Université Paris IX Dauphine
Crédit Agricole S.A

MASEF

Ensembles d'arbres

Théorie et application au scoring

Étudiant:
Guillaume Ausset

Professeur:
Prof. B. Bouchard

Encadrant:
M. J. Brand

Année 2014-2015

Guillaume Ausset : *Ensembles d'arbres* novembre 2015.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

ALGORITHMES

RÉSUMÉ

Dans ce rapport, différentes méthodes pour combiner des modèles de scoring sont étudiées. Une attention particulière est portée à la combinaison d'arbres aléatoires, mais la plupart des méthodes peuvent être adaptées à toute autre méthode de scoring de base. Les différences entre le vote, le stacking et la combinaison bayésienne sont abordées et les performances des différentes méthodes sont étudiées. Enfin un soin tout particulier est porté aux problèmes liés au scoring et aux déséquilibres des données et nous proposons alors différentes mesures de performances et procédures visant à résoudre les problèmes d'équilibrage. Un aperçu assez complet de la littérature est présenté au cours du mémoire afin de donner des idées d'approfondissement futur.

REMERCIEMENTS

Je tiens à remercier toute l'équipe du GRO pour m'avoir accueilli durant ces 6 mois, notamment Joseph Brand qui a dû relire ce rapport plus de fois qu'il n'est raisonnable, et Stephan Kryzaniak, responsable du GRO, sans qui ce stage n'aurait pas été possible. Un grand merci aussi aux autres stagiaires et à leurs présentations et rapports de stage très instructifs et à tous les encadrants de l'Université Dauphine.

Montrouge, novembre 2015 G.A.

INTRODUCTION

Au moment de prendre la décision d'accorder ou non un crédit à un client, le conseiller doit prendre en compte ce qu'il sait sur le client pour inférer si celui-ci va faire défaut ou non. Il n'est bien sûr pas possible d'effectuer ce choix avec une totale certitude, le conseiller va alors chercher à construire un score représentant d'une façon abstraite la certitude qu'il possède que celui-ci fasse faillite ou non et refusera ou acceptera le crédit au-dessus d'un certain seuil fixé à l'avance. Avec l'augmentation colossale et constante de la quantité de données disponible ainsi que l'évolution des capacités de calculs disponibles est apparue la nécessité de développer de nouvelles méthodes statistiques pour les exploiter. La grande quantité d'observations disponibles permet l'utilisation de techniques d'apprentissage statistique non paramétriques plus coûteuses et moins puissantes que les tests statistiques classiques plus contraignants par leurs hypothèses. Ainsi de nombreux types de classifieurs ont vu le jour et les efforts pour améliorer leur précision est le centre même de cette nouvelle branche des statistiques couramment appelée Machine Learning. Les arbres de décision, sous leurs multiples formes, sont grâce à leur simplicité à la base de nombreux algorithmes, mais leurs performances seules sont assez limitées. Il est alors possible de grouper plusieurs arbres pour obtenir de meilleures performances. Cette technique, qui s'applique à n'importe quel classifieur, fera alors ici l'objet d'un exposé plus détaillé à travers l'étude des principales méthodes ensemblistes. Dans ce mémoire nous exposerons certains exemples de la littérature sur les méthodes ensemblistes pour les arbres puis testerons leurs performances sur deux bases de données réelles représentatives des problématiques du scoring. Nous rappellerons dans un premier temps le cadre théorique ainsi que les différentes techniques classiques qui seront utilisées. Puis, puisque pour juger de l'efficacité de nos méthodes sur les données il convient d'utiliser des métriques de performances qui représentent bien ce que l'on cherche à obtenir, nous verrons alors diverses façons de mesurer la qualité d'un classifieur en fonction des critères retenus. Nous développerons ensuite différentes méthodes de rééquilibrage, soit sur les données elles-mêmes soit par une modification judicieuse de l'algorithme d'apprentissage afin de pallier le problème que présente le déséquilibre des classes dans le cas du scoring. La classe intéressante étant en réalité la classe minoritaire, de telles méthodes sont indispensables. Enfin nous aborderons rapidement la question des données manquantes et comment les incorporer au modèle.

LE PREMIER CHAPITRE fait office de rapide introduction au scoring et traite du choix de la mesure des performances.

LE DEUXIÈME CHAPITRE introduit diverses techniques d'apprentissage statistique populaires.

LE TROISIÈME CHAPITRE explique comment différentes méthodes d'agrégation des classifieurs permettent d'améliorer les résultats.

LE QUATRIÈME CHAPITRE introduit les arbres de classification et applique les méthodes vues dans le chapitre précédent .

LE CINQUIÈME CHAPITRE explore différentes techniques de rééquilibrage des données.

LE SIXIÈME CHAPITRE traite différentes méthodes pour prendre en compte les valeurs manquantes.

1 LE SCORING

1.1 PRINCIPES DU SCORING

Le scoring est intimement lié à la classification supervisée en apprentissage statistique, nous utiliserons alors les notations standards en machine learning dans cette section.

Le scoring cherche à classifier des individus $X_i \in \mathcal{X}$ en fonction de leurs caractéristiques afin de déterminer leur appartenance à une classe $Y \in \mathcal{Y}$. Il existe en effet toujours une incertitude dans la classification des individus et il est nécessaire de mettre en place une règle simple permettant de discriminer les individus selon les impératifs légaux, financiers ou réglementaires. Un classifieur qui ne donnerait que 0 ou 1 dans le cas de la classification binaire est trop contraignant puisqu'il ne permet pas par exemple de mettre en place des règles permettant de cibler plus d'individus 0 quitte à cibler à tort des individus 1 (par exemple dans le cas d'une étude marketing ou le diagnostic d'une maladie grave) selon les impératifs financiers ou moraux.

Afin de pouvoir pallier les problèmes précédents, dans le cas de la classification binaire, au lieu de directement classer les individus nous allons chercher à déterminer une variable latente sur laquelle la classification en elle-même pourra se faire par analyse discriminante.

Une fonction de score est alors seulement un classifieur à valeurs dans $\mathbb{R}$, c'est-à-dire une fonction $S : \mathcal{X} \to \mathbb{R}$. Pour qu'il soit possible de construire une règle de décision sur ce score, il faudrait en théorie avoir la propriété suivante :

$$S(x_1) > S(x_2) \Leftrightarrow \mathbb{P}(y_1 = 1 \mid x_1) > \mathbb{P}(y_2 = 1 \mid x_2) \tag{1.1}$$

On se donne alors comme règle de décision :

$$\begin{cases} y = 1 \text{ si } S(x) \geq s \\ y = 0 \text{ si } S(x) < s \end{cases}$$

ou s est un seuil à fixer selon les impératifs extérieurs.

On dira qu'un score est *bien ordonné* s'il répond au plus près au critère 1.1, c'est le critère qui nous intéresse le plus et qui est le moins contraignant puisque toutes les transformations monotones du score nous laissent indifférents dans ce cas. Dans d'autres cas, un autre critère peut être demandé au score :

$$S(x) = \mathbb{P}(y = 1 \mid x)$$

On dira alors que le score est *bien calibré*.

1.2 PERFORMANCES D'UN SCORE

Il existe un grand nombre de façons de construire des classifieurs et donc des scores. Pouvoir comparer les performances de ces différents classifieurs nécessite alors l'utilisation de mesures de performance adaptées aux spécificités du scoring : le scoring n'étant qu'un cas particulier de la classification binaire il convient de s'intéresser à la mesure de ses performances. Pour cela définissons tout d'abord Vrais Positifs (True Positives) (TP), Vrais Negatifs (True Negatives) (TN) la quantité d'individus correctement classés dans leurs classes respectives et Faux Positifs (False Positives) (FP), False Negatives (False Negatives) (FN) ceux incorrectement classés.

La mesure de performance du score la plus intuitive et la plus naïve est :

$$\text{Err} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}$$

Néanmoins cette mesure présente un énorme défaut dans notre cas puisqu'elle est inadaptée à la classification dans le cas ou les classes ne sont pas présentes de façon équilibrée. En effet si seulement 1% de notre population appartient à la classe positive alors le classifieur attribuant toujours la classe négative à toute observation obtiendra un excellent taux d'erreur de 1% alors que la classe positive, qui de plus est souvent la classe qui nous intéresse et donc celle qu'il est le plus important de reconnaître, aura un taux d'erreur de 100%.

Une correction rapide de ce phénomène est possible en utilisant le taux d'erreur pondéré :

$$\beta \frac{\text{FP}}{\text{TN} + \text{FP}} + (1 - \beta) \frac{\text{FN}}{\text{TP} + \text{FN}}$$

On prend souvent $\beta = 0.5$ ce qui implicitement revient à attribuer les mêmes coûts aux deux classes. Il est également possible d'utiliser la moyenne géométrique pour éviter les problèmes du taux d'erreur classique :

$$\sqrt{\frac{\text{FP}}{\text{TN} + \text{FP}} * \frac{\text{FN}}{\text{TP} + \text{FN}}}$$

Dans le cas du scoring il est plus courant de mesurer la performance plutôt que l'erreur, on introduit alors

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{F} - \text{mesure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Néanmoins aucune de ces mesures ne prend en compte les particularités du scoring c'est-à-dire la nécessité de choisir un seuil sur le score

pour répondre à un impératif industriel de détection d'une population minoritaire. Les outils les plus utilisés et ceux que nous retiendrons sont alors la courbe Receiver Operating Characteristic (ROC) et l'Area under curve (AUC). La courbe ROC est la courbe du taux de vrais positifs $\frac{\text{TP}}{\text{TN+FP}}$ par rapport au taux de faux positifs $\frac{\text{FP}}{\text{TN+FP}}$ et représente le compromis à effectuer lors du choix du seuil de score, il s'agit donc d'une mesure tout à fait adaptée au problème du scoring. La courbe n'est de plus pas sensible au déséquilibre des classes.

Si l'on note t le seuil de décision et T la sortie de notre classifieur, on a $\mathbb{P}(T > t \mid Y = 1) \simeq \frac{\text{TP}(t)}{\text{TP}(t)+\text{FN}(t)}$ et de la même façon $\mathbb{P}(T > t \mid Y = 0) \simeq \frac{\text{FP}(t)}{\text{TN}(t)+\text{FP}(t)}$. La courbe ROC est alors la courbe paramétrée :

$$\text{ROC}(t) = \begin{bmatrix} \mathbb{P}(T > t \mid Y = 0) \\ \mathbb{P}(T > t \mid Y = 1) \end{bmatrix}$$

Sous cette forme l'AUC se calcule comme :

$$\begin{aligned} \text{AUC} &= -\int_0^1 \mathbb{P}(T > t \mid Y = 1)\mathbb{P}'(T > t \mid Y = 0)\,\mathrm{d}t \\ &= \int_0^1 \mathbb{P}(T > t \mid Y = 1)p(t \mid Y = 0)\,\mathrm{d}t \end{aligned}$$

En théorie tous les points de l'enveloppe convexe sont des points ROC *admissibles, en réalité le classifieur dont dispose le statisticien ne peut atteindre qu'un nombre fini de valeurs d'ou la forme en escalier.*

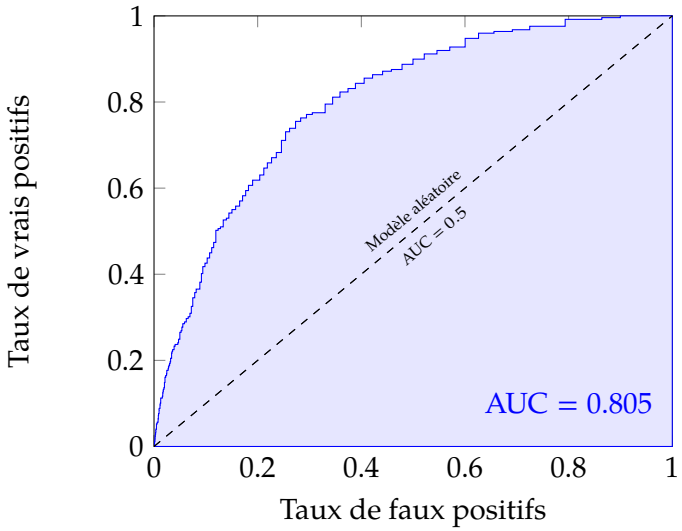

Fig. 1.1 : Courbe ROC et AUC.

On obtient alors une interprétation de l'AUC qui explique tout son intérêt dans le cas du scoring :

$$\begin{aligned} \mathbb{P}(T^i > T^j \mid Y^i = 1, Y^j = 0) &= \iint_{\{(t^i,t^j):t^i>t^j)\}} p(t^i, t^j \mid Y^i = 1, Y^j = 0)\,\mathrm{d}t^i\,\mathrm{d}t^j \\ &= \int_0^1 \left(\int_{t^j}^1 p(t^i \mid Y^i = 1)\,\mathrm{d}t^i \right) p(t^j \mid Y^j = 0)\,\mathrm{d}t^j \\ &= \int_0^1 \mathbb{P}(T > t \mid Y = 1)p(t^j \mid Y^j = 0)\,\mathrm{d}t^j \\ &= \text{AUC} \end{aligned}$$

L'AUC est donc la capacité du classifieur à bien classer les individus, il s'agit exactement de ce que l'on souhaite mesurer pour classer les performances d'un algorithme de scoring. L'AUC permet alors de caractériser à quel point un classifieur est bien ordonné, il reste alors à vérifier si notre fonction score est bien calibrée dans le cas où l'on cherche à prédire des probabilités.

On introduit alors une *règle de scoring*, R, qui va mesurer la calibration en récompensant les prédictions de probabilité qui correspondent à la vraie probabilité. $R(p, y)$ est une fonction de la prédiction et de la réalisation de l'événement telle que le classifieur soit récompensé pour une bonne prédiction. On peut affiner cette définition en définissant une règle de scoring *propre* qui donne la récompense maximale lorsque les probabilités prédites correspondent exactement aux probabilités réelles.

Definition 1 (Régle de scoring propre)**.** Une régle de scoring

$$R : [0,1]^J \times \mathcal{Y} \to \mathbb{R}$$

est dite propre si et seulement si

$$\left(\mathbb{P}(c_1), \dots, \mathbb{P}(c_J)\right) = \arg\max_p \mathbb{E}_Y \left[R(p, y)\right]$$

Parmi les règles de scoring propres les plus utilisées, on peut citer :

RÈGLE LOGARITHMIQUE

$$R(p, y) = \log(p_{i_y})$$

RÈGLE QUADRATIQUE DE BRIER

$$R(p, y) = 2p_{i_y} - \|p\|_2$$

RÈGLE SPHÉRIQUE

$$R(p, y) = \frac{p_{i_y}}{\|p\|_2}$$

où i_y est l'indice de la classe de y.

Le choix du seuil ayant lieu après la sélection du modèle il convient alors de créer une mesure de performance basée sur la courbe ROC qui prend en compte tous les seuils possibles, on utilise alors l'aire sous la courbe ROC ou AUC.

Une fois le seuil choisi l'AUC ou la courbe ROC n'ont plus aucun sens, il faut donc une métrique robuste aux déséquilibres. Dans le cas de la classification binaire il est possible d'utiliser le Coefficient de corrélation de Matthews (MCC) qui est une mesure de la corrélation entre les vraies valeurs et les valeurs prédites.

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}$$

2 APPRENTISSAGE STATISTIQUE

2.1 THÉORIE

2.1.1 L'apprentissage machine

Initialement une branche des statistiques, l'apprentissage statistique s'est rapidement transformé en une discipline à part entière mêlant plusieurs domaines des mathématiques et de l'informatique : l'apprentissage machine.

Le terme *apprentissage statistique* en lui-même est vague et regroupe plusieurs sous-domaines. De façon générale on dispose d'un échantillon $\mathcal{L}$ d'individus possédant des caractéristiques $X_i \in \mathcal{X}$ propres considérées comme déterministes appelées variables et un attribut aléatoire $Y \in \mathcal{Y}$. Si $\mathcal{Y}$ est un ensemble discret on parle de problème de *classification*, s'il est continu on parle alors de problème de *régression*. Il existe un grand nombre d'autres objectifs comme le *clustering*, la *détection de structures* et autres, mais nous ne nous intéresserons ici qu'à ces deux grandes familles en choisissant à chaque fois la tache qui facilite les explications ou est la plus en rapport avec notre problème.

Il est possible de distinguer deux grandes approches pour l'apprentissage machine :

L'ESTIMATION STATISTIQUE est l'approche classique et historiquement la première approche. Ici le but est d'*identifier* la fonction génératrice[1] des données parmi une classe de fonction choisies. Un certain nombre d'hypothèses doit être fait puis à l'aide de techniques d'approximation fonctionnelle le modèle le plus probable est choisi parmi ceux considérés. Il est possible de différencier l'estimation paramétrique ou une classe de fonctions génératrices est fixée et le problème se ramène à l'estimation des paramètres de ces fonctions, et l'estimation non paramétrique ou aucune hypothèse sur la fonction génératrice des données n'est faite. L'estimation non paramétrique paraît a priori plus attirante puisque souvent il est impossible de connaître la famille à laquelle appartient la fonction génératrice, néanmoins les résultats ne sont valides qu'asymptotiquement et un nombre d'observations bien plus important que dans le cas de l'estimation paramétrique est nécessaire.

[1] *Ici on appelle* fonction génératrice *la densité ou probabilité qui donne naissance aux individus, et non la fonction (ou* « système »*) qui lie les caractéristiques d'un individu à son attribut.*

L'APPRENTISSAGE PRÉDICTIF, qui s'occupe d'*imiter* le *système* qui pour chaque individu est capable d'associer l'attribut aux caractéristiques. Le but est ici la création d'un modèle non pas « vrai » mais

qui se *généralise*, c'est-à-dire à même d'obtenir les mêmes performances sur de nouveaux individus non observés que sur l'échantillon $\mathcal{L}$.

Le problème d'apprentissage prédictif est plus simple que le problème classique d'estimation statistique puisque l'on ne cherche pas à trouver le vrai modèle, mais seulement une assez bonne imitation. Les deux approches ne cherchent pas à répondre à la même question et ne peuvent donc pas être comparées directement. Résoudre le problème d'estimation statistique permet de comprendre toutes les caractéristiques du système et les étudier, ce que le problème d'apprentissage statistique n'a pas vocation à faire puisqu'il ne cherche qu'à répliquer le pouvoir prédictif du système. Dans le cas de la classification ou de la régression nous ne sommes intéressés que par la capacité de notre *classifieur* ou *régresseur* à obtenir de bonnes performances sur de nouveaux individus, le concept de *bonnes performances* étant à définir en fonction du problème. Il s'agit donc de l'approche que nous adopterons.

PLUS D'INFORMATIONS

Pour plus de littérature sur l'apprentissage statistique il est possible de se référer à Trevor HASTIE et al. (2008) ou BISHOP (2006) pour un aperçu des différentes méthodes d'apprentissage machine. Un traitement purement statistique de méthodes d'apprentissage ainsi qu'un traitement des techniques nécessaires à leur compréhension est disponible dans WASSERMAN (2004), DEVROYE et al. (1997) traitent de manière rigoureuses les différentes considérations mathématiques de l'apprentissage statistique. Si au contraire le lecteur cherche un aperçu plus général combinant à la fois estimation statistique et apprentissage prédictif en expliquant les motivations sans trop de lourdeur mathématique il est possible de se référer à CHERKASSKY et MULIER (2007). Enfin un traitement rigoureux du cas de l'estimation paramétrique et des problèmes de convergences de telles méthodes est disponible dans TSYBAKOV et ZAIATS (2009).

2.1.2 Minimisation du risque empirique

Supposons que l'on possède un échantillon d'apprentissage $\mathcal{L}$ composé de N individus $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. Il est possible d'essayer de modéliser la loi jointe de $(\mathbf{x}, y)$ mais c'est un problème trop complexe et en réalité pas nécessaire à notre but, il suffit en effet de modéliser le lien entre $\mathbf{x}$ et y, c'est-à-dire considérer le problème comme l'apprentissage d'une fonction f inconnue telle que $y = f(\mathbf{x}) + \varepsilon$ où ε est une *erreur*. Il s'agit d'une forme plus contraignante, mais en réalité plus adaptée à notre problème et beaucoup plus simple à étudier. Puisqu'il est souvent impossible de connaître la forme de la loi de X ou ε, nous

allons effectuer comme unique hypothèse que les variables sont indépendantes et de même loi.

Le problème est alors maintenant un problème d'approximation de f par une famille de fonctions et un algorithme de notre choix. Nous appellerons $\varphi_{\mathcal{L}}$ l'approximation que nous obtenons de cette façon, celle-ci dépend évidemment non seulement de $\mathcal{L}$ mais de l'algorithme d'apprentissage utilisé. Puisqu'il s'agit d'un problème d'approximation fonctionnelle le cadre naturel est de se donner une distance à minimiser, mais puisque nous ne connaissons pas f mais seulement les réalisations de f à certains points il nous faut remplacer la distance entre les fonctions par une forme plus faible : la *perte* L aux points de la fonction. Ainsi la perte au point $(\mathbf{x}_i, y_i)$ est :

$$L(\varphi_{\mathcal{L}}(y_i, \mathbf{x}_i))$$

Enfin puisque nous disposons d'un échantillon d'apprentissage et non pas seulement d'un individu unique il semble naturel de chercher à minimiser la perte moyenne

$$\mathrm{Err}\,(\varphi) = \mathbb{E}_{X,Y}\left[L(\varphi_{\mathcal{L}}(Y, X))\right]$$

Ainsi si un certain algorithme d'apprentissage est capable de modéliser une classe Φ de fonctions, notre problème est le choix d'un $\varphi^{\star}_{\mathcal{L}}$ tel que :

$$\begin{aligned}\varphi^{\star} &= \underset{\varphi_{\mathcal{L}}\in\Phi}{\arg\min}\, \mathbb{E}_{X,Y}\left[L\left(Y, \varphi_{\mathcal{L}}\left(X\right)\right)\right] \\ &\simeq \underset{\varphi_{\mathcal{L}}\in\Phi}{\arg\min}\, \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \varphi_{\mathcal{L}}\left(\mathbf{x}_i\right)\right)\end{aligned}$$

Notons alors

$$\begin{aligned}\mathrm{Err}_N\left(\varphi_{\mathcal{L}}\right) &= \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \varphi_{\mathcal{L}}\left(\mathbf{x}_i\right)\right) \\ \varphi^{\star}_{\mathcal{L}} &= \underset{\varphi\in\Phi}{\arg\min}\, \mathrm{Err}_N\left(\varphi\right)\end{aligned}$$

La question naturelle est alors de se demander si les performances, c'est-à-dire la perte minimale obtenue par notre algorithme d'optimisation sur l'échantillon d'apprentissage, se généralisent sur la population totale. Il est en réalité possible de fournir des garanties pour qu'optimiser en moyenne empirique sur l'échantillon, c'est-à-dire travailler par Empirical Risk Minimization (ERM), revienne à une erreur prés à optimiser en moyenne. On veut alors prouver que notre fonction se *généralise* bien, c'est-à-dire soit assez expressive pour imiter la fonction f mais néanmoins assez restreinte pour ne pas donner un φ qui colle parfaitement à l'échantillon d'apprentissage au détriment d'un pouvoir de généralisation nul.

En effet il est facile de voir que si Φ est l'ensemble de toutes les fonctions alors la perte empirique minimale sera toujours 0 sans aucune

garantie sur la perte réelle. Au contraire si l'on prend à l'inverse le cas très restreint de $\Phi = \{\text{fonctions constantes}\}$ alors les deux pertes seront relativement proches. Il faut alors introduire une notion de classes de complexité des fonctions.

C'est dans cette optique que V. Vapnik et A. Chervonenkis ont introduit la notion de dimension de Vapnik-Chervonenkis (VC-dimension) (Vapnik, 1998, 2000). Commençons par définir la notion de *consistance* :

Definition 2. La procédure ERM est consistante si :

$$\begin{aligned} \text{Err}\left(\varphi_{\mathcal{L}}^{\star}\right) &\xrightarrow[|\mathcal{L}|\to+\infty]{\mathbb{P}} \text{Err}(\varphi^{\star}) \\ \text{Err}_N\left(\varphi_{\mathcal{L}}^{\star}\right) &\xrightarrow[|\mathcal{L}|\to+\infty]{\mathbb{P}} \text{Err}(\varphi^{\star}) \end{aligned}$$

Vapnik et Chervonenkis ont alors montré le théorème suivant :

Théorème 2.1 (Théorème fondamental de l'apprentissage)**.** *Pour une fonction de perte L bornée, la méthode ERM est consistante si et seulement si le risque empirique converge uniformément vers le vrai risque, c'est-à-dire :*

$$\lim_{|\mathcal{L}|\to+\infty} \mathbb{P}\left(\sup_{\varphi\in\Phi} |\text{Err}(\varphi_{\mathcal{L}}) - \text{Err}_N(\varphi_{\mathcal{L}})| > \varepsilon\right) = 0 \tag{2.1}$$

On peut remarquer que 2.1 dépend de la classe de fonction choisie et donc que le supremum de l'erreur dépend de la richesse de la classe de fonction.

Notons alors $N(\Phi, \mathcal{L})$ le nombre de dichotomies que peuvent engendrer les fonctions de Φ, aussi appelé *diversité*. On peut alors définir l'*entropie aléatoire* :

$$H(\Phi, \mathcal{L}) = \ln N(\Phi, \mathcal{L})$$

et l'entropie de Vapnik-Chervonenkis

$$H(\Phi, N) = \mathbb{E}\left[\ln N(\Phi, \mathcal{L})\right]$$

Cette quantité dépend de la distribution de $\mathcal{L}$, on introduit alors la *fonction de croissance* G :

$$G(\Phi, N) = \ln \max_{\mathcal{L}} N(\Phi, \mathcal{L})$$

ainsi que l'entropie de Vapnik-Chervonenkis recuite :

$$H_{\text{recuite}}(\Phi, N) = \ln \mathbb{E}\left[N(\Phi, \mathcal{L})\right]$$

Vapnik et Chervonenkis ont alors montré le résultat suivant :

Proposition 2.2. *La procédure ERM est consistante si et seulement si*

$$\lim_{N\to+\infty} \frac{H(\Phi, N)}{N} = 0$$

Cette propriété n'est en général d'aucune utilité puisqu'elle dépend de la distribution de $\mathcal{L}$ et ne donne aucune garantie sur la vitesse de convergence. On va alors chercher à trouver des conditions pour la convergence rapide :

Definition 3. On dit qu'il y a *convergence rapide* si il existe un $c > 0$ tel que $\forall n > n_0$:

$$\mathbb{P}\left(\sup_{\varphi\in\Phi} |\mathrm{Err}(\varphi_{\mathcal{L}}) - \mathrm{Err}_N(\varphi_{\mathcal{L}})| > \varepsilon\right) \leq e^{-cn\varepsilon^2}$$

On a alors que

$$\lim_{N\to+\infty} \frac{H_{\text{recuite}}(\Phi, N)}{N} = 0$$

est une condition suffisante pour la convergence rapide. Finalement :

Théorème 2.3. *La procédure ERM est consistante et converge rapidement si et seulement si*

$$\lim_{N\to+\infty} \frac{G(\Phi, N)}{N} = 0$$

2.1.3 Dimension de Vapnik-Chervonenkis

Il est en fait possible d'obtenir des bornes de convergence explicite, pour cela il faut définir la VC-dimension.

Definition 4 (Dimension de Vapnik-Chervonenkis)**.** On dit qu'une famille de fonctions indicatrices *pulvérise* $\mathcal{L}$ si toutes les dichotomies de $\mathcal{L}$ sont réalisables par cette famille. C'est à dire dans le cas de la classification binaire que toutes les partitions en 2 sous-ensembles de points sont possibles. On dit qu'une famille d'indicatrices est de VC-dimension h s'il existe un $\mathcal{L}, |\mathcal{L}| = h$ qui puisse être pulvérisé, mais pas de $\mathcal{L}, |\mathcal{L}| = h + 1$. La VC-dimension est donc le nombre maximal de dichotomies que la famille d'indicatrices puisse réaliser. Soit une famille de fonctions f à valeurs réelles indexée par ω. La VC-dimension de (f_ω) est définie comme la VC-dimension de la famille $\left(\mathbb{1}_{f_\omega(\cdot)-\beta>0}\right)_{\omega,\beta}$ indexée par ω et β.

Vapnik et Chervonenkis ont montré que :

$$G(\Phi, N) \leq h\left(1 + \ln\left(\frac{N}{h}\right)\right)$$

Le résultat le plus important, celui qui justifie de se ramener à un problème d'optimisation de la perte empirique en ignorant les habituelles questions statistiques.

Théorème 2.4. *Dans le cas de la classification binaire et avec la perte* $0/1$ *on a :*

$$\mathbb{P}\left(\sup_{\varphi\in\Phi} |\mathrm{Err}_N(\varphi) - \mathrm{Err}(\varphi)| > \varepsilon\right) \leq 8G(\Phi, N)e^{-N\varepsilon^2/32} \tag{2.2}$$

$$\mathbb{E}\left[\sup_{\varphi\in\Phi} |\mathrm{Err}_N(\varphi) - \mathrm{Err}(\varphi)|\right] \leq 2\sqrt{\frac{\log G(\Phi, N) + \log 2}{N}} \tag{2.3}$$

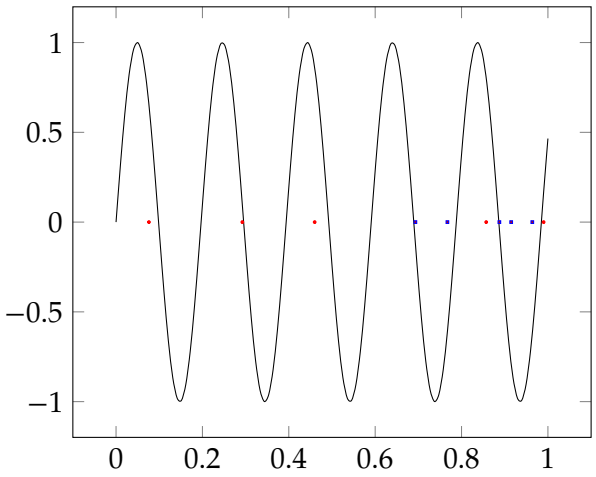

FIG. 2.1: $\sin(\omega x)$ possède un unique paramètre et une dimension VC infinie.

Idée de preuve. La preuve utilise la plupart des techniques courantes pour les preuves de la théorie de Vapnik-Chervonenkis et est donc adaptée et reproduite de DEVROYE et al. (1997) en annexe B. Celle-ci procède en 4 étapes types :

1. Symétrisation par échantillon fantôme.
2. Symétrisation par processus de Rademacher.
3. Conditionnement sur $\mathcal{L}$.
4. Utilisation d'une inégalité de concentration.

□

2.2 SÉLECTION DU MODÈLE

Le choix du modèle à utiliser est un problème complexe qui dépend grandement des données en question, il n'est pas possible de trouver un modèle qui assure les meilleures performances dans tous les cas, il n'y a pas de *Free Lunch* possible (WOLPERT, 1996). Il faut donc bien comprendre comment chaque algorithme obtient de bonnes performances sur un certain jeu de données et de mauvaises sur un autre, mais surtout comprendre quels sont les éléments généraux qui gouvernent la performance des algorithmes.

2.2.1 Dilemme Biais-Variance

Le choix d'une famille de modèles ou même des hyper paramètres au sein de cette famille comme le coefficient de pénalisation, le nombre de fonctions de la base ou la bande passante d'un noyau, influent sur la complexité du modèle, où la complexité réfère ici à l'étendue des

fonctions que le modèle peut représenter. Il est tentant d'augmenter la complexité d'un modèle puisqu'en pouvant représenter plus de fonctions différentes on augmente les chances de bien approcher la fonction sous-jacente, mais cela n'est pas nécessairement judicieux.

Tous les algorithmes d'apprentissage statistique cherchent d'une façon ou d'une autre à minimiser une fonction de perte sur les données d'apprentissage ; le choix d'un modèle complexe et donc d'une large famille d'estimateurs permet alors mécaniquement de réduire l'erreur d'apprentissage, mais ne garantit pas forcément une réduction de l'erreur de généralisation. En effet il est possible de *sur apprendre* les données d'apprentissage (*overfitting*) en construisant un modèle qui s'adapte aux caractéristiques uniques de l'échantillon d'apprentissage et ne se généralise pas du tout à des données nouvelles. Il faut donc trouver un compromis entre sur et sous-apprentissages (over/under-fitting) afin d'obtenir de bonnes performances de généralisation.

Les concepts de sous et sur apprentissage sont intimement liés à la décomposition biais-variance de l'erreur de généralisation. Ainsi pour la perte quadratique on a :

$$\mathrm{Err}\left(\varphi_{\mathcal{L}}(x_0)\right) = \mathrm{Var}[y_0 \mid x_0] + \mathrm{Var}_{\mathcal{L}}[\hat{y}_0] + \left(f(x_0) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}(x_0)\right]\right)^2$$

avec $y_0 = f(x_0) + \varepsilon$ et $\varphi_{\mathcal{L}}$ le classifieur appris sur l'échantillon d'apprentissage.

Prenons par exemple comme classifieur les k-plus proches voisins, on a alors (Trevor Hastie et al., 2008) :

$$\mathrm{Err}\left(\varphi_{\mathcal{L},k}(x_0)\right) = \sigma^2 + \left[f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_{(l)})\right]^2 + \frac{\sigma^2}{k}$$

Dans le cas où la taille de l'échantillon d'apprentissage tend vers $+\infty$ et donc la distance entre x_0 et $x_{(1)}$ tend vers 0 on peut voir que le terme $\left[f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_{(l)})\right]^2$ est minimal pour $k = 1$, puis croit vers $\left[f(x_0) - f(\hat{x})\right]^2$ alors que $\frac{\sigma^2}{k}$ tend vers 0 pour $k \to \infty$. On s'aperçoit ici de la nécessité de choisir une valeur raisonnable de k qui minimise la somme des deux. En effet lors de l'apprentissage l'erreur mesurée est l'erreur sur le premier terme seulement et tend vers 0 si k est choisi trop petit, il faut donc minimiser une approximation de l'erreur de généralisation. Parmi de telles approximations, les plus courantes sont :

ERREUR LEAVE-ONE-OUT (LOO) Ici le classifieur est construit sur tous les individus sauf l'individu i puis l'erreur LOO_i est mesurée sur cet individu. L'approximation de l'erreur est alors $\mathrm{LOO} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{LOO}_i$

ERREUR k-FOLD CROSS VALIDATION (k-CV) L'erreur LOO étant incroyablement coûteuse en terme de calcul il est possible d'utiliser à la place la validation croisée : l'échantillon d'apprentissage est découpé en k strates $\mathcal{L}_1, \ldots, \mathcal{L}_k$ de tailles environ égales, le classifieur

est construit sur $\mathcal{L}_{-i} = \mathcal{L} \setminus \mathcal{L}_i$ et l'erreur CV_i mesurée sur $\mathcal{L}_i$. L'erreur k-CV est alors $\mathrm{CV} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{CV}_i$.

ERREUR k-CV RÉPÉTÉE Dans ce cas, le processus de validation croisée est moyenné sur V répétitions.

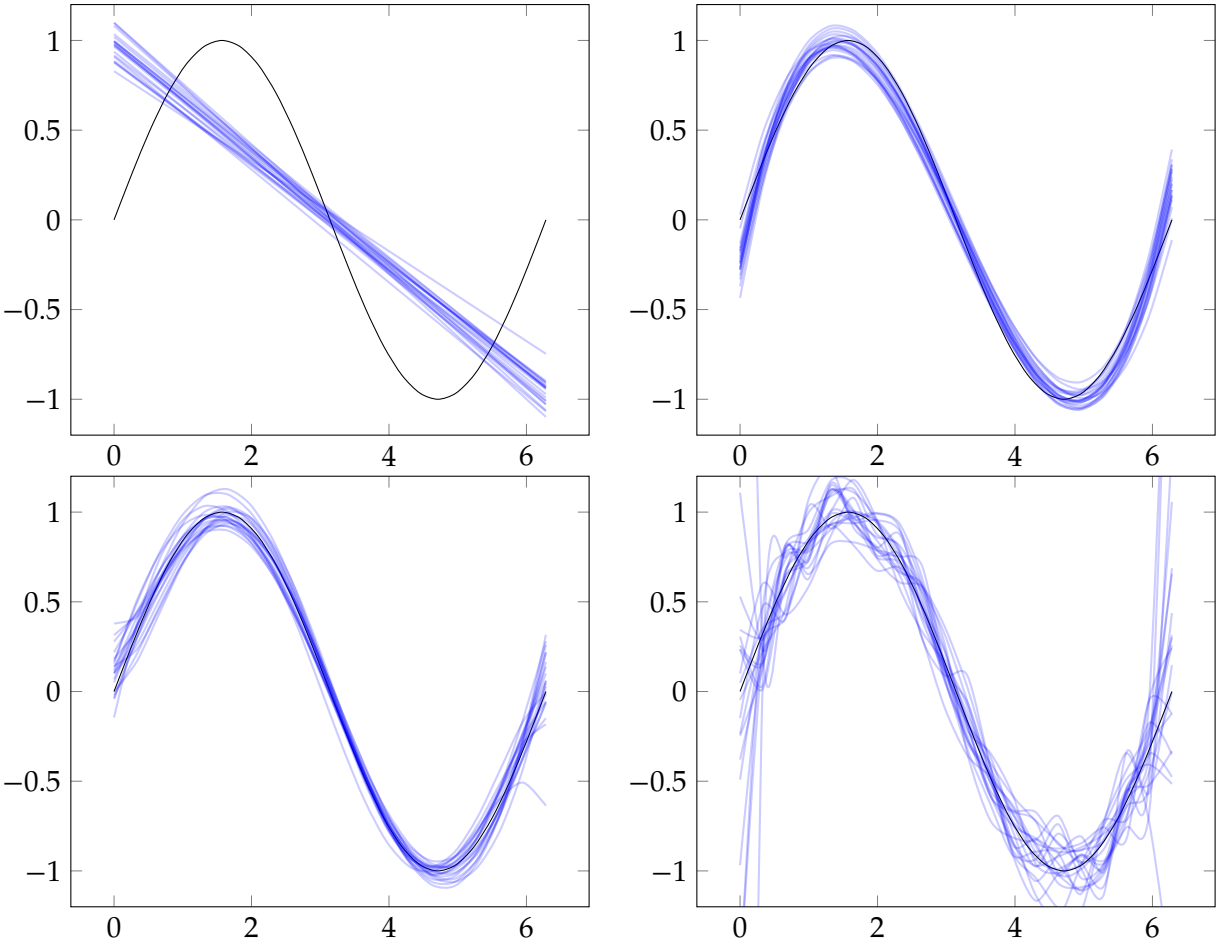

FIG. 2.2: Apprentissage de $\sin(x)$ sur $[0, 2\pi]$ à l'aide de polynômes de degrés 1,3,5 et 20.

