arbre sur cet échantillon avec randomnisation des variables : la recherche de chaque noeud optimal est précédé d'un tirage aléatoire d'un sous-ensemble de q prédicteurs.
- Calculer l'estimation $\widehat{\phi}_B(\mathbf{x}_0) = \dfrac{1}{B}\displaystyle\sum_{b=1}^{B} \widehat{\phi}_{\mathbf{z}_b}(\mathbf{x}_0)$ ou le résultat du vote.

ÉLAGAGE

La stratégie d'élagage peut, dans le cas des forêts aléatoires, être plus élémentaire qu'avec le bagging en se limitant à des arbres de taille q relativement réduite voire même triviale $q \approx 2$.

- Avec le bagging, des arbres limités à une seule fourche risquent d'être très semblables (fortement corrélés) car impliquant les mêmes quelques variables apparaissant comme les plus explicatives.

- La sélection aléatoire d'un nombre réduit de prédicteurs potentiels à chaque étape de construction d'un arbre, accroît significativement la variabilité en mettant en avant nécessairement d'autres variables. Chaque modèle de base est évidemment moins performant mais l'agrégation conduit finalement à de bons résultats.

ÉLAGAGE

Le nombre de variables tirées aléatoirement n'est pas un paramètre sensible :

$$q = \sqrt{p} \text{ si discrimination}$$

et

$$q = \frac{p}{3} \text{ si régression.}$$

Comme pour le bagging, l'évaluation itérative de l'erreur out-of-bag prévient d'un éventuel sur-ajustement si celle-ci vient à se dégrader.

INTERPRÉTATION

Comme pour tout modèle construit par agrégation, il n'y a pas d'interprétation directe.

Plusieurs critères sont proposés pour évaluer l'importance d'une variable

- Le critère Mean Decrease Accuracy repose sur une permutation aléatoire des valeurs de cette variable.
  - Calculer de la moyenne sur les observations *out-of-bag* de la décroissance de leur marge lorsque la variable est aléatoirement perturbée.
  - La marge est la proportion de votes pour la vraie classe d'une observation moins le maximum des proportions de votes pour les autres classes.
  - Mesure globale mais indirecte de l'influence d'une variable sur la qualité des prévisions. Plus la prévision est dégradée par la permutation des valeurs d'une variable, plus celle-ci est importante.

INTERPRÉTATION

- Le critère Mean Decrease Gini est local, basé sur la décroissance d'entropie ou sur la décroissance de l'hétérogénéité définie par le critère de Gini. L'importance d'une variable est alors une somme pondérée des décroissances d'hétérogénéité induites lorsqu'elle est utilisée pour définir la division associée à un noeud.

- Il existe un troisème critère plus rudimentaire qui s'intéresse simplement à la fréquence de chacune des variables apparaissant dans les arbres de la forêt.

INTERPRÉTATION

- Comparaison des 3 critères

  - Les critères 1 et 2 sont très proches, l'importance d'une variable dépend de sa fréquence d'apparition mais aussi des places qu'elle occupe dans chaque arbre. Ils sont pertinents pour une discrimination de 2 classes ou, lorsqu'il y a plus de 2 classes, si celles-ci sont relativement équilibrées.

  - Dans le cas contraire, c-à-d si une des classes est moins fréquente et plus difficile à discriminer, le 3ème critère présente un avantage : il donne une certaine importance aux variables qui sont nécessaires à la discrimination d'une classe difficile alors que celles-ci sont négligées par les 2 autres critères.

Principe : Construction, de façon récurrente, d'une famille de modèles qui sont ensuite agrégés par une moyenne pondérée des estimations ou un vote.

Chaque modèle est une version adaptative du modèle précédent en donnant plus de poids, lors de l'étape suivante, aux observations mal ajustées ou mal prédites.

Idée : Cet algorithme concentre ses efforts sur les observations les plus difficiles à ajuster et l'agrégation de l'ensemble des modèles permet d'échapper au sur-ajustement.

Les algorithmes de boosting diffèrent par différents caractéristiques :

- la façon de pondérer, c-à-d de renforcer l'importance des observations mal estimées lors de l'itération précédente,
- leur objectif selon le type de la variable à prédire Y : binaire, qualitative à k modalités, réelle,
- la fonction de perte, qui peut être choisie plus ou moins robuste aux valeurs atypiques, pour mesurer l'erreur d'ajustement,
- la façon d'agréger, ou plutôt de pondérer les modèles de base successifs.

