Apprentissage statistique

Apprentissage supervisé: suite

Cours pour non-spécialistes

Aurélien Garivier

Support Vector Machines

- 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 $Y \in \mathbb{R}$.

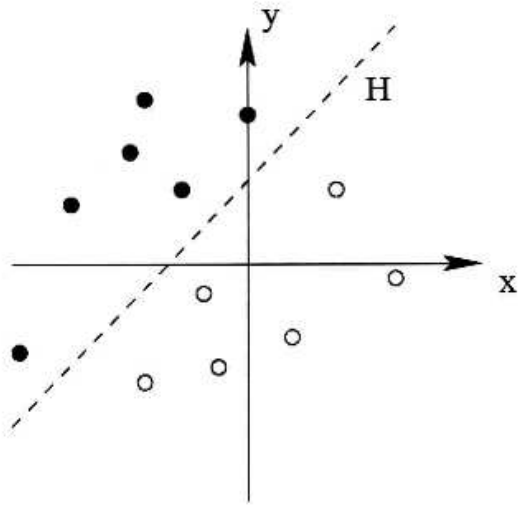
La distribution jointe de $(\mathbf{X}, Y)$, notée $\mathbb{P}$, est inconnue.

SVM est une méthode de classification binaire par apprentissage supervisé.

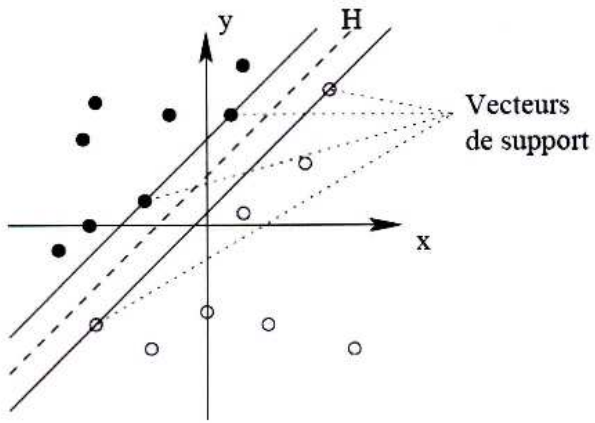
Cette méthode repose sur

- 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.

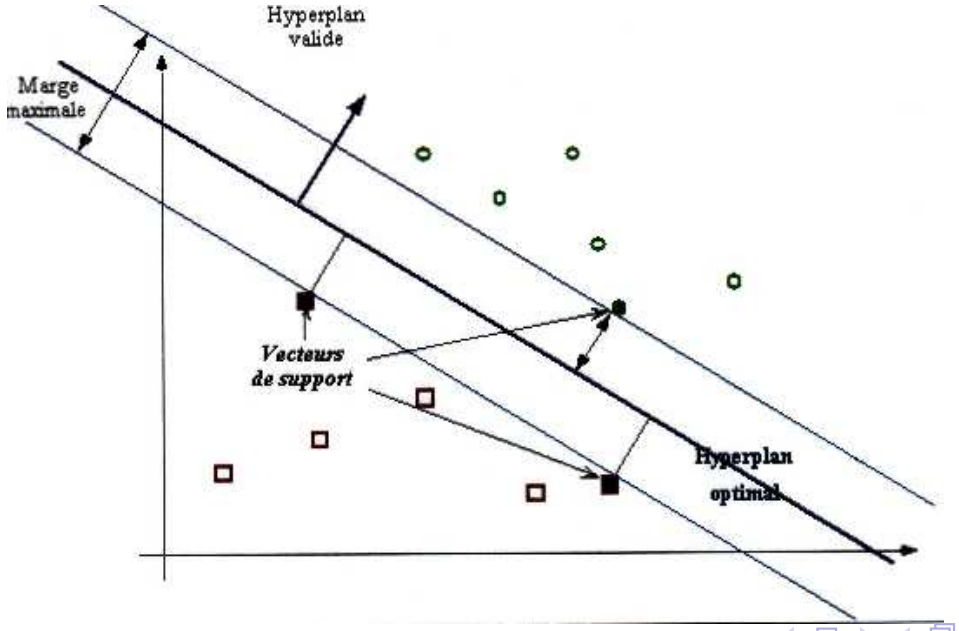


Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support.

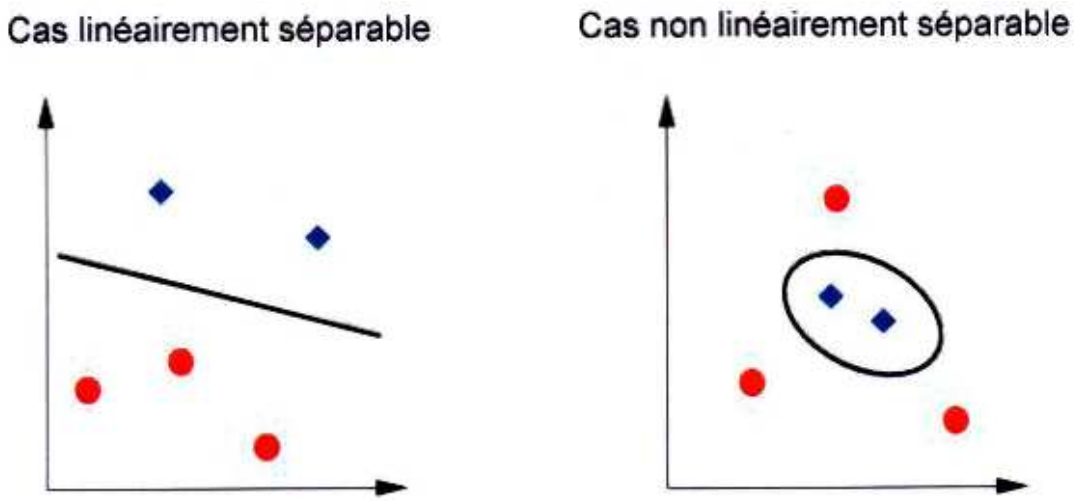


Intuition : Parmi les hyperplans valides, chercher l'hyperplan le "plus sûr".
En supposant qu'un exemple n'ait pas été parfaitement décrit, une petite variation ne modifiera pas sa classification si sa distance à l'hyperplan est grande.