2.2.2 Fléau de la dimension

Un grand nombre d'algorithmes d'apprentissage repose sur l'utilisation d'un voisinage du point à estimer. Ces techniques, bien qu'intuitives et efficaces, voient leurs performances chuter de façon dramatique avec la dimension de $\mathcal{X}$, l'espace ambiant. En effet, supposons par exemple que les individus soient uniformément distribués sur $[0, 1]^p$, alors pour créer un voisinage (au sens L^1 par exemple) d'une observation contenant une fraction r des observations totales il faut en moyenne couvrir sur chaque dimension $r^{1/p}$. On voit ainsi dans 2.3 que très rapidement lorsque p augmente il devient nécessaire de prendre en compte la majorité des réalisations de chaque variable.

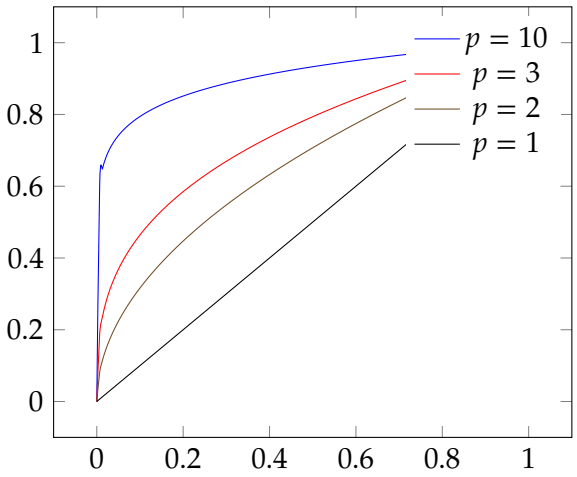

FIG. 2.3: Évolution de la longueur moyenne des voisinages dans chaque dimension

Il est, en partie également possible d'avoir l'intuition du dilemme biais-variance dans la formulation du théorème 2.4 qu'une augmentation de la complexité du modèle résulte en une réduction de l'erreur empirique (biais) mais une augmentation de la borne de généralisation (assimilable à la variance). Il faut donc fixer un modèle de VC-dimension la plus faible possible à erreur fixée, ce qui est le principe de Structural Risk Minimization (SRM).

Qui plus est la densité de l'échantillon est proportionnelle à $N^{1/p}$. Un échantillon dense en dimension 1 pourra alors être épars en dimension 10.

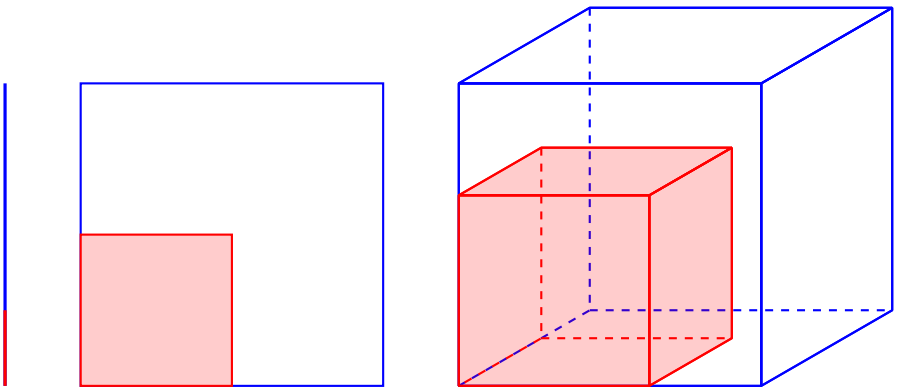

FIG. 2.4 : Un voisinage de 1/4 de l'espace en dimensions 1,2 et 3.

Il est possible de se rendre compte de l'impact de la dimension sur différentes méthodes d'apprentissage classique (Trevor HASTIE et al., 2008). Ainsi dans le cas des plus proches voisins, si l'on fait l'hypothèse que les observations sont distribuées uniformément sur $[0,1]^p$, la distance médiane du plus proche voisin de l'origine est de $\left(1 - \left(\frac{1}{2}\right)^{1/N}\right)^{1/p}$. En grande dimension, les observations sont alors plus proches de la frontière que de toute autre observation.

Même un algorithme aussi simple que la régression linéaire n'est pas immunisée contre le fléau de la dimension (Trevor Hastie et al., 2008), supposons par exemple que les données suivent un vrai modèle linéaire c'est-à-dire que :

$$Y = X\beta + \varepsilon$$

avec $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$. On a alors $\hat{y}_0 = x\hat{\beta}$ et donc $\hat{y}_0 = x_0\beta + x_0(X^\top X)^{-1}X^\top\varepsilon$ on obtient la décomposition biais-variance (voir 3.1) on a alors

$$\mathrm{Err}(x_0) = \sigma^2 + \mathbb{E}_{\mathcal{L}}\left[x_0^\top \left(X^\top X\right)^{-1} x_0\right]\sigma^2 + 0^2$$

Donc sous l'hypothèse $\mathbb{E}[X] = 0$ on a que $X^\top X \sim N\,\mathrm{Cov}(X)$ pour N grand, alors :

$$\begin{aligned}
\mathbb{E}_{x_0}\left[\mathrm{Err}(x_0)\right] &\sim \mathbb{E}_{x_0}\left[\frac{x_0^\top \mathrm{Cov}(X)^{-1}x_0}{N} + \sigma^2\right] \\
&= \mathbb{E}_{x_0}\left[\frac{\sigma^2}{N}\sum_i\sum_j \left[\mathrm{Cov}(x)^{-1}\right]_{ij}[x_0]_i[x_0]_j + \sigma^2\right] \\
&= \sigma^2\left(\frac{\mathrm{trace}\left(\mathrm{Cov}(x)^{-1}\mathrm{Cov}(x_0)\right)}{N} + 1\right) \\
&= \sigma^2\left(\frac{\mathrm{trace}\left(\mathbb{I}_p\right)}{N} + 1\right) \\
&= \sigma^2\left(\frac{p}{N} + 1\right)
\end{aligned}$$

On retrouve alors ce que nous avons observé plus tôt : plus la dimension est grande plus le nombre d'observations nécessaire est important.

Nous verrons néanmoins que certains modèles résistent particulièrement bien aux hautes dimensions, notamment les forêts aléatoires dont la vitesse de convergence ne dépend que du nombre de variables informatives (Biau, 2010 ; Breiman, 2004).

2.3 MODÈLES LINÉAIRES ET LOGISTIQUES

Bien que nous nous intéressions à la classification, il est plus simple d'étudier la régression linéaire puis de se ramener à ce cas. Bien que très simples, les méthodes linéaires possèdent de nombreuses généralisations qui étendent leur pouvoir de prédiction.

Nous cherchons alors $y = f(x)$ sous une forme bien particulière :

$$f(x) = \beta_0 + \sum_{k=1}^{p} x_k\beta_k$$

Le problème est alors une estimation des paramètres β_k optimaux. Dans le cas de la perte L^2, qui est le cas le plus courant de par ses caractéristiques théoriques intéressantes et sa facilité de calcul on peut

trouver directement l'estimateur. En effet, en écrivant le problème sous forme matricielle :

$$\min_{\beta}(y - x\beta)^\top (y - x\beta)$$

on trouve comme solution

$$\hat{\beta} = (x^\top x)^{-1} x^\top y$$

dont on peut donner[2] la variance si $y = x\beta + \varepsilon$ avec $\mathbb{V}[\varepsilon] = \sigma^2 \mathbb{I}$:

$$\mathrm{Var}(\hat{\beta}) = (x^\top x)^{-1}\sigma^2$$
$$\hat{\sigma}^2 = \frac{1}{N - p - 1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

[2] *Le calcul ne présente aucune difficulté mais est donné en annexe.*

Le choix de l'estimateur linéaire obtenu par régression L^2 est dû au théorème de Gauss-Markov :

Definition 5. Supposons que le modèle réel est :

$$y = x\beta + \varepsilon$$
$$\mathbb{E}(\varepsilon_i) = 0, \forall i$$
$$\mathrm{Var}(\varepsilon) = \sigma^2 \mathbb{I}, \sigma^2 < +\infty$$

Si θ est un estimateur linéaire de β, c'est-à-dire $\theta = Hy$ et non biaisé c'est-à-dire $\mathbb{E}[\theta] = \beta$ alors nécessairement :

$$\mathrm{Var}[\theta] \geq \mathrm{Var}[\hat{\beta}]$$

où $\hat{\beta}$ est l'estimateur des moindres carrés.

Comme vu précédemment la décomposition biais-variance permet de décomposer l'erreur :

$$\mathrm{Err}_{L^2}(\theta) = \mathrm{Var}[\theta] + [\mathbb{E}[\theta] - \beta]^2$$

donc le théorème de Gauss-Markov implique que parmi les estimateurs linéaires non biaisés, celui des moindres carrés possède la perte quadratique la plus faible. Il n'est pourtant pas toujours judicieux de ne considérer que des estimateurs non biaisés, en effet le dilemme biais-variance s'applique toujours et nous verrons qu'il existe des estimateurs biaisés qui présentent une plus faible variance et peuvent donc éventuellement obtenir une perte totale plus faible.

Restriction à un sous-ensemble de variables

Une méthode courante visant à diminuer la complexité du modèle et améliorer son pouvoir de généralisation est la restriction à un sous-ensemble bien choisi de variables sur lesquelles effectuer la régression. La sélection d'un sous-ensemble optimal de variables de taille donnée

étant un problème NP-difficile il convient de mettre en place des stratégies heuristiques.

La méthode la plus couramment utilisée est la sélection de variables par étape *forward* ou *backward*. Dans la sélection *forward*, l'algorithme commence par estimer l'origine β_0 puis, à chaque étape, ajoute la variable qui maximise la réduction de l'erreur. La procédure s'arrête une fois le nombre de variables souhaitées atteint. La sélection *backward* procède de la même façon, mais en retirant la variable contribuant le moins à l'ensemble en partant de l'ensemble total des variables.

Il existe également une version plus générale de l'algorithme sélection des variables : Least Angle Regression (LARS) (Efron et al., 2003) qui peut être utilisée non seulement pour reconstruire l'algorithme *forward step-wise* mais également le chemin complet du *Lasso*.

Restriction par pénalisation

Une autre approche pour contrôler la complexité du modèle est d'ajouter une pénalisation sur la norme du vecteur des coefficients, et donc une restriction sur les valeurs admissibles des coefficients.

Dans le cas général, cela revient à résoudre le problème suivant

$$\arg\min_{\beta} \left\{ (Y - X\beta)^\top (Y - X\beta) + \lambda \|\beta\|_q \right\}$$

pour une norme $q \geq 1$. Dans le cas $q = 2$ on obtient la pénalisation d'arête (*ridge*) ou régularisation de Tychonoff qui possède une solution explicite et est donc facile à déterminer. Dans le cas $q = 1$ on obtient la régression *LASSO*, il s'agit d'un problème de minimisation quadratique et peut donc être également aisément calculé. Le *LASSO* présente une particularité intéressante : les coefficients seront souvent fixés à une valeur exactement égale à 0, et cette régularisation en plus d'améliorer le pouvoir de généralisation de la régression effectue donc une forme de sélection des variables automatique. Ce phénomène peut se comprendre en examinant la pénalisation sous un point de vu Bayesien (Park et Casella, 2008), on peut rapidement le comprendre en réécrivant le problème comme :

$$\arg\max_{\beta} - \left\{ (Y - X\beta)^\top (Y - X\beta) - \lambda \|\beta\|_q \right\}$$

Or on rappelle que l'on a dans le cadre Bayesien :

$$\log(\text{a posteriori}) \sim \log(\text{vraisemblance}) + \log(\text{a priori})$$

On cherche donc le mode de l'a posteriori. Dans le cas d'une erreur Gaussienne on peut donc identifier les moindres carrés à la vraisemblance et la pénalisation à l'a priori d'une certaine loi de la famille exponentielle. Lorsque $q = 1$ on obtient un a priori de Laplace 2.5.

Bien que les $q \in]1, 2[$ soient intéressants, car correspondant à des a priori bayesiens différents, la résolution du problème est bien trop

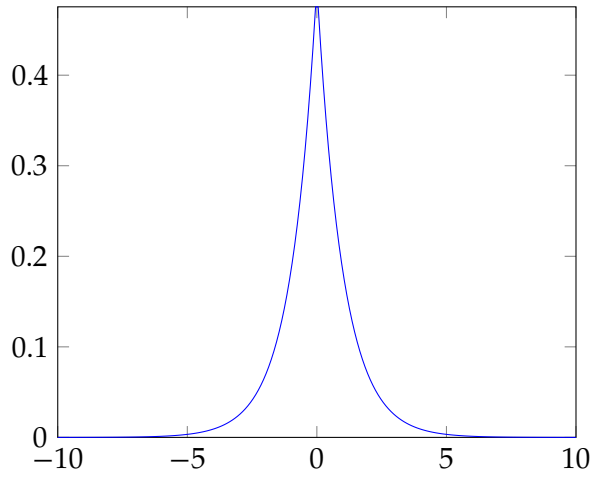

FIG. 2.5 : Densité de la loi de Laplace

difficile, il est alors souvent choisi comme compromis la pénalisation *Elastic-Net* :

$$\arg\min_{\beta} \left\{ (Y - X\beta)^\top (Y - X\beta) + \lambda(\alpha\|\beta\|_2 + (1-\alpha)\|\beta\|_1) \right\}$$

Régression logistique

Il est possible d'étendre les modèles linéaires par une transformation non linéaire de l'objectif. Ainsi le nouveau modèle est

$$y = g^{-1}(x\beta)$$

où g est appelée la fonction de lien. Dans le cas de la classification binaire, on considère les réalisations y comme une variable de Bernouilli dont on veut estimer la probabilité de réussite. On choisit comme fonction de lien

$$g(y) = \log\left(\frac{y}{1-y}\right)$$

En effet cette fonction, appelée *logit*, mesure les *odds-ratio* de la variable de Bernouilli et fournit donc une interprétation par analyse discriminante évidente : lorsque l'*odds-ratio* est supérieur à 1 on associe un succès et un échec dans le cas inférieur à 1.

Il n'existe pas dans ce cas de formule analytique comme dans le cas linéaire simple, l'estimation de β est alors effectué par maximum de vraisemblance et nécessite donc des calculs de minimisation par méthode de Newton-Rhapson par exemple, qui sont bien plus coûteux.

2.4 MACHINES À VECTEURS DE SUPPORT

Dans le cas de la discrimination, on a jusqu'à présent cherché à chaque fois à se ramener à un problème de séparation des données par un hyperplan d'une façon ou d'une autre. Il paraît donc légitime d'essayer

de poser le problème sous la forme de la recherche d'un hyperplan de séparation *optimal* en un certain sens à définir.

2.4.1 Séparation par un hyperplan optimal

Cas séparable

On cherche à minimiser l'erreur de classification, c'est-à-dire la probabilité $\mathbb{P}(Y \neq \varphi(X)|\mathcal{L})$ de mal classifier un nouveau vecteur inconnu étant donné un ensemble de vecteurs d'apprentissage et une fonction de classification. La théorie de Vapnik-Chervonenkis donne un majorant de cette erreur qui dépend de la complexité de la classe de fonction de séparation choisie : la dimension de Vapnik-Chervonenkis. Vapnik a montré que l'on pouvait minimiser la dimension VC en maximisant la **marge** de l'hyperplan de séparation (Cortes et Vapnik, 1995), c'est-à-dire la distance maximale entre les deux hyperplans limite de décision, dans le cas particulier des Machine à vecteurs de support (SVM). La distance entre un point x et l'hyperplan (w, b) est $\frac{\|w \cdot x + b\|}{\|w\|}$. On choisi la représentation dite canonique de l'hyperplan pour laquelle le discriminant, $f(x) = w^\top x + b = 0$, est égal à 1 en valeur absolue sur les vecteurs dits *supports* les plus proches pour l'échantillon d'apprentissage. Donc la distance au point le plus proche devient :

$$\frac{\|w \cdot x + b\|}{\|w\|} = \frac{1}{\|w\|}$$

Et la marge : $\frac{2}{\|w\|}$

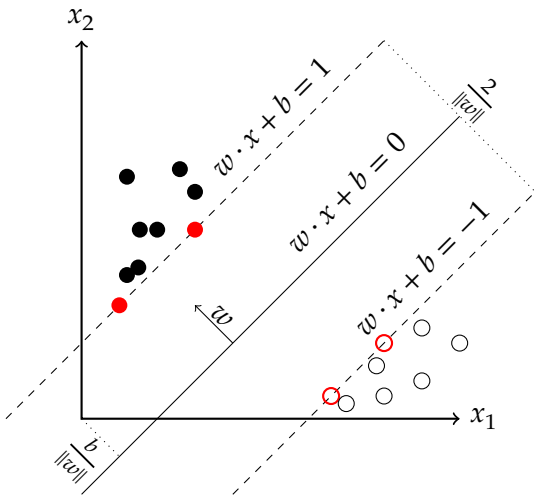

FIG. 2.6 : Hyperplan optimal au sens de la marge

Soit $\{x_i, y_i\}_{i=1..N}$ l'échantillon d'apprentissage avec $x_i \in \mathbb{R}^n$ les vecteurs de données et $y_i \in \{-1, 1\}$ sa classe. Si l'on suppose que le jeu de donné $\{x_i, y_i\}_{i=1..N}$ est linéairement séparable c'est-à-dire

$$\mathcal{U}_{\text{adm}} = \left\{(w, b)|\forall i \; y_i(w^\top x_i = b) \geq 1\right\} \neq \emptyset$$

alors on peut définir l'hyperplan :

$$\{x : f(x) := w^\top x + b = 0\}$$

Donc la règle de classification biclasse est $\text{sgn}[f(x)]$. On appelle alors hyperplan optimal un hyperplan qui maximise la marge, on à alors le problème d'optimisation convexe suivant :

$$\begin{array}{ll} \text{minimiser} & J_{w,b}(w) = \frac{1}{2}\|w\|^2 \\ \text{sous contraintes} & y_i\left(w^\top x_i + b\right) \geq 1,\ \forall i \leq N \end{array}$$

où la solution existe bien puisque $\mathcal{U}_{\text{adm}} \neq \emptyset$. On peut alors appliquer le théorème de Karush-Kuhn-Tucker (KKT) pour caractériser les solutions w^*, b^*, λ^* de ce problème et le Lagrangien correspondant est :

$$L(w, b, \lambda) = \frac{1}{2}\|w\|^2 + \sum_{j=1}^{n} \lambda_j (1 - y_j(w^\top x_j + b))$$

$$\lambda_i \geq 0$$

La fonction duale du Lagrangien est :

$$\Theta(\lambda) = \min_{w,b} L(w, b, \lambda) = L(w^*, b^*, \lambda)$$

Pour trouver w^* et b^* on calcule le gradient et on obtient les conditions suivantes :

$$\nabla_w L(w, b, \lambda) = w - \sum_{j=1}^{n} \lambda_j y_j x_j = 0$$

Ce qui implique :

$$w^* = \sum_{j=1}^{n} \lambda_j y_j x_j$$

On a aussi :

$$\frac{\partial L(w^*, b^*, \lambda)}{\partial b} = -\sum_{j=1}^{n} \lambda_j y_j = 0$$

En réinjectant dans L on a :

$$\begin{aligned} L(w^*, b^*, \lambda) &= \sum_{j=1}^{n} \lambda_j - b\sum_{j=1}^{n} \lambda_j y_j - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^\top x_j \\ &= \sum_{j=1}^{n} \lambda_j - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^\top x_j \end{aligned}$$

Dans le cas non séparable, il n'existe pas de solution au problème puisque la solution est nécessairement un hyperplan de séparation.

Cas non séparable

On relâche alors la contrainte en autorisant certaines variables à se trouver dans la marge.

$$\begin{aligned} &\text{minimiser} && J_{w,b,\xi}(w) = \frac{1}{2}\|w\|^2 + \overbrace{C\sum_{i=1}^{n}\xi_i}^{\text{pénalisation}} \\ &\text{sous contraintes} && y_i\left(w^\top x_i + b\right) \geq 1 - \xi_i,\ \forall i \leq N \\ & && \xi_i \geq 0 \end{aligned}$$

Le Lagrangien $L(w, b, \xi, \alpha, \lambda)$ est :

$$L(w,b,\xi,\alpha,\lambda) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - y_j(w^\top x_i + b) - \xi_i) + \sum_{i=1}^{n}\lambda_i(-\xi)$$

Sous contraintes $\alpha_i > 0$ et $\lambda_i > 0$. On dérive alors par rapport à w et on obtient la condition :

$$\begin{aligned} &\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0 \\ \textit{i.e.}\quad &w = \sum_{i=1}^{n}\alpha_i y_i x_i \end{aligned}$$

En dérivant par rapport à b on obtient la condition :

$$\frac{\partial L}{\partial b} = \sum_{i}^{n}\alpha_i y_i = 0$$

Et par rapport à ξ_i on a enfin :

$$\begin{aligned} &\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i = 0 \\ \textit{i.e.}\quad &\alpha_i = C - \lambda_i \end{aligned}$$

En réinjectant, on trouve alors :

$$\begin{aligned} &\text{maximiser} && L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle \\ &\text{sous contraintes} && 0 \leq \alpha_i \leq C,\ \forall i \leq N \\ & && \sum_{i=1}^{N}\alpha_i y_i = 0 \\ &\text{La solution vérifie} && w^* = \sum_{i=1}^{N}\alpha_i^* y_i x_i \\ & && f(x) = \sum_{i=1}^{N}\alpha_i^* y_i \langle x_i, x\rangle + b \end{aligned}$$

On peut résoudre ce problème en réappliquant le théorème de KKT sur ce problème plus simple ne comportant plus qu'une variable au lieu de 3 ou par tout autre algorithme de résolution numérique comme Sequential minimal optimization tel que dans Platt (1998).

2.4.2 Non-linéarité et noyaux

Motivation

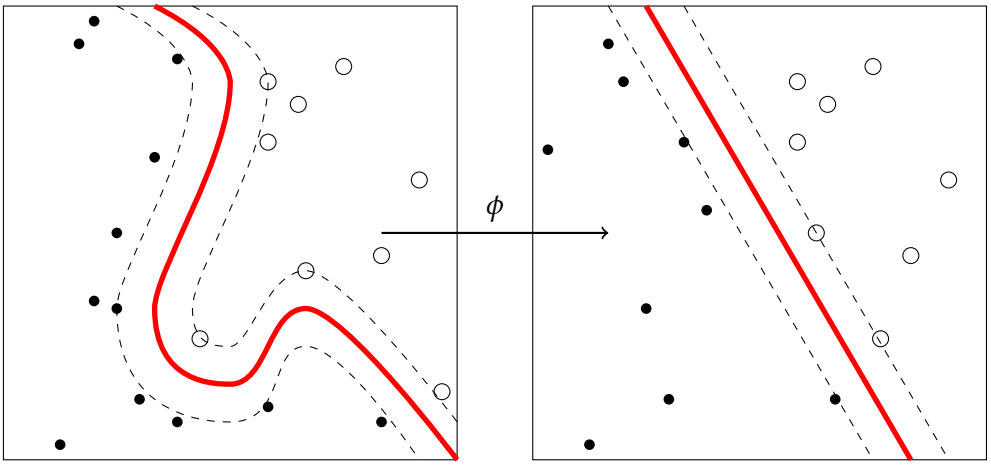

Fig. 2.7: Transformation d'un problème non linéaire en problème linéaire dans un espace de dimension supérieure

Il est également possible de séparer les classes non linéairement en se plaçant sur un espace de dimension supérieure. En effet on a le théorème suivant dû à Cover (1965).

Théorème 2.5 (Cover). *La probabilité que les classes soient linéairement séparables augmente quand les vecteurs sont non linéairement appliqués à un espace de dimension supérieure.*

Idée de preuve. Pour N individus distincts dans un espace l-dimensionnel, le nombre de groupements linéairement séparables est :

$$O(N,l) = 2\sum_{i=0}^{l}\binom{N-1}{i}$$

alors que le nombre total de groupements est 2^N. Alors la probabilité que l'échantillon soit linéairement séparable est le rapport

$$P_N^l = \frac{O(N,l)}{2^N}$$

□

En effet une surface de séparation admissible dans cet espace de dimension supérieure reste une surface de séparation dans notre espace de départ lorsque transportée par une fonction injective.

SVMs non linéaires

Il est alors légitime d'appliquer notre séparation SVM dans un espace de dimension supérieur. Notre problème devient alors :

$$\begin{aligned}
&\text{maximiser} && L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle \\
&\text{sous contraintes} && 0 \leq \alpha_i \leq \gamma, \ \forall i \leq N \\
& && \sum_{i=1}^{N} \alpha_i y_i = 0 \\
&\text{solution} && w^* = \sum_{i=1}^{N} \alpha_i^* y_i \phi(x_i) \\
& && f(x) = \sum_{i=1}^{N} \alpha_i^* y_i \langle \phi(x_i), \phi(x) \rangle + b
\end{aligned}$$

Néanmoins cette technique comporte de nombreux inconvénients ; en se plaçant en haute dimension le calcul de ϕ devient de plus en plus complexe. Qui plus est, cela suppose que l'on soit capable de trouver ϕ manuellement, c'est notamment le cas lorsque $\langle \phi(x_i), \phi(x_j) \rangle$ peut s'écrire $K(x_i, x_j)$, c'est ce qu'on appellera un noyau.

Donnons néanmoins un exemple simple pour lequel il est possible de trouver explicitement ϕ. Le cas de deux disques de points concentriques séparables par un ellipsoïde. On choisit comme noyau :

$$K(x, y) = \left(x^\top y\right)^2$$

que l'on peut associer à la fonction de transport ϕ suivante :

$$\phi(x) = \left(x_1^2, x_2^2, \sqrt{2} x_1 x_2\right)$$

Kernel Trick

On peut remarquer que le programme d'optimisation linéaire ne fait intervenir que le produit scalaire des $\phi(x_i)$. On va donc montrer l'existence de fonctions ϕ telles que :

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

où K est une fonction donnée que l'on appelle noyau. On peut donc se contenter dans ce cas de faire tous nos calculs dans l'espace d'origine à l'aide de K. Une condition suffisante pour l'existence de ϕ est que K soit symétrique défini positif.

Definition 6 (Définie Positivité)**.** Une fonction symétrique $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est définie positive si pour toute suite finie de points $x_1, \ldots, x_n$ de $\mathcal{X}$ et tout choix des réels $c_1, \ldots, c_n$ on a :

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i K(x_i, x_j) c_j > 0$$

2.4.3 Espace de Hilbert à noyau reproduisant

Un Espace de Hilbert à noyau reproduisant (RKHS) (Aronszajn, 1950), est un espace de Hilbert $\mathcal{H}$ dans lequel toutes les fonctions d'évaluation de Dirac de $\mathcal{H}$ sont bornées et continues pour la norme L^2. Par le théorème de Riesz, on a l'existence d'un unique $k_x \in \mathcal{H}$ tel que $\delta_x f = f(x) = \langle f, k_x \rangle_{\mathcal{H}}$ pour tout $f \in \mathcal{H}$. On définit alors le *Noyau Reproduisant* K de $\mathcal{H}$ par : $K(x,x') = \langle k_x, k_{x'} \rangle_{\mathcal{H}}$. La propriété importante des RKHS est que $\langle f, K(x,\cdot)\rangle_{\mathcal{H}} = f(x)$. Qui plus est, k_x est défini comme étant la fonction $y \to K(x,y)$ et donc $\langle K(x,\cdot), K(y,\cdot)\rangle_{\mathcal{H}} = K(x,y)$

Créer un RKHS à partir d'un noyau

On a vu que les noyaux reproduisants étaient définis positifs puisqu'ils étaient des produits scalaires, nous allons montrer qu'en réalité toute fonction définie positive est le noyau reproduisant d'un certain espace $\mathcal{H}_K$ unique à un isomorphisme près.

Considérons $\mathcal{H}_K = \overline{\text{vect}\, S}$ ou $S = \{k_{x_i} = K(x_i,\cdot) : x_i \in X\}$. On définit alors un produit scalaire sur $\mathcal{H}_K$ de la manière suivante : soient f,g tels que :

$$f(x) = \sum_{i=1}^{\infty} \alpha_i K(x, x_i) \in \mathcal{H}_K$$
$$g(x) = \sum_{i=1}^{\infty} \beta_i K(x, x'_i) \in \mathcal{H}_K$$

alors :

$$\langle f, g \rangle_{\mathcal{H}_K} := \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \alpha_i \beta_j K(x_i, x'_j)$$

Il est aisé de voir que $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$ définit un produit scalaire. Donc on a :

$$\begin{aligned} \langle f, K(\cdot, x)\rangle_{\mathcal{H}_K} &= \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i K(x, x_i) \\ &= f(x) \end{aligned}$$

Et en particulier

$$\langle K(\cdot, x), K(\cdot, y)\rangle_{\mathcal{H}_K} = K(x,y)$$

Donc dans cette construction on a : $\phi(x) = K(\cdot, x)$

Construction de Mercer

On rappelle d'abord le théorème dû à Mercer (1908)

Théorème 2.6 (Theorème de Mercer)**.** *Supposons que K est un noyau continu symétrique défini positif sur X. Si la fonction k ou $k(x) = K(x,x)$ est $L^1_\mu(X)$ alors il existe une base orthonormée $(e_i)_{i\in\mathbb{N}}$ de $L^2_\mu(X)$ de fonctions propres de T_K*

$$[T_K\varphi](x) = \int_\mu K(x,s)\varphi(s)\ \mathrm{d}\mu(s)$$

telles que la suite de valeurs propres $(\lambda_i)_{i\in\mathbb{N}}$ soit positive. Les fonctions propres associées à des valeurs propres non nulles sont continues sur X et K possède la représentation suivante :

$$K(s,t) = \sum_{j=1}^{\infty} \lambda_j\, e_j(s)\, e_j(t)$$

On peut alors utiliser cette représentation bien particulière pour construire un RKHS. Considérons un noyau $K(x,x')$ semi-défini positif. D'après le théorème de Mercer, il existe une famille orthonormée par rapport à une mesure μ quelconque de fonctions propres ψ_i telle que

$$K(x,x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(x')$$

Considérons maintenant l'espace de Hilbert :

$$\mathcal{H}_K = \{f | f(x) = \sum_{i=1}^{\infty} f_i\psi_i(x), \sum_{i=1}^{\infty} f_i^2/\lambda_i < \infty\}$$

muni du produit scalaire suivant :

$$\langle f, g\rangle_{\mathcal{H}} = \sum_{i=1}^{\infty} \frac{f_i g_i}{\lambda_i}$$

Le caractère reproduisant en découle aisément :

$$\begin{aligned}\langle f(\cdot), K(\cdot,x)\rangle &= \sum_{i=1}^{\infty} \frac{f_i\lambda_i\psi_i(x)}{\lambda_i} \\ &= f(x)\end{aligned}$$

De la même façon :

$$\begin{aligned}\langle K(\cdot,x), K(\cdot,x')\rangle &= \sum_{i=1}^{\infty} \frac{\lambda_i\psi_i(x)\lambda_i\psi_i(x')}{\lambda_i} \\ &= K(x,x')\end{aligned}$$

De plus $K(x,\cdot) \in \mathcal{H}$ puisque $\|K(x,\cdot)\| = K(x,x) < \infty$. Et on pose enfin

$$\phi(x) = \left(\sqrt{\lambda_1}\psi_1(x), \dots, \sqrt{\lambda_n}\psi_n(x)\right)$$

2.4.4 Théorème de représentation

On vient donc de voir l'utilité des RKHS dans le cas bien précis des SVM. Mais le théorème de représentation prouvé par Scholkopf et al. (2001) permet de généraliser cette méthode à d'autres problèmes d'optimisation similaires.

Théorème 2.7 (Théorème de représentation). *Soit $\mathcal{X}$ un ensemble non vide et K un noyau réel symétrique non défini positif sur $\mathcal{X} \times \mathcal{X}$ avec un RKHS correspondant $\mathcal{H}_K$. Étant donné un échantillon d'apprentissage $(x_i, y_i)_i \in \mathcal{X} \times \mathbb{R}$, une fonction réelle strictement croissante $g : [0, \infty) \to \mathbb{R}$, et une fonction de risque empirique quelconque $E : (\mathcal{X} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$, alors pour n'importe quel $f^* \in H_k$ satisfaisant*

$$f^* = \underset{f \in H_k}{\arg\min} \{E((x_1, y_1, f(x_1)), ..., (x_n, y_n, f(x_n))) + g(\|f\|)\}$$

f^* admet une représentation de la forme :

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i K(\cdot, x_i)$$

où $\alpha_i \in \mathbb{R}$ pour tout $1 \leq i \leq n$.

On peut donc en réalité étendre grâce aux RKHS les techniques étudiées précédemment à un bien plus vaste champ de problèmes d'optimisation et donc de techniques de classification. Les très bons résultats des SVMs en font une technique de choix pour un très grand nombre de problèmes, mais ceux-ci ne sont pas exempts de problèmes, en effet les performances dépendent grandement du noyau choisi et il est donc nécessaire d'adapter la structure du classifieur au jeu de données et au problème étudié. De plus les SVMs restent une technique "boîte noire" qui ne fournit aucune information explicative sur le modèle sous-jacent et le contributions des variables. Enfin comme beaucoup de modèles, elles sont très sensibles à l'overfitting si un noyau trop discriminant est choisi ainsi qu'au déséquilibre des données.

2.5 LES RÉSEAUX DE NEURONES

Les réseaux de neurones, ainsi nommés, car apparus simultanément et indépendamment dans la communauté neuro-scientifique et statistique, peuvent être vus comme une généralisation des méthodes linéaires. En effet, là où celles-ci considèrent des combinaisons linéaires de fonctions des entrées comme première généralisation, les réseaux de neurones répètent ce principe n fois en considérant chaque nouvelle sortie comme une nouvelle entrée.

2.5.1 Principe

Un réseau de neurones est organisé en un certain nombre de *couches* de neurones, ces derniers étant définis par $\left(\omega_{ik}^{l+1}\right)_i$ le vecteur des poids de sorties du $k^{\text{ème}}$ neurone de la couche l et $b_i k^l$ son biais. Nous ne considérons ici que les réseaux entièrement connectés c'est-à-dire tels que chaque neurone de la couche l pointe vers tous les neurones de la couche $l+1$. Ainsi chaque neurone reçoit les sorties de tous les neurones précédents et fournit en sortie à tous les neurones suivant la valeur a_k^l :

$$a_k^l = \sigma\left(\sum_j \omega_{kj}^l a_j^{l-1} + b_k^l\right)$$

Pour simplifier les notations, on notera par ω^l la matrice des poids de la couche l, b^l le vecteur des biais et a^l le vecteur des sorties. Ici σ est une fonction à choisir en fonction du problème appelée fonction d'activation.

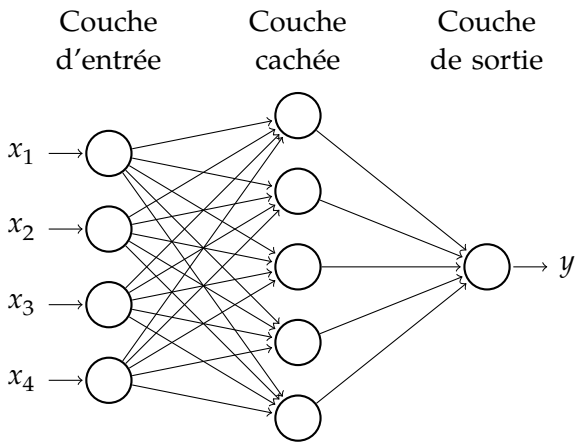

FIG. 2.8 : Réseau de neurones artificiel $4 \times 5 \times 1$

Ainsi la sortie $\hat{y} = a^L$ est définie de manière récursive :

$$\begin{aligned} a^1 &= x \\ a^l &= \sigma(\omega^l a^{l-1} + b^l) \end{aligned}$$

Comme dans les autres algorithmes d'apprentissage, on va alors chercher à apprendre ω^l et b^l afin de minimiser une certaine fonction de perte $J(y, \hat{y})$, la plupart du temps la perte L^2. En effet cette tache est légitime puisqu'il est possible de montrer qu'un tel réseau de neurones est capable d'approcher toute fonction continue sur un compact sous certaines hypothèses (HORNIK, 1991).

Théorème 2.8 (CYBENKO, 1989)**.** *Soit σ une fonction sigmoïde continue. Soit $I_m = [0,1]^m$ l'hypercube de dimension m et $C(I_m)$ l'espace des fonc-*

tions continues sur I_m. Alors pour tout $f \in C(I_m)$, $\varepsilon > 0$ il existe un entier N et $v_i, b_i \in \mathbb{R}$, $\omega_i \in \mathbb{R}^m$ tel que :

$$|\varphi(x) - f(x)| < \varepsilon, \forall x \in I_m$$
$$\text{avec} \quad \varphi(x) = \sum_{i=1}^{N} v_i \sigma(\omega_i^\top + b_i)$$

Démonstration. La preuve B de Cybenko (1989) procède en deux étapes en montrant tout d'abord que les fonctions discriminantes pour les mesures sont denses dans $C(I_m)$ puis que les fonctions sigmoïdes sont discriminantes pour les mesures. □

Le problème d'apprentissage des réseaux de neurones n'est alors qu'un problème d'optimisation en les ω_i, b_i que l'on peut résoudre par descente de gradient. Néanmoins le nombre colossal de coefficients et biais à optimiser résulte en autant de dérivées partielles à calculer, le problème ne peut donc pas être résolu de manière naïve en un temps raisonnable. Nous allons donc voir en quoi la forme particulière du réseau de neurones et donc de la fonction de sortie permet de mettre en place un algorithme d'optimisation rapide.