ALGORITHME DE BASE

Adaboost discret : Version originale du boosting pour un problème de discrimination élémentaire à deux classes en notant δ la fonction de discrimination à valeurs dans $\{-1, 1\}$.

Algorithme

- Soient $\mathbf{x}_0$ à prévoir et $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ un échantillon.
- Initialiser les poids $\omega = \{\omega_i = 1/n; i = 1, \ldots, n\}$.
- Pour $m = 1, \ldots, M$
  - Estimer δ_m sur l'échantillon pondéré par ω.
  - Calculer le taux d'erreur apparent

$$\widehat{\mathcal{E}}_P = \frac{\sum_{i=1}^{n} \omega_i \ \mathbb{1}_{\delta_m(\mathbf{x}_i) \neq y_i}}{\sum_{i=1}^{n} \omega_i}.$$

  - Calculer les logit : $c_m = \log[(1 - \widehat{\mathcal{E}}_P)/\widehat{\mathcal{E}}_P]$.
  - Calculer les nouvelles pondérations : $\omega_i := \omega_i \exp[c_m \ \mathbb{1}_{\delta_m(\mathbf{x}_i) \neq y_i}]; i = 1, \ldots, n.$
- Résultat du vote : $\widehat{\phi}_M(\mathbf{x}_0) = signe \left[\sum_{m=1}^{M} c_m \delta_m(\mathbf{x}_0) \right]$.

RÉGRESSION

- Soit $\mathbf{x}_0$ à prévoir et $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ un échantillon.
- Initialiser $\mathbf{p}$ par la distribution uniforme $\mathbf{p} = (p_i = 1/n; i = 1, \ldots, n)$.
- Pour $m = 1$ à M
  - Tirer avec remise dans $\mathbf{z}$ un échantillon $\mathbf{z}^*_m$ suivant $\mathbf{p}$.
  - Estimer $\widehat{\phi}_m$ sur l'échantillon initial $\mathbf{z}$ avec Q fonction de perte :

$$l_m(i) = Q\left(y_i, \widehat{\phi}_m(\mathbf{x}_i\right), \quad L_m = \sup_{i=1,\cdots n} l_m(i), \quad \widehat{\epsilon}_m = \sum_{i=1}^{m} p_i l_m(i),$$

$$\omega_i = \beta_m^{1-l_m(i)/L_m} p_i \quad \text{où} \quad \beta_m = \frac{\widehat{\epsilon_m}}{L_m - \widehat{\epsilon}_m}.$$

  - Calculer les nouvelles probabilités : $p_i := \dfrac{\omega_i}{\sum_{i=1}^{n} \omega_i}$.
- Calculer $\widehat{\phi}(\mathbf{x}_0)$ moyenne ou médiane des prévisions $\widehat{\phi}_m(\mathbf{x}_0)$ pondérées par des coefficients $\log\left(\dfrac{1}{\beta_m}\right)$.

RÉGRESSION

- Dans cet algorithme, la fonction perte Q peut être exponentielle, quadratique ou, plus robuste, la valeur absolue.
- Une condition supplémentaire peut être ajoutée. L'algorithme est arrêté ou réinitialisé à des poids uniformes si l'erreur se dégrade trop : $\widehat{\epsilon}_m < 0,5 L_m$.

L'algorithme génère M prédicteurs construits sur des échantillons bootstrap $\mathbf{z}^*_m$ dont le tirage dépend des probabilités $\mathbf{p}$ mises à jour à chaque itération, mise à jour dépendant

- d'un paramètre β_m indicateur de la performance sur l'échantillon $\mathbf{z}$ du $m^{\text{ème}}$ prédicteur estimé sur l'échantillon $\mathbf{z}^*_m$,
- de la qualité relative $l_m(i)/L_m$ de l'estimation du $i^{\text{ème}}$ individu.

L'estimation finale est enfin obtenue à la suite d'une moyenne ou médiane des prévisions pondérées par la qualité respective de chacune des prévisions.