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.



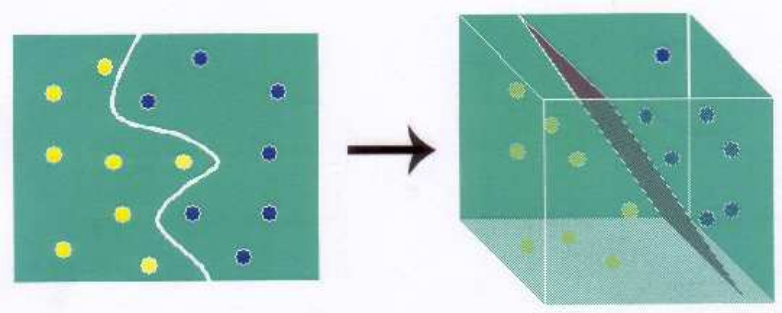
Parmi les modèles des SVMs, on distingue les cas linéairement séparables et les cas non-linéairement séparables. Dans les premiers cas, il est facile de trouver le classifieur linéaire.



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.

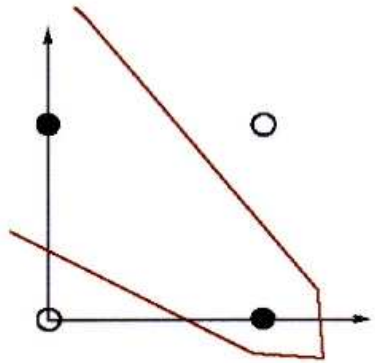


- Transformation d'un problème de séparation non-linéaire dans l'espace de représentation en un problème de séparation linéaire dans un espace de représentation de plus grande dimension.

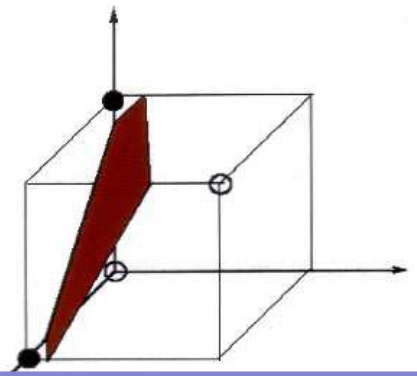
- Cette transformation non-linéaire est réalisée via une fonction noyau. En pratique, quelques familles de fonctions noyau paramétrables sont connues : polynômiale, gaussien, sigmoïde et laplacien. Il revient à l'utilisateur de SVM d'effectuer des tests pour déterminer celle qui convient le mieux pour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



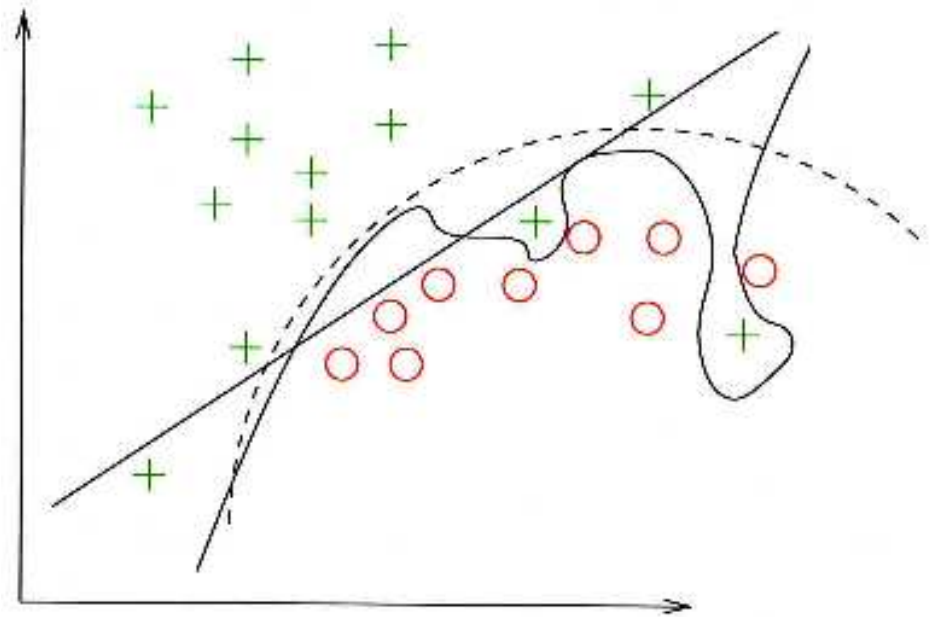
Si on prend une fonction polynômiale $(x,y) \mapsto (x,y,x \times y)$ qui fait passer d'un espace de dimension 2 à un espace de dimension 3, on obtient un problème en dimension 3 linéairement séparable.



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

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

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

Agrégation de modèles

Roadmap

Support Vector Machines

Agrégation de modèles

- Bagging, Boosting, Random Forest

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 $\widehat{\phi}_B(\mathbf{x}_0) = \dfrac{1}{B}\displaystyle\sum_{b=1}^{B} \widehat{\phi}_{\mathbf{z}_b}(\mathbf{x}_0)$ 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