2.5.2 Rétropropagation

On se place ici dans le cas où la fonction de perte pour l'échantillon d'apprentissage entier s'exprime comme la moyenne des pertes pour chaque individu et où la perte pour un seul individu peut s'écrire comme fonction des sorties du réseau. On suppose de plus que la perte est différentiable. C'est par exemple le cas pour la perte quadratique que nous utiliserons à partir d'ici. Nous n'exposerons qu'une version simplifiée des réseaux de neurones *feed-forward* sans aborder les subtilités telles que le *dropout* et la régularisation. Pour de plus amples informations il est possible de consulter (Bengio et al., 2015 ; Nielsen, 2015 ; Rojas, 1996).

On cherche à optimiser les poids et biais afin de minimiser la perte, on procède alors par descente de gradient. Il faut alors pour cela calculer le gradient de la perte par rapport aux poids et biais. Le calcul du gradient pourrait se faire par différence finie, ce qui serait non seulement très coûteux, car nécessitant $O(n)$ opérations où n est le nombre de paramètres a optimiser, mais également imprécis. Il est également possible d'effectuer une différentiation symbolique soit à la main, soit par ordinateur. Dans notre cas il est finalement possible d'exploiter la structure très particulière du problème.

Si l'on appelle θ l'ensemble des paramètres, la procédure est alors :

$$\Delta\theta = -\gamma \nabla_\theta C$$

ou γ est la vitesse d'apprentissage.

Or un réseau de neurones représente un graphe de composition de fonctions, nous allons donc utiliser les propriétés de la différentielle d'une composée de fonctions pour faciliter les calculs.

Notons $z_j^l = \sum_k \omega_{jk}^l a_k^{l-1} + b_j^l$ et l'*erreur* locale $\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$.

En utilisant la formule de la dérivée d'une fonction composée on trouve les équations de la rétropropagation qui permettent de calculer l'*erreur* et donc les dérivées partielles à chaque nœud en remontant le graphe de manière rétrograde.

Théorème 2.9 (Équations de la rétropropagation)**.**

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{2.4}$$

$$\delta^l = \left(\left(\omega^{l+1}\right)^\top \delta^{l+1}\right) \odot \sigma'(z^l) \tag{2.5}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{2.6}$$

$$\frac{\partial C}{\partial \omega_{jk}^l} = a_k^{l-1} \delta_j^l \tag{2.7}$$

Ou $\odot$ est le produit d'Hadamard ou de Schur, c'est à dire le produit composante par composante.

En utilisant les équations de rétropropagation il est alors possible de calculer le gradient par rapport à tous les paramètres en un seul parcours du graphe et de façon indépendante au sein de chaque couche, donc entièrement en parallèle au sein de chaque couche. Ainsi le problème d'optimisation d'origine qui paraissait intraitable en un temps raisonnable peut maintenant être résolu en un temps tolérable. De plus, la capacité à hautement paralléliser l'algorithme permet, contrairement à d'autres problèmes en apparence plus simple, de diminuer le temps de calcul de façon arbitraire en rajoutant des unités de calcul, la seule contrainte étant alors les moyens financiers dont on dispose.

2.5.3 Différentiation automatique

Nous avons vu une méthode pour calculer les dérivées d'une fonction en apparence très complexe en un temps de calcul limité et sans commettre aucune erreur due aux approximations numériques. Nous allons maintenant voir que cette technique n'est pas limitée aux réseaux de neurones et peut s'appliquer sous deux formes à différents problèmes de calcul de gradient ou dérivées supérieures tels que présents en finance, voir par exemple Homescu (2011), Giles et Glasserman (2005) ou Xu et al. (2014).

En effet, de la même façon que dans le cas des réseaux de neurones il est toujours possible d'exprimer une fonction complexe comme un graphe de composition de fonctions de base. Nous allons donc utiliser les propriétés de la différentiation de composées de fonctions.

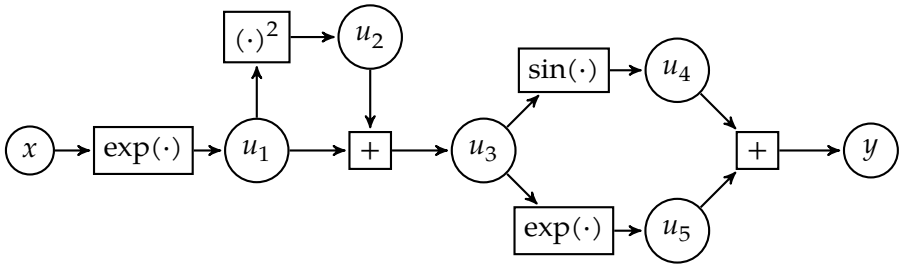

FIG. 2.9: Graphe de $f(x) = \exp(\exp(x) + (\exp(x))^2) + \sin(\exp(x) + (\exp(x))^2)$

Prenons par exemple le cas de 2.9. Nous pouvons traverser le graphe de gauche à droite, ou de droite à gauche.

Backward

On peut alors réécrire le graphe sous forme équationelle :

$$
\begin{aligned}
y &= f_6(u_5, u_4) & \mathrm{d}y &= \frac{\partial f_6}{\partial u_5}\,\mathrm{d}u_5 + \frac{\partial f_6}{\partial u_4}\,\mathrm{d}u_4 \\
u_5 &= f_5(u_3) & \mathrm{d}u_5 &= \frac{\partial f_5}{\partial u_3}\,\mathrm{d}u_3 \\
u_4 &= f_4(u_3) & \mathrm{d}u_4 &= \frac{\partial f_4}{\partial u_3}\,\mathrm{d}u_3 \\
u_3 &= f_3(u_2, u_1) & \mathrm{d}u_3 &= \frac{\partial f_3}{\partial u_2}\,\mathrm{d}u_2 + \frac{\partial f_3}{\partial u_1}\,\mathrm{d}u_1 \\
u_2 &= f_2(u_1) & \mathrm{d}u_2 &= \frac{\partial f_2}{\partial u_1}\,\mathrm{d}u_1 \\
u_1 &= f_1(x) & \mathrm{d}u_1 &= \frac{\partial f_1}{\partial u_1}\,\mathrm{d}u_1
\end{aligned}
$$

Il est ainsi aisé de voir que l'on peut obtenir $\frac{\partial y}{\partial x}$ en remontant le graphe par la droite, ce sens de calcul des dérivées dans le sens opposé du calcul de la fonction donne alors la différentiation automatique *backward*. L'algorithme s'applique à toutes les fonctions $f : \mathbb{R}^n \to \mathbb{R}^m$, la différentiation automatique *backward* est capable de calculer le gradient de chaque application partielle en une seule passe, mais nécessite m passes pour le gradient complet. L'algorithme est donc intéressant dans le cas où $n \gg m$.

Forward

Nous venons de voir une méthode permettant de calculer les dérivées en parcourant le graphe dans le sens inverse de l'évaluation de la fonction, mais il est également possible de mettre à jour l'état actuel de la dérivée en même temps que l'on traverse le graphe pour calculer la fonction. Puisqu'ici les deux calculs ont lieu dans le même sens on parle de différentiation automatique *forward*. Pour cela on augmente les

nombres réels en les munissant de ε avec $\varepsilon^2 = 0$. $\mathbf{x}$ appartient à la nouvelle algèbre obtenue appelée algèbre duale si $\mathbf{x} = x + x'\varepsilon, x, x' \in \mathbb{R}$.

Il est aisé de voir que si P est un polynôme réel alors :

$$P(x + x'\varepsilon) = P(x) + P'(x)x'\varepsilon$$

Ainsi en étendant les fonctions analytiques à cette nouvelle algèbre nous avons un moyen d'obtenir à la fois la valeur de la fonction et sa dérivée en ce point en l'évaluant dans l'algèbre augmentée. Ainsi si $\bar{f_i}$ est l'extension de f_i à notre nouvelle algèbre alors calculer $f = f_n \circ f_{n-1} \circ \ldots \circ f_1$ ainsi que sa dérivée en x revient à calculer $\bar{f_n} \circ \bar{f_{n-1}} \circ \ldots \circ \bar{f_1}(x + 1\varepsilon)$

La différentiation automatique *forward* est très facile à implémenter et, comme la version *backward*, s'étend au cas multidimensionnel. Néanmoins l'algorithme part ici d'une valeur d'entrée x_i pour obtenir toutes les dérivées des fonctions partielles par rapport à celle-ci, il faut donc n passes pour calculer le gradient entier et l'algorithme est intéressant dans le cas où $n \ll m$.

3 ENSEMBLES DE CLASSIFIEURS

3.1 DÉCOMPOSITION BIAIS–VARIANCE

3.1.1 Dans le cas de la régression L^2

On rappelle que l'erreur de prédiction à un certain point $X = x$ est :

$$\mathrm{Err}(\varphi_{\mathcal{L}}(x)) = \mathbb{E}_{Y|X=x}\left[L(Y, \varphi_{\mathcal{L}}(x))\right] \tag{3.1}$$

Dans le cas de la régression le choix le plus courant et naturel pour L est la perte quadratique. On peut alors en notant φ_B le modèle idéal théorique de Bayes :

$$\begin{aligned}
\mathrm{Err}(\varphi_{\mathcal{L}}(x)) &= \mathbb{E}_{Y|X=x}\left[(Y - \varphi_{\mathcal{L}}(x))^2\right] \\
&= \mathbb{E}_{Y|X=x}\left[(Y - \varphi_B(x) + \varphi_B(x) - \varphi_{\mathcal{L}}(x))^2\right] \\
&= \mathbb{E}_{Y|X=x}\left[(Y - \varphi_B(x))^2\right] + \mathbb{E}_{Y|X=x}\left[(\varphi_B(x) - \varphi_{\mathcal{L}}(x))^2\right] \\
&\quad + \mathbb{E}_{Y|X=x}\left[2(Y - \varphi_B(x))(\varphi_B(x) - \varphi_{\mathcal{L}}(x))\right] \\
&= \mathrm{Err}\left(\varphi_B(x)\right) + \left(\varphi_B(x) - \varphi_{\mathcal{L}}(x)\right)^2
\end{aligned}$$

Par définition du classifieur de Bayes on a $\mathbb{E}_{Y|X=x}\left[Y - \varphi_B(x)\right] = 0$ Si de plus on considère que l'échantillon d'apprentissage $\mathcal{L}$ est une variable aléatoire et que l'algorithme d'apprentissage est déterministe, on peut écrire l'erreur moyenne due à l'échantillonnage

$$\begin{aligned}
&\mathbb{E}_{\mathcal{L}}\left[(\varphi_B(x) - \varphi_{\mathcal{L}}(x))^2\right] \\
&= (\varphi_B(x) - \mathbb{E}_{\mathcal{L}}[\varphi_{\mathcal{L}}(x)])^2 + \mathbb{E}_{\mathcal{L}}\left[(\mathbb{E}_{\mathcal{L}}[\varphi_{\mathcal{L}}(x)] - \varphi_{\mathcal{L}}(x))^2\right]
\end{aligned} \tag{3.2}$$

On obtient alors la décomposition biais-variance due à Geman et al. (1992) :

Théorème 3.1 (Décomposition biais-variance)**.** *Dans le cas de la perte L^2 on a :*

$$\mathbb{E}_{\mathcal{L}}\left[\mathrm{Err}(\varphi_{\mathcal{L}}(x))\right] = \mathrm{bruit}(x) + \mathrm{biais}^2(x) + \mathrm{var}(x) \tag{3.3}$$

où

$$\begin{aligned}
\mathrm{bruit}(x) &= \mathrm{Err}(\varphi_B(x)) \\
\mathrm{biais}^2(x) &= \left(\varphi_B(x) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}(x)\right]\right)^2 \\
\mathrm{var}(x) &= \mathbb{E}_{\mathcal{L}}\left[(\varphi_{\mathcal{L}}(x) - \mathbb{E}_{\mathcal{L}}[\varphi_{\mathcal{L}}(x)])^2\right]
\end{aligned}$$

3.1.2 Extension à la classification binaire

Dans le cas de la perte 0-1, c'est-à-dire

$$\mathbb{E}_{\mathcal{L}}\left[\mathbb{E}_{Y|X=x}[\mathbb{1}_{\varphi_{\mathcal{L}}(x)\neq Y}]\right] = \mathbb{P}_{\mathcal{L},Y|X=x}(\varphi_{\mathcal{L}}(x) \neq Y)$$

il est possible d'utiliser le fait que de nombreux algorithmes de classification calculent en réalité l'estimateur $\hat{p}_{\mathcal{L}}(Y = c \mid X = x)$ et en déduisent la classe par la règle :

$$\varphi_{\mathcal{L}}(x) = \arg\max_{c\in\mathcal{Y}} \hat{p}_{\mathcal{L}}(Y = c \mid X = x)$$

On trouve alors de la même façon que précédemment la décomposition

$$\begin{aligned}&\mathbb{E}_{\mathcal{L}}\left[\mathbb{E}_{Y|X=x}\left[\mathbb{1}_{\varphi_{\mathcal{L}}(x)\neq Y}\right]\right]\\ &= \mathbb{P}\left(\varphi_B(x) \neq Y\right) + \mathbb{P}_{\mathcal{L}}\left(\varphi_{\mathcal{L}}(x) \neq \varphi_B(x)\right)\left(2\mathbb{P}\left(\varphi_B(x) = Y\right) - 1\right)\end{aligned}$$

où $\mathbb{P}$ est à chaque fois $\mathbb{P}_{Y|X=x}$. La prévision du modèle diffère de la prévision optimale de Bayes lorsque

$$\mathbb{P}_{\mathcal{L}}\left(\varphi_{\mathcal{L}}(x) \neq \varphi_B(x)\right) = \mathbb{P}_{\mathcal{L}}\left(\hat{p}_{\mathcal{L}}(Y = \varphi_B(x)) < 0.5\right)$$

Si l'on suppose alors que $\hat{p}_{\mathcal{L}}(Y = \varphi_B(x))$ est normalement distribuée alors on obtient le théorème suivant

Théorème 3.2 (Louppe, 2014)**.** *Pour la perte 0-1 dans le cas de la classification, l'erreur de généralisation attendue* $\mathbb{E}_{\mathcal{L}}\left[\mathrm{Err}(\varphi_{\mathcal{L}}(x))\right]$ *en* $X = x$ *admet la décomposition suivante :*

$$\begin{aligned}&\mathbb{E}_{\mathcal{L}}\left[\mathrm{Err}(\varphi_{\mathcal{L}}(x))\right] = \mathbb{P}(\varphi_B(x) \neq Y) + \cdots\\ &\cdots + \Phi\left(\frac{0.5 - \mathbb{E}_{\mathcal{L}}[\hat{p}_{\mathcal{L}}(Y = \varphi_B(x))]}{\sqrt{\mathbb{V}_{\mathcal{L}}\hat{p}_{\mathcal{L}}(Y = \varphi_B(x))}}\right)(2\mathbb{P}(\varphi_B(x) = Y) - 1)\end{aligned}$$

On déduit de ce théorème deux cas importants :

- Lorsque $\mathbb{E}_{\mathcal{L}}\left[\hat{p}_{\mathcal{L}}(Y = \varphi_B(x))\right] > 0.5$ alors une réduction de la variance réduit l'erreur vers l'erreur de Bayes.
- Lorsque $\mathbb{E}_{\mathcal{L}}\left[\hat{p}_{\mathcal{L}}(Y = \varphi_B(x))\right] < 0.5$ une réduction de la variance augmente l'erreur.

3.2 RÉDUCTION DE LA VARIANCE

3.2.1 Combinaison linéaire de classifieurs

On suppose être capable de créer des classifieurs aléatoires $\varphi_{\mathcal{L},m}$ tous appris sur le même échantillon et prédisant le même modèle. On veut alors combiner ces classifieurs en *ensemble de classifieurs* afin d'éventuellement réduire l'erreur de généralisation par rapport à un classifieur

unique. La méthode la plus naturelle pour agréger les prédictions des classifieurs est une combinaison linéaire des prédictions, ainsi dans le cas de la régression on adopte :

$$\psi_{\mathcal{L},1,\dots,M}(x) = \frac{1}{M}\sum_{m=1}^{M}\varphi_{\mathcal{L},m}(x)$$

Dans le cas de la classification, si les classifieurs ne fournissent que des prévisions 0-1 on utilise le *vote de majorité* pour effectuer la prédiction finale

$$\psi_{\mathcal{L},1,\dots,M}(x) = \arg\max_{c\in\mathcal{Y}}\frac{1}{M}\sum_{m=1}^{M}\mathbb{1}_{\varphi_{\mathcal{L},m}(x)=c}$$

Dans le cas où les classifieurs donnent en sortie $\hat{p}_{\mathcal{L},m}(Y = c \mid X = x)$ on peut adopter comme règle de vote

$$\psi_{\mathcal{L},1,\dots,M}(x) = \arg\max_{c\in\mathcal{Y}}\frac{1}{M}\sum_{m=1}^{M}\hat{p}_{\mathcal{L},m}(Y = c \mid X = x)$$

Il est alors possible de montrer le théorème suivant :

Théorème 3.3 (Louppe, 2014)**.** *Dans le cas de la régression L^2, la décomposition biais-variance de l'erreur de généralisation attendue $\mathbb{E}_{\mathcal{L}}\left[\text{Err}\left(\psi_{\mathcal{L},1,\dots,M}(x)\right)\right]$ d'un ensemble de M classifieurs aléatoires $\varphi_{\mathcal{L},m}$ est :*

$$\mathbb{E}_{\mathcal{L}}\left[\text{Err}(\psi_{\mathcal{L},1,\dots,M}(x))\right] = \text{bruit}(x) + \text{biais}^2(x) + \text{var}(x)$$

avec

$$\begin{aligned}
\text{bruit}(x) &= \text{Err}(\varphi_B(x)) \\
\text{biais}^2(x) &= \left(\varphi_B(x) - \mathbb{E}_{\mathcal{L},i}\left[\varphi_{\mathcal{L},i}(x)\right]\right)^2 \\
\text{var}(x) &= \rho(x)\sigma^2_{\mathcal{L},i}(x) + \frac{1-\rho(x)}{M}\sigma^2_{\mathcal{L},i}(x)
\end{aligned}$$

Avec

$$\begin{aligned}
\sigma^2_{\mathcal{L},i}(x) &= \mathbb{V}_{\mathcal{L},i}[\varphi_{\mathcal{L},i}(x)] \\
\mu_{\mathcal{L},i}(x) &= \mathbb{E}_{\mathcal{L},i}[\varphi_{\mathcal{L},i}(x)] \\
\rho(x) &= \frac{\mathbb{E}_{\mathcal{L},i,i'}[\varphi_{\mathcal{L},i}(x)\varphi_{\mathcal{L},i'}(x)] - \mu^2_{\mathcal{L},i}(x)}{\sigma^2_{\mathcal{L},i}(x)}
\end{aligned}$$

Cette décomposition montre l'un des éléments cruciaux à vérifier pour la construction de nos classifieurs : ceux-ci doivent être le plus faiblement corrélés possible pour réduire l'erreur, mais cela doit se faire sans augmenter de façon trop importante le biais des classifieurs.

3.2.2 Combinaisons bayésiennes

Les méthodes de combinaisons de différents classifieurs cherchent toutes à modéliser la présence d'incertitude dans les prédictions de chacun des classfieurs et donc a réduire cette incertitude *a posteriori* en prenant en compte l'information apprise grâce à l'échantillon d'apprentissage, les prédictions individuelles et le nouvel individu à classer. Le problème de création d'un ensemble, une fois reformulé de cette façon semble naturellement être un problème Bayesien. Ainsi en utilisant la formule des probabilités totales on obtient en notant (φ_m) l'ensemble des modèles construits :

$$\mathbb{P}(\zeta \mid \mathcal{L}) = \sum_{m=1}^{M} \mathbb{P}(\zeta \mid \varphi_m, \mathcal{L}) \mathbb{P}(\varphi \mid \mathcal{L}) \tag{3.4}$$

où ζ est une quelconque valeur d'intérêt, dans la plupart des cas $\zeta = y$ et on a donc :

$$\mathbb{P}(y_i \mid x_i, \mathcal{L}) = \sum_{m=1}^{M} \mathbb{P}(y_i \mid x_i, \varphi_m, \mathcal{L}) \mathbb{P}(\varphi \mid \mathcal{L}) \tag{3.5}$$

En utilisant encore une fois le théorème de Bayes, on peut aussi écrire :

$$\mathbb{P}(\varphi \mid \mathcal{L}) \propto \mathbb{P}(\varphi_m) \mathbb{P}(\mathcal{L} \mid \varphi_m) \tag{3.6}$$

$$\propto \mathbb{P}(\varphi_m) \int \mathbb{P}(\mathcal{L} \mid \theta_m, \varphi_m) \mathbb{P}(\theta_m \mid \varphi_m) \, \mathrm{d}\theta_m \tag{3.7}$$

ou $\mathbb{P}(\theta_m \mid \varphi_m)$ est définie *a priori*.

Néanmoins cette méthode, dite de Bayesian Model Averaging (BMA) présente deux inconvénients majeurs mis en avant dans Minka (2002). Tout d'abord la forme de 3.4 suppose que l'on fait l'hypothèse que exactement un modèle de l'espace des modèles considéré est correct, et non une combinaison de modèles. L'espace d'hypothèses sur lequel on cherche la solution est donc nettement plus restreint que l'espace des combinaisons linéaires de modèles comme dans le cas du **B**ootstrap **Agg**rega**ting** (Bagging) par exemple. D'autre part puisqu'il est nécéssaire de tirer un nombre discret d'hypothèses dans l'espaces des hypothèses, l'algorithme de BMA a tendance lorsque $|\mathcal{L}| \to \infty$ à converger vers le modèle le plus probable ; tous les poids sont alors concentrés sur un unique modèle et la combinaison est dégénérée. Paradoxalement, un apport de données peut diminuer les performances de l'ensemble et l'algorithme de BMA se résume à sélectionner le meilleur modèle.

Pour pallier ces 2 problèmes, Monteith et al. (2011) proposent une modification du BMA. Ainsi, au lieu de considérer seulement comme espace d'hypothèse l'espace des modèles possibles $\mathcal{M}$, ils considèrent l'espace des combinaisons de $\mathcal{M}$ que l'on notera $C(\mathcal{M})$. Le problème devient alors :

$$\mathbb{P}(y_i \mid x_i, \mathcal{L}) = \sum_{c \in C(\mathcal{M})} \mathbb{P}(y_i \mid x_i, \varphi_m, \mathcal{L}) \mathbb{P}(c \mid \mathcal{L}) \tag{3.8}$$

Les difficultés liées au BMA et Bayesian Model Combination (BMC) sont multiples et sont exposées avec différentes solutions par Hoeting et al. (1999) :

- Le nombre de termes dans 3.8 est infini et celui dans 3.8 potentiellement très grand, il n'est pas possible de tous les énumérer et une restriction est nécessaire.
- Les intégrales présentes de façon implicite dans 3.8 et 3.4, c'est-à-dire dans 3.7 sont difficiles à calculer et un échantillonnage par Markov Chain Monte Carlo (MCMC) est nécessaire.
- Le choix des a priori $\mathbb{P}(\theta_m \mid \varphi_m)$ et $\mathbb{P}(\varphi_m)$ est difficile.

3.3 RÉDUCTION DU BIAIS

3.3.1 Boosting

Au lieu de combiner des modèles forts, mais à forte variance il est également possible de combiner des modèles faibles afin de les transformer en un classifieur fort. Ainsi Breiman (1998) puis Schapire (1990) introduisent le *boosting* afin de combiner plusieurs instances d'un même classifieur de biais élevé de façon adaptative pour en diminuer le biais. Cette technique de boosting consiste à entraîner des classifieurs successifs en se concentrant à chaque étape sur les valeurs mal classifiées puis à faire une moyenne pondérée par la puissance prédictive respective de chacun de ces classifieurs.

Dans le cas de la classification binaire, quitte à renommer les modalités, nous considérons ici que $\mathcal{Y} = \{-1, 1\}$ et le classifieur faible de base $B : (\mathcal{X} \times \mathcal{Y}) \times \mathbb{R}^N \times \mathcal{X} \to \{-1, 1\}$ construit en prenant en compte les poids. L'algorithme de boosting original introduit par Freund et al. (2003) est alors :

Algorithme 1 AdaBoost.M1

procedure AdaBoost.M1($\mathcal{L}, B$)
  $\omega_i \leftarrow \frac{1}{N}, i = 1, 2, \dots, N$
  pour $k = 1, \dots, K$ **faire**
    $B_k(x) \leftarrow B(\mathcal{L}, (\omega_i), x)$
    $\text{Err}_k \leftarrow \frac{\sum_{i=1}^{N} \omega_i \mathbb{1}_{y_i \neq B_k(x_i)}}{\sum_{i=1}^{N} \omega_i}$
    $\alpha_k \leftarrow \log\left((1 - \text{Err}_k)/\text{Err}_k\right)$
    $\omega_i \leftarrow \omega_i \exp\left(\alpha_m \mathbb{1}_{y_i \neq B_m(x_i)}\right)$
  fin pour
  renvoyer $\varphi(x) = \text{sign}\left[\sum_{k=1}^{K} \alpha_k B_k(x)\right]$
fin procedure

L'algorithme `AdaBoost.M1` minimise en réalité la perte exponentielle de manière gloutonne[1]. Trevor Hastie et al. (2008) donnent une forme

[1] Un algorithme glouton est un algorithme qui cherche à trouver une solution a priori globale en trouvant des solutions heuristiques locales.

plus générale du principe sous le terme *Induction par ajout d'un modèle additif (Forward Stagewise Additive Modeling)* détaillé dans l'algorithme 2.

Le but étant d'approximer f par une combinaison linéaire de fonctions $b(x, \gamma_k)$, c'est à dire

$$f(x) = \sum_{k=1}^{K} \beta_k b(x, \gamma_k)$$

Il s'agit d'une forme classique et intuitive pour la construction des classifieurs puisque l'on peut par exemple représenter les réseaux de neurones ou les arbres sous cette forme respectivement comme somme de sigmoïdes et somme d'indicatrices d'appartenance aux feuilles. Trouver la meilleure combinaison est un problème compliqué et généralement trop coûteux en temps de calcul. On se ramène donc à un algorithme inductif glouton pour approcher la solution.

Algorithme 2 Forward Stagewise Additive Modeling

procedure FORWARD STAGEWISE ADDITIVE MODELING
  $f_0(x) \leftarrow 0$
  pour $k = 1, \dots, K$ **faire**
    $(\beta_k, \gamma_k) \leftarrow \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L\left(y_i, f_{k-1}(x_i) + \beta\right)$
    $f_k(x) \leftarrow f_{k-1}(x) + \beta_k b(x, \gamma_k)$
  fin pour
  renvoyer $f_K(x)$
fin procedure

Dans le cas de l'algorithme `AdaBoost.M1` le critère de perte choisi est la perte exponentielle

$$L(y, f(x)) = \exp(-yf(x))$$

La procédure *Forward Stagewise Additive Modeling* revient alors à déterminer

$$(\beta_k, B_k) = \arg\min_{\beta,B} \sum_{i=1}^{N} \exp\left(-y_i(f_{k-1}(x_i) + \beta B(x_i))\right) \tag{3.9}$$

En résolvant 3.9 on trouve (Trevor HASTIE et al., 2008) que :

$$\begin{aligned}
& & B_k &= \arg\min_{B} \sum_{i=1}^{N} \omega_i^{(k)} \mathbb{1}_{y_i \neq B(x_i)} \\
\text{avec} & & \omega_i^{(k)} &= \exp\left(-y_i f_{k-1}(x_i)\right) \\
& & \beta_k &= \frac{1}{2} \log \frac{1 - \text{err}_k}{\text{err}_k} \\
\text{où} & & \text{err}_k &= \frac{\sum_{i=1}^{N} \omega_i^{(k)} \mathbb{1}_{y_i \neq B(x_i)}}{\sum_{i=1}^{N} \omega_i^{(k)}}
\end{aligned}$$

La perte exponentielle est choisie, car elle est plus sensible aux changements dans les probabilités estimées des classes que la simple erreur de classification.

Il est possible de choisir d'autres fonctions de perte, mais il faut prendre en compte leur robustesse notamment par rapport à la marge $yf(x)$. Ainsi une fonction de perte robuste pénalisera plus les marges négatives, c'est-à-dire les éléments mal classés, que les marges positives qui sont déjà bien classées. Dans le cas de la régression il est nécessaire d'utiliser une perte résistante aux données aberrantes comme la perte L^1 tout en présentant des performances proches de la perte L^2 comme la perte de Huber HUBER (1964) :

$$L(y,f(x)) = \begin{cases} |y-f(x)|^2 & \text{pour } |y-f(x)| < \delta \\ 2\delta|y-f(x)| - \delta^2 & \text{sinon} \end{cases}$$

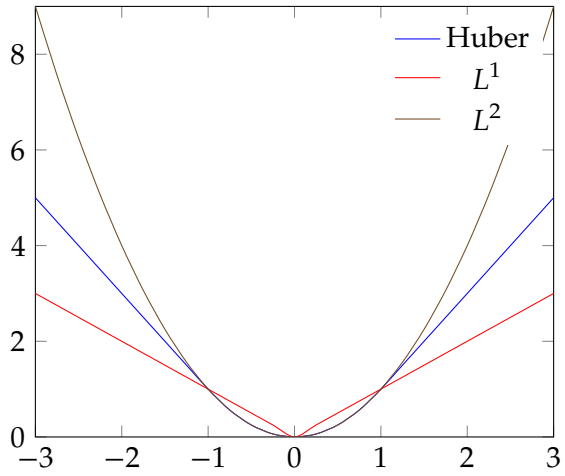

FIG. 3.1 : Comparaison des pertes Huber, L^1 et L^2

Gradient Boosting

Résoudre 3.9 est également possible grâce à des méthodes numériques approchées, en effet le programme d'optimisation est

$$\begin{aligned} &\text{minimiser} && L(f) = \sum_{i=1}^{N} L\left(y_i, f(x_i)\right) \\ &\text{sous contrainte} && f(x) = \sum_{k=1}^{K} \beta_k b(x, \gamma_k) \end{aligned} \tag{3.10}$$

On peut alors appliquer notre procédure de boosting dans le cas plus général d'une perte L différentiable, ainsi la méthode de la descente de

gradient est elle-même une procédure gloutonne qui résout 3.10 sous la forme :

$$[g_k]_i = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i)=f_{k-1}(x_i)} \tag{3.11}$$

$$\rho_k = \arg\min_{\rho} L(f_{k-1} - \rho g_k) \tag{3.12}$$

Et l'approximation courante est mise à jour avec

$$f_k = f_{k-1} - \rho_k g_k$$

Cela ne correspond pas exactement à 3.10 puisque f doit être contraint, on choisit donc d'entraîner un classifieur sous notre forme contrainte afin d'approcher au plus les valeurs du gradient souhaité. On obtient alors l'algorithme 3 pour le gradient boosting.

Algorithme 3 Gradient Boosting

procedure GRADIENT BOOSTING($\mathcal{L}, L$)
  $f_0(x) \leftarrow \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$
  pour $k = 1, \ldots, K$ **faire**
    $[r_m]_i \leftarrow -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i)=f_{k-1}(x_i)}$
    Entraîne $B_k(x)$ sur les pseudo-résidus $\{(x_i, [r_k]_i\}$
    $\gamma_k \leftarrow \arg\min_\gamma \sum_{i=1}^{N} L(y_i, f_{k-1}(x_i) + \gamma B_k(x_i))$
    $f_k(x) \leftarrow f_{k-1} + \gamma_k B_k(x)$
  fin pour
  renvoyer $f_K(x)$
fin procedure

Les arbres de classification présentent d'excellentes performances en tant que classifieur de base pour le boosting, il faut néanmoins porter une attention particulière à leurs tailles afin de pouvoir capturer suffisamment d'interactions entre les variables sans augmenter l'erreur de généralisation.

3.3.2 Stacking

Nous avons vu plusieurs techniques visant à améliorer les performances d'un classifieur en combinant les résultats de plusieurs classifieurs existants. Néanmoins toutes ces méthodes combinent les résultats de classifieurs issus du même algorithme, une extension naturelle serait alors de créer des classifieurs radicalement différents et combiner leurs résultats pour en obtenir de nouveaux. Il est alors possible de voir cette seconde étape comme une procédure d'apprentissage machine sur un nouveau *méta* échantillon composé de l'ensemble des prédictions de nos classifieurs. Cette méthode appelée *Stacking* ou *Blending*

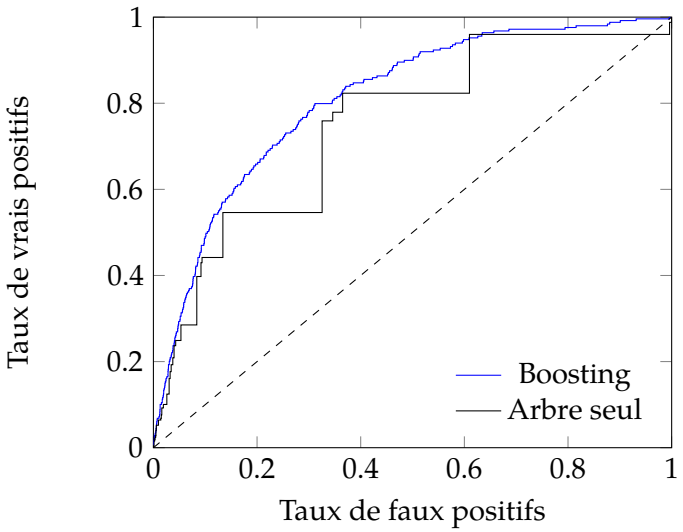

FIG. 3.2 : Performances du boosting d'arbres (AUC = 0.8103).

introduite par WOLPERT (1992) a pris de l'importance ces dernières années en présentant d'excellentes performances sur des problèmes industriels réels comme le problème de recommandation de Netflix (Robert M BELL et al., 2007 ; Robert M. BELL et al., 2008 ; JAHRER et al., 2010 ; TÖSCHER et al., 2009).

Dans van der LAAN et al. (2007), les auteurs introduisent un algorithme de *super apprentissage* et montrent que ce nouveau *super classifieur* est asymptotiquement au moins aussi bon que le meilleur des classifieurs le constituant. Leur algorithme procède en plusieurs étapes :

1. L'échantillon d'apprentissage est découpé en N tranches T_i de tailles proches

2. Pour chaque tranche T_i tous les classifieurs φ_i^j sont construits sur l'ensemble $\mathcal{L} \setminus T_i$ et leurs prédictions sur T_i agrégées dans un nouvel échantillon $\mathcal{M}$ avec les vraies étiquettes de chaque individu.

3. Une fois cette procédure répétée N fois le modèle d'agrégation choisi construit un classifieur Ψ sur $\mathcal{M}$

4. Chaque classifieur φ^j est reconstruit sur $\mathcal{L}$

Le super classifieur étant alors $\{(\varphi^1, \dots, \varphi^M), \Psi\}$

Le choix de l'algorithme de meta-apprentissage Ψ influe également grandement sur les performances du super classifieur, il est possible de choisir une fonction aussi simple que le vote majoritaire ou la moyenne non pondérée ou arbitrairement complexe comme un réseau de neurones artificiels profond, le choix étant alors à faire en fonction de jeux de données, de la mesure de performance choisie et le temps de calcul disponible. JAHRER et al. (2010) fournit un aperçu de différentes techniques de mélange et leurs performances.

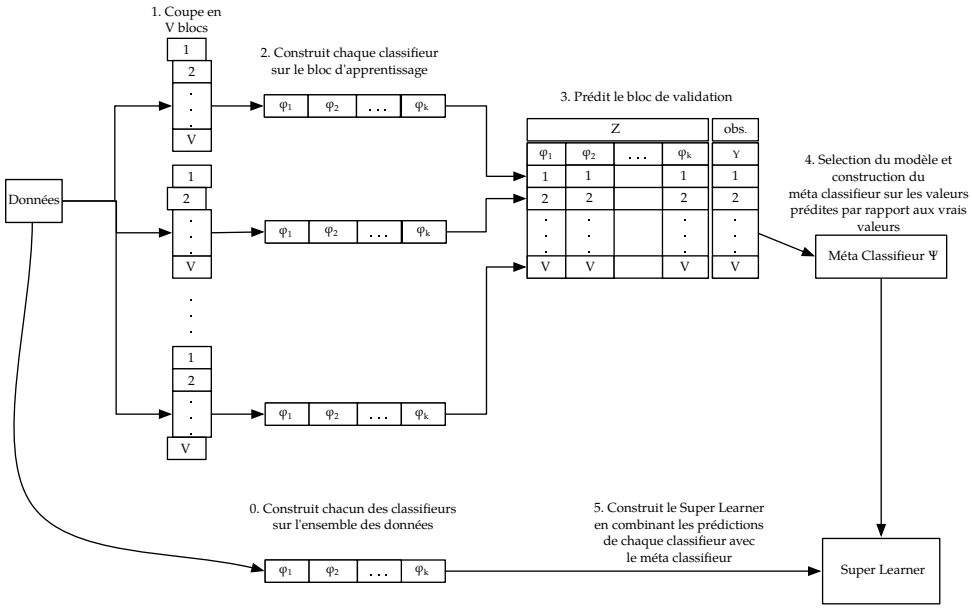

FIG. 3.3: Fonctionnement de l'algorithme SuperLearner, adapté de van der LAAN et al. (2007)

Ces méthodes de mélange considèrent tous les modèles disponibles quelque soit leur contribution aux performances, pour pallier ce problème et permettre une optimisation de l'ensemble de modèles par rapport à la métrique de son choix. CARUANA et al. (2004) proposent une méthode de construction d'ensemble par étapes où seuls les classifieurs améliorant la qualité de l'ensemble sont rajoutés. En ne prenant comme fonction de mélange que la moyenne, ils permettent des combinaisons très rapides, ce qui autorise l'utilisation de milliers de classifieurs de base. Mais il est tout à fait envisageable d'utiliser un meta classifieur plus performant.

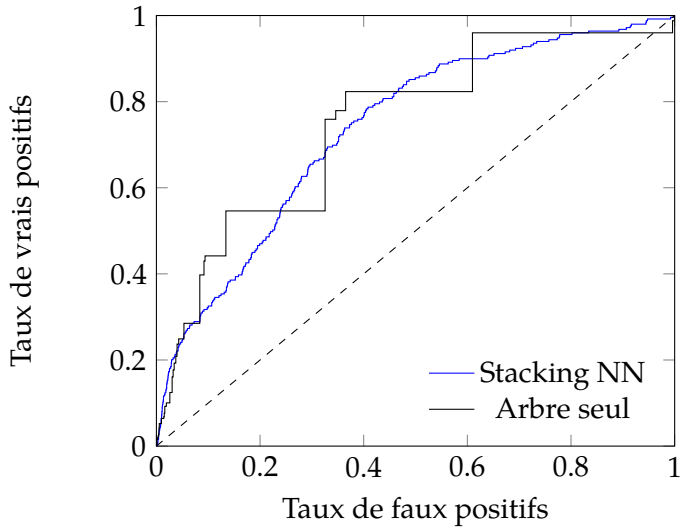

Fig. 3.4: Performances du stacking avec réseaux de neurones (AUC = 0.7372).