- Sur-ajustement :
  - Le nombre d'itérations peut être contrôlé par un échantillon de validation. Comme pour d'autres méthodes (perceptron), il suffit d'arrêter la procédure lorsque l'erreur estimée sur cet échantillon arrive à se dégrader.
  - Une autre possibilité consiste à ajouter un coefficient de rétrécissement. Compris entre 0 et 1, celui-ci pénalise l'ajout d'un nouveau modèle dans l'agrégation. Il joue le rôle d'un taux d'apprentissage du perceptron et si sa valeur est petite ($< 0,1$), cela conduit à accroître le nombre d'arbres mais entraîne des améliorations sensibles de la qualité de prédiction.
- Instabilité :
  - Les modèles construits à base d'arbres sont très instables : une légère modification des données est susceptible d'engendrer de grandes modifications dans les paramètres (les seuils et feuilles) du modèle. C'est justement cette propriété qui rend cette technique très appropriée à une amélioration par agrégation.

- Interprétabilité :
  - L'interprétabilité des arbres de décision sont une des raisons de leur succès. Cette propriété est évidemment perdue par l'agrégation d'arbres ou de tout autre modèle. Néanmoins, surtout si le nombre de variables est très grand, il est important d'avoir une indication de l'importance relative des variables entrant dans la modélisation.

  - Un critère est calculé pour chaque variable j à partir des valeurs $D_j^2(l, m)$, calculées pour chaque noeud l de chaque arbre m. Cette quantité est la décroissance optimale de déviance produite par la segmentation associée à ce noeud par le choix de la variable j. Ces valeurs sont sommées par arbre sur l'ensemble des noeuds puis moyennées sur l'ensemble des arbres. Une normalisation fixe à 100 la plus grande valeur correspondant à la variable la plus influente.

Qualités des classifieurs obtenus par agrégation

Interprétabilité : NON !

Critique : NON

Consistance : NON (mais empiriquement efficace)

Minimax : NON

Parameter-free : NON

Vitesse : NON

Online : OUI

- Introduction
- SVM : Principe de fonctionnement général
  - Hyperplan, marge et support vecteur.
  - Linéairement séparable. Non-linéairement séparable.
  - Cas non linéaire. Le cas XOR.
- Fondements mathématiques
  - Problème d'apprentissage.
  - Cas linéaire. Maximisation de la marge.
  - Cas linéaire. Marges larges relaxées.
  - De la linéarité à la non-linéarité : Fonction noyau et Condition de Mercer.
- Conclusion

Objectif : Explication d'une variable Y par p variables X^j, $j = 1, \ldots, p$ observées sur un même échantillon Ω de taille n. On note $\mathbf{X}$ la matrice des variables explicatives.

On dispose d'un nouvel individu (ou de plusieurs) sur lequel on a observé les X^j mais pas Y. Il s'agit de prédire la valeur de Y de ce nouvel individu.

Seul le cas $Y \in \{-1, 1\}$ sera traité ici. On peut facilement étendre à $\text{card}(Y) > 2$ et à $Y \in \mathbb{R}$.

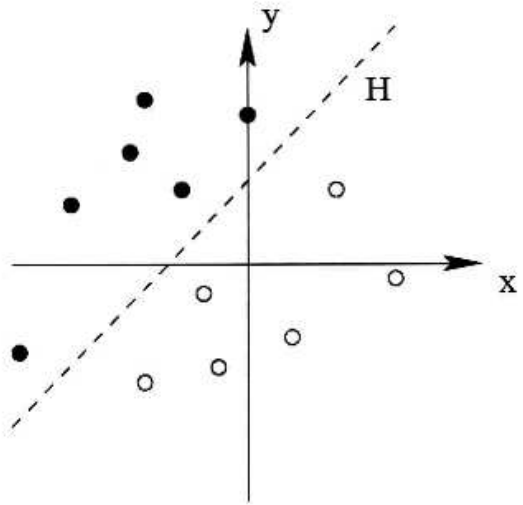
La distribution jointe de $(\mathbf{X}, Y)$, notée $\mathbb{P}$, est inconnue.

SVM est une méthode de classification binaire par apprentissage supervisé.