4 ARBRES ET FORÊTS

4.1 ARBRES DE DÉCISION

La performance pure des méthodes statistiques a toujours été l'un des principaux moteurs de la recherche dans le domaine, mais cela n'est pas le seul critère de la qualité d'un modèle. En effet, l'interprétabilité d'un modèle est dans beaucoup de domaines aussi, si ce n'est plus, important que la performance du modèle. Parmi ces modèles explicatifs, l'un sort du lot par sa combinaison de bonnes performances et facilité de compréhension : les arbres de décision. Nous suivrons les notations et définitions de Louppe (2014), et renvoyons à sa thèse pour un aperçu exhaustif de la littérature existante ainsi qu'une analyse de l'implémentation et la complexité algorithmique des arbres et forêts aléatoires, ce chapitre n'a pas donc vocation a apporter un point de vu original. Historiquement introduit par Morgan et Sonquist en 1963 la version moderne de ces arbres a été introduite indépendamment par Breiman, Friedman et Quinlan. Plus précisément c'est le modèle Classification and Regression Tree (CART) de Breiman et al. (1984) qui introduit de façon rigoureuse et unifiée la théorie des arbres de décision. Avant même d'en étudier leurs caractéristiques, il est possible de donner plusieurs raisons expliquant la popularité des arbres de décision

- Ils sont non-paramétriques.
- Ils peuvent traiter les données hétérogènes (mélange de variables qualitatives et quantitatives) ainsi que les données manquantes.
- Ils effectuent automatiquement une sélection des variables et sont donc en partie résistants aux variables inutiles ou bruitées.
- Ils sont résistants aux outliers ou erreurs d'étiquetage.
- Ils sont faciles d'interprétation.

4.1.1 Structure

De la façon la plus générale, on peut voir un arbre de décision comme une partition récursive de l'espace ce qui est tout particulièrement adapté aux problèmes de classification. En effet si $\mathcal{Y} = \{c_1, \dots, c_j\}$ est l'espace des classes, alors on peut voir notre problème de classification supervisée sur Ω comme une partition

$$\Omega = \Omega_{c_1} \cup \dots \cup \Omega_{c_j}$$

avec Ω_{c_k} l'ensemble des individus de classe c_k. On peut donc voir notre classifieur φ comme une partition de l'espace puisqu'il définit une approximation $\hat{Y}$ de Y, à l'exception que cette partition se fait sur $\mathcal{X}$ au lieu de Ω directement. On a alors :

$$\mathcal{X} = \mathcal{X}^{\varphi}_{c_1} \cup ... \cup \mathcal{X}^{\varphi}_{c_j}$$

où $\mathcal{X}^{\varphi}_{c_k}$ est l'ensemble des $x \in \mathcal{X}$ tels que $\varphi(x) = c_k$. Notre problème d'apprentissage peut alors se réécrire comme un problème d'approximation de la partition optimale de Bayes :

$$\mathcal{X} = \mathcal{X}^{\varphi_B}_{c_1} \cup ... \cup \mathcal{X}^{\varphi_B}_{c_j}$$

Nous pouvons donc maintenant créer nos arbres comme des partitions récursives de l'espace.

Definition 7 (Arbre enraciné)**.** Un arbre enraciné est un graphe acyclique orienté $G = (V, E)$ où l'un des nœuds est désigné comme racine et toutes les arrêtes fuient la racine.

Definition 8 (Nœuds)**.** S'il existe une arrête de t_1 vers t_2 alors t_1 est appelé nœud père de t_2 et t_2 nœud fils de t_1.

Definition 9. Un nœud d'un arbre enraciné est dit interne s'il possède au moins un fils et terminal (ou feuille) sinon.

Definition 10. Un arbre binaire est un arbre enraciné où chaque nœud interne possède exactement 2 fils : le fils gauche et le fils droit notés à partir d'ici respectivement t_L et t_R.

Dans ces conditions, un *arbre de décision* est un classifieur $\varphi : \mathcal{X} \to \mathcal{Y}$ représenté par un arbre enraciné où chaque nœud t représente un sous-espace $\mathcal{X}_t \subseteq \mathcal{X}$ avec la racine t_0 représentant $\mathcal{X}$ tout entier. Les fils $t_i, ..., t_j$ de t représentent alors une partition disjointe de $\mathcal{X}_t$. Chaque feuille est alors étiquetée avec $\hat{y}_t \in \mathcal{Y}$, la meilleure approximation possible de y. Si $\mathcal{Y} = \{c_1, ..., c_j\}$ alors l'arbre est dit de classification, si $\mathcal{Y} = \mathbb{R}$ il est dit de régression, ce ne sont pas les seuls types d'arbres possibles et l'on peut notamment citer les arbres de quantiles (Meinshausen, 2006), de densités (Ram et al., 2011) ou de survie (Ishwaran, Kogalur, Blackstone et al., 2008). Alors $\varphi(x)$ est l'étiquette de la feuille atteinte à partir de t_0 en suivant les règles de coupure. Le choix de représenter un problème de classification par un graphe acyclique est tout à fait raisonnable. En effet, on cherche à modéliser un système complexe par les interactions entre les différentes variables aléatoires qui le composent, ce qui correspond exactement au cadre général plus vaste des *modèles graphiques*. Néanmoins en restreignant le type de graphe considéré on se ramène à un problème de partitionnement bien plus facile à étudier que ce soit de manière théorique ou dans la résolution pratique.

Le choix de représenter le problème sous la forme d'arbres est arbitraire et ne sert qu'à rendre les calculs plus faciles, il est tout à fait

envisageable de détendre les restrictions sur la forme du graphe et autoriser par exemple plusieurs ancêtres. Les performances peuvent ainsi être améliorées au détriment de la facilité de calcul (Shotton et al., 2013).

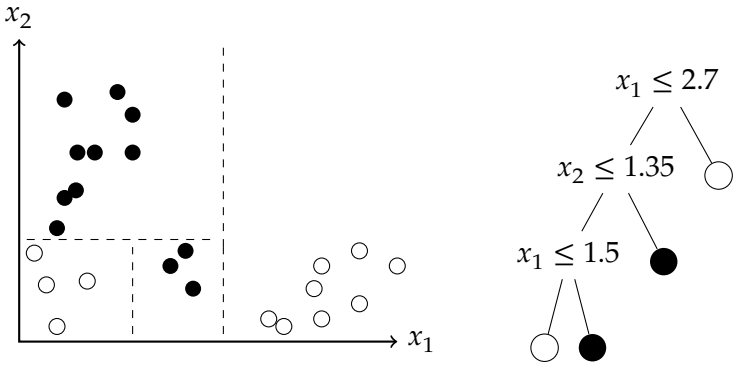

Fig. 4.1: Équivalence entre la partition du plan et un arbre de décision binaire

4.1.2 Critère d'impureté

Apprendre un arbre de décision revient alors à déterminer la plus petite des partitions la plus proche de la partition engendrée par Y sur $\mathcal{X}$, ce problème est en réalité NP-complet et trouver la meilleure approximation est impossible en un temps raisonnable. Il est donc nécessaire de développer une heuristique permettant de construire l'arbre par induction en maximisant une fonction "objectif" assurant de faire un choix judicieux à chaque étape. Ainsi Breiman et al. (1984) définissent une notion de mesure d'*impureté* $i(t)$ qui évalue la qualité de la partition courante en t. Dans ce cadre, plus $i(t)$ est petit, plus le nœud est *pur* et meilleure est la prédiction $\hat{y}_t(x)$, où $(x, y) \in \mathcal{L}_t$ avec

$$\mathcal{L}_t = \{(x, y) \in \mathcal{L} \mid x \in \mathcal{X}_t\}$$

et $\mathcal{X}_t$ le sous-ensemble de l'espace représenté par le nœud t (donc $\mathcal{X}_{t'} \subset \mathcal{X}_t$ si t' est un descendant de t et $\mathcal{X} = \bigcup \mathcal{X}_t$).

On obtient ainsi tout naturellement un algorithme glouton de construction de l'arbre : en partant d'une racine représentant l'ensemble d'apprentissages tout entier (et donc l'espace tout entier) chaque itération divise la population du nœud courant de façon à minimiser l'impureté des nouveaux nœuds, l'algorithme s'arrête lorsque l'impureté ne peut plus être réduite. Définissons alors la diminution de l'impureté pour une coupure binaire s en t :

Definition 11. La *réduction d'impureté pondérée* d'une partition binaire $s \in Q$ divisant t en t_L et t_R est :

$$\Delta i(s, t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

où p_L (resp. p_R) est le rapport $\frac{N_{t_L}}{N_t}$ (resp. $\frac{N_{t_R}}{N_t}$) d'échantillons d'apprentissage de $\mathcal{L}_t$ allant dans t_L (resp. t_R) avec $N_t = |\mathcal{L}_t|$

Classification

Dans le cas de la classification, le rôle de l'arbre est la minimisation de l'erreur de généralisation. Il peut donc sembler naturel de prendre pour $i(t)$ l'estimation locale de la certitude de classification en t.

Definition 12. Dans le cas de la classification, l'*impureté* $i_R(t)$ basée sur l'estimation locale de la certitude de classification définie sur la perte 0-1 est :

$$i_R(t) = 1 - p(\hat{y}_t \mid t) = 1 - \max_{c \in \mathcal{Y}} p(c \mid t)$$

Ce choix est naturel puisqu'il revient à choisir la coupure qui minimise l'erreur d'apprentissage. Néanmoins cette notion d'impureté a plusieurs défauts :

1. $\Delta i(s,t)$ vaut 0 pour toutes les coupures pour lesquelles tous les fils gardent la même classe majoritaire, c'est-à-dire si $y_t = y_{t_L} = y_{t_R}$. Toutes les coupures sont alors considérées comme aussi bonnes.
2. Il y a peu d'impact de la distribution a posteriori des classes $p(y \mid t_L)$ et $p(y \mid t_R)$.

Ces défauts sont principalement dus au processus de création de l'arbre : puisque l'arbre est construit de manière gloutonne, il faudrait que notre notion d'impureté prenne en compte les possibilités de réduction future de l'impureté dans les étapes suivantes. Pour éviter les défauts de $i_R(t)$ le choix de $i(t)$ doit être fait de telle sorte que celui-ci décroît au fur et à mesure que les nœuds t deviennent plus homogènes. Dans l'algorithme CART, Breiman et al. (1984) identifient une famille de fonctions d'impureté $i(t)$ qui répond à ce critère :

Théorème 4.1. *Soit* $\Phi(p_1, \dots, p_J)$ *une fonction strictement concave définie sur* $0 \leq p_k \leq 1$, *pour* $k = 1, \dots, J$ *et* $\sum_{k=1}^{J} p_k = 1$ *telle que :*

- $\Phi(1, \dots, 0) = \Phi(0, 1, \dots, 0) = \cdots = \Phi(0, \dots, 1)$ *soit minimale*
- $\Phi\left(\frac{1}{J}, \dots, \frac{1}{J}\right)$ *soit maximale*

Alors pour $i(t) := \Phi\left(p(c_1 \mid t), \dots, p(c_J \mid t)\right)$ *et pour n'importe lequel s,*

$$\Delta i(s,t) \geq 0$$

avec égalité si et seulement si $p(c_k \mid t_L) = p(c_k \mid t_R) = p(c_k \mid t)$ *pour* $k = 1, \dots, J$

Les critères d'impuretés les plus courants dans le cas des arbres de classifications sont l'entropie de Shannon et l'indice de Gini :

Definition 13. La fonction d'impureté $i_H(t)$ basée sur l'entropie de Shannon (Shannon, 1948) est :

$$i_H(t) = -\frac{1}{2} \sum_{k=1}^{J} p(c_k \mid t) \log_2 \left(p(c_k \mid t)\right)$$

Definition 14. La fonction d'impureté $i_G(t)$ basée sur l'indice de Gini est :

$$i_G(t) = \sum_{k=1}^{J} p\left(c_k \mid t\right)\left(1 - p(c_k \mid t)\right)$$

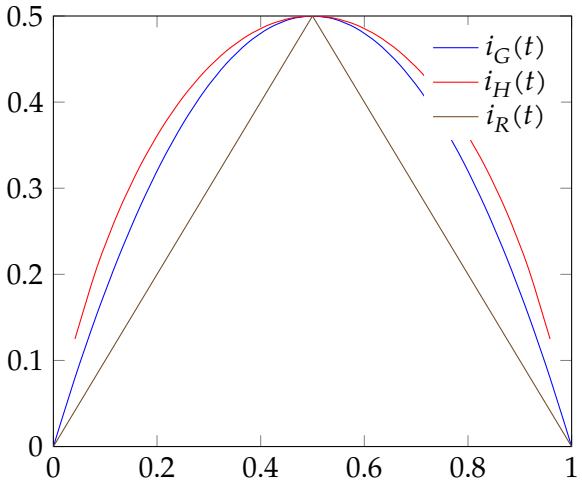

FIG. 4.2 : Comparaison de i_R, i_H et i_G

i_H et i_R répondent toutes les deux aux conditions du théorème 4.1 mais représentent des quantités différentes. i_H qui est basée sur l'entropie mesure l'incertitude de Y en t, avec ce critère d'impureté $\Delta i_H(s,t)$ qu'on appellera alors *gain d'information*. Il représente l'information apprise sur Y en coupant t en t_L et t_R. L'impureté de Gini i_G mesure quant à elle avec quelle fréquence un objet $x \in \mathcal{L}_t$ choisi au hasard serait incorrectement classifié s'il était étiqueté par $c \in \mathcal{Y}$ au hasard suivant la loi $p(y \mid t)$. Ces deux notions d'impureté ne sont pourtant pas parfaites, en effet elles présentent toutes les deux la propriété dite End Cut Property (ECP) qui peut rendre un arbre profond et donc difficilement interprétable. De plus les coupes sont biaisées vers la variable possédant le plus de réalisations. De nombreuses alternatives de critères d'impureté existent pour tenter de remédier à ces problèmes telles que le *ratio de gain d'information* utilisé dans les Arbres Extrêmement Randomisés (ExtRa-Trees), mais les changements dans la structure de l'arbre induits par le choix de la mesure d'impureté n'ont en réalité que peu d'impact sur la performance en terme de classification (LOUPPE, 2014).

Régression

Dans le cas de la régression, c'est-à-dire quand la variable Y est quantitative, il est possible de tout simplement choisir l'erreur L^2 locale sans rencontrer les problèmes du cas de la classification.

Definition 15. En régression, la mesure d'impureté $i_R(t)$ issue de l'estimation locale de l'erreur L^2 est définie par :

$$i_R(t) = \frac{1}{N_t} \sum_{x,y \in \mathcal{L}_t} (y - \hat{y}_t)^2$$

Cela revient donc à minimiser la variance de chacun des noeuds. Il est intéressant de remarquer que le problème de classification binaire avec critère de Gini est en réalité strictement équivalent à un problème de régression avec erreur L^2. En effet puisque $\hat{y}_t = \frac{1}{N_t} \sum_{x,y \in \mathcal{L}_t} y = p(c_2 \mid t) = 1 - p(c_1 \mid t)$ on a :

$$\begin{aligned} i_R(t) &= \frac{1}{N_t} \sum_{x,y \in \mathcal{L}_t} (y - \hat{y}_t)^2 \\ &= \hat{y}_t - 2\hat{y}_t^2 + \hat{y}_t^2 \\ &= \hat{y}_t(1 - \hat{y}_t) \\ &= p(c_2 \mid t)(1 - p(c_2 \mid t)) \\ &= \frac{1}{2} i_G(t) \end{aligned}$$

4.1.3 Sélection du point de coupe

Familles $\mathcal{Q}$ de règles de partition

Comme vu précédemment une coupure s du nœud t est une partition disjointe de $\mathcal{X}_t$ où chaque partie disjointe correspond à l'un des fils de t. Dans le cas où s partitionne en deux parties disjointes on appelle s une coupure *binaire* de fils t_L et t_R, cependant dans le cas plus général, s peut partitionner $\mathcal{X}_t$ en N avec tout autant de fils. L'ensemble S de toutes les coupures s est l'ensemble de toutes les partitions disjointes de $\mathcal{X}_t$, qui est de cardinal infini dès que les variables peuvent prendre un nombre infini de valeurs, néanmoins il n'est utile de considérer que les partitions qui engendrent une partition de $\mathcal{L}_t$ de parties non vides. Ainsi dans le cadre de l'apprentissage, le problème de trouver la meilleure coupure s^* de $\mathcal{X}_t$ se résume à trouver la meilleure partition du sous-échantillon d'apprentissage $\mathcal{L}_t$. Si l'on considère que les N_t échantillons d'apprentissage sont distincts, le nombre de partitions de $\mathcal{L}_t$ en k sous-ensembles non vides est donné par Knuth (1992) :

$$S(N_t, k) = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} k^{N_t}$$

qui vaut $2^{N_t - 1} - 1$ pour une partition binaire. Il est clair que tester naïvement toutes les coupures possibles et ne garder que la meilleure est impossible. Nous allons donc effectuer des hypothèses limitantes sur s^* c'est-à-dire chercher s^* dans une famille $\mathcal{Q} \subseteq S$ de coupures possibles qui donne toujours une approximation suffisamment bonne de

la partition optimale. Le choix habituel naturel pour $\mathcal{Q}$ est la famille des coupures binaires définies sur une variable unique et sans sous-ensembles vides :

$$\mathcal{Q} = \left\{ s \mid \bigcup_{j=1}^{p} \mathcal{Q}(X_j), \mathcal{L}_{t_L} \neq \emptyset, \mathcal{L}_{t_R} \neq \emptyset \right\}$$

- Si X_j est une variable ordonnée à valeurs dans $\mathcal{X}_j$ alors l'ensemble des coupures binaires sur X_j est :

$$\mathcal{Q}(X_j) = \left\{ \left(\left\{ x \mid x_j \leq v \right\}, \left\{ x \mid x_j \geq v \right\} \right) \mid v \in \mathcal{X}_j \right\}$$

  Il s'agit géométriquement de partitions de $\mathcal{X}$ à l'aide d'hyperplans parallèles aux axes.

- Si X_j est une variable catégorielle non ordonnée à valeurs dans $\mathcal{X}_j = \{b_1, \dots, b_L\}$ alors on prend pour $\mathcal{Q}(X_j)$:

$$\mathcal{Q}(X_j) = \left\{ \left(\left\{ x \mid x_j \in \mathcal{B} \right\}, \left\{ x \mid x_j \in \overline{\mathcal{B}} \right\} \right) \mid \mathcal{Q} \subset \{b_1, \dots, b_L\} \right\}$$

Notre problème de minimisation de l'*impureté* est alors :

$$\begin{cases} s^* = \underset{\substack{s_j^* \\ j=1,\dots,p}}{\operatorname{argmax}} \Delta i(s_j^*, t) \\ s_j^* = \underset{\substack{s \in \mathcal{Q}(X_j) \\ \mathcal{L}_{t_L}, \mathcal{L}_{t_R} \neq \emptyset}}{\operatorname{argmax}} \Delta i(s, t) \end{cases}$$

Bien que le type de partition donné précédemment soit le plus couramment utilisé et celui adopté par Breiman dans les forêts aléatoires classiques et leurs variantes, on notera qu'il n'est pas nécessaire de se restreindre à des hyperplans parallèles aux axes et alors opter pour des coupures dites obliques. On peut même éventuellement utiliser des surfaces de décision non linéaires ou lever la restriction sur le caractère disjoint de la partition.

Recherche de la coupure binaire optimale

Maintenant que le problème d'optimisation est bien défini, il convient de trouver le point de coupure optimal. Il faut alors faire la distinction entre les variables ordonnées et catégorielles. Louppe (2014) propose un algorithme pour chacun des cas afin de simplifier la recherche du point de coupure optimal dans la majorité des cas.

Dans le cas d'une variable ordonnée, il remarque en effet qu'il est possible de calculer $\Delta i(s_j^{v'_{k+1}}, t)$ à partir de $\Delta i(s_j^{v'_k}, t)$ où $\{v'_k\}$ est la famille des points de coupure potentiels ordonnés que l'on choisit comme étant les réalisations dans le cas où $\mathcal{X}_j$ est discret et le point médian entre les réalisations dans le cas où $\mathcal{X}_j$ est continue.

Dans le cas d'une variable catégorielle et pour la classification binaire une approche naïve forcerait à évaluer $2^{L-1} - 1$ partitions, mais Breiman et al. (1984) prouvent le théorème suivant permettant de se ramener à $L - 1$ partitions :

Théorème 4.2. *Quitte à réordonner les X_j tel que :*

$$p(c_1 \mid t, X_j = b_{l_1}) \leq p(c_1 \mid t, X_j = b_{l_2}) \leq \ldots \leq p(c_1 \mid t, X_j = b_{l_L})$$

Si $i(t)$ est tel que dans le théorème 4.1 alors une des $L - 1$ sous-parties $\mathcal{B} = \{b_{l_1}, \ldots, b_{l_h}\}$, $h = 1, \ldots, L - 1$ définit une partition binaire de l'échantillon du nœud en :

$$\mathcal{L}^{\mathcal{B}}_{t_L} = \{(x, y) \mid (x, y) \in \mathcal{L}_t, x_j \in \mathcal{B}\}$$
$$\mathcal{L}^{\mathcal{B}}_{t_R} = \{(x, y) \mid (x, y) \in \mathcal{L}_t, x_j \in \bar{\mathcal{B}}\}$$

On traite alors X_j comme une variable ordonnée en remplaçant les b_l par $p(c_1 \mid t, X_j = b_l)$. Malheureusement le théorème 4.2 ne s'étend pas au cas de la classification multiclasse où la seule solution reste d'évaluer les $2^{L-1} - 1$ partitions ou, à défaut, un sous-ensemble aléatoire.

Coupures aléatoires et ECP

La recherche du point de coupure aléatoire est très chère et le gain apporté par un choix optimal dépend fortement de la régularité de la fonction que l'on cherche à approcher, des caractéristiques de la fonction d'impureté choisie et des caractéristiques de l'espace d'apprentissage $\mathcal{X}$. Au vu de la difficulté de s'assurer que couper de façon optimale apporte un quelconque gain, il peut sembler judicieux d'économiser du temps de calcul en coupant de manière aléatoire. Il est possible de réduire l'espace de recherche de deux façons :

- En ne considérant qu'un sous-ensemble choisi aléatoirement de variables à considérer. Il s'agit de la méthode Random Subspaces (RS) introduite par Ho (1998)
- En choisissant parmi un ensemble de points de coupures tirés aléatoirement

Ces techniques de randomisation sont d'une grande importance pour la construction des forêts aléatoires comme nous le verrons ensuite, on peut donc donner quelques exemples d'arbres aléatoires

EXTRA! (EXTRA!) Introduits par Geurts et al. (2006), ici à chaque nœud K variables sont choisies aléatoirement et K points de coupures sont tirés uniformément parmi l'ensemble des valeurs réalisées de chacune de ces variables, la coupure qui minimise l'impureté est alors choisie.

ARBRE PUREMENT ALÉATOIRE NON-ÉQUILIBRÉ (UBPRT) Introduit par BREIMAN (2001) pour ses caractéristiques théoriques, ici on choisit aléatoirement le nœud à partitionner, puis on choisit aléatoirement la variable de coupure et enfin le point de coupure est choisi aléatoirement parmi les réalisations.

ARBRE PARFAITEMENT ALÉATOIRE (PERT) Dans cette construction de CUTLER et ZHAO (2001) il n'y a plus de critère de réduction de l'impureté ; à chaque nœud 2 individus $x_{1,j}$ et $x_{2,j}$ sont tirés aléatoirement de façon à ce qu'ils soient d'une classe différente, puis une variable de coupure est tirée elle aussi aléatoirement. La dernière étape consiste à tirer uniformément sur $[x_{1,j}, x_{2,j}]$ un point de coupure. La procédure de construction de l'arbre s'arrête lorsque tous les nœuds actuels sont purs.

ARBRES UNIFORMÉMENT ALÉATOIRES Ici CISS (2013) tire $\lceil \beta d \rceil$ variables avec remise ($\beta > 0$) puis les points de coupures sont choisis uniformément sur le support de X_j. Le critère de maximisation de la réduction d'impureté est ici utilisé comme précédemment.

Comme nous le verrons plus tard ces arbres aléatoires sont importants pour la création de forêts aléatoires et présentent souvent de bonnes performances pratiques en évitant les éventuelles caractéristiques problématiques des données. Néanmoins ISHWARAN (2014) montre que, du moins dans certains cas, les coupures aléatoires sont moins bonnes que celles vues précédemment. Commençons tout d'abord par introduire la notion de variable de bruit et de propriété ECP.

Definition 16. Si X est la caractéristique à p dimensions et Y la variable à prédire on appelle *variable bruyante* une variable $X_j \subseteq \mathcal{X}$ si la loi conditionnelle de Y sachant X est indépendante de X_j. Elle est appelée *variable forte* dans le cas contraire.

Definition 17. Une règle de coupure a la propriété ECP si elle tend à couper aux bords pour les variables bruyantes. C'est-à-dire si $\hat{s}_N$ est le point de coupure optimal parmi les points de coupures possibles $x_1 \leq ... \leq x_N$ alors si X_j est bruyante $\hat{s}_N$ tend vers x_1 ou x_N.

La propriété ECP a longtemps été considérée comme indésirable, car ayant tendance à provoquer des arbres très profonds, mais selon Ishwaran celle-ci est en réalité bénéfique puisqu'en coupant aux bords pour les variables bruyantes on ne dilue pas l'échantillon pour des coupures inutiles et il sera possible de recouper sur des variables fortes plus profondément dans l'arbre.

On peut de plus calculer le point de coupure optimal en fonction de la fonction à estimer f ($Y = f(x) + \varepsilon$) dans le cadre d'une population de nœuds infinie.

Théorème 4.3. *Soit $\varphi(s) = \mathbb{P}[Y = 1 \mid X = s]$. Si $\varphi(s)$ est continu sur $t = [a, b]$ et $\mathbb{P}$ admet une densité continue positive sur t alors la coupure*

s optimale au sens de la réduction de l'impureté de Gini dans le cas de la classification binaire est solution de

$$2\varphi(s) = \int_a^s \varphi(x)\mathbb{P}_{t_L}(\mathrm{d}x) + \int_s^b \varphi(x)\mathbb{P}_{t_R}(\mathrm{d}x), a \leq s \leq b \tag{4.1}$$

Théorème 4.4. *Le processus de coupure basé sur l'impureté de Gini a la propriété ECP.*

4.1.4 Critère d'arrêt

Plus un arbre de décision est profond, plus son erreur d'apprentissage est faible, mais l'erreur d'apprentissage ne reflète pas forcément l'erreur de généralisation, c'est-à-dire l'erreur sur un échantillon de taille infinie distribué selon la vraie distribution des individus. En effet, un tel arbre a tendance à surapprendre (*overfitting*) et à capturer le bruit plus que l'information. Il est alors nécessaire de trouver un compromis entre sur et sous-apprentissage et donc entre un arbre trop peu profond (l'extrême étant un arbre de profondeur 1, aussi appelé *souche de décision*) et un arbre trop profond qui obtiendrait alors des performances parfaites sur l'échantillon d'apprentissage et très mauvaises sur l'échantillon test.

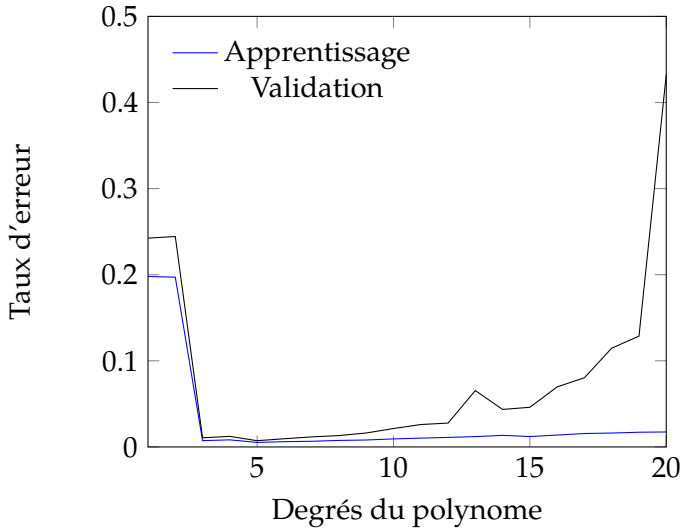

FIG. 4.3 : Sur-apprentissage dans le problème d'apprentissage de $sin(x)$ 2.2

Il est donc nécessaire d'avoir une règle d'arrêt de la procédure de coupe récursive. Notons tout d'abord que la procédure de partition possède deux critères d'arrêt inhérents à la procédure :

1. lorsque $\mathcal{L}_t$ est homogène, c'est-à-dire que tous les individus ont la même étiquette

2. lorsque les variables de tous les individus de $\mathcal{L}_t$ sont identiques

Pour empêcher le sur-apprentissage un certain nombre de critères d'arrêt heuristiques peuvent être ajoutés :

1. Fixer une taille $N_{\min}$ de l'échantillon pour la partition.
2. Fixer une profondeur $d_{\max}$ de l'arbre.
3. Fixer une réduction β minimum de l'impureté.
4. Fixer une taille N_{feuille} minimum pour chacun des fils en cas de coupure.

Toutes ces variables forment un jeu d'hyper-paramètres qu'il est possible d'optimiser pour obtenir les performances désirées.

4.1.5 Importance des variables

Bien que le but principal d'un algorithme de classification supervisée soit l'étiquetage de nouveaux individus en fonctions de leurs variables, une question naturelle et très importante dans de nombreux domaines comme la bio-informatique est la question de l'importance des variables. Mesurer l'importance des variables dans l'explication du facteur à prédire fournit ainsi une forme d'explication du mécanisme régissant ce facteur. Les arbres aléatoires sont facilement interprétables et il est donc possible d'établir plusieurs mesures de l'importance des variables.

Diminution de l'impureté

L'importance de X_j par diminution de l'impureté dans le cas d'un arbre seul est introduite par Breiman et al. (1984) comme la diminution totale de l'impureté si l'on effectuait les coupures uniquement sur X_j à chaque nœud en imitant la coupure réelle au mieux possible. C'est-à-dire

$$\text{Imp}(X_j) = \sum_{t \in \varphi} \Delta I(\tilde{s}_t^j, t)$$

ou $\tilde{s}_t^j$ est la coupure suppléante en t sur X_j, c'est à dire la coupure qui minimise la distance entre les distributions des classes dans les noeuds fils. On remplace toutes les coupures de l'arbre par leur coupure suppléante sur X_j de façon à minimiser les effets de masquage. En effet une variable peut être importante, mais toujours donner une diminution de l'impureté légèrement inférieure aux autres, cette variable ne sera alors jamais sélectionnée et apparaîtrait comme d'importance nulle.

Diminution de l'erreur

Il est également possible de mesurer l'importance d'une variable en mesurant sa contribution à la précision de la prédiction. Pour cela on

permute les valeurs des observations de la variable X_j d'intérêt, et l'on mesure la diminution de la précision.

$$\begin{aligned}\text{Imp}(X_j) = &\frac{1}{N} \sum_{(x'_i, y_i) \in \pi_j(\mathcal{L})} \mathbb{1}_{\varphi(x'_i) \neq y_i} \\ &- \frac{1}{N} \sum_{(x_i, y_i) \in \mathcal{L}} \mathbb{1}_{\varphi(x_i) \neq y_i}\end{aligned}$$

où $\pi_j(\mathcal{L})$ est $\mathcal{L}$ où l'on a permuté les valeurs de X_j. Si X_j est une variable prédictive alors on s'attend à ce que changer ses valeurs détériore le pouvoir prédictif de la forêt.

Profondeur minimale

Dans Ishwaran (2007), l'auteur introduit la notion de v sous-arbre maximal pour étudier les propriétés théoriques de l'importance des variables par diminution de l'impureté. Un v sous-arbre est un sous-arbre dont la racine définit une coupe selon la variable v, il est maximal s'il n'est pas lui même v sous-arbre d'un autre v sous-arbre. Dans Ishwaran, Kogalur, Gorodeski et al. (2010) les auteurs utilisent cette notion de sous arbre pour définir une nouvelle importance de la variable v comme étant la profondeur minimale des v sous-arbres de l'arbre. Il est alors possible de définir l'équivalent pour les forêts en prenant comme importance pour v la profondeur minimale moyenne des v sous-arbres dans la forêt.

Les variables importantes ayant une plus grande chance d'être retenues parmi les différentes variables considérées aléatoirement, on s'attend à ce que les variables importantes soient tirées tôt, les variables peu importantes n'étant tirées qu'en présence uniquement de variables moins importantes, ce qui devrait se produire peu souvent et donc n'arriver en moyenne que profondément dans l'arbre.

4.1.6 Effeuillage

Afin d'éviter le sur-apprentissage il est essentiel de limiter la complexité de l'arbre. En effet il est trivial en développant l'arbre jusqu'à avoir des feuilles ne comportant qu'un seul individu d'obtenir une erreur nulle sur l'échantillon d'apprentissage. Pour autant l'erreur sur l'échantillon de test explosera. On a précédemment introduit des critères d'arrêts qui permettent d'éviter ce sur-apprentissage durant la procédure de création de l'arbre mais il est également possible d'effectuer la réduction de la complexité une fois l'arbre entièrement construit.

On procède alors à un *effeuillage* de l'arbre, c'est à dire la suppression des feuilles considérées comme contribuant au sur-apprentissage.

COMPARAISON DE DIFFÉRENTES MÉTHODES
La procédure d'effeuillage n'est pas nécéssaire dans le cas des forêts aléatoires, nous n'exposerons donc ici que une seule méthode possible afin de donner une idée mais il existe d'autres méthodes possibles. Nous renvoyons à MINGERS (1989) pour une comparaison des méthodes d'effeuillage d'arbres de décisions seuls.

Puisque l'on cherche à trouver un compromis entre erreur et complexité, si l'on considère que le nombre de feuilles reflète la complexité d'un arbre alors BREIMAN et al. (1984) propose l'algorithme d'*effeuillage coût-complexité*. On définit par (φ_m) l'ensemble des sous-arbres de φ, ou un sous-arbre est un arbre obtenu par effeuillage successif de φ. On cherche alors :

$$\arg\min_{\varphi_m} \frac{\mathrm{Err}\left[\varphi_m, \mathcal{L}'\right] - \mathrm{Err}\left[\varphi, \mathcal{L}'\right]}{|\varphi_m|_F - |\varphi|_F}$$

avec $\mathcal{L}'$ un échantillon de test et $|\varphi_m|_F$ le nombre de feuilles de φ_m.

4.1.7 Consistance

Bien que de nombreuses variantes des arbres (et par extension forêts) aléatoires sont introduites et étudiées expérimentalement, la théorie permettant d'expliquer les performances est très faible. Le problème principal est la preuve de la consistance des arbres aléatoires de Breiman.

La consistance des arbres aléatoires est due à BIAU, DEVROYE et al. (2008) ; DEVROYE et al. (1997) et se base sur le théorème suivant :

Théorème 4.5 (DEVROYE et al. (1997), page 94, théorème 6.1)**.** *Soit une règle de partionnement telle que :*

$$\varphi_n(\mathbf{x}) = \begin{cases} 0 \text{ si } \sum_{i=1}^n \mathbb{1}_{y_i=1}\mathbb{1}_{\mathbf{x}_i \in \mathcal{X}^\varphi(\mathbf{x})} \leq \sum_{i=1}^n \mathbb{1}_{y_i=0}\mathbb{1}_{\mathbf{x}_i \in \mathcal{X}^\varphi(\mathbf{x})} \\ 1 \text{ sinon.} \end{cases}$$

Avec $\mathcal{X} = \bigcup \mathcal{X}^\varphi(x_i)$ la partition (finie) de l'espace définie par le classifieur. Notons aussi

$$N(\mathbf{x}) = \sum_{i=1}^n \mathbb{1}_{\mathbf{x}_i \in \mathcal{X}^\varphi(\mathbf{x})}$$

le nombre de voisins de x, et :

$$\mathrm{diam}(\mathcal{X}^\varphi) = \sup_{x,y \in \mathcal{X}^\varphi} \|x - y\|$$

Alors $\mathrm{Err}_n(\varphi_n) \to \mathrm{Err}(\varphi_B)$ *si*

$$\mathrm{diam}(\mathcal{X}^\varphi(\mathbf{x})) \xrightarrow{\mathbb{P}} 0$$
$$N(\mathbf{x}) \xrightarrow{\mathbb{P}} +\infty$$

4.2 FORÊTS ALÉATOIRES

On a vu dans le chapitre 3 comment la création d'ensembles de classifieurs à forte variance permet la construction d'un ensemble de classifieurs de meilleure qualité. Les arbres de classification sont un candidat idéal pour les méthodes ensemblistes grâce à leur capacité à modéliser des relations non linéaires complexes, leurs faibles biais et dans le cas des arbres non effeuillés leur instabilité très élevée. Nous avons également vu qu'il existait plusieurs méthodes pour construire de tels ensembles, néanmoins la plus grande partie des travaux sur les forêts aléatoires porte sur des variantes de la méthode proposée par Breiman (2001). Nous regrouperons alors toutes ces méthodes sous le terme de Forêts de Breiman.

Plus de littérature
Durant la rédaction de ce rapport un résumé de la littérature sur les forêts aléatoires par Biau et Scornet (2015) est apparu. Plusieurs autres méthodes sont abordées et les aspects théoriques en particulier font l'objet d'une grande attention. Le rapport technique de Criminisi et al. (2012) porte lui plus sur les aspects pratiques des forêts aléatoires, mais donne un aperçu intéressant des très nombreux problèmes pouvant être résolus par l'apprentissage machine en général et les forêts aléatoires en particulier.

4.2.1 Forêts de Breiman et dérivés

On a vu dans 3.2.1 que la combinaison linéaire de classifieurs faiblement corrélés permettait la création d'un nouveau classifieur plus puissant. Il est donc nécessaire de construire une famille d'arbres aléatoires les moins corrélés possible avant d'envisager la construction de l'ensemble en lui-même.

La première façon, dite des RS, d'introduire de l'aléa a été introduite par Ho (1998), elle consiste à chaque étape de création d'une coupure, à tirer aléatoirement un sous-ensemble de K variables de $\mathcal{X}$ puis à appliquer la procédure de choix de la coupe. Le bénéfice de cette technique est double : d'une part en introduisant de l'aléa les arbres sont en partie décorrélés, et d'autre part la réduction du nombre de coupures à considérer accélère grandement la construction de l'arbre. Néanmoins les arbres restent fortement corrélés et pour pallier ce problème Breiman (1996) introduit le Bagging. Un échantillon bootstrap de l'échantillon d'apprentissage est tiré avec remise et l'arbre est construit sur ce nouvel échantillon tiré aléatoirement. Ici encore, l'ajout d'aléa permet de décorréler les arbres et de réduire la complexité de construction des arbres d'environ 1/3 en moyenne. Une autre des propriétés importantes issues de cette technique est l'existence d'un échantillon Out-of-Bag

(OOB). En effet la procédure de tirage avec remise induit qu'en moyenne $1 - \left(1 - \frac{1}{n}\right)^n \approx 63\%$ des observations sont utilisées pour construire chaque arbre, il est donc possible de mesurer l'erreur OOB comme approximation de l'erreur de généralisation.

Definition 18 (Erreur OOB)**.** On appelle erreur OOB :

$$\widehat{\text{Err}}^{\text{OOB}}(\psi_{\mathcal{L}}) = \frac{1}{N} \sum_{(x_i, y_i) \in \mathcal{L}} \left(\frac{1}{M^{-i}} \sum_{l=1}^{M^{-i}} \mathbb{1}_{\varphi_{\mathcal{L}^{m_{k_l}}}(x_i) \neq y_i} \right)$$

ou $m_{k_1}, ..., m_{k_{M^{-i}}}$ sont les M^{-i} arbres construits à partir d'échantillons bootstrap ne contenant pas (x_i, y_i)

Cette approximation de l'erreur de généralisation est au moins aussi bonne que celle obtenue par k-validation croisée (Wolpert et Macready, 1999) tout en étant bien moins coûteuse numériquement.

La version de Breiman possède un bon compromis entre biais et variance et a montré de façon empirique d'excellents résultats sur un grand nombre de problèmes aussi bien artificiels que réels (Fernández-Delgado et al., 2014).

Le caractère guidé, c'est-à-dire la procédure d'optimisation du point du coupe, et le choix de la meilleure coupe parmi K coupes donne aux forêts aléatoires une propriété intéressante qui peut expliquer en partie les très bonnes performances observées, notamment dans le cas des grandes dimensions. En effet Breiman, 2004 montre que la convergence des forêts aléatoires ne dépend que des variables dites *fortes* et non pas les variables *faibles* bruitées. Il montre que la convergence pour une version simplifiée des forêts aléatoires vers l'erreur de Bayes est de l'ordre de

$$N^{-\frac{3}{4|S|+3}}$$

avec S les variables fortes définies telles que $\mathbb{E}[y \mid \mathbf{x}]$ ne dépend que de S. Ce résulat se retrouve pour d'autre types de forêts (Biau, 2010) et semble donc se généraliser.

Il existe néanmoins un certain nombre de jeux de données pour lesquels le caractère guidé de l'algorithme CART ne présente pas une réduction suffisante du biais, il peut alors être plus efficace de rajouter de l'aléa afin de bénéficier de la réduction de la variance. On peut donc citer deux modifications de l'algorithme de Breiman cherchant à ajouter de l'aléa au détriment du biais.

Forêts extrêmement aléatoires

Cette variante de l'algorithme original introduite dans Geurts et al. (2006) apporte deux modifications lors de la construction des arbres au niveau du critère de sélection et du choix de la coupure.

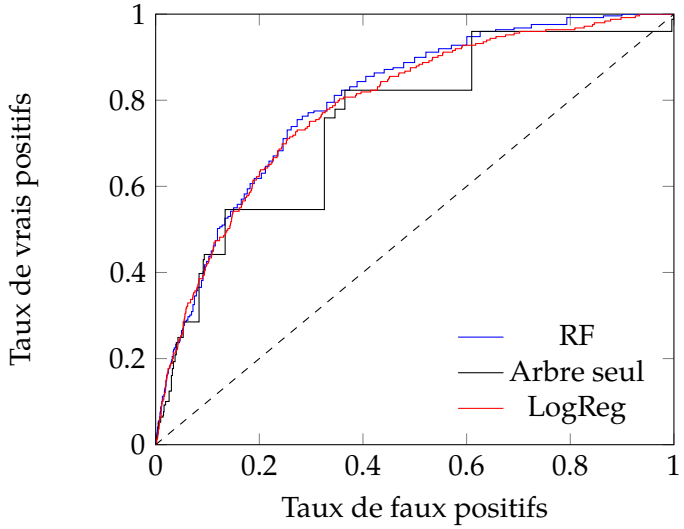

FIG. 4.4: Performances de RF (AUC = 0.805) par rapport à la régression logistique (AUC = 0.7889).

CHOIX DU POINT DE COUPE Ici, le point de coupe de la variable x_i est choisi aléatoirement au lieu de maximiser un critère de réduction d'impureté. Dans le cas ou x_i est numérique on tire un point de coupure aléatoirement sur $\left[\min_{\mathcal{L}_t} x_i, \max_{\mathcal{L}_t} x_i\right]$. Dans le cas d'une variable catégorique, on énumère $\mathcal{X}^i_{\mathcal{L}_t}$ le sous-ensemble des valeurs de x_i présentent dans $\mathcal{L}_t$, on tire aléatoirement une sous-partie $\mathcal{X}^i_1 \subset \mathcal{X}^i_{\mathcal{L}_t}$ et $\mathcal{X}^i_2 \subset \mathcal{X}^i \setminus \mathcal{X}^i_{\mathcal{L}_t}$, la coupure choisie est alors $\left(\mathcal{X}^i_1 \cup \mathcal{X}^i_2, (\mathcal{X}^i_1 \cup \mathcal{X}^i_2)^\top\right)$.

CRITÈRE DE SÉLECTION DU POINT DE COUPE Ici, l'impureté de Gini est remplacée par le gain d'information, on choisit comme score :

$$\text{Score}(s, \mathcal{L}_t) = \frac{2I^S_C(\mathcal{L}_t)}{H_S(\mathcal{L}_t) + H_c(\mathcal{L}_t)}$$

Forêts uniformément aléatoires

Cette variante principalement utile pour ses propriétés théoriques étudiées dans Ciss (2013) présente malgré des hypothèses très restrictives et des simplifications importantes de bonnes performances pratiques de par la forte décorrélation entre les différents arbres. Comme dans le cas des forêts extrêmement aléatoires, les changements dans la construction ont lieu au niveau du choix du point de coupe et du critère de sélection, ainsi que le choix des variables à envisager.

CHOIX DES VARIABLES DE NŒUDS On tire à chaque nœud $\lceil \beta M \rceil$ variables à considérer avec $\beta \in [0, 1]$.

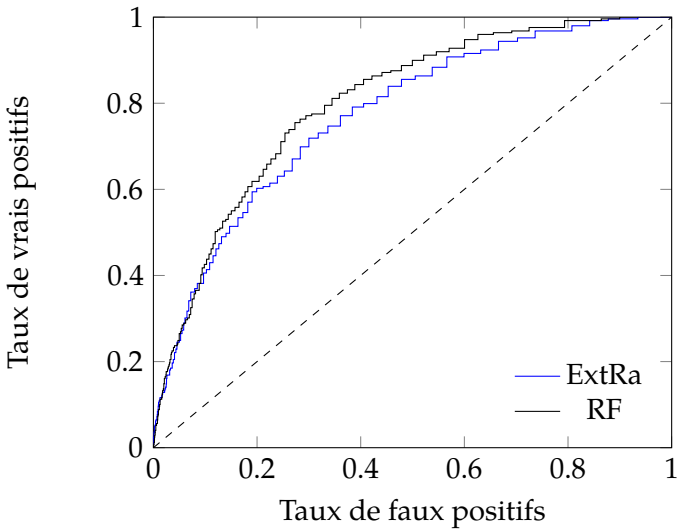

FIG. 4.5 : Performances de ExtRa-Trees (AUC = 0.7786).

CHOIX DU POINT DE COUPE On tire le point de coupe uniformément sur le support des observations ; dans le cas des variables catégorique celles-ci sont converties en variables numériques $1, 2, \dots, M$.

CRITÈRE DE SÉLECTION DU POINT DE COUPE Le critère à optimiser est ici le Gain d'information (Information Gain) (IG) défini par

$$\mathrm{IG}(Y, X) = \mathrm{H}(y) - [\mathrm{H}(Y \mid X \leq s) + \mathrm{H}(Y \mid X > s)]$$

avec H l'entropie de Shannon :

$$\mathrm{H}(Y) = \mathbb{E}\left[-\log\left(\mathbb{P}(Y = y)\right)\right]$$

Consistance des forêts aléatoires

La consistance des forêts aléatoires est un problème complexe. La légitimité de leur utilisation n'est pas affectée puisque si l'on fixe un nombre maximal de feuilles, les forêts aléatoires ont une VC-dimension finie et donc se généralisent bien, mais prouver la convergence vers l'erreur de Bayes ou la convergence vers la vraie distribution dans le cas où l'on estime des probabilités nécessite des techniques particulières.

De nombreux résultats sur la consistance de variantes très aléatoires ou indépendantes des données existent, mais la consistance du classifieur de Breiman n'a été établi que récemment par SCORNET et al. (2014) dans le cas de la régression (on rappelle que la classification binaire peut être vue comme un problème de régression) :

Théorème 4.6 (SCORNET et al., 2014). *Supposons que :*

(H1) $\quad Y = \sum_{j=1}^{p} m_j(x_j) + \varepsilon$

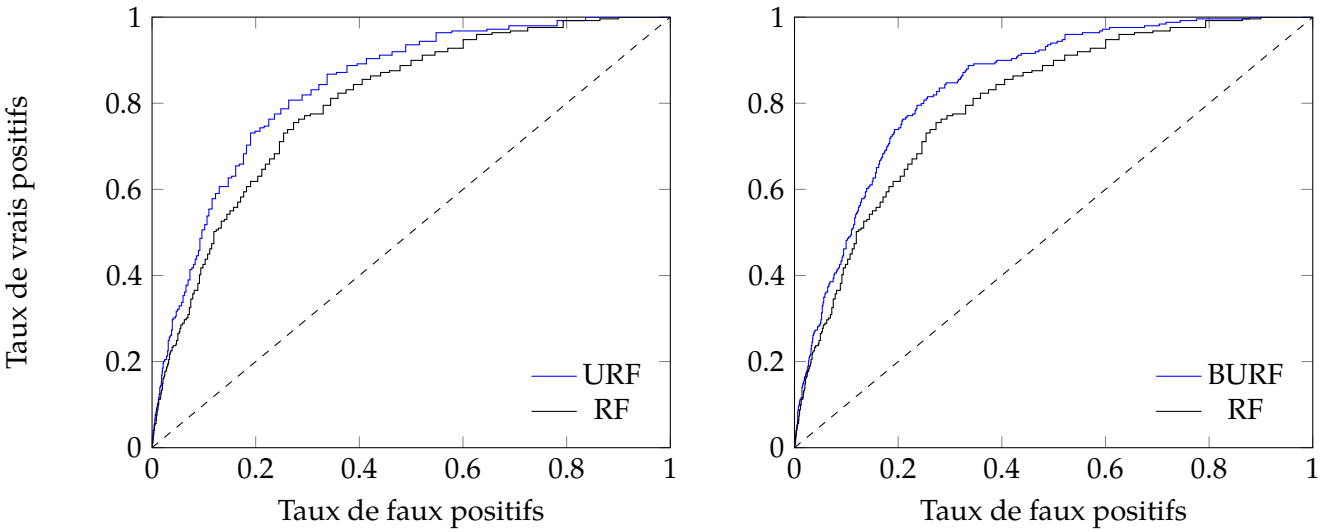

FIG. 4.6: Performances de URF (AUC = 0.8432) et URF Équilibré (AUC = 0.8452)

ou $\mathbf{x} = (x_1, \cdots, x_p)$ *suit une loi uniforme sur* $[0,1]^p$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ *indépendante et tel que* $0 < \sigma^2 < +\infty$, *et* m_j *continus. Supposons aussi* ***(H2)*** *: Il existe une suite* $(\gamma_n)_n \to 0$ *telle que* $\forall n \in \mathbb{N}^*$ *et* $1 \leq i, j \leq n$ *avec* $j \neq i$ *:*

$$\left| \mathbb{E}\left[Y_i - \mathbb{E}\left[Y \mid X = \mathbf{x}_i \right] \mid \mathbf{x}_i, \mathbf{x}_j, y_j, \mathbb{1}_{\mathbf{x} \in \mathcal{X}^{\varphi}(\mathbf{x}_i)}, \mathbb{1}_{\mathbf{x} \in \mathcal{X}^{\varphi'}(\mathbf{x}_j)} \right] \right| \leq \gamma_n \; p.s$$

avec φ' *une copie identique mais indépendante de* φ *et* $\mathbb{1}_{\mathbf{x} \in \mathcal{X}^{\varphi}(\mathbf{x}_i)}$ *l'indicatrice que* $\mathbf{x}$ *tombe dans la même feuille que* $\mathbf{x}_i$ *dans l'arbre* φ. *Supposons de plus qu'il existe une constante* $\sigma'^2 > 0$ *telle que* $\forall 1 \leq i \leq n$ *:*

$$\mathbb{E}\left[(Y_i - \mathbb{E}\left[Y \mid X = \mathbf{x}_i \right])^2 \mid \mathbf{x}_i, \mathbb{1}_{\mathbf{x} \in \mathcal{X}^{\varphi}(\mathbf{x}_i)} \right] \leq \sigma'^2 \; p.s$$

Alors sous ***(H1)*** *et* ***(H2)*** *on a que si :*

$$a_n \to +\infty \; et \; a_n \frac{\log n}{n} \to 0$$

avec a_n *le nombre d'individus tirés sans remise, alors la forêt aléatoire est consistante.*

On retrouve en général dans tous les théorèmes de consistance des forêts aléatoires le même type d'hypothèse pour la consistance :

$$k \xrightarrow[n \to +\infty]{} +\infty \text{ et } \frac{k}{n} \xrightarrow[n \to +\infty]{} 0$$

avec k le nombre d'individus dans chaque feuille. Cela se comprend assez bien : on cherche à avoir des feuilles représentant des hypercubes assez petits pour capturer l'ensemble de la distribution tout en ayant assez d'individus dans chaque hypercube pour que l'estimation locale de la probabilité soit elle-même convergente.

Scornet (2014) prouve également que le cas des forêts avec un nombre fini d'arbres présente les mêmes caractéristiques que les forêts infinies ce qui nous permet donc de considérer les forêts aléatoires comme une méthode d'apprentissage statistique avec des garanties théoriques suffisantes pour être valide dans la plupart des cas. (Biau, Devroye et al., 2008 comporte un contre-exemple de densité atomique dans lequel les forêts ne sont pas consistantes, il faut donc toujours être vigilant.)

4.2.2 Forêts obliques

Coupes obliques

Jusqu'à présent, tous les arbres ont utilisé des coupures parallèles aux axes de forme $\{X_i \leq s\}$, mais il n'est pas obligatoire de se restreindre à des coupures de ce type. Une extension naturelle à ces coupures selon les axes, c'est-à-dire selon un hyperplan formé sur une seule variable, est d'envisager les coupures *obliques* du plan, c'est-à-dire selon des hyperplans reposant sur une combinaison linéaire des variables.

Ainsi il est possible de modifier simplement l'algorithme des forêts classiques pour considérer les coupures multivariées ce qui est l'approche retenue par Menze et al. (2011). Ici de la même façon que dans la version standard à chaque nœud, seulement K variables sont considérées, mais au lieu de choisir la meilleure coupure sur une seule variable, une coupure oblique sur l'ensemble de ces K variables est considérée. Trois exemples sont considérés dans Menze et al. (2011) :

ORF-RND Cette version introduite par Breiman (2001) tire L variables au hasard parmi l'ensemble des variables et crée F combinaisons linéaires de coefficients tirés aléatoirement sur $[0, 1]$ de celles-ci. La meilleure coupure est sélectionnée parmi les F.

ORF-LDA Ici K variables sont tirées aléatoirement et une surface de décision est créée par Linear Discriminant Analysis (LDA)

ORF-RIDGE La surface de décision est créée par régression d'arête (régression linéaire avec régularisation de Tychonoff) sur K variables tirées aléatoirement.

Il est bien sûr possible d'utiliser d'autres algorithmes donnant un hyperplan de séparation, il faut toutefois veiller à la complexité d'un tel algorithme puisque la recherche d'un hyperplan optimal au sens de la mauvaise classification est un problème NP-difficile.

Barros et al. (2011) proposent un algorithme d'induction d'arbres obliques original puisqu'à l'inverse de tous les algorithmes présentés jusque-là, la procédure d'induction de l'arbre commence par les feuilles (*bottom-up*) au lieu de la racine (*top-down*). En commençant l'induction par les feuilles on se libère de la nécessité de posséder une mesure

de l'impureté puisque la distribution finale est connue a priori, l'algorithme *BUTIA* donné par les auteurs procède alors de la façon suivante :

1. Les individus sont réunis en K ensembles S_i purs avec K le nombre de classe.
2. L_i clusters sont construits par Expectation Maximization (EM) sur chaque S_i et les centroïdes de chaque cluster sont calculés.
3. Chaque cluster devient un nœud, les paires des nœuds de centroïdes les plus proches sont fusionnées si elles sont de classes différentes.
4. Le nouveau nœud se voit attribuer une nouvelle méta classe et un nouveau centroïde et l'hyperplan de séparation optimal est construit par SVM
5. Les procédures 1 à 4 sont répétées jusqu'à obtenir la racine.

Bien que les auteurs n'envisagent que le cas de la construction d'un arbre de classification simple, il est toujours possible de construire chaque arbre sur un échantillon boostrap des données et de ne considérer qu'un sous-ensemble k de variables pour la construction des hyperplans à chaque nœud.

Rotation de l'espace

L'un des principes de base des forêts de Breiman est la sélection d'un sous-espace de coupure à chaque nœud et nous avons vu précédemment qu'il est possible d'effectuer des coupes obliques comme combinaison linéaire des variables, il est en fait possible de combiner ces deux étapes en une seule et proposer une version plus générale des forêts aléatoires. Cet algorithme introduit par Tomita et al. (2015) utilise des projections de l'espace $\mathcal{X}$ sur un espace de dimension plus petit à chaque étape d'induction de nœuds. Une telle projection possède bien les deux caractéristiques précédentes recherchées : le nouvel espace est de dimension inférieure, une coupe parallèle aux axes dans le nouvel espace est une coupe oblique dans l'espace de base.

Dans l'esprit des forêts aléatoires, afin de garder des arbres faiblement corrélés et donc de bonnes performances de l'ensemble, les projections sont aléatoires. Cette idée est motivée par le théorème de Johnson-Lindenstrauss (Achlioptas, 2003) selon lequel un petit ensemble de points dans un espace de grande dimension peut être plongé par une projection orthogonale dans un espace de plus petite dimension tout en presque conservant la distance entre les points. C'est à dire :

Lemme 4.7 (Johnson-Lindenstrauss en distribution)**.** *$\forall \varepsilon > 0$, $\delta < 1/2$ et $d \in \mathbb{N}^+$ il existe une distribution μ sur $\mathcal{R}^{k\times d}$ avec $k = O(\varepsilon^{-2}\log(1/\delta))$ et $A \sim \mu$ tel que :*

$$\forall x, \|x\| = 1 : \mathbb{P}\left(|\|Ax\|_2^2 - 1| > \varepsilon\right) < \delta$$

Un choix possible pour A qui possède l'avantage d'accélérer grandement le produit matriciel est donné par Achlioptas (2003) :

$$A_{i,j} = \sqrt{3}\begin{cases} +1 \text{ avec probabilité } 1/6 \\ 0 \text{ avec probabilité } 2/3 \\ -1 \text{ avec probabilité } 1/6 \end{cases}$$

Néanmoins les avantages des forêts aléatoires standard étant la robustesse aux données aberrantes et l'invariance aux transformations monotones des variables, il semble naturel de chercher à conserver ces avantages tout en essayant d'obtenir l'invariance par rotation.

Une méthode pour obtenir l'invariance par rotation est la projection Principal Component Analysis (PCA) mais celle-ci est coûteuse en temps de calcul. Tomita et al. (2015) proposent donc d'utiliser des Projections Très Creuses (Li et al., 2006) : d valeurs dans $\{-1, 1\}$ sont tirées avec mêmes probabilités et insérées aléatoirement uniformément dans A, la matrice qui en résulte est très creuse, ce qui facilite les calculs, et presque invariante par rotation. Malheureusement, cette projection a pour inconvénient de faire perdre l'invariance par transformation monotone puisque la magnitude relative des variables joue maintenant lorsque celles-ci sont combinées. On peut alors observer que l'algorithme de coupure ne s'intéresse pas aux valeurs en elles-mêmes, mais à leur ordre relatif. Alors avant d'effectuer la projection, la matrice des observations de base est transformée en la matrice des rangs des observations.

Algorithme 4 Randomer Forest

procedure Randomer Forest($\mathcal{L}, \mu(\mathcal{L})$)
  pour chaque arbre n **faire**
    Création du sous-échantillon $\mathcal{L}_n$
    pour chaque nœud i de l'arbre **faire**
      $\tilde{X}_{i,n} \leftarrow A_{i,n}^{\top} X_n$ avec $A_{i,n} \sim \mu(\mathcal{L}_{i,n})$
      $s^*_{i,n} \leftarrow$ MeilleurCoupure($\tilde{X}_{i,n}$)
      Coupe $\mathcal{L}_{i,n}$ en $\mathcal{L}_{l_i,n}$ et $\mathcal{L}_{r_i,n}$ selon la règle de coupure $s^*_{i,n}$
    fin pour
  fin pour
  renvoyer $\left(\text{Arbres}_n, (A_{i,n})\right)$
fin procedure

4.2.3 Forêts bayésiennes

Nous avons vu dans la partie 3.2.2 qu'il était possible de combiner les prévisions d'un ensemble d'arbres dans un modèle bayésien. Chipman et al. (1998) proposent alors une méthode de combinaison bayésienne d'arbres CART, mais comme vu en 3.2.2, le BMA présente certains inconvénients qui impactent les performances de manière négative. Les

auteurs proposent alors (Chipman et al., 2008) la construction d'un modèle bayesien sous la forme d'une somme d'arbres, se rapprochant alors de l'esprit du BMC.

Les auteurs construisent se placent dans le modèle probit pour la classification. On cherche alors à modéliser

$$p(x) \equiv \mathbb{P}(Y = 1 \mid x) = \Phi(\varphi(x)) \tag{4.2}$$

où Φ est la fonction de distribution de la loi normale $\mathcal{N}(0,1)$ et

$$\phi(x) = \sum_{i=1}^{m} \varphi(x, T_i, M_i)$$

Ici T_i est l'arbre i, c'est-à-dire l'ensemble des variables et règles de coupures, et $M_i = (\mu_{1i}, \dots, \mu_{bi})$ le vecteur des étiquettes de ses feuilles et φ la fonction de prédiction d'un arbre.

Puisqu'il s'agit d'un modèle bayésien il convient de définir un a priori sur la distribution des arbres. Afin de faciliter les calculs, l'a priori est fixé sous une forme simple :

$$\mathbb{P}((T_1, M_1), \dots, (T_m, M_m)) = \prod_j \left((T_j) \prod_i (\mu_{ij} \mid T_j) \right)$$

A priori sur T_i

Les auteurs définissent l'a priori $\mathbb{P}(T_i)$ par :

$\alpha(1+d)^{-\beta}$ la probabilité qu'un nœud soit terminal.

$\mathcal{U}_V$, loi uniforme sur les variables, la probabilité de choisir une variable pour la coupure.

$\mathcal{U}_{[\min X_j, \max X_j]}$, loi uniforme sur les réalisations discrètes de X_j, la probabilité de tirer une coupure.

Le processus permettant de tirer un arbre est décrit dans Chipman et al. (1998). Les auteurs prennent soin de faire en sorte que le processus de Markov résultant soit réversible. Ainsi le processus de transition $q(T, T')$ qui à partir d'un arbre T donne un nouvel arbre T' tire l'une des 4 modifications suivantes :

$\mathbb{P}(\textsf{AJOUTE}) = 0.25$ Une feuille est tirée uniformément et une coupure est générée aléatoirement selon la loi a priori de coupure.

$\mathbb{P}(\textsf{RETIRE}) = 0.25$ Le père d'une feuille est tiré aléatoirement et ses deux fils supprimés.

$\mathbb{P}(\textsf{CHANGE}) = 0.4$ Un nœud est tiré aléatoirement et une nouvelle règle de coupure lui est attribué.

$\mathbb{P}(\textsf{PERMUTE}) = 0.1$ Un couple père-fils interne est tiré aléatoirement et leurs règles de coupure sont échangées. Si les deux fils avaient la même règle de coupure alors les deux fils ont leurs règles échangées à la fois.

Ayant ainsi défini un a priori sur les arbres et un processus pour générer des arbres il est possible d'utiliser l'algorithme de Metropolis-Hastings A.2 pour obtenir une suite d'arbres distribuée selon notre a posteriori.

A priori sur $\mu_{ij} \mid T_j$

Encore de façon à faciliter les calculs, les auteurs choisissent

$$\mu_{ij} \mid T_j \sim \mathcal{N}(\mu_\mu, \sigma_\mu^2)$$

Comme il en découle que l'a priori sur $\mathbb{E}[Y \mid \mathbf{x}]$ est $\mathcal{N}(m\mu_\mu, m\sigma_\mu^2)$, on choisit μ_μ et σ_μ^2 tels que :

$$m\mu_\mu - k\sqrt{m}\sigma_\mu = \Phi(-3) \tag{4.3}$$
$$m\mu_\mu + k\sqrt{m}\sigma_\mu = \Phi(3) \tag{4.4}$$

pour un k au choix.

Le modèle possède ainsi un très grand nombre de paramètres qui rendent une approche naïve impossible, les auteurs exploitent donc la forme particulière du problème pour mettre en place l'algorithme de *Bayesian Backfitting*, introduit dans T Hastie et Tibshirani (2000). Cet algorithme, tout comme celui de Metropolis-Hastings, repose sur la théorie des MCMC exposée dans l'annexe A. Afin de pouvoir appliquer le Bayesian backfitting, qui consiste à construire la somme progressivement sur les résidus. Dans le cas de la classification il n'y a pas de résidus explicites mais l'on peut réécrire le problème 4.2 en introduisant n variables latentes $Z_i \sim \mathcal{N}(\varphi(x))$ i.i.d telles que $Y_i = 1$ si $Z_i > 0$ et $Y_i = 0$ sinon.

En effet :

$$\begin{aligned}
\mathbb{P}(Y_i = 1 \mid x) &= \mathbb{P}(Z_i > 0 \mid x) \\
&= \mathbb{P}(N + \phi(x) > 0)) \quad \text{avec } N \sim \mathcal{N}(0,1) \\
&= \mathbb{P}(N > -\phi(x)) \\
&= 1 - \Phi(-\phi(x)) \\
\text{(par symétrie)} \quad &= \Phi(\phi(x))
\end{aligned}$$

On a de plus $Z_i \mid y_i = 1 \sim \max\{\mathcal{N}(g(x),1), 0\}$, et $Z_i \mid y_i = 0 \sim \min\{\mathcal{N}(g(x),1), 0\}$.

Le problème se ramène alors à un problème de régression et il est possible d'effectuer la procédure de *bayesian backfitting* sur les résidus de Z.

L'algorithme BART propose à chaque nouvelle étape un arbre obtenu comme modification locale ce qui réduit la capacité de la chaîne à explorer les zones de faible probabilité, la chaîne se concentrant autour des maximums locaux. Il est de plus nécessaire de recalculer les probabilités a priori et les vraisemblances à chaque étape ce qui est très coûteux. Pour tenter de pallier ces problèmes Lakshminarayanan, D. M. Roy et al. (2015) propose de remplacer l'algorithme de Gibbs / Metropolis-Hastings par une méthode de Monte-Carlo séquentielle, ou filtre particulaire. Cette modification n'est pas étudiée ici, mais pourrait permettre l'utilisation des forêts bayésiennes sur de grandes bases de données.

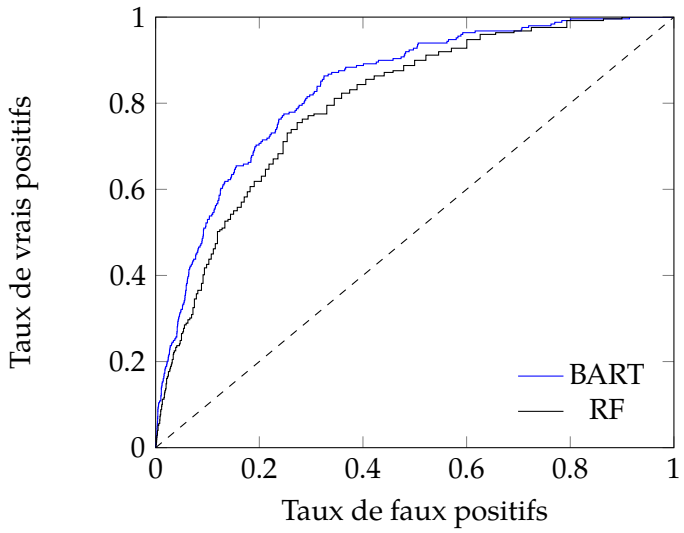

Fig. 4.7 : Performances de BART (AUC = 0.8365).

Notons également que dans l'algorithme BART, on considère la forêt en elle-même comme le processus de génération des données, mais il est également possible de décider d'un processus de génération des données mêmes et la forêt comme une fonction de ce processus. C'est par exemple le point de vue adopté par Taddy et al. (2015). Ici les auteurs construisent une forêt bayésienne « classique » comme agrégation d'arbres et non pas comme somme d'arbres. Il ne s'agit donc pas du tout du même modèle et les performances sont inférieures, mais cette formulation leur permet de définir une adaptation de leur algorithme pour les très grands jeux de données.

4.2.4 Amélioration globale de l'erreur des forêts d'arbres

La procédure standard de forêts aléatoires construit tous les arbres indépendamment sans prendre en compte les prédictions des autres arbres. L'algorithme minimise alors une fonction de perte arbre par arbre avant de les combiner alors qu'idéalement la perte serait minimisée sur la forêt entière.

Une façon de voir la construction d'un arbre aléatoire, et donc d'une forêt aléatoire, est de voir la procédure de construction de l'arbre comme accomplissant deux tâches distinctes :

1. L'apprentissage du « meilleur » partitionnement de l'espace c'est-à-dire l'apprentissage d'une famille d'indicatrices
2. L'apprentissage du meilleur étiquetage de chacune des feuilles c'est-à-dire l'apprentissage du meilleur coefficient devant chaque indicatrice.

Ainsi la procédure d'induction d'un arbre consiste à trouver une approximation

$$\hat{f}(x) = \sum_{k=1}^{K} \beta_k B_k(x) \tag{4.5}$$

avec K le nombre de feuilles de l'arbre. De la même façon pour une forêt composée de N arbres

$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K_i} \beta_k B_{i,k}(x) \tag{4.6}$$

La procédure d'induction des arbres de classification fournit à la fois le partitionnement et l'étiquetage, mais il est également possible de séparer les deux étapes d'apprentissage.

Ainsi Ren et al. (2015) proposent d'optimiser de façon globale la perte des forêts en séparant ces deux étapes d'apprentissage. Une fois les arbres construits il est possible de les interpréter comme des vecteurs indicatrices $\varphi : \mathcal{X} -> \mathbb{R}^K$, ainsi $\varphi(x)$ représente la feuille dans laquelle x tombe. Sous cette forme, on peut alors donner l'étiquetage sous la forme $y_i = \omega\varphi(x_i)$, et apprendre l'étiquetage revient à apprendre le vecteur des poids optimal. Soit pour un arbre seul :

$$\begin{aligned} &\text{minimiser} && \frac{1}{N} \sum_{i=1}^{N} l(y_i, \hat{y}_i) \\ &\text{tel que} && y_i = \omega\varphi(x_i), \forall i \in [1, N] \end{aligned}$$

Sous cette forme apprendre les feuilles d'une forêt est alors

$$\begin{aligned} &\text{minimiser} && \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} l(y_i^t, \hat{y}_i) \\ &\text{tel que} && y_i^t = \omega_t \varphi_t(x_i), \forall i \in [1, N], \forall t \in [1, T] \end{aligned}$$

Mais cette forme ne correspond pas à l'esprit des forêts aléatoires, on minimise ici la moyenne des pertes sur les arbres au lieu de minimiser la perte de la forêt, il est donc judicieux de modifier notre programme d'optimisation. Pour cela, introduisons le vecteur indicatrice de la forêt globale $\Phi(x) = [\varphi_1(x), \dots, \varphi_T(x)]$ et la matrice des poids des feuilles

$W = [\omega_1, \dots, \omega_T]$. Il est alors possible d'exprimer la prédiction y de la forêt sous la forme $y = W\Phi(x)$. Il est maintenant possible de minimiser la perte globale de la forêt, afin de se ramener à un problème déjà résolu. Ren et al. (2015) donne le problème d'apprentissage des feuilles sous la forme suivante :

$$\begin{aligned} &\text{minimiser} && \frac{1}{2}\|W\|_2^2 + \frac{C}{N}\sum_{i=1}^{N} l(y_i, \hat{y}_i) \\ &\text{tel que} && y_i = W\Phi(x_i), \forall i \in [1, N] \end{aligned} \tag{4.7}$$

La facilité de résolution de 4.7 vient du fait qu'il s'agit exactement d'un problème de SVM avec régularisation L^2 pour éviter l'overfitting, on choisit alors pour la perte l dans le cas de la classification la perte levier (en se ramenant à des classes $\{-1, 1\}$)

$$l(x, y) = \max(0, 1 - x \cdot y)$$

Une conséquence de l'optimisation globale des feuilles est qu'il devient alors possible, de la même façon qu'avec les arbres seuls, d'effeuiller la forêt de façon globale pour non seulement réduire de façon considérable la taille de la forêt, mais potentiellement en améliorer le pouvoir de généralisation. La procédure d'effeuillage se fait par étapes successives :

1. Calcul du vecteur W selon 4.7
2. La somme des normes l^2 de toutes les feuilles de même père est calculée
3. Une proportion α des paires les plus petites sont combinées et le vecteur indicatrice est mis à jour
4. Répéter la procédure jusqu'à atteindre le critère d'arrêt choisi

Apprentissage des feuilles par maximisation de l'AUC

AUC ET ERM

Nous avons jusqu'à présent, à travers différentes méthodes, cherché à minimiser une perte empirique, souvent l'erreur empirique. Toutes ces approches étaient naturellement justifiées par la théorie de Vapnik. Malheureusement l'AUC ne peut pas s'écrire comme la moyenne empirique d'une perte sur l'échantillon puisqu'il s'agit d'une statistique de l'ensemble tout entier et non individu par individu. Il est toutefois possible de se ramener au cadre ERM, nous renvoyons alors le lecteur vers Clémençon et al. (2006).

En considérant le problème sous cette forme, on peut également chercher à optimiser les étiquettes des feuilles pour optimiser le critère qui

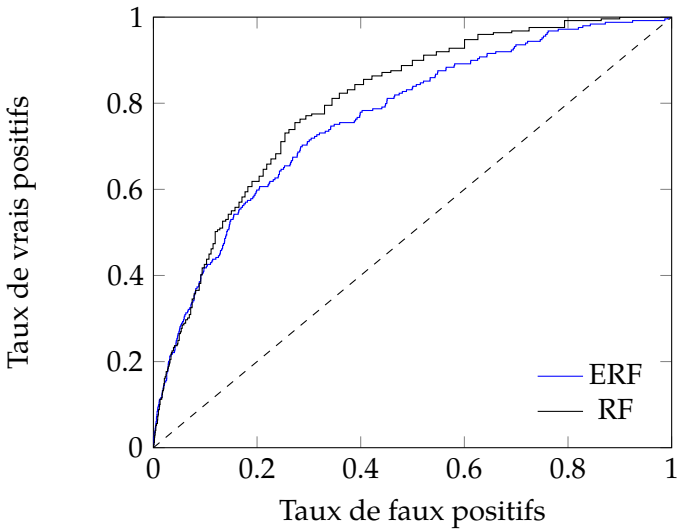

FIG. 4.8 : Performances de ERF (AUC = 0.7664).

nous intéresse. En effet, là où l'approche précédente cherche à maximiser la marge et réduire l'erreur de classification, il paraît plus intéressant de chercher à maximiser directement notre critère de performance : l'AUC.

Optimiser l'AUC présente deux inconvénients majeurs :

1. L'AUC est une statistique globale de la population et non individu par individu. La calculer nécessite un tri coûteux.
2. L'AUC n'est pas différentiable et se prête donc mal aux techniques d'optimisation.

Il faut donc non seulement trouver un substitut différentiable, mais aussi un substitut qui permette de faciliter les calculs pour rendre l'optimisation possible. Dans CALDERS et JAROSZEWICZ (2007), les auteurs proposent ainsi deux approximations polynomiales permettant non seulement d'obtenir un substitut différentiable, mais aussi de faire apparaître plusieurs statistiques constantes n'ayant pas besoin d'être mises à jour à chaque étape.