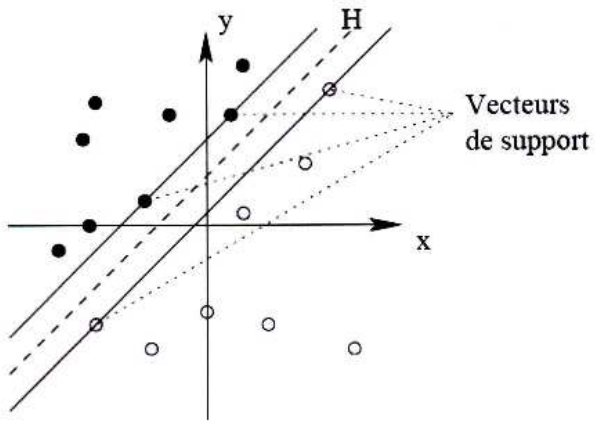
Cette méthode repose sur

- l'existence d'un classifieur linéaire dans un espace approprié,
- l'utilisation de fonction noyau qui permettent une séparation optimale des données.

But : Trouver un classifieur linéaire (hyperplan) qui va séparer les données et maximiser la distances entre ces 2 classes.

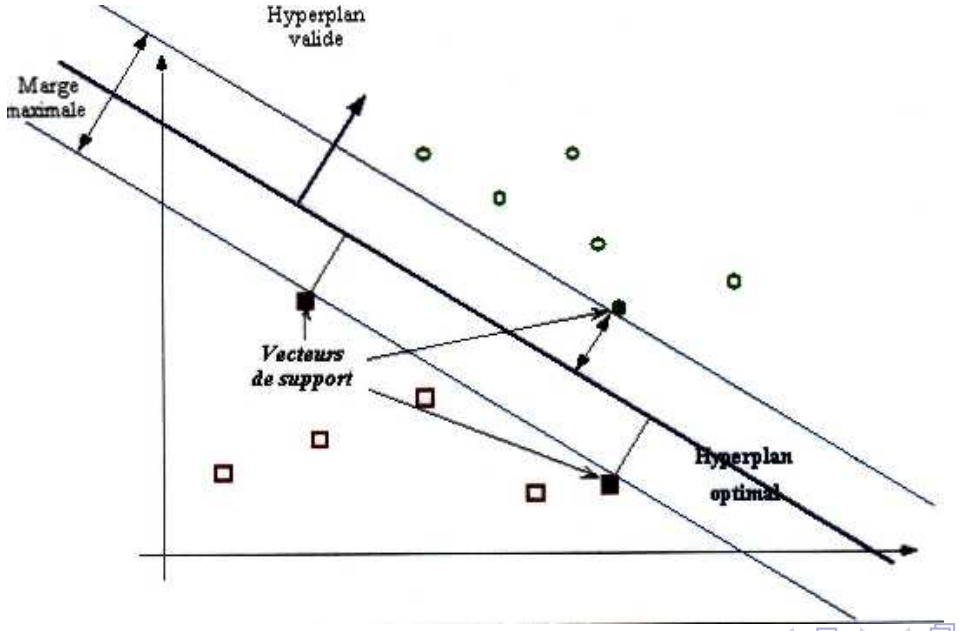


Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support.

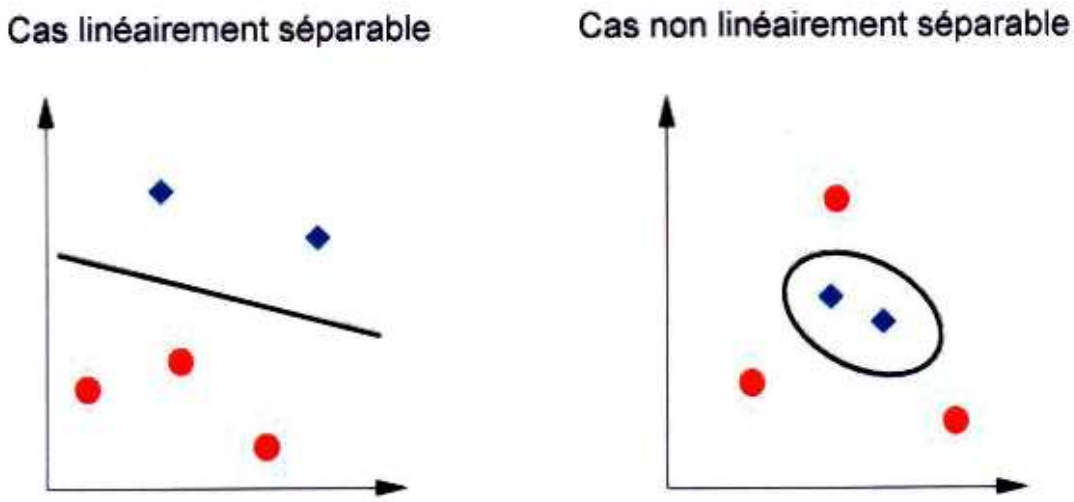


Intuition : Parmi les hyperplans valides, chercher l'hyperplan le "plus sûr".
En supposant qu'un exemple n'ait pas été parfaitement décrit, une petite variation ne modifiera pas sa classification si sa distance à l'hyperplan est grande.

Objectif : Parmi les hyperplans valides, chercher l'hyperplan dont la distance minimale aux exemples d'apprentissage est maximale. Cette distance est appelée distance marge entre l'hyperplan et les exemples.



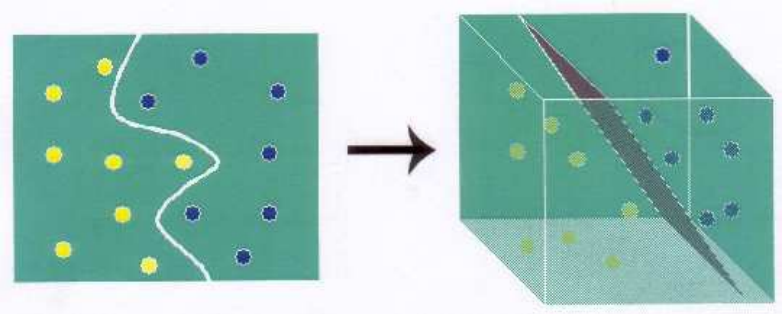
Parmi les modèles des SVMs, on distingue les cas linéairement séparables et les cas non-linéairement séparables. Dans les premiers cas, il est facile de trouver le classifieur linéaire.



Dans la plupart des problèmes réels, il n'y a pas de séparation linéaire possible entre les données.

Idée : changer l'espace des données afin de surmonter l'inconvénient des cas non-linéairement séparables. La transformation non-linéaire des données peut permettre une séparation linéaire des exemples dans un nouvel espace, espace de représentation.

Intuitivement, plus la dimension de l'espace de représentation est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée.

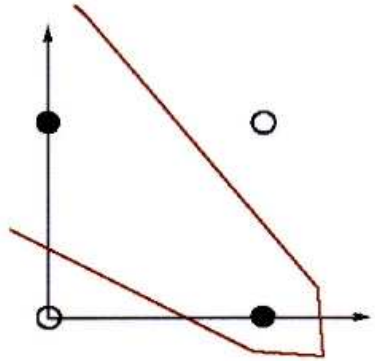


- Transformation d'un problème de séparation non-linéaire dans l'espace de représentation en un problème de séparation linéaire dans un espace de représentation de plus grande dimension.

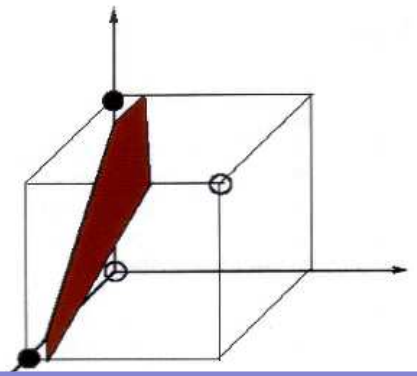
- Cette transformation non-linéaire est réalisée via une fonction noyau. En pratique, quelques familles de fonctions noyau paramétrables sont connues : polynômiale, gaussien, sigmoïde et laplacien. Il revient à l'utilisateur de SVM d'effectuer des tests pour déterminer celle qui convient le mieux pour son application.

Le cas XOR

Le cas XOR $\{(0,0);(0,1);(1,0),(1,1)\}$ n'est pas linéairement séparable, si on place les points dans un plan à dimension 2.