Notons $\mathcal{L} = \mathcal{L}_0 \cup \mathcal{L}_1$ avec $\mathcal{L}_0$ les individus négatifs et $\mathcal{L}_1$ ceux positifs. On a alors :

$$\mathrm{AUC}(\varphi, \mathcal{L}) = \frac{\sum_{k_0 \in \mathcal{L}_0} \sum_{k_1 \in \mathcal{L}_1} \mathbb{1}_{\varphi(x_0) < \varphi(x_1)}}{|\mathcal{L}_0||\mathcal{L}_1|}$$

La première approche consiste donc à approcher $\mathbb{1}$ par une fonction polynomiale, néanmoins il est aussi possible de choisir un substitut dérivable en changeant la fonction :

$$\mathrm{AUC}(\varphi, \mathcal{L}) = \frac{\sum_{k_0 \in \mathcal{L}_0} \sum_{k_1 \in \mathcal{L}_1} \mathrm{sigmoïde}_\beta\left(\varphi(x_1) - \varphi(x_0)\right)}{|\mathcal{L}_0||\mathcal{L}_1|}$$

$$\mathrm{sigmoïde}_\beta = \frac{1}{1 + e^{-\beta x}}$$

Ce substitut coûte plus cher à calculer, mais présente des avantages propres attrayants selon les auteurs puisqu'il mesure d'une certaine façon la marge en donnant un coût aux points très mal classés et reste une bonne approximation puisque $\text{sigmoïde}_\beta(x) \to H(x)$ lorsque $\beta \to \infty$. Les auteurs proposent donc de l'estimer par une fonction polynomiale. De façon plus générale, on veut donc approcher

$$\text{AUC}(\varphi, \mathcal{L}) = \frac{\sum_{k_0 \in \mathcal{L}_0} \sum_{k_1 \in \mathcal{L}_1} f\left(\varphi(x_1) - \varphi(x_0)\right)}{|\mathcal{L}_0||\mathcal{L}_1|}$$

En effet nous allons voir qu'utiliser une approximation polynomiale facilite grandement les calculs numériques. Soit $P(x) = \sum_k^d \lambda_k x^k$ une approximation polynomiale de la fonction f que l'on souhaite approcher (fonction de Heaviside ou sigmoïde). Alors

$$\begin{aligned} f\left(\varphi(x_1) - \varphi(x_0)\right) &\simeq \sum_{k=0}^{d} \lambda_k \left(\varphi(x_1) - \varphi(x_0)\right)^k \\ &\simeq \sum_{k=0}^{d} \lambda_k \sum_{l=0}^{k} \binom{k}{l} \varphi(x_1)^l (-\varphi(x_0))^{k-l} \\ &\simeq \sum_{k=0}^{d} \alpha_{kl} \sum_{l=0}^{k} \varphi(x_1)^l \varphi(x_0)^{k-l} \end{aligned}$$

avec $\alpha_{kl} = \lambda_k \binom{k}{l} (-1)^{k-l}$ et donc :

$$\text{AUC}(\varphi, \mathcal{L}) \simeq \frac{1}{n_0 n_1} \sum_{k=0}^{d} \alpha_{kl} \sum_{l=0}^{k} \left(\sum_{x_1 \in \mathcal{L}_1} \varphi(x_1)^l \right) \left(\sum_{x_1 \in \mathcal{L}_1} \varphi(x_0)^{k-l} \right)$$

Suivant les notations des auteurs on note

$$s(\varphi, \mathcal{L}) = \sum_{x \in \mathcal{L}} f(x)$$

En se replaçant dans le cadre précédent d'apprentissage des feuilles on peut alors effectuer une descente de gradient par rapport à $\mathbf{W}$, en notant AUC_p l'approximation polynomiale de AUC

$$\begin{aligned} \frac{\partial \text{AUC}_p(\varphi)}{\partial \mathbf{W}_i} = \frac{1}{n_0 n_1} \sum_{k=0}^{d} \sum_{l=0}^{k} \alpha_{kl} (l \cdot s(\Phi(x)_i \left(\mathbf{W}\Phi(x)\right)^{l-1}, \mathcal{L}_1) \cdot s((\mathbf{W}\Phi(x))^{k-l}, \mathcal{L}_0) \\ + (k-l) \cdot s((\mathbf{W}\Phi(x))^l, \mathcal{L}_1) \cdot s(\Phi(x)_i \left(\mathbf{W}\Phi(x)\right)^{k-l-1}, \mathcal{L}_0)) \end{aligned}$$

On peut donc maintenant effectuer la descente de gradient $\mathbf{W}' \leftarrow \mathbf{W} + \gamma \mathbf{g}$, avec $\mathbf{g} = \nabla \text{AUC}$. Nous utilisons alors le fait que l'AUC est indépendant de la norme de $\mathbf{W}$, donc au lieu de chercher γ optimal on peut se contenter de chercher $\mathbf{W}' \leftarrow \cos(\alpha)\mathbf{W} + \sin(\alpha)\nabla\mathbf{g}$. Les auteurs

proposent alors une méthode astucieuse exploitant cette forme particulière pour éviter les calculs coûteux.

$$\mathrm{AUC}(\varphi_{\mathbf{W}'}, \mathcal{L}) = \frac{1}{n_0 n_1} \sum_{k=0}^{d} \sum_{l=0}^{k} \alpha_{kl} \left(\sum_{x \in \mathcal{L}_1} \left(\frac{\cos(\alpha)}{A} \mathbf{W}\Phi(x) + \frac{\sin(\alpha)}{A} \mathbf{g}\Phi(x) \right)^{l} \right) \left(\sum_{x \in \mathcal{L}_0} \left(\frac{\cos(\alpha)}{A} \mathbf{W}\Phi(x) + \frac{\sin(\alpha)}{A} \mathbf{g}\Phi(x) \right)^{k-l} \right)$$

où A est un coefficient de renormalisation permettant de rester dans la zone de bonne approximation de notre polynôme. En développant les deux sommes de la même façon que précédemment on retrouve une expression en fonction des fonctions s qui peut être calculée en une seule passe.

4.2.5 Forêts en ligne

Jusqu'à présent toutes les méthodes présentées possédaient un point commun : l'algorithme prend tout $\mathcal{L}$ en entrée et fournit un classifieur en sortie. Il se peut que tous les individus ne soient pas observables immédiatement ou que $\mathcal{L}$ soit trop grand pour permettre l'utilisation de l'algorithme. Dans ce cas, il serait alors préférable de pouvoir mettre à jour le classifieur de façon incrémentale au fur et à mesure que l'on observe les individus. Ce type d'algorithme, appelé *online* et se différenciant des algorithmes *batch*, est d'une importance capitale dès lors que l'apprentissage doit se faire en temps réel par exemple ou dans un environnement aux performances restreintes.

Les forêts aléatoires étant largement utilisées pour de nombreux problèmes, il paraît naturel de tenter de les étendre au problème de l'apprentissage en ligne.

Nous n'exposerons pas ici les différentes méthodes existantes et renvoyons par exemple à Saffari et al. (2009) et Denil et al. (2013) qui proposent des modifications légères de l'algorithme de base où un certain nombre de statistiques descriptives de l'arbre et des individus déjà rencontrés sont gardés en mémoire. À chaque nouvel individu, ces statistiques sont mises à jour et la structure de l'arbre adaptée en conséquence. Lakshminarayanan, D. Roy et al. (2014) définissent une distribution sur les arbres construits sur $N + 1$ dont la loi conditionnelle par rapport a l'arbre construit sur N individus est explicite. Cela permet donc à chaque nouvelle observation de générer une nouvelle forêt aléatoire.

Ces algorithmes n'ont pas étés implémentés et testés car leurs performances sont toujours au mieux aussi bonne que la version de base. Leur intérêt est néanmoins évident dans le cadre des bases de données massives.

5 DONNÉES DÉSÉQUILIBRÉES

Le principal obstacle dans la classification en scoring est le déséquilibre souvent très important entre la classe d'intérêt et la classe majoritaire. Ce déséquilibre a pour conséquence de biaiser la performance du classifieur vers la classe majoritaire au détriment de la classe minoritaire puisqu'il n'est pas rare d'observer des ratios de 99% contre 1% dans le milieu du credit scoring par exemple. Il est donc primordial de mettre en place des méthodes visant à améliorer les performances réelles du classifieur en présence de classes déséquilibrées.

5.1 SOUS-ÉCHANTILLONNAGE

Une première méthode possible pour réduire le déséquilibre entre les classes est tout simplement de supprimer un certain nombre d'observations de la classe majoritaire pour se ramener à une distribution plus équilibrée. Une simple suppression aléatoire de données majoritaires est possible, mais présente l'inconvénient extrêmement pénalisant d'ignorer la majorité des données lors de la construction du modèle. Cette perte d'information peut conduire à une diminution des performances qui serait assez importante pour ne pas contrebalancer le gain amené par le rééquilibrage. Il convient donc d'introduire des méthodes plus intelligentes qui évitent de supprimer l'information utile.

5.1.1 Sous-échantillonnage informé

Definition 19 (Liens de Tomek)**.** Soit x et y deux individus de classes différentes et soit $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ une distance sur les individus. (x, y) est appelé lien de Tomek si $\nexists z, d(x, z) < d(x, y)$ ou $d(y, z) < d(x, y)$. Autrement dit si x et y sont plus proches voisins.

Il est alors possible de sous-échantillonner en supprimant tous les individus de la classe majoritaire membre d'un lien de Tomek.

Definition 20 (Condensed Nearest Neighbor (CNN), Hart, 1968 ; Kubat et Matwin, 1997)**.** Le but est la construction d'un sous-ensemble consistant au sens de Hart (1968). Tout d'abord un ensemble $\mathcal{L}'$ contenant tous les individus de la classe minoritaire et un individu de la classe majoritaire tiré au hasard est construit, puis la règle de classification 1-NN est utilisée pour classifier tous les individus. Chaque individu mal classé est ajouté à $\mathcal{L}'$ et la procédure est répétée jusqu'à ce qu'il n'y ait plus d'individus mal classés.

Definition 21 (Edited Nearest Neighbor (ENN), Laurikkala, 2001 ; D. L. Wilson, 1972)**.** Pour chaque $\mathbf{x}_i$ ses trois plus proches voisins sont recherchés. Si $\mathbf{x}_i$ appartient à la classe majoritaire et est mal classé par ses voisins alors celui-ci est supprimé. S'il appartient à la classe minoritaire et est mal classé, alors ses trois voisins sont retirés.

Definition 22 (One-sided selection (OSS),Kubat et Matwin, 1997)**.** La méthode OSS est la combinaison des liens de Tomek suivi de ENN. Ainsi les individus considérés comme *bruyants* sont retirés par l'élimination des liens de Tomek puis les individus à la frontière par ENN. Il est également possible d'effectuer l'opération dans l'autre sens tel que proposé par Batista et al. (2004).

Ces méthodes de rééquilibrage procèdent à une forme de régularisation a priori de la surface de séparation en supprimant les exemples les plus à même de grandement modifier la surface. On comprend pourquoi il est légitime d'espérer une amélioration de l'erreur de généralisation.

Les résultats de Chen et al. (2004) ; He et Garcia (2009) semblent indiquer un fort potentiel de ces techniques pour les données purement théoriques mais toutes ces méthodes utilisent à une étape la distance entre les individus. Une telle distance est difficile à définir dans le cas des données hétérogènes et diminue fortement les performances que l'on obtient en pratique. Nous introduisons dans 6.1 des distances adaptées à nos données pour essayer de résoudre ce problème.

5.1.2 Balanced et Roughly Balanced Random Forest

Le sous-échantillonnage naïf provoque une perte d'information trop importante, mais les forêts aléatoires présentent déjà une procédure de sous-échantillonnage à travers le Bagging ; il est donc possible d'adapter le Bagging afin de rétablir l'équilibre des classes.

La première méthode proposée par Breiman (2001) est la méthode Balanced Random Forest (BRF) qui consiste à effectuer le tirage bootstrap sur l'échantillon d'observations de la classe minoritaire et celui de la classe majoritaire de manière indépendante. Ainsi si l'on note $\mathcal{L}_m$ les individus de la classe minoritaire et $\mathcal{L}_M$ ceux de la classe majoritaire on construit l'échantillon bootstrap pour chaque arbre en tirant $|\mathcal{L}_m|$ individus de $\mathcal{L}_m$ avec remise et le même nombre d'individus de $\mathcal{L}_M$. Ainsi chaque arbre est construit sur un échantillon de taille $2|\mathcal{L}_m|$ très réduit, mais parfaitement équilibré. Contrairement au sous-échantillonnage naïf où la perte d'information est dommageable ici le sous-échantillonnage a lieu à chaque arbre, mais tout l'échantillon est considéré pour la forêt. Ainsi, quitte à augmenter le nombre d'arbres, il n'y a pas de perte d'information.

Une version modifiée de cet algorithme est l'algorithme Roughly Balanced Random Forest (RBRF) introduit dans Hido et al. (2009). Ici la

procédure de Bagging est répliquée en ajoutant un a priori sur la distribution des classes afin éventuellement de mieux capturer les avantages du Bagging pour les forêts aléatoires. On tire toujours $|\mathcal{L}_m|$ individus bootstrap de $\mathcal{L}_m$ mais le nombre d'individus tirés dans $\mathcal{L}_M$ est aléatoire. On tire alors $\text{BN}(|\mathcal{L}_m|, 0.5)$ individus de $\mathcal{L}_M$ où $\text{BN}(n, q)$ est la loi binomiale négative avec n le nombre d'échecs et q la probabilité d'échec.

En effet la procédure de bootstrap tire chaque point $(x, y) \in \mathcal{L}$ avec probabilité $p(x, y)$. En utilisant la formule de Bayes on a $p(x, y) = p(y)p(x \mid y)$. Sous cette décomposition, tirer un échantillon bootstrap revient à d'abord tirer n_y sous la loi binomiale $\text{B}(|\mathcal{L}|, p(y))$ puis n_y individus de la classe y sont tirés uniformément. Il est alors possible de choisir l'a priori $p(y) = 0.5$ pour effecteur le tirage. Cela revient donc à chaque tirage, à sélectionner avec probabilité $p(y) = 0.5$ la classe puis à choisir uniformément un individu au sein de celle-ci. Puisque le nombre d'individus minoritaires est faible, on ajoute une seconde restriction en arrêtant le tirage une fois que $|\mathcal{L}_m|$ individus de la classe minoritaire ont été tirés, ce qui revient à choisir le nombre d'individus à tirer selon une loi binomiale négative.

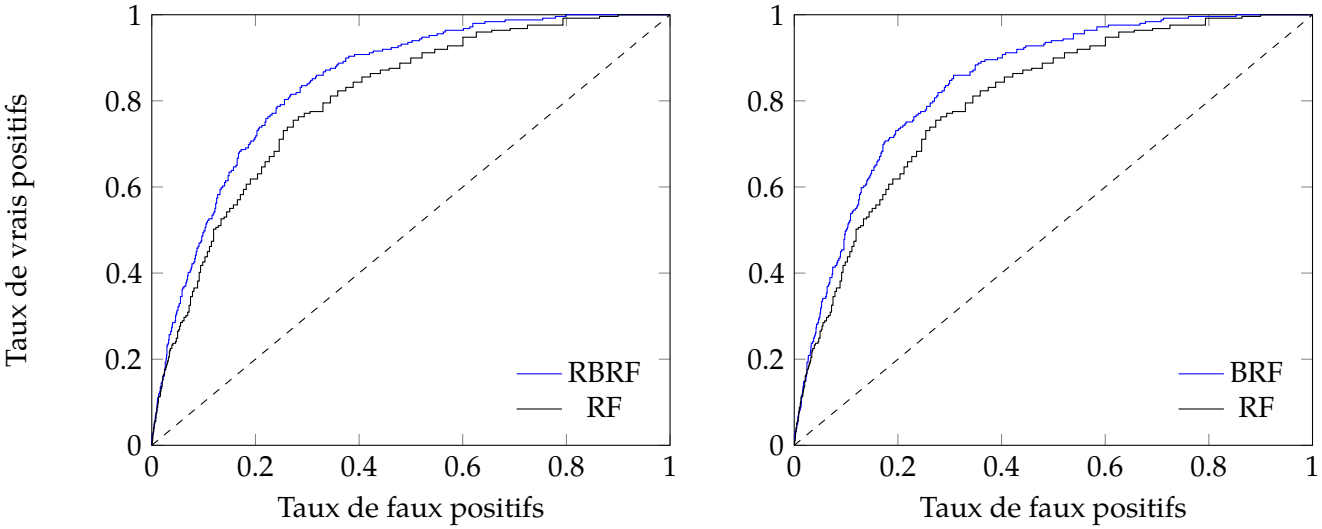

FIG. 5.1 : Performances de BRF (AUC = 0.8432) et RBRF (AUC = 0.8452)

Ces deux modifications du Bagging cherchent, au moins en moyenne, à égaliser les classes. Néanmoins DUPRET et KODA (2001) montrent qu'un rééquilibrage parfait n'est pas optimal, nous introduisons alors deux variantes des algorithmes précédents.

Si q_a est la proportion d'individus appartenant à la classe a dans l'échantillon d'apprentissage alors on fixe α comme proportion d'individus de la classe a tirés durant le bagging :

$$\alpha = \frac{1}{2} + \frac{2q_a - 1}{4}$$

On introduit alors l'algorithme Nearly Balanced Random Forest (NBRF) comme modification de BRF :

Algorithme 5 Algorithme NBRF pour les forêts aléatoires

procedure NBRF($\mathcal{L}$, règle arrêt, M)
  $q_a \leftarrow \frac{|\mathcal{L}_m|}{|\mathcal{L}|}$
  $\alpha \leftarrow \frac{1}{2} + \frac{1-2q_a}{4}$
  pour chaque arbre $i < M$ **faire**
    $\mathcal{L}_n \leftarrow |\mathcal{L}_m|$ individus minoritaires et $|\alpha\mathcal{L}_m|$ majoritaires
    pour chaque nœud j et règle d'arrêt non remplie **faire**
      K variables tirées aléatoirements
      Choix de la meilleur coupure
    fin pour
  fin pour
  renvoyer (Arbres_i)
fin procedure

Afin de conserver l'esprit de l'algorithme RBRF on introduit Very Roughly Balanced Random Forest (VRBRF)

Algorithme 6 Algorithme VRBRF pour les forêts aléatoires

procedure VRBRF($\mathcal{L}$, règle arrêt, M)
  $q_a \leftarrow \frac{|\mathcal{L}_m|}{|\mathcal{L}|}$
  $\alpha \leftarrow \frac{1}{2} + \frac{1-2q_a}{4}$
  pour chaque arbre $i < M$ **faire**
    $\mathcal{L}_n \leftarrow |\mathcal{L}_m|$ minoritaires et $\text{BN}(\alpha|\mathcal{L}_n|, 0.5)$ majoritaires
    pour chaque nœud j et règle d'arrêt non remplie **faire**
      Tirer K variables aléatoirements
      Choisir la meilleure coupure
    fin pour
  fin pour
  renvoyer (Arbres_i)
fin procedure

5.2 SUR-ÉCHANTILLONNAGE

De la même façon qu'il est possible de rétablir l'équilibre en diminuant la proportion d'individus de la classe majoritaire, il est possible d'augmenter le nombre d'individus de la classe minoritaire. La méthode naïve consiste alors à tout simplement dupliquer les individus minoritaires, mais le risque de sur-apprentissage augmente alors en conséquence. On peut alors pour diminuer ce risque introduire des méthodes de sur-échantillonnage dites *synthétiques*.

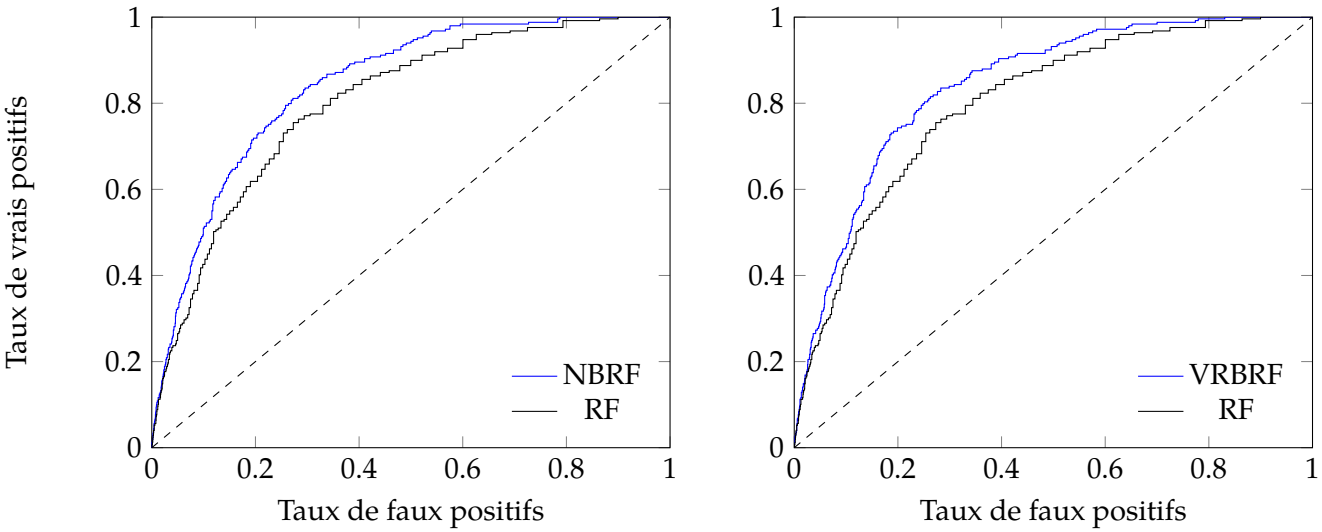

FIG. 5.2 : Performances de NBRF (AUC = 0.8427) et VRBRF (AUC = 0.8421)

5.2.1 SMOTE

Le principe de SMOTE tel qu'introduit par CHAWLA et al. (2002) est simple : augmenter le nombre d'observations minoritaires sans duplication tout en restant dans la zone de l'espace où l'on s'attendrait à trouver de telles observations. Pour cela, ils créent des observations synthétiques en extrapolant entre des observations minoritaires proches déjà présentes. Les observations considérées pour l'extrapolation sont choisies par k-plus proches voisins (k-NN) et tout simplement linéairement interpolées.

Cette méthode est très simple d'application dans le cas où les variables sont homogènes numériques mais ne peut pas s'appliquer telle quelle à un échantillon hétérogène puisque les notions de distance et d'interpolation ne sont pas bien définies. Le cas de l'interpolation est simplement résolu en fixant un seuil et en décidant par exemple que si x_i est une variable catégorielle alors le nouvel individu synthétique $\mathbf{x}_s$ interpolé à partir de $\mathbf{x}_1$ et $\mathbf{x}_2$ prend comme valeur pour $\mathbf{x}_{s,i}$, $\mathbf{x}_1$ si le coefficient d'interpolation est inférieur à 0.5, $\mathbf{x}_2$ sinon.

Borderline-SMOTE

HAN et al. (2005) proposent une modification de SMOTE cherchant à concentrer la création d'individus synthétiques aux zones *frontières* puisque ce sont ceux-ci qui majoritairement détermineront la surface de décision. La modification par rapport à SMOTE est alors de rajouter une étape permettant de décider si un individu minoritaire est à la frontière ou non.

1. Pour chaque $\mathbf{x}_i$ de la classe minoritaire on cherche les k plus proches voisins, on note k_M le nombre de ces voisins appartenant à la classe majoritaire.

2. Si $k_M = k$ alors l'individu est considéré comme du bruit et est sauté, si $k/2 \leq k_M \leq k$ alors $\mathbf{x}_i$ est considéré comme étant à la frontière F.

3. Pour chaque individu de F, on construit des individus synthétiques à partir de ses k plus proches voisins dans $\mathcal{L}_1$ en utilisant un coefficient d'interpolation aléatoire compris entre 0 et 1.

Les auteurs proposent une autre variante qui, au lieu de ne considérer que les voisins appartenant à $\mathcal{L}_1$, considère tout $\mathcal{L}$. Dans ce cas, si l'interpolation entre $\mathbf{x} \in \mathcal{L}_1$ et $\mathbf{x}' \in \mathcal{L}_0$ se fait à l'aide d'un coefficient aléatoire compris entre 0 et 1/2.

ADA-SYN

Dans cette variante proposée par He, Bai et al. (2008), le nombre d'individus synthétiques à créer pour chaque individu de la classe majoritaire est pondéré par $\frac{k_M^i}{\sum_{i=1}^{|\mathcal{L}_1|} k_M^i}$ où k_M^i est le nombre d'individus de la classe majoritaire présent parmi les k plus proches voisins de $\mathbf{x}_i \in \mathcal{L}_1$

5.2.2 Distances hétérogènes

Dans le cas des données hétérogènes, il est nécéssaire d'introduire de nouvelles métriques. Il en existe de nombreuses (Lumijärvi et al., 2004 ; Rodriguez et al., 2008 ; D. R. Wilson et Martinez, 1997) mais nous ne donnons ici que deux exemples simples.

Definition 23 (Distance Heterogeneous Euclidean-Overlap Metric (HEOM))**.** On appelle HEOM la distance définie par $\text{HEOM}(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{j=1}^{p} \text{d}_{X_j}(\mathbf{x}_j, \mathbf{x}'_j)}$ avec :

$$\text{d}_{X_j}(x_j, x'_j) = \begin{cases} \text{d}_Q(x_j, x'_j) \text{ si } X_j \text{ qualitative} \\ \text{d}_N(x_j, x'_j) \text{ si } X_j \text{ quantitative} \end{cases}$$

$$\text{d}_Q(x_j, x'_j) = \begin{cases} 1 \text{ si } x_j = x'_j \\ 0 \text{ sinon} \end{cases}$$

$$\text{d}_N(x_j, x'_j) = \frac{|x_j - x'_j|}{\max X_j - \min X_j}$$

Definition 24 (Distance Heterogeneous Value Difference Metric (HVDM))**.** On appelle HVDM la distance définie par $\text{HVDM}(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{j=1}^{p} \text{d}'_{X_j}(\mathbf{x}_j, \mathbf{x}'_j)}$ avec :

$$\text{d}'_{X_j}(x_j, x'_j) = \begin{cases} \text{VDM}_Q(x_j, x'_j) \text{ si } X_j \text{ qualitative} \\ \text{VDM}_N(x_j, x'_j) \text{ si } X_j \text{ quantitative} \end{cases}$$

$$\text{VDM}_Q(x_j, x'_j) = \frac{1}{2} \sum_{c=1}^{J} \left| \mathbb{P}\left(Y = c \mid X_j = x_j\right) - \mathbb{P}\left(Y = c \mid X_j = x'_j\right) \right|^q$$

$$\text{VDM}_N(x_j, x'_j) = \frac{|x_j - x'_j|}{4\mathbb{V}\left[X_j\right]}$$

Sur le même principe que HVDM, nous introduisons dans 6.1 une distance adaptée à notre cas particulier de la classification binaire.

5.3 FONCTION DE COÛT

La plupart des algorithmes de classification cherchent à minimiser l'erreur 0-1, c'est-à-dire le nombre de mauvaises prédictions. Néanmoins dans un grand nombre de cas réels, le coût de mauvaise classification est plus élevé pour une des classes (souvent la classe minoritaire) que pour les autres classes. En effet en prenant par exemple le cas de la classification des mails en Spam ou Non-Spam, mal classifier un mail indésirable en tant que Non-Spam ne coûtera que quelques minutes de votre temps, à l'inverse mal classifier un mail de votre employeur en tant que Spam aura un coût potentiellement très élevé. Il convient alors au lieu de minimiser le taux d'erreur de minimiser le coût à partir d'une matrice de coûts C où $C(i, j)$ est le coût de classification d'un élément de vraie classe i en classe j.

5.3.1 Weighted Random Forest

Une première solution pour prendre en compte le coût de mauvaise classification est d'intégrer celui-ci directement à la procédure d'apprentissage des arbres aléatoires. Pour cela Breiman (2001) propose d'attribuer un poids aux observations en donnant un poids plus fort aux classes que l'on souhaite mieux classer, c'est-à-dire celles qui possèdent le plus fort coût de mauvaise classification. Dans la procédure standard d'induction de l'arbre, l'impureté de Gini d'un nœud est

$$i_G = \sum_i p_i(1 - p_i) = 1 - \sum_i p_i^2$$

où p_i est la proportion de l'échantillon du nœud qui appartient à la classe i. Dans ce cas l'impureté, est maximale quand la distribution des classes est uniforme, ce qui est implicitement considéré comme la

distribution a priori des classes. Dans le cas où les classes ne sont pas uniformes, on veut alors modifier la fonction d'impureté pour refléter ce déséquilibre et placer le maximum d'impureté à la distribution a priori des classes. Si l'on note w_i le poids de la classe i et n_i le nombre d'observations de i dans le nœud fils t_R alors le poids total dans le nœud t_R est

$$w_R = \sum_i w_i n_i$$

et l'impureté est alors

$$i_G(t_R) = 1 - \sum_i \left(\frac{w_i n_i}{t}\right)^2$$

Donc l'impureté de la coupure est :

$$i_G(t) = \frac{w_L}{w} i_G(t_L) + \frac{w_R}{w} i_G(t_R)$$

avec w le poids total dans le nœud parent t.

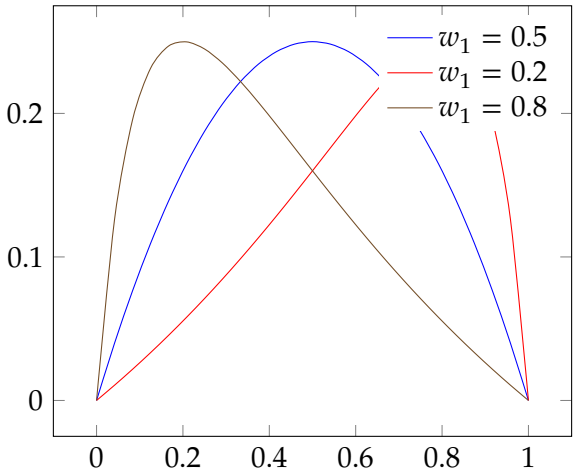

FIG. 5.3 : Impuretés selon les poids relatifs

5.3.2 Algorithme Metacost

Une approche possible pour la création d'un algorithme d'apprentissage prenant en compte le coût est de modifier l'algorithme pour incorporer directement les coûts dans la construction, néanmoins cela reste une solution difficile à mettre en oeuvre et qu'il est nécessaire de répéter pour chaque algorithme que l'on veut adapter. Pour éviter ce travail long et fastidieux, DOMINGOS (1999) propose l'algorithme MetaCost qui permet de transformer tout algorithme qui peut fournir une approximation des probabilités de classification en un algorithme de minimisation des coûts. Le but dans le cas de la minimisation du coût est de minimiser le risque conditionnel

$$R(i \mid x) = \sum_j \mathbb{P}(j \mid x) C(i, j) \tag{5.1}$$

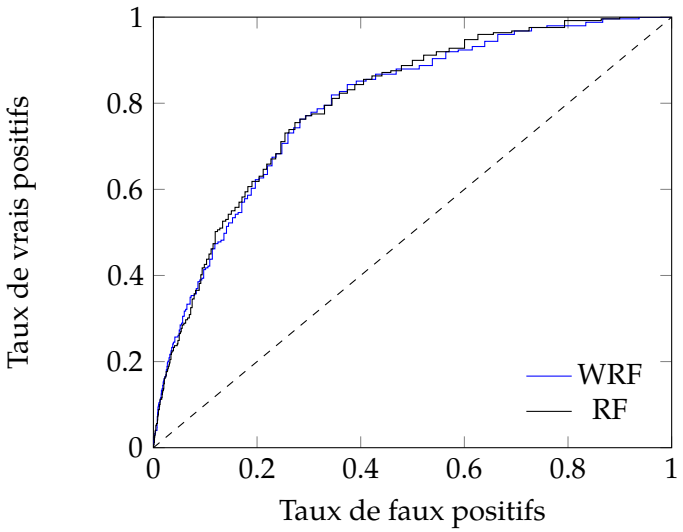

FIG. 5.4 : Performances de WRF (AUC = 0.8006).

Cette règle partitionne l'espace $\mathcal{X}$ en j régions de décision optimales. Notre but est alors de déterminer les frontières de ces espaces afin de construire notre classifieur. Il s'agit d'un problème difficile puisqu'il est possible que les étiquettes des individus d'apprentissage elles-mêmes ne soient pas optimales selon cette règle. L'idée de l'algorithme MetaCost est alors de réétiqueter l'échantillon d'apprentissage avec les classes optimales estimées et de réapprendre sur ce nouveau meta-échantillon. Il faut tout d'abord un moyen d'estimer la probabilité conditionnelle d'appartenir à la classe j de chaque individu d'apprentissage x, on utilise donc pour cela notre algorithme en lui-même. On peut ensuite calculer les nouvelles étiquettes optimales grâce à 5.1, réétiqueter l'échantillon d'apprentissage, et enfin apprendre notre classifieur final sur celui-ci.

Dans le cas des Forêts Aléatoires on possède déjà une bonne approximation des probabilités conditionnelles grâce à l'échantillon OOB. Il n'est donc pas nécessaire d'utiliser la procédure de Bagging, ici redondante, proposée par DOMINGOS (1999). Dans les rares cas où x n'est jamais OOB, on utilise comme approximation la probabilité issue de la forêt entière.

La procédure MetaCost pour les forêts aléatoires est résumée dans 7

Dans le cadre des forêts aléatoires, le coût de la procédure n'est alors que deux fois plus élevé. Dans le cas où l'algorithme ne fournit pas de probabilités, il est toujours possible d'obtenir une approximation de celles-ci en procédant à un bagging comme dans le cas des forêts aléatoires, le coût est alors N fois plus grand où N est le nombre d'échantillons bootstrap construits.

Algorithme 7 Algorithme Metacost pour les forêts aléatoires

procedure METACOST-RF(S, C)
  `RF` $\leftarrow$ `ForêtsAléatoire(S)`
  pour $(y, x) \in S$ **faire**
    pour chaque classe j **faire**
      si $x \in$ OOB(`RF`) **alors**
        $\hat{p}(j \mid x) \leftarrow \mathbb{P}_{\text{OOB}}(j \mid x)$
      sinon
        $\hat{p}(j \mid x) \leftarrow$ `RF`$(j \mid x)$
      fin si
    fin pour
    $y \leftarrow \arg\min_i \sum_j \hat{p}(j \mid x) C(i, j)$
  fin pour
  renvoyer S
fin procedure

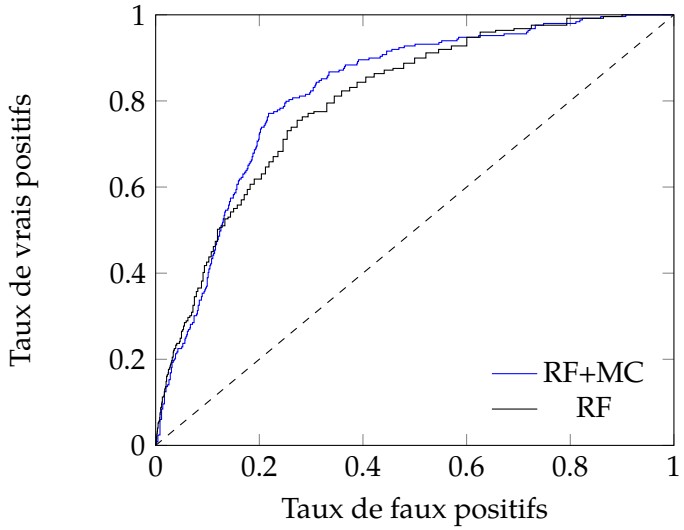

FIG. 5.5 : Performances de MetaCost + RF (AUC = 0.8217).

6 PRISE EN COMPTE DES VALEURS MANQUANTES

Il est courant que les jeux de données réelles comportent un grand nombre de données manquantes, cela peut être dû à une absence d'information pour l'individu concerné, à une erreur causée par une entrée manuelle des données ou d'autres processus comme la corruption des données. A priori, une observation dont il manque certaines valeurs n'est pas utilisable et doit être ignorée lors de la création du modèle, ainsi les données éventuellement déjà peu nombreuses se trouvent être encore plus réduites et la performance du modèle diminue. Il semble légitime de vouloir quand même utiliser les valeurs non manquantes de l'observation pour améliorer les performances sans trop les diminuer en traitant la ou les valeurs manquantes. Il est possible de modifier l'algorithme d'apprentissage pour directement prendre en compte les valeurs manquantes, par exemple dans le cas des arbres en propageant les valeurs manquantes dans les deux fils à la fois ou en utilisant un arbre ternaire, l'un des trois fils récupérant les valeurs manquantes, néanmoins ces techniques restent très spécifiques et ne peuvent pas se généraliser à tous les algorithmes. La procédure standard est alors l'estimation des valeurs manquantes pour se ramener à un échantillon d'apprentissage sans valeurs manquantes sur lequel appliquer nos algorithmes.

Le problème d'estimation des valeurs manquantes est en lui même un problème d'apprentissage machine, nous allons donc utiliser des techniques d'apprentissage machine standard pour estimer les valeurs manquantes à partir des valeurs que l'on connaît.

6.1 ESTIMATION PAR LES k–PLUS PROCHES VOISINS

L'une des méthodes les plus simples pour inférer les valeurs manquantes est de les remplacer par une valeur renseignée chez un autre individu *proche*. Pour diminuer le risque d'overfitting il est possible de baser la prédiction sur k individus proches, on se ramène ainsi à un problème d'estimation des k-plus proches voisins.

Afin d'appliquer l'algorithme k-NN, il est nécessaire de se munir d'une métrique. On rencontre alors deux problèmes : toutes les variables n'ont pas la même échelle et donc peuvent prendre un poids disproportionné dans le calcul des distances, et les distances courantes ne sont pas adaptées pour les variables qualitatives. Le premier problème est aisé-

ment résolu en renormalisant et standardisant chacune des variables, mais le second nécessite la création de mesures spécifiques (voir D. R. Wilson et Martinez (1997), Lumijärvi et al. (2004) ou Rodriguez et al. (2008)). Puisque nous nous intéressons au cas de la classification binaire, nous allons exploiter les caractéristiques particulières du problème pour obtenir une mesure de similarité plus simple.