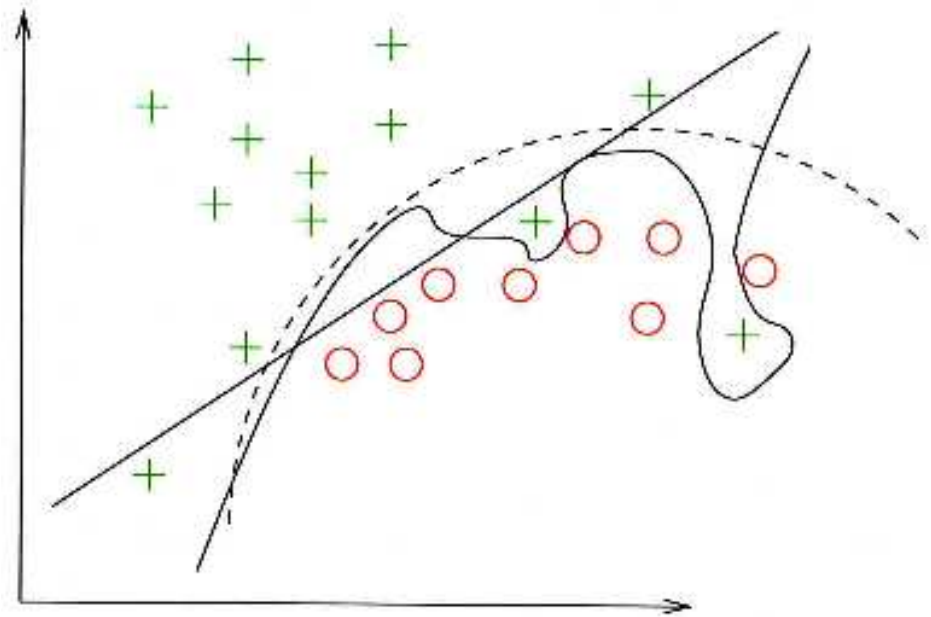
Si on prend une fonction polynômiale $(x,y) \mapsto (x,y,x \times y)$ qui fait passer d'un espace de dimension 2 à un espace de dimension 3, on obtient un problème en dimension 3 linéairement séparable.



Objectif : Sachant qu'on observe un échantillon $\{(\mathbf{X}_1, Y_1); \ldots; (\mathbf{X}_n, Y_n\})$ de n copies indépendantes de $(\mathbf{X}, Y)$, on veut construire une fonction $f : \mathbf{X} \mapsto Y$ telle que

$$\mathbb{P}[f(\mathbf{X}) \neq Y] = \mathbb{P}[f(\mathbf{X})Y \leq 0] \text{ soit minimale.}$$

Comme à chaque fois, il faut éviter le sous et le sur-apprentissage. Il faut trouver un compromis entre adéquation aux données et complexité du modèle afin de pouvoir généraliser.



Un hyperplan est défini à l'aide du produit scalaire de $\mathcal{X}$ par

$$< \mathbf{w}, \mathbf{x} > + b = 0$$

où $\mathbf{w}$ est un vecteur orthogonal au plan.
Le signe de $f(\mathbf{x}) =< \mathbf{w}, \mathbf{x} > + b$ indique le côté où se trouve le point $\mathbf{x}$.
Un point est bien classé si et seulement si

$$yf(\mathbf{x}) > 0.$$

Mais comme $(\mathbf{w}, b)$ caractérise le plan à un coefficient multiplicatif près, on s'impose

$$yf(\mathbf{x}) \geq 1.$$

Un plan $(\mathbf{w}, b)$ est un séparateur si

$$\forall i \in \{1, \cdots, n\}, \quad y_i f(\mathbf{x}_i) \geq 1.$$

La distance d'un point $\mathbf{x}_0$ au plan $(\mathbf{w}, b)$ est donnée par

$$d(\mathbf{x_0}) = \frac{|\mathbf{w}.\mathbf{x_0} + b|}{||\mathbf{w}||} = \frac{|f(\mathbf{x})|}{||\mathbf{w}||}.$$

La marge du plan a pour valeur $\frac{2}{||\mathbf{w}||^2}$.

Chercher le plan séparateur de marge maximale revient à résoudre le problème d'optimisation sous contraintes

$$\begin{cases} \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 \\ \text{avec } \forall i \in \{1, \cdots, n\}, \quad y_i(< \mathbf{w}, \mathbf{x}_i > + b) \geq 1. \end{cases}$$

La solution est fournie par le point-selle $(\mathbf{w}^*, b^*, \lambda^*)$ du lagrangien

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \lambda_i \left[y_i(< \mathbf{w}, \mathbf{x}_i > +b) - 1\right].$$

Ce point-selle vérifie

$$\forall i \in \{1, \cdots, n\}, \quad \lambda_i^* \left[y_i(< \mathbf{w}^*, \mathbf{x}_i > +b^*) - 1\right] = 0.$$

Les vecteurs supports sont les vecteurs $\mathbf{x}_i$ pour lesquels la contrainte est active, i.e. les plus proches du plan :

$$y_i(< \mathbf{w}^*, \mathbf{x}_i > +b^*) = 1.$$

En calculant les dérivées partielles du lagrangien, avec les λ_i^* non nuls seulement pour les points supports, on obtient :

$$\mathbf{w}^* = \sum_{i=1}^{n} \lambda_i^* y_i \mathbf{x}_i \quad \text{et} \quad \sum_{i=1}^{n} \lambda_i^* y_i = 0.$$

Ces contraintes d'égalité permettent d'exprimer la formule duale du lagrangien

$$W(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j > .$$

Pour trouver le point-selle, il suffit alors de maximiser $W(\lambda)$ avec $\lambda_i \geq 0, \forall i \in \{1, \cdots, n\}$.

La résolution de ce problème d'optimisation quadratique de taille n fournit l'équation de l'hyperplan optimal :

$$\sum_{i=1}^{n} \lambda_i^* y_i < \mathbf{x}_i, \mathbf{x}_j > + \mathbf{b}^* = 0$$

avec

$$b^* = -\frac{1}{2} \left[< \mathbf{w}^*, sv_{class+1} > + < \mathbf{w}^*, sv_{class-1} > \right].$$

Pour une nouvelle observation **x** non apprise présentée au modèle, il suffit de regarder le signe de

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i^* y_i < \mathbf{x}, \mathbf{x}_i > + \mathbf{b}^*,$$

pour savoir dans quel demi-espace cette forme se trouve et donc quelle classe lui sera attribuée.

Inconvénient de cette méthode :

- Très sensible aux outliers.

Lorsque les observations ne sont pas séparables par un plan, il est nécessaire d'"assouplir" les contraintes par l'introduction de termes d'erreur ξ qui en contrôlent le dépassement :

$$y_i(< \mathbf{w}, \mathbf{x}_i > +b) \geq 1 - \xi_i, \ \forall i \in \{1, \cdots, n\}.$$

Le modèle attribue ainsi une réponse fausse à un vecteur $\mathbf{x}_i$ si le ξ_i correspondant est supérieur à 1. La somme de tous les ξ_i représente donc une borne du nombre d'erreurs.

Le problème de minimisation est réécrit en introduisant une pénalisation par le dépassement de la contrainte

$$\begin{cases} \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \delta \sum_{i=1}^{n} \xi_i \\ \text{avec } \forall i \in \{1, \cdots, n\}, \quad y_i(< \mathbf{w}, \mathbf{x}_i > +b) \geq 1 - \xi_i. \end{cases}$$

Remarques :

- La constante δ est maintenant un paramètre de l'algorithme que l'on choisit en pratique par validation croisée ou par approximations. Plus δ est grand et plus cela revient à attribuer une forte importance à l'ajustement. Ce paramètre ajuste le compromis entre bon ajustement et bonne généralisation.

- La différence avec le cas séparable, c'est que les coefficients λ_i sont tous bornés par le paramètre δ.

Au lieu de chercher un hyperplan dans l'espace des entrées, on passe d'abord dans un espace de représentation intermédiaire (feature space) de grande dimension

$$\Phi : \begin{array}{l} \mathbb{R}^d \longrightarrow \mathcal{F} \\ \mathbf{x} \mapsto \Phi(\mathbf{x}). \end{array}$$

La formulation du problème de minimisation ainsi que sa solution

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i^* y_i < \mathbf{x}, \mathbf{x}_i > + b^*$$

ne font intervenir $\mathbf{x}$ et $\mathbf{x}'$ que par l'intermédiaire de produits scalaires $< \mathbf{x}, \mathbf{x}' >$.