À partir de maintenant nous supposons que $\mathcal{Y} = \{0, 1\}$ avec 1 la classe d'intérêt. De la même façon que dans 4.1.3, on munit les variables qualitatives d'une relation d'ordre $\preceq$, ainsi si a et b sont deux catégories de X_i, on a :

$$a \preceq b \Leftrightarrow \mathbb{P}\left[Y = 1 \mid X_i = a\right] \leq \mathbb{P}\left[Y = 1 \mid X_i = b\right]$$

Maintenant que toutes nos variables sont munies de relations d'ordre totales, on pose

$$x \to F_i(x) := \begin{cases} \mathbb{P}\left[X_i \leq x \mid Y_i = 1\right] \text{ si } X_i \text{ quantitative} \\ \mathbb{P}\left[X_i \preceq x \mid Y_i = 1\right] \text{ si } X_i \text{ qualitative} \end{cases}$$

La mesure de similarité est alors, avec $x^1 = (x_1^1, \dots, x_M^1), x^2 = (x_1^2, \dots, x_M^2)$, $\lambda \in [0, 1]$ et $q > 0$:

$$d(x^1, x^2) = \left(\sum_{i=1}^{M} \left(\lambda + \mathbb{1}_{\substack{x_i^1 \neq \text{NA} \\ x_i^2 \neq \text{NA}}} (|F_i(x_i^1) - F_i(x_i^2)| - \lambda) \right)^q \right)^{\frac{1}{q}}$$

Ici λ est une pénalisation des valeurs manquantes afin d'accorder plus d'importances aux individus sans valeurs manquantes. De plus la transformation par F_i donne une standardisation efficace puisque tous nos nouveaux points sont distribués identiquement uniformément sur $[0, 1]$.

Enfin si x^1 possède x_j^1 comme valeur manquante on choisit comme estimation

$$m(x_j^1, d, k) = \text{médiane}\left(\left[\underset{\substack{x^2 \in \text{k-NN}(x^1) \\ x^2 \in \mathcal{L}}}{\arg\min} \; d(x^1, x^2) \right]_j \right)$$

6.2 ESTIMATION PAR PRÉDICTIONS RÉPÉTÉES

6.2.1 Méthode de Breiman

Les forêts aléatoires peuvent fournir une mesure de similarité. En effet il est possible de voir un arbre comme une fonction indicatrice qui à chaque individu leur attribue une feuille. Ainsi du point de vue d'un arbre deux individus tombant dans la même feuille sont similaires, il

est alors possible de moyenner la décision *similaire* / *dissimilaire* sur l'ensemble des arbres de la forêt pour obtenir une mesure de similarité.

$$\text{similarité}(x_1, x_2) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t \in \tilde{\varphi}_{\mathcal{L},m}} \mathbb{1}_{x_1, x_2 \in \mathcal{X}_t}$$

avec $\tilde{\varphi}_{\mathcal{L},m}$ l'ensemble des feuilles de l'arbre $\varphi_{\mathcal{L},m}$. La mesure de similarité est sensible à la taille des feuilles, en effet si l'on autorise les arbres avec des feuilles pures la majorité des individus seront classés comme *dissimilaires*.

L'idée de Breiman (2001) pour traiter les valeurs manquantes repose sur l'utilisation de cette mesure de dissimilarité. Les valeurs manquantes sont tout d'abord remplacées par une estimation naïve et rapide comme la moyenne ou le mode, une forêt est ensuite construite sur cet échantillon et la similarité des individus possédant des valeurs manquantes à tous les autres individus est calculée. Les valeurs manquantes sont alors re-estimées comme étant la moyenne pondérée par les similarité des observations. Cette étape est répétée jusqu'a convergence des valeurs manquantes. Dans le cas des variables qualitatives on peut procéder à un vote pondéré.

6.2.2 MissForest

Bien qu'il soit possible comme précédemment d'utiliser un algorithme spécifique pour l'estimation des valeurs manquantes, nous possédons déjà un algorithme d'apprentissage performant capable d'estimer à la fois les variables qualitatives et quantitatives et s'adaptant à nos données : les forêts aléatoires. Il est ainsi possible de voir le problème d'estimation des valeurs manquantes comme P problèmes d'apprentissage machine avec P le nombre de variables possédant des valeurs manquantes dans l'échantillon d'apprentissage tel que dans la méthode `MissForest` (Stekhoven et Bühlmann, 2012). En choisissant la colonne x_j comme cible d'apprentissage et le reste, c'est-à-dire (y, x_{-j}) comme observation, on peut alors construire une forêt, s'en servir pour estimer les valeurs manquantes puis passer à la variable suivante. Il s'agit d'une méthode intuitive et simple à mettre en œuvre mais possédant un coût de calcul très élevé.

Il faut se fixer un critère d'arrêt, les auteurs proposent d'arrêter la procédure lorsque la différence entre les deux jeux de données augmente pour la première fois par rapport non seulement aux variables quantitatives mais aussi qualitatives. Le critère d'arrêt est donc :

$$\begin{cases} \Delta_{\mathcal{M}}^k > 0 \\ \Delta_{\mathcal{Q}}^k > 0 \end{cases}$$

Algorithme 8 Algorithme MissForest

procedure MissForest($\mathcal{L}$, Critère)
  Compléter $\mathcal{L}$ naïvement (k-NN par exemple)
  K ← indices des variables triées par ordre croissant de valeurs manquantes
  tant que Critère non rempli **faire**
    pour $k \in K$ **faire**
      ForêtAléatoire($x^k \sim \left(y, x^{-k}\right)$)
      Prédit $x^k_{\text{manquantes}}$ avec $\left(y, x^{-k}\right)_{\text{manquantes}}$
      $\mathcal{L}^{\text{imp}} \leftarrow \mathcal{L}^{\text{imp}}$ avec x^k mis à jour
    fin pour
  fin tant que
  renvoyer $\mathcal{L}^{\text{imp}}$
fin procedure

Avec NA l'ensemble des indices des valeurs manquantes, $\mathcal{M}$ l'ensemble des variables quantitative et $\mathcal{Q}$ les variables qualitatives :

$$\Delta^k_{\mathcal{M}} = \frac{\sum_{j \in \mathcal{M} \cap K} \sum_{i \in \text{NA} \cap [1,\dots,N]} (x^j_{i,k} - x^j_{i,k-1})^2}{\sum_{j \in \mathcal{M} \cap K} \sum_{i \in \text{NA} \cap [1,\dots,N]} (x^j_{i,k})^2}$$

$$\Delta^k_{\mathcal{Q}} = \frac{\sum_{j \in \mathcal{Q} \cap K} \sum_{i \in \text{NA} \cap [1,\dots,N]} \mathbb{1}_{x^j_{i,k} \neq x^j_{i,k-1}}}{|\text{NA}|}$$

6.3 MODIFICATION ALGORITHMIQUE

Au lieu de chercher à estimer les valeurs manquantes, il est possible de modifier les algorithmes d'apprentissage en eux-mêmes pour prendre en compte les valeurs manquantes. La première méthode consiste à considérer les valeurs manquantes comme un type de valeur contenant de l'information. Il existe en effet un grand nombre de cas dans lesquels le fait que la variable soit non renseignée est informative ; en détection des fraudes sur internet on peut par exemple considérer qu'un champ `Adresse` non renseigné est suspect et donc augmente le risque. Dans cette optique, une modification possible de l'algorithme d'induction d'arbre est d'opter pour des arbres ternaires au lieu de binaire, la nouvelle branche correspondant aux valeurs manquantes. Cette méthode a pour inconvénient de multiplier le nombre de branches et donc en diluant d'une certaine façon l'information d'augmenter le risque de sur-apprentissage. Une autre solution est de répercuter les variables manquantes à chaque nœud : les observations manquantes sont copiées dans les deux nœuds fils et l'algorithme continue. L'inconvénient de cette méthode étant qu'un même individu sera présent dans plusieurs feuilles à la fois. La représentation de l'arbre comme partition

binaire récursive n'est plus alors valide et la probabilité empirique donnée par les feuilles plus explicitement interprétable.

7 CONCLUSION

Nous avons vu que les méthodes ensemblistes peuvent améliorer les performances par rapport à un modèle simple, souvent de façon très importante. Il apparaît que les méthodes d'agrégations de nombreux modèles fortement randomisés présentent de très bonnes performances dans le cas des données utilisées (voir tableau C.2).

Il est aussi encourageant, bien que surprenant, de voir que les corrections les plus simples pour les données déséquilibrées c'est-à-dire un rééquilibrage par sous-échantillonnage bootstrap semblent être les plus efficaces, en effet non seulement leur implémentation est triviale, mais elles ont pour conséquence secondaire de fortement accélérer les temps de calcul.

Les méthodes de types *stacking* ont montré sur nos données des résultats assez faibles et sont plus coûteuses à mettre en place tout en nécessitant une certaine « ingénierie » de la part du praticien dans le choix des modèles retenus, algorithmes de stacking et autres paramètres, ce qui augmente grandement la complexité.

Les méthodes bayésiennes ont atteint des performances bien supérieures à celles attendues, et ce sans prendre en compte aucune correction pour le déséquilibre des données. Le principal inconvénient rencontré par les méthodes bayésiennes est le temps de calcul colossal et la mémoire machine nécessaire lors de la construction de la forêt et dans une moindre mesure lors de la prédiction d'une nouvelle observation. Néanmoins il est possible que dans certains cas l'avantage apporté par la connaissance de la densité tout entière et non pas une statistique précise soit assez grand pour justifier le temps de calcul. Il semble donc intéressant de continuer à étudier les méthodes bayésiennes, en ajoutant des a priori adaptés au cas déséquilibré par exemple ou en améliorant les performances des algorithmes afin de rendre leur utilisation possible en grande dimension ou pour des échantillons de grande taille.

Nous précisons que les méthodes de remplacement des données manquantes n'ont amélioré les résultats que de manière anecdotique et leurs résultats ne sont pas présentés, car identiques aux cas sans traitement des données manquantes. Qui plus est, les algorithmes du type MissForest sont bien trop coûteux pour être appliqués sur une base « normale ». Néanmoins l'application de l'algorithme dans le cas de la classification binaire est simple et rapide et ne semble présenter aucun inconvénient ou dégradations des performances. Il serait donc judicieux de l'appliquer en permanence dans les cas où l'échantillon possède peu d'individus.

Enfin il semble important de concentrer les efforts futurs sur les méthodes de *ranking*, et l'optimisation directe de métriques telles que l'AUC. En effet la plupart des algorithmes classiques optimisent d'une façon ou d'une autre la perte 0-1 qui n'est pas celle qui est dans notre cas d'intérêt, il semble donc logique d'optimiser le critère que l'on mesure. Qui plus est, bien qu'il semble que les gains en AUC se fassent souvent au détriment du score propre (Brier par exemple), les deux ne sont pas liés et il est possible de d'abord optimiser l'AUC puis calibrer les probabilités estimées puisqu'une transformation monotone ne modifie pas l'AUC.

A MARKOV CHAIN MONTE-CARLO

A.1 STATISTIQUES BAYESIENNES ET CHAINES DE MARKOV

Pour approfondir
Pour une introduction rapide aux techniques MCMC il est possible de voir Gonçalves (2015), Lam (2008) où Andrieu et al. (2003). Pour une véritable justification, voir Robert et Casella (2004).

Nous avons vu que toutes les méthodes bayésiennes possédaient le même fonctionnement de base : la spécification d'un a priori $\mathbb{P}(\theta)$ sur les paramètres d'intérêt, puis la mise à jour de cet a priori en fonction des observations en utilisant la loi de Bayes afin d'obtenir un a posteriori $\mathbb{P}(\theta \mid \mathbf{x})$.

$$\mathbb{P}(\theta) \propto \mathbb{P}(\mathbf{x} \mid \theta)\mathbb{P}(\theta)$$

Enfin, puisque souvent la valeur qui nous intéresse est $f(\theta)$, on obtient l'estimateur de Bayes associé :

$$\mathbb{E}\left[f(\theta) \mid \mathbf{x}\right] \tag{A.1}$$

Cet estimateur est intéressant puisqu'admissible, convergeant en probabilité et asymptotiquement normale.

La procédure standard serait alors de chercher à estimer A.1 par intégration de Monte-Carlo :

$$\mathbb{E}\left[f(\theta) \mid \mathbf{x}\right] = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(\tilde{\theta}_i)$$
$$\tilde{\theta}_i \sim \theta \mid \mathbf{x}$$
$$\tilde{\theta}_i \text{ indépendants}$$
$$\mathbb{E}\left[|\tilde{\theta}_1|\right] < +\infty$$

Il existe néanmoins un obstacle de taille à la mise en place de ce calcul puisque le calcul de $\mathbb{E}\left[f(\theta) \mid \mathbf{x}\right]$ suppose que l'on sache simuler des réalisations de $\mathbb{P}(\theta \mid \mathbf{x})$ indépendantes. La simulation d'une telle loi est dans certains cas possibles :

- la loi est une loi déjà connue et simulable directement.
- la loi est associée à une fonction de répartition inversible ou pseudo-inversible.

- on connaît une autre loi qui borne celle-ci et permet donc de procéder à une procédure d'acceptation-rejet.

Pour les lois complexes, ce sera presque toujours le cas, il est impossible d'avoir recours à de telles méthodes. Obtenir des réalisations indépendantes des $\tilde{\theta}_i$ est trop contraignant et nous pourrons au mieux obtenir des réalisations corrélées de $\theta \mid \mathbf{x}$. Il semble alors nécessaire de posséder une généralisation de la loi forte des grands nombres (LFGN) dans le cas où les observations ne sont pas i.i.d.

Théorème A.1 (Théorème Ergodique)**.** *Soient $\hat{\theta}_1, \dots, \hat{\theta}_m$ une suite de valeurs d'une chaîne de Markov de noyaux de transition g, c'est à dire*

$$\mathbb{P}\left(\hat{\theta}_i \mid \hat{\theta}_{i-1}, \dots, \hat{\theta}_1\right) = g(\hat{\theta}_{i-1}, \hat{\theta}_i)$$

et matrice de transition P, telle que la chaîne soit :

APÉRIODIQUE

$$\exists i, \forall j > i, g\left(\hat{\theta}_j \mid \hat{\theta}_i\right) > 0$$

IRRÉDUCTIBLE

$$\forall i, j \; g\left(\hat{\theta}_j \mid \hat{\theta}_j\right) > 0$$

POSITIVE RÉCURRENTE

$$\forall i, \mathbb{E}\left[T_i\right] < \infty$$
$$T_i = \inf\left\{j \geq 1 : \hat{\theta}_j = i \mid \hat{\theta}_0 = i\right\}$$

Alors si $\mathbb{E}\left[f(\theta)\right] < \infty$ alors

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} f(\hat{\theta}_i) \to \int f(\theta)\pi(\theta)\,\mathrm{d}\theta\right)$$

où π est la distribution stationnaire de la chaîne de Markov c'est-à-dire $\pi = \pi P$

Nous allons donc chercher à créer une chaîne de Markov répondant aux hypothèses de A.1 de distribution stationnaire égale à $\mathbb{P}(\theta \mid \mathbf{x})$.

A.2 ÉCHANTILLONAGE DE METROPOLIS-HASTINGS

Il semble naturel de se poser la question « Est-il toujours possible de construire une chaîne de Markov répondant aux hypothèses du théorème A.1 ? ». La réponse est oui. En effet une condition suffisante (mais non nécessaire) pour assurer que $p(\theta)$ est la distribution invariante de

notre chaîne est la condition de réversibilité suivante, aussi appelée équilibre détaillé :

$$p(\theta_i)g(\theta_{i-1} \mid \theta_i) = p(\theta_{i-1})g(\theta_i \mid \theta_{i-1}) \tag{A.2}$$

L'algorithme de Metropolis-Hastings utilise alors une densité de proposition $q(\theta' \mid \theta)$, qui joue le rôle de chaîne de Markov continue. À partir de θ, une valeur θ' est proposée puis acceptée comme nouvel état avec probabilité :

$$A(\theta, \theta') = \min\left\{1, \frac{p(\theta')q(\theta \mid \theta')}{p(\theta)q(\theta' \mid \theta}\right\}$$

L'algorithme de Metropolis-Hastings est alors :

Algorithme 9 Metropolis-Hastings

procedure METROPOLIS-HASTINGS(q, θ_0, n)
  pour $i \in [0, n-1]$ **faire**
    Tire $u \sim \mathcal{U}_{[0,1]}$
    Tire $\theta' \sim q(\theta' \mid \theta_i)$
    si $u < \min\left\{1, \frac{p(\theta')q(\theta_i|\theta')}{p(\theta_i)q(\theta'|\theta_i)}\right\}$ **alors**
      $\theta_{i+1} = \theta'$
    sinon
      $\theta_{i+1} = \theta_i$
    fin si
  fin pour
  renvoyer $(\theta_0, \cdots, \theta_{n-1})$
fin procedure

On a ici que

$$g(\theta_i, \theta_{i+1}) = q(\theta_{i+1} \mid \theta_i)A(\theta_i, \theta_{i+1}) + \delta_{\theta_i}(\theta_{i+1})R(\theta_i)$$

Avec $R\theta_i$ le terme de rejet :

$$R(\theta_i) = \int q(\theta \mid \theta_i)(1 - A(\theta_i, \theta))\,\mathrm{d}\theta$$

Par construction g remplis la propriété A.2. De plus, le terme de rejet garantit l'apériodicité et il suffit que le support de q inclue le support de p pour garantir l'irréductibilité.

La présence de p non seulement au numérateur, mais aussi au dénominateur permet de se débarrasser de la constante de renormalisation, souvent l'un des éléments les plus contraignants.

A.3 ÉCHANTILLONNAGE DE GIBBS

Il existe un cas dans lequel l'algorithme de Metropolis-Hastings possède une forme particulière : le cas où l'on connaît les lois conditionnelles complètes de tous les paramètres, c'est-à-dire si l'on sait tirer une valeur de $\mathbb{P}(\theta_i \mid \theta_{-i}, \mathbf{x})$, $\forall i$.

Dans ce cas il est possible de poser :

$$q(\theta' \mid \theta_i) = \begin{cases} p(\theta'^{(j)} \mid \theta_i^{(-j)}) \text{ si } \theta'^{(-j)} = \theta_i^{(-j)} \\ 0 \text{ sinon.} \end{cases}$$

Et la probabilité d'acceptation est alors

$$\begin{aligned} A(\theta_i, \theta') &= \min\left\{1, \frac{p(\theta')q(\theta_i \mid \theta')}{p(\theta_i)q(\theta' \mid \theta_i)}\right\} \\ &= \min\left\{1, \frac{p(\theta')p(\theta_i^{(j)} \mid \theta'^{(-j)})}{p(\theta_i)p(\theta'^{(j)} \mid \theta_i^{(-j)})}\right\} \\ &= \min\left\{1, \frac{p(\theta^{(-j)})p(\theta')\overbrace{p(\theta_i^{(j)} \mid \theta'^{(-j)})p(\theta'^{(-j)})}^{p(\theta_i)}}{p(\theta'^{(-j)})p(\theta_i)\underbrace{p(\theta'^{(j)} \mid \theta_i^{(-j)})p(\theta^{(-j)})}_{p(\theta')}}\right\} \\ &= \min\left\{1, \frac{p(\theta^{(-j)})}{p(\theta'^{(-j)})}\right\} \\ (\text{car } \theta'^{(-j)} = \theta_i^{(-j)}) \qquad &= 1 \end{aligned}$$

L'algorithme d'échantillonnage de Gibbs est, en notant $\theta = (\theta_1, \dots, \theta_p)$, le vecteur des paramètres et θ^i la $i^{\text{ème}}$ itération.

- Choix de valeurs initiales pour θ^0.
- Tirage de $\theta_1^1 \sim \theta_1 \mid \theta_{-1}^0$
- θ^0 est mis à jour à l'aide du tirage précédent.
- Les θ_i^1 sont tirés de la même façon jusqu'a ce qu'ils aient tous été mis à jour afin d'obtenir θ^1.
- La procédure est répétée M fois.

La suite $\left(\theta^i\right)$ qui en résulte répond aux propriétés de Markov souhaitées.

B | PREUVES

Preuve du théorème 2.4, adaptée de Devroye et al. (1997). Soit $\mathcal{X} = \mathbb{R}^d$ et $\mathcal{Y} = \{0, 1\}$. Ici $L(y, \varphi(\mathbf{x})) = \mathbb{1}_{\varphi(\mathbf{x}) \neq y}$ et donc $\mathrm{Err}(\varphi(X)) = \mathbb{P}(\varphi(X) \neq Y)$. Supposons que $N\varepsilon^2 > 2$ et introduisons $\mathcal{L}'$ un nouvel échantillon de $(\mathbf{x}'_i, y'_i)$ de taille N, indépendant de $\mathcal{L}$ et identiquement distribué. Soit

$$\mathrm{Err}'_N(\varphi) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\varphi(\mathbf{x}'_i) \neq y'_i}$$

Montrons d'abord que :

$$\mathbb{P}\left(\sup_{\varphi \in \Phi} |\mathrm{Err}_N(\varphi) - \mathrm{Err}(\varphi)| > \varepsilon \right) \leq 2\mathbb{P}\left(\sup_{\varphi \in \Phi} |\mathrm{Err}_N(\varphi) - \mathrm{Err}'_N(\varphi)| > \varepsilon/2 \right)$$

Soit $\tilde{\varphi}$ tel que $|\mathrm{Err}_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| > \varepsilon$. C'est une fonction de $\mathcal{L}$, mais nous ne l'écrivons pas pour simplifier les notations.

$$\begin{aligned}
&\mathbb{P}\left(\sup_{\varphi \in \Phi} |\mathrm{Err}_N(\varphi) - \mathrm{Err}'_N(\varphi)| > \varepsilon/2 \right) \geq \mathbb{P}\left(|\mathrm{Err}_N(\tilde{\varphi}) - \mathrm{Err}'_N(\tilde{\varphi})| > \varepsilon/2 \right) \\
&\geq \mathbb{P}\left(|\mathrm{Err}_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| > \varepsilon \text{ et } |\mathrm{Err}'_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| < \varepsilon/2 \right) \\
&\geq \mathbb{E}\left[\mathbb{1}_{|\mathrm{Err}_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| > \varepsilon} \mathbb{1}_{|\mathrm{Err}'_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| < \varepsilon/2} \right] \\
&\geq \mathbb{E}\left[\mathbb{1}_{|\mathrm{Err}_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| > \varepsilon} \mathbb{E}\left[\mathbb{1}_{|\mathrm{Err}'_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| < \varepsilon/2} \mid \mathcal{L} \right] \right] \\
&\geq \mathbb{E}\left[\mathbb{1}_{|\mathrm{Err}_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| > \varepsilon} \mathbb{P}\left(\mathbb{1}_{|\mathrm{Err}'_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| < \varepsilon/2} \mid \mathcal{L} \right) \right]
\end{aligned}$$

Notons alors

$$U_i = \mathbb{1}_{\tilde{\varphi}(X'_i) \neq Y'_i} - \mathbb{E}\left[\mathbb{1}_{\tilde{\varphi}(X'_i) \neq Y'_i} \mid \mathcal{L} \right]$$

qui sont des variables aléatoires centrées, indépendantes et identiquement distribuées. Alors conditionnellement à $\mathcal{L}$:

$$\mathrm{Err}'_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi}) = \frac{1}{N} \sum_{i=1}^{N} U_i$$

On a alors :

$$\begin{aligned}
\mathbb{P}\left(|\mathrm{Err}'_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| < \varepsilon/2 \mid \mathcal{L}\right) &= \mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N} U_i\right| < \frac{\varepsilon}{2} \mid \mathcal{L}\right) \\
&= \mathbb{P}\left(\left|\sum_{i=1}^{N} U_i\right| < \frac{N\varepsilon}{2} \mid \mathcal{L}\right) \\
\text{(Par l'inégalité de Chebyshev)} \qquad &\geq 1 - \frac{4}{n^2\varepsilon^2}\mathbb{V}\left[\left|\sum_{i=1}^{N} U_i\right| \mid \mathcal{L}\right] \\
\text{(Car il s'agit d'une variable binaire)} \qquad &\geq 1 - \frac{4}{n^2\varepsilon^2}\frac{1}{4} \\
\text{(Car } N\varepsilon^2 \geq 2\text{)} \qquad &\geq \frac{1}{2}
\end{aligned}$$

Donc

$$\begin{aligned}
&\mathbb{P}\left(\sup_{\varphi\in\Phi} |\mathrm{Err}_N(\varphi) - \mathrm{Err}'_N(\varphi)| > \frac{\varepsilon}{2}\right) \\
&\geq \mathbb{E}\left[\mathbb{1}_{|\mathrm{Err}_N(\tilde{\varphi})-\mathrm{Err}(\tilde{\varphi})|>\varepsilon}\mathbb{P}\left(|\mathrm{Err}'_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| < \frac{\varepsilon}{2} \mid \mathcal{L}\right)\right] \\
&\geq \frac{1}{2}\mathbb{E}\left[\mathbb{1}_{|\mathrm{Err}_N(\tilde{\varphi})-\mathrm{Err}(\tilde{\varphi})|>\varepsilon}\right] \\
&\geq \frac{1}{2}\mathbb{P}\left(|\mathrm{Err}_N(\tilde{\varphi}) - \mathrm{Err}(\tilde{\varphi})| > \varepsilon\right) \\
&\geq \frac{1}{2}\mathbb{P}\left(\sup_{\varphi\in\Phi} |\mathrm{Err}_N(\varphi) - \mathrm{Err}(\varphi)| > \varepsilon\right)
\end{aligned}$$

On va maintenant procéder à une seconde symétrisation par un processus de Rademacher. On peut effet écrire :

$$\mathbb{P}\left(\sup_{\varphi\in\Phi} |\mathrm{Err}_N(\varphi) - \mathrm{Err}'_N(\varphi)| > \frac{\varepsilon}{2}\right) = \mathbb{P}\left(\sup_{\varphi\in\Phi} \frac{1}{N}\left|\sum_{i=1}^{N} (\mathbb{1}_{\varphi(X_i)\neq Y_i} - \mathbb{1}_{\varphi(X'_i)\neq Y'_i})\right| > \frac{\varepsilon}{2}\right)$$

Par construction $\mathbb{1}_{\varphi(X_i)\neq Y_i} - \mathbb{1}_{\varphi(X'_i)\neq Y'_i}$ est de moyenne nulle et symétrique. Introduisons $\sigma_1, \cdots, \sigma_N$ variables aléatoires i.i.d indépendantes de $\mathcal{L}$ et $\mathcal{L}'$ telles que $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$.

$$
\begin{aligned}
&\mathbb{P}\left(\sup_{\varphi\in\Phi}|\mathrm{Err}_N(\varphi)-\mathrm{Err}'_N(\varphi)|>\frac{\varepsilon}{2}\right)\\
&=\mathbb{P}\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}(\mathbb{1}_{\varphi(X_i)\neq Y_i}-\mathbb{1}_{\varphi(X'_i)\neq Y'_i})\right|>\frac{\varepsilon}{2}\right)\\
&=\mathbb{P}\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i(\mathbb{1}_{\varphi(X_i)\neq Y_i}-\mathbb{1}_{\varphi(X'_i)\neq Y'_i})\right|>\frac{\varepsilon}{2}\right)\\
&\leq\mathbb{P}\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i(\mathbb{1}_{\varphi(X_i)\neq Y_i}-\mathbb{1}_{\varphi(X'_i)\neq Y'_i})\right|>\frac{\varepsilon}{2}\right)\\
&\leq\mathbb{P}\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X_i)\neq Y_i}\right|-\left|\sum_{i=1}^{N}\mathbb{1}_{\varphi(X'_i)\neq Y'_i}\right|\right|>\frac{\varepsilon}{2}\right)\\
&\leq\mathbb{P}\Bigg(\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X_i)\neq Y_i}\right|>\frac{\varepsilon}{2}\text{ et }\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X'_i)\neq Y'_i}\right|>\frac{\varepsilon}{4}\right)\\
&\text{ou }\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X_i)\neq Y_i}\right|>\frac{\varepsilon}{4}\text{ et }\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X'_i)\neq Y'_i}\right|>\frac{\varepsilon}{2}\right)\Bigg)\\
&\leq\mathbb{P}\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X'_i)\neq Y'_i}\right|>\frac{\varepsilon}{4}\text{ ou }\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X_i)\neq Y_i}\right|>\frac{\varepsilon}{4}\right)\\
&\leq 2\mathbb{P}\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X'_i)\neq Y'_i}\right|>\frac{\varepsilon}{4}\right)
\end{aligned}
$$

On a donc

$$
\mathbb{P}\left(\sup_{\varphi\in\Phi}|\mathrm{Err}_N(\varphi)-\mathrm{Err}(\varphi)|>\varepsilon\right)\leq 4\mathbb{P}\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(X'_i)\neq Y'_i}\right|>\frac{\varepsilon}{4}\right)
$$

On va maintenant conditionner sur $\mathcal{L}$ pour faire apparaître la dimension de Φ par rapport à $\mathcal{L}$. Soit $\Phi_{\mathcal{L}}$ la plus petite partie de Φ telle que

$$
N(\Phi,\mathcal{L})=N(\Phi_{\mathcal{L}},\mathcal{L})
$$

On a alors $N(\Phi_{\mathcal{L}}, \mathcal{L}) \leq G(\Phi, \mathcal{L})$.

$$
\begin{aligned}
&\mathbb{P}\left(\sup_{\varphi\in\Phi}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{\varepsilon}{4}\right)\\
&= \mathbb{P}\left(\max_{\varphi\in\Phi_{\mathcal{L}}}\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{\varepsilon}{4}\right)\\
&= \mathbb{P}\left(\bigcup_{\varphi\in\Phi_{\mathcal{L}}}\left\{\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{\varepsilon}{4}\right\}\right)\\
&\leq \sum_{\varphi\in\Phi_{\mathcal{L}}}\mathbb{P}\left(\left\{\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{\varepsilon}{4}\right\}\right)\\
&\leq |\Phi_{\mathcal{L}}|\sup_{\varphi\in\Phi_{\mathcal{L}}}\mathbb{P}\left(\left\{\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{\varepsilon}{4}\right\}\right)\\
&\leq G(\Phi, N)\sup_{\varphi\in\Phi_{\mathcal{L}}}\mathbb{P}\left(\left\{\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{\varepsilon}{4}\right\}\right)\\
&\leq G(\Phi, N)\sup_{\varphi\in\Phi}\mathbb{P}\left(\left\{\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{\varepsilon}{4}\right\}\right)
\end{aligned}
$$

On peut enfin conclure par l'utilisation d'une inégalité de concentration. En notant que les $Z_i = \sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}$ sont des variables aléatoires centrées, i.i.d et bornées par -1 et 1 on a

$$
\begin{aligned}
\mathbb{P}\left(\left\{\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{\varepsilon}{4}\right\}\right) &\leq \mathbb{P}\left(\left\{\left|\sum_{i=1}^{N}\sigma_i\mathbb{1}_{\varphi(\mathbf{x}_i)\neq y_i}\right| > \frac{N\varepsilon}{4}\right\}\right)\\
\text{(Inégalité de Hoeffding)} \qquad &\leq 2e^{-2\frac{(N\varepsilon/4)^2}{\sum_{i=1}^{N}(1-(-1))^2}}\\
&\leq 2e^{-\frac{N\varepsilon^2}{32}}
\end{aligned}
$$

□

Preuve de 3. On suppose que $y = x\beta + \varepsilon$, avec $\mathbb{E}[\varepsilon] = 0$ et $\mathbb{V}[\varepsilon] = \sigma^2\mathbb{I}$. Alors :

$$
\begin{aligned}
\mathbb{V}\left[\hat{\beta}\right] &= \mathbb{E}\left[\hat{\beta}\hat{\beta}^\top\right] - \mathbb{E}\left[\hat{\beta}\right]\mathbb{E}\left[\hat{\beta}^\top\right]\\
&= \mathbb{E}\left[\left((x^\top x)^{-1}x^\top y\right)\left((x^\top x)^{-1}x^\top y\right)^\top\right] - \beta\beta^\top\\
&= \mathbb{E}\left[\left((x^\top x)^{-1}x^\top(x\beta+\varepsilon)\right)\left((x^\top x)^{-1}x^\top(x\beta+\varepsilon)\right)^\top\right] - \beta\beta^\top\\
&= \beta\beta^\top + \mathbb{E}\left[\beta\left((x^\top x)^{-1}x^\top\varepsilon\right)^\top\right] + \mathbb{E}\left[\left((x^\top x)^{-1}x^\top\varepsilon\right)\beta^\top\right]\\
&+ \mathbb{E}\left[\left((x^\top x)^{-1}x^\top\varepsilon\right)\left((x^\top x)^{-1}x^\top\varepsilon\right)^\top\right] - \beta\beta^\top\\
&= \beta\underbrace{\mathbb{E}\left[\varepsilon^\top\right]}_{0}\left((x^\top x)^{-1}x^\top\right)^\top + \left((x^\top x)^{-1}x^\top\right)\underbrace{\mathbb{E}\left[\varepsilon\right]}_{0}\beta^\top\\
&+ (x^\top x)^{-1}x^\top\mathbb{E}\underbrace{\left[\varepsilon\varepsilon^\top\right]}_{\sigma^2\mathbb{I}}\left(x(x^\top x)^{-1}\right)\\
&= \left(x^\top x\right)^{-1}\sigma^2
\end{aligned}
$$

□

Théorème d'approximation universelle de CYBENKO *(1989).* Nous allons tout d'abord montrer que les fonctions G du type

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^\top x + \theta_j)$$

avec σ discriminante, c'est-à-dire tel que pour tout $\mu \in \mathcal{M}(I_M)$ mesure de Borel sur I_M on a :

$$\int_{I_M} \sigma(y_j^\top x + \theta_j)\, \mathrm{d}\mu(x) = 0, \forall y \in \mathbb{R}^M, \theta \in \mathbb{R} \implies \mu = 0$$

sont denses dans $\mathcal{C}(I_M)$.

Soit

$$S = \text{vect}\left\{\sigma(y_j^\top x + \theta_j), \sigma \text{ discriminante}\right\}$$

Il faut alors montrer que $\bar{S} = \mathcal{C}(I_M)$. Supposons alors par l'absurde que $\bar{S} \neq \mathcal{C}(I_M)$, alors $\bar{S}$ strictement inclue dans $\mathcal{C}(I_M)$ et par le théorème de Hahn-Banach (voir BREZIS (2011) par exemple) il existe une application linéaire L bornée, sur $\mathcal{C}(I_M)$ telle que $L \neq 0$ et $L(S) = L(\bar{S}) = 0$. Par le théorème de représentation de Riesz, on a :

$$L(h) = \int_{I_M} h(x)g(x)\, \mathrm{d}x = \int_{I_M} h(x)\, \mathrm{d}\mu(x)$$

pour un certain g et donc μ. En particulier on a :

$$\int_{I_M} \sigma(y^\top x + \theta)\, \mathrm{d}\mu(x) = 0, \forall y, \theta$$

Et comme σ supposée discriminante on a $\mu = 0$, donc $L = 0$ ce qui est impossible. On a alors montré que $\bar{S} = \mathcal{C}(I_M)$.

Reste alors à montrer que toute fonction *sigmoïdale*, mesurable, bornée, est discriminante. On appelle ici fonction *sigmoïdale* la classe très large des fonctions σ du type :

$$\sigma(t) \to \begin{cases} 1 \text{ quand } t \to +\infty \\ 0 \text{ quand } t \to -\infty \end{cases}$$

Posons $\sigma_\lambda = \sigma(\lambda(y^\top x + \theta) + \varphi)$.

$$\sigma(\lambda(y^\top x + \theta) + \varphi) \xrightarrow[\lambda \to \infty]{} \begin{cases} 1 \text{ lorsque } y^\top x + \theta > 0 \\ 0 \text{ lorsque } y^\top x + \theta < 0 \\ \sigma(\varphi) \text{ lorsque } y^\top x + \theta = 0 \end{cases}$$

(σ_λ) converge simplement vers

$$x \to \mathbb{1}_{y^\top x + \theta > 0} + \sigma(\varphi)\delta_{y^\top x + \theta = 0}$$

De plus (σ_λ) est dominée puisque σ bornée, donc par le théorème de convergence dominée de Lebesgue on a :

$$\begin{aligned}
0 &= \lim_{\lambda\to+\infty} \int_{I_M} \sigma_\lambda(x)\,\mathrm{d}\mu(x) \qquad \text{car } \sigma \text{ discriminante} \\
&= \int_{I_M} \lim_{\lambda\to+\infty} \sigma_\lambda(x)\,\mathrm{d}\mu(x) \\
&= \int_{I_M} \left(\mathbb{1}_{y^\top x+\theta>0}(x) + \sigma(\varphi)\delta_{y^\top x+\theta=0}(x)\right)\mathrm{d}\mu(x) \\
&= \sigma(\varphi)\mu(\Pi_{y,\theta}) + \mu(H_{y,\theta}), \forall \varphi, \theta, y
\end{aligned}$$

avec

$$\begin{aligned}
\Pi_{y,\theta} &= \{x \mid y^\top x + \theta = 0\} \\
H_{y,\theta} &= \{x \mid y^\top x + \theta > 0\}
\end{aligned}$$

En faisant tendre φ vers $+\infty$ on a alors :

$$\mu(\Pi_{y,\theta}) + \mu(H_{y,\theta}) = 0 \qquad \forall \theta, y$$

et avec $\varphi \to -\infty$

$$\mu(H_{y,\theta}) = 0 \qquad \forall \theta, y$$

Il faut maintenant montrer que cette condition implique $\mu = 0$, il faut en effet se rappeler que ici μ est une mesure signée. Introduisons, pour y fixé, l'application linéaire F bornée sur $L^\infty(\mathbb{R})$:

$$F(h) = \int_{I_M} h(y^\top x)\,\mathrm{d}\mu(x)$$

On a alors

$$\begin{aligned}
F(\mathbb{1}_{[\theta,\infty[}) &= \int_{I_M} \mathbb{1}_{[\theta,\infty[}(x)\,\mathrm{d}\mu(x) = \mu(\Pi_{y,-\theta}) + \mu(H_{y,-\theta}) \\
&= 0 \\
F(\mathbb{1}_{]\theta,\infty[}) &= \int_{I_M} \mathbb{1}_{[\theta,\infty[}(x)\,\mathrm{d}\mu(x) = \mu(H_{y,-\theta}) \\
&= 0
\end{aligned}$$

Puisque F est linéaire, on a alors $F(h) = 0$ pour tout h indicatrice d'un intervalle et donc $F(h) = 0$ pour toute fonction simple. Par densité des fonctions simples dans $L^\infty(\mathbb{R})$ on a alors $F = 0$. Posons :

$$h(x) = \cos(x) + i\sin(x) \in L^\infty(\mathbb{R})$$

On a alors :

$$F(h) = \int_{I_M} \exp(iy^\top x)\,\mathrm{d}\mu(x) = 0$$

La transformée de Fourrier de μ est alors 0 donc $\mu = 0$.

On a donc bien prouvé que σ est discriminant, et donc S est dense dans $C(I_M)$. □

C RÉSULTATS

Les deux critères retenus pour la mesure des performances sont l'AUC et le Score de Brier.

Dans le cas de la base A, l'échantillon de validation est un échantillon « out-of-time », c'est-à-dire de données acquises après les données d'apprentissage. Dans le cas de la base B, les échantillons d'apprentissage et validation ont été créés en effectuant une coupe aléatoire stratifiée sur la variable y afin de garder le même équilibre dans les deux bases. La coupe a été effectuée dans une proportion de 2/3 pour l'échantillon d'apprentissage et 1/3 pour l'échantillon de validation.

La base A est un problème de classification binaire, la variable à prédire étant `Défaut` ou `Non Défaut`. La base B est elle aussi un problème de classification binaire, on cherche à prédire si les individus sont `Appétent` ou `Non Appétent`. Les caractéristiques globales des bases d'apprentissages sont données dans le tableau C.1. Les bases de validation possèdent globalement les mêmes caractéristiques.

	Obs.	NAs	Quantitatives	Qualitatives	% Positifs
Base A	26906	7834	57	5	2.86
Base B	18605	0	85	16	2.44

Tab. C.1 : Caractéristiques des bases d'apprentissage

Les bases ne sont que des échantillons de bases de plusieurs millions d'observations. Le choix a été fait de travailler sur des bases de tailles restreintes afin non seulement de pouvoir comparer les résultats avec ceux obtenus par le GRO précédemment, mais aussi afin de rendre l'apprentissage des différents algorithmes possible sur des machines de travail normales, et ce en un temps raisonnable compatible avec le processus d'expérimentation.

	Base A		Base B	
Classifieur	AUC	Brier[1]	AUC	Brier
Régression logistique	0.7889	0.0314	0.6727	0.02356
CART	0.79649	0.03194	0.6542	0.02489
Forêts aléatoires	0.805	0.02864	0.699	0.02346
Boosting Arbres	0.8103	0.0295	0.6345	0.0247
Stacking ∼ Réseau de neurones	0.7372	0.0399	0.6324	0.02477
Balanced RF	0.843	0.1509	0.7162	0.14497
Weighted RF	0.8006	0.02861	0.7017	0.02345
Roughly Balanced RF	0.8432	0.15395	0.7211	0.14597
Nearly Balanced RF	0.8427	0.1334	0.7235	0.1216
Very Roughly Balanced RF	0.8421	0.1297	0.7205	0.12123
RF + Metacost	0.8217	0.2938	0.69787	0.18886
RF + Feuilles SVMs	0.7664	0.11437	0.5898	0.0246
RF Uniformes	0.8423	0.0276	0.7053	0.02365
RF Uniformes Equilibrées	0.84346	0.2152	0.7305	0.2082
ExtRa-Trees[2]	0.7786	0.02853	0.6753	0.0251
BART	0.8365	0.02745	0.7321	0.02328

TAB. C.2: Performances des différentes méthodes sur 2 bases de validation différentes

[1] Dans le cas de Brier il s'agit d'une perte et non d'un score. Donc on souhaite la plus petite valeur possible.

[2] Dans le cas des ExtRa-Trees l'algorithme effectue un 1-hot dummy encoding, les résultats ne sont donc pas complètement comparables.

NOTATIONS

b_l	La $l^{\text{ième}}$ valeur d'une variable catégorielle
$\text{BN}(n, q)$	Loi binomiale négative avec n le nombre d'échecs et q la probabilité d'échec.
c_k	La $k^{\text{ième}}$ classe
$\mathbb{E}$	Espérance
$\overline{E}(\varphi_{\mathcal{L}}, \mathcal{L}')$	L'erreur de prédiction moyenne $\varphi_{\mathcal{L}}$ sur $\mathcal{L}'$
$Err(\varphi_{\mathcal{L}})$	L'erreur de généralisation de $\varphi_{\mathcal{L}}$
$H(X)$	L'entropie de Shannon de X
$H(X\|Y)$	L'entropie de Shannon de X conditionnellement à Y
$\mathcal{H}$	L'espace des modèles candidats, le dictionnaire
$i(t)$	Impureté du nœud t
$i_R(t)$	Impureté du nœud t basée sur l'estimation locale de l'erreur de rebalancement
$i_H(t)$	L'impureté issue de l'entropie du nœud t
$i_G(t)$	L'impureté de Gini du nœud t
$\Delta i(s, t)$	La diminution d'impureté due à la coupe s au nœud t
$I(X; Y)$	L'information mutelle entre X et Y
$\text{Imp}(X_j)$	L'importance de la variable X_j
J	Le nombre de classes
K	Nombre de strates dans la validation croisée
$K(\mathbf{x}_i, \mathbf{x}_j)$	Le noyau de $\mathbf{x}_i$ et $\mathbf{x}_j$
L	Une fonction de perte
$\mathcal{L}$	Un échantillon d'apprentissage $(\mathbf{X}, \mathbf{y})$
$\mathcal{L}_i$	La $i^{\text{ième}}$ strate de $\mathcal{L}$ lors de la validation croisée
$\mathcal{L}_{-i}$	$\mathcal{L} \setminus \mathcal{L}_i$
$\mathcal{L}^m$	Le $m^{\text{ième}}$ échantillon bootstrap issu de $\mathcal{L}$
$\mathcal{L}_t$	Le sous-échantillon d'observations présent au nœud t
M	Le nombre de modèles dans l'ensemble
$\mu_{\mathcal{L},\theta_m}(\mathbf{x})$	La prédiction moyenne en $X = \mathbf{x}$ de $\varphi_{\mathcal{L},\theta_m}$
N	Le nombre d'observations dans $\mathcal{L}$
N_t	Le nombre d'observations présentes au nœud t
N_{ct}	Le nombre d'observations de classe c au nœud t
Ω	L'univers des observations
p	Le nombre de variables des observations de $\mathcal{L}$
p_L	La proportion des échantillons du nœud allant dans t_L
p_R	La proportion des échantillons du nœud allant dans t_R
$p(t)$	L'estimation de la probabilité $p(X \in \mathcal{X}_t) = \frac{N_t}{N}$
$p(c\|t)$	L'estimation empirique de la probabilité $p(Y = c\|X \in \mathcal{X}_t) = \frac{N_{ct}}{N_t}$ de classe c au nœud t

$\hat{p}_{\mathcal{L}}$	L'estimation empirique à partir de l'échantillon $\mathcal{L}$ de la probabilité
$P(X, Y)$	La loi jointe de $X = (X_1, \dots, X_p)$ et Y
$\mathcal{P}_k(V)$	L'ensemble des parties de V de taille k
φ	Un classifieur ou une fonction $\mathcal{X} \mapsto \mathcal{Y}$
$\widetilde{\varphi}$	L'ensemble des feuilles de φ
$\varphi(\mathbf{x})$	La prédiction de φ en $\mathbf{x}$
$\varphi_{\mathcal{L}}$	Un classifieur construit à partir de $\mathcal{L}$
$\varphi_{\mathcal{L},\theta}$	Un classifieur construit à partir de $\mathcal{L}$ avec graine aléatoire θ
φ_B	Le classifieur optimal de Bayes
$\psi_{\mathcal{L},\theta_1,\dots,\theta_M}$	Un ensemble de M modèles construits à partir de $\mathcal{L}$ et graine $\theta_1, \dots, \theta_M$
$\mathcal{Q}$	Un ensemble $\mathcal{Q} \subseteq \mathcal{S}$ de coupures d'une certaine structure
$\mathcal{Q}(X_j)$	L'ensemble $\mathcal{Q}(X_j) \subseteq \mathcal{Q}$ des coupures univariées définies sur X_j
$\rho(\mathbf{x})$	Le coefficient de corrélation entre les prédiction en $X = \mathbf{x}$ de deux modèles aléatoires
s	Une coupure
s^*	La coupure optimale
s_j^*	La coupure optimale par rapport à X_j
s_j^v	La coupure optimale $(\{\mathbf{x} \mid x_j \leq v\}, \{\mathbf{x} > v\})$ définie sur X_j avec seuil de discrétisation v
s_t	La coupure du nœud t
$\tilde{s}_t^j$	La coupure suppléante optimale s_t par rapport à X_j
$\mathcal{S}$	L'ensemble des coupures s possibles
$\sigma^2_{\mathcal{L},\theta_m}(\mathbf{x})$	La variance de la prédiction en $X = \mathbf{x}$ of $\varphi_{\mathcal{L},\theta_m}$
t	Un nœud d'un arbre
t_L	Fils gauche du nœud t
t_R	Fils droit du nœud t
θ	Un vecteur d'hyper-paramètres
θ^*	Le vecteur d'hyper-paramètres optimal
$\hat{\theta}^*$	The approximately optimal hyper-parameters
θ_m	La graine du $m^{\text{ième}}$ modèle d'un ensemble
v	Seuil de discrétisation d'une coupure binaire
v_k	La $k^{\text{ième}}$ valeur d'une variable ordonnée, lorsque l'échantillon est ordonné croissant
v'_k	Le point de coupe médian entre v_k et v_{k+1}
V	L'ensemble $\{X_1, \dots, X_p\}$ des variables de $\mathcal{L}$
V^{-j}	$V \setminus \{X_j\}$
$\mathbb{V}$	Variance
$\mathbf{w}$	Vecteur des poids de l'hyperplan de séparation
$\mathbf{x}$	Un individu, une observation $(x_1, \dots, x_p)$
$\mathbf{x}_i$	Le $i^{\text{ième}}$ individu de $\mathcal{L}$
x_j	La valeur de la variable X_j pour l'individu $\mathbf{x}$

$\mathbf{X}$	La matrice $N \times p$ représentant les valeurs des N individus pour chacune des p variables
X_j	La $j^{\text{ième}}$ variable
X	Le vecteur aléatoire $(X_1, \dots, X_p)$
$\mathcal{X}_j$	L'espace de la variable X_j
$\mathcal{X}$	L'espace des variables $\mathcal{X}_1 \times \dots \times \mathcal{X}_p$
$\mathcal{X}_t$	Le sous-espace $\mathcal{X}_t \subseteq \mathcal{X}$ représenté par le nœud t
y	Valeur de l'observation de sortie Y
$\hat{y}_t$	L'étiquette du nœud t
$\hat{y}_t^*$	L'étiquette optimale du nœud t
$\mathbf{y}$	Les valeurs de sorties $(y_1, \dots, y_N)$
Y	La variable de réponse
$\mathcal{Y}$	L'espace de la variable Y

ACRONYME

BIBLIOGRAPHIE

Achlioptas, Dimitris

2003 « Database-friendly random projections : Johnson-Lindenstrauss with binary coins », *Journal of Computer and System Sciences*, 66, 4, p. 671–687, ISSN : 00220000, DOI : 10.1016/S0022-0000(03)00025-4. (Cf. p. 62, 63.)

Andrieu, Christophe, Nando De Freitas, Arnaud Doucet et Michael I. Jordan

2003 « An introduction to MCMC for machine learning », *Machine Learning*, 50, 1-2, p. 5–43, ISSN : 08856125, DOI : 10.1023/A:1020281327116, arXiv : 1109.4435v1. (Cf. p. 91.)

Aronszajn, N.

1950 « Theory of reproducing kernels », *Transactions of the American Mathematical Society*, 68, 3 (mar. 1950), p. 337–337, ISSN : 0002-9947, DOI : 10.1090/S0002-9947-1950-0051437-7, http://www.ams.org/tran/1950-68-03/S0002-9947-1950-0051437-7/. (Cf. p. 23.)

Barros, Rodrigo C., Ricardo Cerri, Pablo a. Jaskowiak et André C P L F De Carvalho

2011 « A bottom-up oblique decision tree induction algorithm », *International Conference on Intelligent Systems Design and Applications, ISDA*, p. 450–456, ISSN : 21647143, DOI : 10.1109/ISDA.2011.6121697. (Cf. p. 61.)

Batista, Gustavo E. A. P. A, Ronaldo C Prati et Maria Carolina Monard

2004 « A study of the behavior of several methods for balancing machine learning training data », *ACM SIGKDD Explorations Newsletter*, 6, 1, p. 20, ISSN : 19310145, DOI : 10.1145/1007730.1007735. (Cf. p. 74.)

Bell, Robert M, Yehuda Koren et Chris Volinsky

2007 « The BellKor solution to the Netflix Prize », *KorBell Teams Report to Netflix*, 2, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.9009%5C&rep=rep1%5C&type=pdf. (Cf. p. 39.)

Bell, Robert M., Yehuda Koren et Chris Volinsky

2008 *The BellKor 2008 Solution to the Netflix Prize*, rapp. tech., DOI : 10.1111/j.1754-4505.2009.00107.x. (Cf. p. 39.)

Bengio, Yoshua, Ian J. Goodfellow et Aaron Courville

2015 *Deep Learning*, MIT Press, http://www.iro.umontreal.ca/~bengioy/dlbook. (Cf. p. 27.)

Biau, Gérard

2010 « Analysis of a Random Forests Model », *Mathematics Subject Classification*, p. 1–40, arXiv : arXiv :1005.0208v3. (Cf. p. 14, 57.)

Biau, Gérard, Luc Devroye et Gábor Lugosi

2008 « Consistency of random forests and other averaging classifiers », 506778, ISSN : 1532-4435, DOI : 10.1145/1390681.1442799, http://eprints.pascal-network.org/archive/00004623/. (Cf. p. 55, 61.)

Biau, Gérard et Erwan Scornet

2015 « A random forest guided tour ». (Cf. p. 56.)

Bishop, Christopher M.

2006 *Pattern Recognition and Machine Learning*, ISBN : 9780387310732, DOI : 10.1641/B580519, arXiv : 0-387-31073-8, http://www.library.wisc.edu/selectedtocs/bg0137.pdf. (Cf. p. 6.)

Breiman, Leo

1996 « Out-of-bag estimation », p. 1–13, ISSN : 00313203, DOI : 10.1016/j.patcog.2009.05.010, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.3712%5C&rep=rep1%5C&type=pdf. (Cf. p. 56.)

1998 « Arcing classifier (with discussion and a rejoinder by the author) », *The Annals of Statistics*, 26, 3, p. 801–849, ISSN : 2168-8966, DOI : 10.1214/aos/1024691079. (Cf. p. 35.)

2001 « Random forests », *Machine Learning*, 45, 1, p. 5–32, ISSN : 08856125, DOI : 10.1023/A :1010933404324, arXiv : /dx.doi.org/10.1023\%2FA\%3A1010933404324 [http :]. (Cf. p. 51, 56, 61, 74, 79, 85.)

2004 « Consistency for a simple model of random forests », *Technical Report 670*, http://scholar.google.com/scholar?hl=en%5C&btnG=Search%5C&q=intitle :CONSISTENCY+FOR+A+SIMPLE+MODEL+OF+RANDOM+FORESTS%5C#0. (Cf. p. 14, 57.)

Breiman, Leo, Jerome H Friedman, R A Olshen et C J Stone

1984 *Classification and Regression Trees*, t. p, p. 368, ISBN : 0412048418, http://scholar.google.com/scholar?hl=en%5C&btnG=Search%5C&q=intitle :Classification+and+regression+trees%5C#0. (Cf. p. 43, 45, 46, 50, 53, 55.)

Brezis, H

2011 *Functional analysis, Sobolev spaces and partial differential equations*, ISBN : 9780387709130, DOI : 10.1007/978-0-387-70914-7, http://link.springer.com/content/pdf/10.1007/978-0-387-70914-7.pdf. (Cf. p. 99.)

CALDERS, Toon et Szymon JAROSZEWICZ

2007 « Efficient AUC optimization for classification », *Knowledge Discovery in Databases*, p. 42–53, ISSN : 0302-9743, DOI : 10.1007/978-3-540-749756-9_8, http://link.springer.com/chapter/10.1007/978-3-540-74976-9%5C_8. (Cf. p. 69.)

CARUANA, Rich, Alexandru NICULESCU-MIZIL, Geoff CREW et Alex KSIKES

2004 « Ensemble Selection from Librairies of Models », *ICML*, 34, p. 1–21, DOI : 10.1145/1015330.1015432, papers://5e3e5e59-48a2-47c1-b6b1-a778137d3ec1/Paper/p1953. (Cf. p. 40.)

CHAWLA, Nitesh V., Kevin W. BOWYER, Lawrence O. HALL et W. Philip KEGELMEYER

2002 « SMOTE : Synthetic Minority Over-sampling Technique », 16, p. 321–357. (Cf. p. 77.)

CHEN, Chao, Andy LIAW et Leo BREIMAN

2004 « Using random forest to learn imbalanced data », *University of California, Berkeley*, 1999, p. 1–12. (Cf. p. 74.)

CHERKASSKY, Vladimir et Filip MULIER

2007 *Learning from Data*, ISBN : 9780471681823. (Cf. p. 6.)

CHIPMAN, Hugh a., Edward I. GEORGE et Robert E. MCCULLOCH

1998 « Bayesian CART Model Search », *Journal of the American Statistical Association*, 93, 443, p. 935–948, ISSN : 0162-1459, DOI : 10.2307/2669832, http://www.jstor.org/stable/2669832$%5Cbackslash$nhttp://www.jstor.org.libproxy1.nus.edu.sg/stable/pdfplus/2669832.pdf?acceptTC=true. (Cf. p. 63, 64.)

2008 « BART : Bayesian additive regression trees », *Annals of Applied Statistics*, 6, 1, p. 266–298, ISSN : 19326157, DOI : 10.1214/09-AOAS285, arXiv : 0806.3286. (Cf. p. 64.)

CISS, Saïp

2013 *Forêts uniformément aléatoires*, thèse de doct., p. 1–63. (Cf. p. 51, 58.)

CLÉMENÇON, Stéphan, Gábor LUGOSI et Nicolas VAYATIS

2006 « Ranking and empirical minimization of U-statistics », arXiv : 0603123 [math], http://arxiv.org/abs/math/0603123. (Cf. p. 68.)

CORTES, Corinna et Vladimir VAPNIK

1995 « Support-vector networks », *Machine Learning*, 20, 3 (sept. 1995), p. 273–297, ISSN : 0885-6125, DOI : 10.1007/BF00994018, http://link.springer.com/10.1007/BF00994018. (Cf. p. 18.)

Cover, Thomas M.

1965 « Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition », *IEEE Transactions on Electronic Computers*, EC-14, 3, p. 326–334, ISSN : 0367-7508, DOI : 10.1109/PGEC.1965.264137, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4038449. (Cf. p. 21.)

Criminisi, Antonio, Jamie Shotton et Ender Konukoglu

2012 « Decision Forests : A Unified Framework for Classification , Regression , Density Estimation , Manifold Learning and Semi-Supervised Learning », 7, 2011, p. 81–227, DOI : 10.1561/0600000035. (Cf. p. 56.)

Cutler, Adele et G Zhao

2001 « PERT - Perfect Random Tree Ensembles », *Computing Science and Statistics*, 33, p. 490–497, http://www.interfacesymposia.org/I01/I2001Proceedings/ACutler/ACutler.pdf. (Cf. p. 51.)

Cybenko, George

1989 « Approximation by superpositions of a sigmoidal function », *Mathematics of Control, Signals, and Systems*, 2, 4 (déc. 1989), p. 303–314, ISSN : 0932-4194, DOI : 10.1007/BF02551274, http://link.springer.com/10.1007/BF02551274. (Cf. p. 26, 27, 99.)

Denil, Misha, David Matheson et Nando De Freitas

2013 « Consistency of Online Random Forests », arXiv : arXiv:1302.4853v2. (Cf. p. 71.)

Devroye, Luc, László Györfi et Gábor Lugosi

1997 *A Probabilistic Theory of Pattern Recognition*, p. 661, ISBN : 9780387946184, DOI : 10.1007/978-1-4612-0711-5. (Cf. p. 6, 10, 55, 95.)

Domingos, Pedro

1999 « MetaCost : A General Method for Making Classifiers Cost-Sensitive », *Fifth International Conference on Knowledge Discovery and Data Mining*. (Cf. p. 80, 81.)

Dupret, Georges et Masato Koda

2001 « Bootstrap re-sampling for unbalanced data in supervised learning », *European Journal of Operational Research*, 134, 1, p. 141–156, ISSN : 03772217, DOI : 10.1016/S0377-2217(00)00244-7. (Cf. p. 75.)

Efron, Bradley, Trevor Hastie, Iain Johnstone et Robert Tibshirani

2003 « Least Angle Regression », *Annals of Statistics*, p. 1–44. (Cf. p. 16.)

Fernández-Delgado, Manuel, Eva Cernadas, Senén Barro et Dinani Amorim

2014 « Do we Need Hundreds of Classifiers to Solve Real World Classification Problems ? », *Journal of Machine Learning Research*, 15, p. 3133–3181. (Cf. p. 57.)

Freund, Yoav, Raj Iyer, Robert E Schapire et Yoram Singer

2003 « An Efficient Boosting Algorithm for Combining Preferences », *The Journal of Machine Learning Research*, 4, p. 933–969, issn : 15324435, doi : 10.1162/jmlr.2003.4.6.933. (Cf. p. 35.)

Geman, Stuart, Elie Bienenstock et René Doursat

1992 *Neural Networks and the Bias/Variance Dilemma*, doi : 10.1162/neco.1992.4.1.1. (Cf. p. 31.)

Geurts, Pierre, Damien Ernst et Louis Wehenkel

2006 « Extremely randomized trees », *Machine Learning*, 63, 1, p. 3–42, issn : 08856125, doi : 10.1007/s10994-006-6226-1. (Cf. p. 50, 57.)

Giles, M. et P. Glasserman

2005 « Smoking Adjoints : fast evaluation of Greeks in Monte Carlo calculations », 05, p. 1–15, http://eprints.maths.ox.ac.uk/1138/. (Cf. p. 28.)

Gonçalves, Paulo

2015 *Echantillonneurs de Monte Carlo Application à l'Estimation Statistique Bayésienne*, rapp. tech. (Cf. p. 91.)

Han, Hui, Wy Wen-Yuan Wy Wang et Bh Bing-huan Mao

2005 « Borderline-SMOTE : A new over-sampling method in imbalanced data sets learning », *Advances in intelligent computing*, 17, 12, p. 878–887, issn : 1941-0506, doi : 10.1007/11538059_91, http://dx.doi.org/10.1007/11538059%5C_91$%5Cbackslash$nhttp://link.springer.com/chapter/10.1007/11538059%5C_91$%5Cbackslash$nhttp://link.springer.com/chapter/10.1007/11538059%5C_91$%5Cbackslash$nhttp://www.bmva.org/bmvc/1988/avc-88-023.html. (Cf. p. 77.)

Hart, P.

1968 « The condensed nearest neighbor rule (Corresp.) » *IEEE Transactions on Information Theory*, 14, 3, p. 1966–1967, issn : 0018-9448, doi : 10.1109/TIT.1968.1054155. (Cf. p. 73.)

Hastie, T et R Tibshirani

2000 « Bayesian backfitting », *Statistical Science*, 15, 3, p. 196–213, issn : 0883-4237, %3CGo%20to%20ISI%3E://000166404100002. (Cf. p. 65.)

Hastie, Trevor, Robert Tibshirani et Jerome Friedman

2008 *The Elements of Statistical Learning*, 3e éd., http://statweb.stanford.edu/~tibs/ElemStatLearn/printings/ESLII%5C_print10.pdf. (Cf. p. 6, 11, 13, 14, 35, 36.)

He, Haibo, Yang Bai, Edwardo a. Garcia et Shutao Li

2008 « ADASYN : Adaptive synthetic sampling approach for imbalanced learning », *Proceedings of the International Joint Conference on Neural Networks*, 3, p. 1322–1328, ISSN : 1098-7576, DOI : 10.1109/IJCNN.2008.4633969. (Cf. p. 78.)

He, Haibo et Edwardo A. Garcia

2009 « Learning from Imbalanced Data », *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 21, 9, http://www.aaai.org/Papers/Workshops/2000/WS-00-05/WS00-05-003.pdf. (Cf. p. 74.)

Hido, Shohei, Hisashi Kashima et Yutaka Takahashi

2009 « Roughly balanced Bagging for Imbalanced data », *Statistical Analysis and Data Mining*, 2, 5-6, p. 412–426, ISSN : 19321872, DOI : 10.1002/sam.10061. (Cf. p. 74.)

Ho, Tin Kam

1998 « The random subspace method for constructing decision forests », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 8, p. 832–844, ISSN : 01628828, DOI : 10.1109/34.709601. (Cf. p. 50, 56.)

Hoeting, Jennifer a, David Madigan, Adrian E Raftery et C T Volinsky

1999 « Bayesian model averaging : a tutorial », *Statistical Science*, 14, 4, p. 382–417, ISSN : 08834237, DOI : 10.2307/2676803, http://www.jstor.org/stable/2676803. (Cf. p. 35.)

Homescu, Cristian

2011 « Adjoints and Automatic (Algorithmic) Differentiation in Computational Finance », p. 23, arXiv : 1107.1831, http://arxiv.org/abs/1107.1831. (Cf. p. 28.)

Hornik, Kurt

1991 « Approximation capabilities of multilayer feedforward networks », *Neural Networks*, 4, 2, p. 251–257, ISSN : 08936080, DOI : 10.1016/0893-6080(91)90009-T. (Cf. p. 26.)

Huber, Peter J.

1964 « Robust Estimation of a Location Parameter », *The Annals of Mathematical Statistics*, 35, 1, p. 73–101, ISSN : 0003-4851, DOI : 10.1214/aoms/1177733256. (Cf. p. 37.)

Ishwaran, Hemant

2007 « Variable importance in binary regression trees and forests », 1, p. 519–537, ISSN : 1935-7524, DOI : 10.1214/07-EJS039, arXiv : 0711.2434, http ://arxiv.org/abs/0711.2434. (Cf. p. 54.)

2014 « The effect of splitting on random forests », *Machine Learning*, 99, 1, p. 75–118, ISSN : 0885-6125, DOI : 10.1007/s10994-014-5451-2, http ://link.springer.com/10.1007/s10994-014-5451-2. (Cf. p. 51.)

Ishwaran, Hemant, Udaya B. Kogalur, Eugene H. Blackstone et Michael S. Lauer

2008 « Random survival forests », *Annals of Applied Statistics*, 2, 3, p. 841–860, ISSN : 19326157, DOI : 10.1214/08-AOAS169, arXiv : 0811.1645. (Cf. p. 44.)

Ishwaran, Hemant, Udaya B. Kogalur, Eiran Z. Gorodeski, Andy J. Minn et Michael S. Lauer

2010 « High-Dimensional Variable Selection for Survival Data », *Journal of the American Statistical Association*, 105, 489, p. 205–217, ISSN : 0162-1459, DOI : 10.1198/jasa.2009.tm08622. (Cf. p. 54.)

Jahrer, Michael, Andreas Töscher et Robert Legenstein

2010 « Combining Predictions for an accurate Recommender System », ISSN : 1450300553, DOI : 10.1145/1835804.1835893, http ://eprints.pascal-network.org/archive/00006084/. (Cf. p. 39.)

Knuth, Donald E.

1992 « Two notes on notation », p. 1–23, ISSN : 00029890, DOI : 10.2307/2325085, arXiv : 9205211 [math], http ://arxiv.org/abs/math/9205211. (Cf. p. 48.)

Kubat, M. et S. Matwin

1997 « Addressing the curse of imbalanced training sets : one-sided selection », *Icml*, 97, p. 179–186, DOI : 10.1.1.43.4487. (Cf. p. 73, 74.)

Lakshminarayanan, Balaji, Daniel M Roy et Yee Whye Teh

2015 « Particle Gibbs for Bayesian Additive Regression Trees », *Proceedings of the 18th International Con- ference on Artificial Intelligence and Statistics*, 38. (Cf. p. 66.)

Lakshminarayanan, Balaji, DM Roy et YW Teh

2014 « Mondrian Forests : Efficient Online Random Forests », *arXiv preprint arXiv :1406.2673*, arXiv : arXiv :1406.2673v1, http ://arxiv.org/abs/1406.2673. (Cf. p. 71.)

Lam, Patrick

2008 « MCMC Methods : Gibbs Sampling and the Metropolis-Hastings Algorithm ». (Cf. p. 91.)

Laurikkala, Jorma

2001 « Improving Identification of Difficult Small Classes by Balancing Class Distribution », *8th Conference on Artificial Intelligence in Medicine in Europe*, p. 63–66, DOI : 10.1007/3-540-48229-6_9. (Cf. p. 74.)

Li, Ping, Trevor J. Hastie et Kenneth W. Church

2006 « Very sparse random projections », *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, p. 287, DOI : 10.1145/1150402.1150436, http://portal.acm.org/citation.cfm?doid=1150402.1150436. (Cf. p. 63.)

Louppe, G

2014 *Understanding random forests from theory to practice*, thèse de doct., arXiv : 1407.7502, http://arxiv.org/abs/1407.7502. (Cf. p. 32, 33, 43, 47, 49.)

Lumijärvi, Janne, Jorma Laurikkala et Martti Juhola

2004 « A comparison of different heterogeneous proximity functions and Euclidean distance. » *Studies in health technology and informatics*, 107, Pt 2, p. 1362–1366, ISSN : 0926-9630. (Cf. p. 78, 84.)

Meinshausen, Nicolai

2006 « Quantile Regression Forests », *Journal of Machine Learning Research*, 7, p. 983–999, ISSN : 15410420, DOI : 10.1111/j.1541-0420.2010.01521.x. (Cf. p. 44.)

Menze, Bjoern H., B. Michael Kelm, Daniel N. Splitthoff, Ullrich Koethe et Fred A. Hamprecht

2011 « On oblique random forests », in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, PART 2, t. 6912 LNAI, p. 453–469, ISBN : 9783642237829, DOI : 10.1007/978-3-642-23783-6_29. (Cf. p. 61.)

Mercer, J.

1908 « Functions of Positive and Negative Type and their Connection with the Theory of Integral Equations », *Philosophical Transactions of the Royal Society of London*, Series A, (cf. p. 23.)

Mingers, J.

1989 « An Empirical Comparison of Pruning Methods for Decision Tree Induction », 243, p. 227–243, ISSN : 0885-6125, DOI : 10.1023/A:1022604100933, http://dx.doi.org/10.1023/A:1022604100933. (Cf. p. 55.)

Minka, Thomas P.

2002 « Bayesian model averaging is not model combination », 2000, p. 1–2. (Cf. p. 34.)

Monteith, Kristine, James L. Carroll, Kevin Seppi et Tony Martinez
2011 « Turning Bayesian model averaging into Bayesian model combination », *The 2011 International Joint Conference on Neural Networks*, p. 2657–2663, ISSN : 2161-4393, DOI : 10.1109/IJCNN.2011.6033566. (Cf. p. 34.)

Nielsen, Michael A.
2015 *Neural Networks and Deep Learning*, Determination Press, http://neuralnetworksanddeeplearning.com/. (Cf. p. 27.)

Park, Trevor et George Casella
2008 « The Bayesian Lasso », *Journal of the American Statistical Association*, 103, 482, p. 681–686, ISSN : 0162-1459, DOI : 10.1198/016214508000000337. (Cf. p. 16.)

Platt, John C.
1998 « Sequential Minimal Optimization : A Fast Algorithm for Training Support Vector Machines », http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.4376. (Cf. p. 21.)

Ram, Parikshit, Atlanta Ga, Alexander G Gray, I Pattern Recognition Models et G Probability
2011 « Density Estimation Trees », *Methods*, p. 627–635, DOI : 10.1145/2020408.2020507, http://users.cis.fiu.edu/~lzhen001/activities/KDD2011Program/docs/p627.pdf. (Cf. p. 44.)

Ren, Shaoqing, Xudong Cao, Yichen Wei et Jian Sun
2015 « Global Refinement of Random Forest », *CVPR*. (Cf. p. 67, 68.)

Robert, Christian P et George Casella
2004 *Monte Carlo Statistical Methods*, t. 95, p. 1089–1100, ISBN : 9781441919397, DOI : 10.1007/978-1-4757-4145-2, http://books.google.com/books?hl=en%5C&lr=%5C&id=HfhGAxn5GugC%5C&oi=fnd%5C&pg=PR9%5C&dq=Monte+Carlo+Statistical+Methods%5C&ots=Bzt%5C_0RaSUz%5C&sig=VCjwbF3WjleVSr63xpMO5oyZXpI$%5Cbackslash$nhttp://link.springer.com/10.1007/978-1-4757-4145-2. (Cf. p. 91.)

Rodriguez, Yanet, Bernard De Baets, Maria M Garcia, Carlos Morell et Ricardo Grau
2008 « A Correlation-Based Distance Function for Nearest Neighbor Classification », Dvdm, p. 284–291. (Cf. p. 78, 84.)

Rojas, Raúl
1996 *Neural Networks*, 1, Springer Berlin Heidelberg, Berlin, Heidelberg, t. 7, p. 509, ISBN : 9783540605058, DOI : 10.1007/978-3-642-61068-4, http://page.mi.fu-berlin.de/rojas/neural/index.html.html%20http://link.springer.com/10.1007/978-3-642-61068-4. (Cf. p. 27.)

Saffari, Amir, Christian Leistner, Jakob Santner, Martin Godec et Horst Bischof

2009 « On-line random forests », in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*, p. 1393–1400, ISBN : 9781424444427, DOI : 10.1109/ICCVW.2009.5457447. (Cf. p. 71.)

Schapire, Robert E.

1990 « The strength of weak learnability », *Machine Learning*, 5, 2, p. 197–227, ISSN : 08856125, DOI : 10.1007/BF00116037. (Cf. p. 35.)

Scholkopf, Bernhard, Ralf Herbrich et Alex J. Smola

2001 *A Generalized Representer Theorem*, sous la dir. de David Helmbold et Bob Williamson, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, t. 2111, ISBN : 9783540423430, DOI : 10.1007/3-540-44581-1, http://www.springerlink.com/index/10.1007/3-540-44581-1. (Cf. p. 25.)

Scornet, Erwan

2014 « On the asymptotics of random forests », arXiv : 1409.2090, http://arxiv.org/abs/1409.2090. (Cf. p. 61.)

Scornet, Erwan, Gérard Biau et Jean-Philippe Vert

2014 « Consistency of Random Forests », arXiv : arXiv:1405.2881v1, http://arxiv.org/abs/1405.2881. (Cf. p. 59.)

Shannon, C.

1948 « A mathematical theory of communication », *Bell System Technology Journal*, 27, 379 :423, 623–656, ISSN : 07246811, DOI : 10.1145/584091.584093, arXiv : 9411012 [chao-dyn]. (Cf. p. 46.)

Shotton, Jamie, Sebastian Nowozin, Toby Sharp, John Winn, Pushmeet Kohli et Antonio Criminisi

2013 « Decision Jungles : Compact and Rich Models for Classification », *Advances in Neural …*, p. 1–9, http://papers.nips.cc/paper/5199-decision-jungles-compact-and-rich-models-for-classification. (Cf. p. 45.)

Stekhoven, Daniel J. et Peter Bühlmann

2012 « Missforest-Non-parametric missing value imputation for mixed-type data », *Bioinformatics*, 28, 1, p. 112–118, ISSN : 13674803, DOI : 10.1093/bioinformatics/btr597, arXiv : 1105.0828. (Cf. p. 85.)

Taddy, Matt, Chun-sheng Chen, Chunschen Ebay Com et Mitch Wyle

2015 « Bayesian and Empirical Bayesian Forests », 37, arXiv : arXiv:1502.02312v2. (Cf. p. 66.)

Tomita, Tyler M, Mauro Maggioni et Joshua T Vogelstein

2015 « Randomer Forests », p. 1–9, arXiv : arXiv : 1506 . 03410v1. (Cf. p. 62, 63.)

Töscher, Andreas, Michael Jahrer et Robert M. Bell

2009 *The BigChaos Solution to the Netflix Grand Prize*, rapp. tech., p. 1–52. (Cf. p. 39.)

Tsybakov, A et V Zaiats

2009 *Introduction to nonparametric estimation*, isbn : 9780387790510, http ://link.springer.com/content/pdf/10.1007/b13794.pdf. (Cf. p. 6.)

Van der Laan, Mark J, Eric C Polley et Alan E Hubbard

2007 « Super Learner », *Statistical applications in genetics and molecular biology*, 6, 1, Article25, issn : 1544-6115, doi : 10.2202/1544-6115.1309. (Cf. p. 39, 40.)

Vapnik, Vladimir

1998 *Statistical Learning Theory*, isbn : 9780471030034. (Cf. p. 8.)

2000 *The Nature of Statistical Learning Theory*, p. 314, isbn : 0387987800, http ://books.google.com/books?hl=es%5C&id=sna9BaxVbj8C%5C&pgis=1$%5Cbackslash$nhttp ://infoscience.epfl.ch/record/82790/files/com02-04.pdf. (Cf. p. 8.)

Wasserman, Larry

2004 *All of Statistics*, Springer Texts in Statistics, Springer New York, New York, NY, isbn : 9781441923226, doi : 10 . 1007 / 978 - 0 - 387-21736-9, http ://link.springer.com/10.1007/978-0-387-21736-9. (Cf. p. 6.)

Wilson, D. Randall et Tony R. Martinez

1997 « Improved heterogeneous distance functions », *Journal of Artificial Intelligence Research*, 6, p. 1–34, issn : 10769757, doi : 10.1613/jair.346, arXiv : 9701101 [cs]. (Cf. p. 78, 84.)

Wilson, Dennis L.

1972 « Asymptotic Properties of Nearest Neighbor Rules Using Edited Data », *IEEE Transactions on Systems, Man, and Cybernetics*, 2, 3, p. 408–421, issn : 0018-9472, doi : 10 . 1109 / TSMC . 1972 . 4309137. (Cf. p. 74.)

Wolpert, David H.

1992 « Stacked generalization », *Neural Networks*, 5, 2, p. 241–259, issn : 08936080, doi : 10.1016/S0893-6080(05)80023-1. (Cf. p. 39.)

1996 « The Existence of A Priori Distinctions Between Learning Algorithms », *Neural Computation*, 8, 7, p. 1391–1420, issn : 0899-7667, doi : 10.1162/neco.1996.8.7.1391. (Cf. p. 10.)

Wolpert, David H. et William G. Macready

1999 « Efficient method to estimate Bagging's generalization error », *Machine Learning*, 35, 1, p. 41–55, ISSN : 08856125, DOI : 10.1023/A :1007519102914. (Cf. p. 57.)

Xu, Wei, Xi Chen et Thomas F Coleman

2014 « The Efficient Application of Automatic Differentiation for Computing Gradients in Financial Applications * », p. 1–24. (Cf. p. 28.)