Idée : Inutile d'expliciter Φ à condition de savoir exprimer les produits scalaires dans $\mathcal{F}$ à l'aide d'une fonction $k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ symétrique appelée noyau telle que

$$k(\mathbf{x}, \mathbf{x}') = < \Phi(\mathbf{x}), \Phi(\mathbf{x}') > .$$

Une fonction $k(.,.)$ symétrique est un noyau si pour toutes les $\mathbf{x}_i$ possibles, $(k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$ est une matrice définie positive.
Dans ce cas, il existe un espace $\mathcal{F}$ et une fonction Φ tels que

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}).\Phi(\mathbf{y}).$$

Remarques :

- Ces conditions ne donnent pas d'indication pour la construction des noyaux.
- Elles ne permettent pas de savoir comment est Φ.
- En pratique, on combine des noyaux simples pour en obtenir des plus complexes.

Pourquoi le fait d'envoyer les données dans un espace de grande dimension (éventuellement infinie) ne conduirait-il pas à de l'overfitting (sur-apprentissage des données) ?
La dimension ne joue pas de rôle, la seule quantité contrôlant la complexité est la marge.

- Linéaire

$$k(\mathbf{x}, \mathbf{y}) = < \mathbf{x}, \mathbf{y} > .$$

- Polynômial

$$k(\mathbf{x}, \mathbf{y}) = (< \mathbf{x}, \mathbf{y} >)^d \text{ ou } (c + < \mathbf{x}, \mathbf{y} >)^d.$$

- Gaussien

$$k(\mathbf{x}, \mathbf{y}) = e^{-||\mathbf{x}-\mathbf{y}||^2/\sigma^2}.$$

- Laplacien

$$k(\mathbf{x}, \mathbf{y}) = e^{-||\mathbf{x}-\mathbf{y}||_1/\sigma^2}.$$

- La séparation claire en deux composantes (fonctionnelle à optimiser et noyau) permet de séparer les tâches. En pratique, on peut se concentrer sur la construction d'un noyau adéquat et le reste est confié à une "boîte noire".

- La formulation comme problème d'optimisation quadratique sous contraintes linéaires permet l'application d'une batterie d'algorithmes d'optimisation efficaces, permettant de s'attaquer à des problèmes de grande taille (10^6 exemples d'apprentissage).

- On peut encore reformuler le problème sous la forme d'une recherche d'un estimateur non paramétrique basé sur une minimisation du risque empirique pénalisé. Cela permet d'étudier les performances de l'algorithme (bornes de risques, vitesses de convergence...) dans un cadre mathématique bien défini.

Qualités des classifieurs SVM

Interprétabilité : NON !

Critique : possible

Consistance : OUI

Minimax : possible

Parameter-free : NON !

Vitesse : NON

Online : NON

Big Data

Big Data = données massives, grand défi de la décennie 2010-2020 :

- *Big Science :* environnement, physique, génomique, épidémiologie...
- *Technologies de l'information :* réseaux de capteurs, internet...

Ce qui change :

- *Volume :* p grandit avec n
  - régression en grande dimension, sparsité
  - tests multiples (faux positifs), matrices aléatoires
- *Vitesse*
  - parallélisation massive, paradigme "map-reduce"
  - traitement décentralisé avec coût de communication élevé
  - nécessité de spécifier niveau de confiance, diagnostic du modèle
- *Variabilité*
  - dérive temporelle, détection de ruptures, robustesse
  - lois de Pareto (cf Zipf, langages naturels, etc.)
  - problème du transfert, apprentissage supervisé faible
  - aspect séquentiel, boucle fermée

Big Data : perspectives

- Nécessité d'algorithmes statistiques passant à l'échelle
- Nouvelle décomposition du risque :

$$\text{risque} = \text{approximation} + \text{estimation} + \text{optimisation}$$

- Les données comme ressources (et plus comme charge de travail) :
  - Sous-échantillonnage intelligent (exemple : régression sous-échantillonnée en observations et prédicteurs)
  - Optimisation par *gradient stochastique*, apprentissage séquentiel
- Utilisation de modèles *non paramétriques* (bayésiens)

$\Longrightarrow$ Forte imbrication de problématiques mathématiques et informatiques