Construction de noyaux pour l'apprentissage supervisé à partir d'arbres aléatoires

Vincent Pisetta*, Pierre-Emmanuel Jouve**
Djamel.A Zighed***

*RITHME, 59 bd Vivier Merle 69003 Lyon
vpisetta@rithme.eu,
**FENICS, 59 bd Vivier Merle 69003 Lyon
pjouve@fenics.com
***Laboratoire ERIC, 5 av. Pierre Mendès France 69500 Bron
abdelkader.zighed@univ-lyon2.fr

Résumé. Nous montrons qu'un ensemble d'arbres de décision avec une composante aléatoire permet de construire un noyau efficace destiné à l'apprentissage supervisé. Nous étudions théoriquement les propriétés d'un tel noyau et montrons que sous des conditions très souvent rencontrées en pratique, il existe une séparabilité linéaire entre exemples de classes distinctes dans l'espace induit par celui-ci. Parallèlement, nous observons également que le classique *vote à la majorité* d'un ensemble d'arbres est un hyperplan (sans garantie d'optimalité) dans l'espace induit par le noyau. Enfin, comme le montrent nos expérimentations, l'utilisation conjointe d'un ensemble d'arbres et d'un séparateur à vaste marge (SVM) aboutit à des résultats extrêmement encourageants.

1 Introduction

Parmi les techniques d'apprentissage statistique les plus performantes se trouvent les méthodes à base de noyaux dont le représentant le plus célèbre est très certainement le Séparateur à Vaste Marge (Vapnik, 1996). Son emploi, motivé initialement par les premiers résultats théoriques de l'apprentissage statistique s'est encore plus largement répandu suite aux nombreux succès empiriques apportés par cet apprenant (Pavlidis et al. (2004); Markowska-Kaczmar et Kubacki (2005); Polat et Günes (2007)). L'une des particularités du SVM est d'utiliser la fameuse *astuce du noyau* pour introduire de la non-linéarité dans la frontière de décision sans augmenter la complexité algorithmique de l'apprentissage. Ainsi, le choix du noyau est sans doute le paramètre le plus important de l'algorithme, différents noyaux pouvant aboutir à des résultats très différents.

Plusieurs indicateurs ont été proposés dans la littérature afin d'évaluer la qualité d'un noyau a priori, autrement dit avant même l'exécution de l'apprentissage. Le plus connu est probablement le *Kernel Target Alignment* (KTA) (Cristianini et al., 2002), bien que d'autres tels que FSM (Nguyen et Ho, 2008) ou la *polarisation* (Baram., 2005) aient également montré des résultats intéressants. L'intérêt de ces indicateurs est qu'ils permettent d'évaluer la pertinence

d'un noyau de manière relativement rapide avec une complexité en $O(n^2)$, où n est le nombre d'exemples, et ce, sans recourir au calcul du modèle. Outre l'évaluation, l'intérêt pratique de ces mesures est qu'elles premettent de construire de manière supervisée un noyau adapté à un jeu de données spécifique. Le principal paradigme utilisant ce principe est le *Multiple Kernel Learning* (MKL) basé sur l'hypothèse que la combinaison de plusieurs noyaux peut aboutir à un résultat plus intéressant que la seule sélection du meilleur. Ainsi, l'objectif du MKL est de chercher une combinaison linéaire (à coefficients positifs) de plusieurs noyaux distincts de manière à maximiser KTA (ou n'importe quel autre indicateur approprié) via par exemple la programmation semidéfinie (Lanckriet et al., 2004) ou l'optimisation séquentielle (Bach et al., 2004). D'autres méthodes (Sonnenburg et al. (2006); Rakotomamonjy et al. (2008)) incorporent le problème d'optimisation du noyau directement dans le problème d'optimisation global du SVM. Toutes ces stratégies ont montré d'excellents résultats et une capacité d'apprentissage très élevée. Cependant, elles souffrent de plusieurs problèmes. Tout d'abord, elles ne résolvent pas le problème du choix des noyaux initiaux (ceux qui doivent être combinés). Généralement, des noyaux gaussiens sont utilisés bien qu'aucune justification théorique n'ait été apportée à ce choix. Ensuite, elles souffrent toutes d'une complexité algorithmique élevée (par exemple, la méthode de (Lanckriet et al., 2004) a une compexité en $O(n^6)$). Enfin, elles supposent toutes que les descripteurs utilisés pour l'apprentissage sont continus alors qu'il arrive fréquemment que des variables catégorielles soient présentes dans les données réelles.

Nous proposons dans cet article d'utiliser des arbres de décision aléatoires afin de générer des noyaux efficaces pour l'apprentissage. Les célèbres Forêts aléatoires (Breiman, 2001) sont un cas particulier de cette méthodologie et se révèlent en fait être des constructrices de noyaux très performantes. En effet, elles ont au fond une motivation similaire au principe du MKL, à savoir l'utilisation d'un ensemble (de classifieurs) plutôt que la recherche du meilleur. En section 2, après avoir brièvement rappelé quelques généralités sur l'apprentissage par SVM, nous décrivons un cadre théorique permettant d'obtenir un bon noyau pour l'apprentissage. Après avoir rappelé le mécanisme des forêts aléatoires, nous verrons comment construire des noyaux à partir de celles-ci (section 3). Nous montrerons théoriquement que, sous des conditions très faibles, le noyau construit par un tel ensemble induit un espace contenant un séparateur linéaire entre exemples de classes distinctes (section 3 et 4). Nous montrerons également que la célèbre décision du *vote à la majorité* n'est rien d'autre qu'un hyperplan dans l'espace généré par la Forêt (section 4). Toutes ces observations suggèrent que l'utilisation d'un SVM dans l'espace généré par un ensemble d'arbres aléatoires peut se révéler intéressante. En section 5, nous montrons expérimentalement le bien fondé de cette hypothèse. Enfin, en section 6, nous concluons.

2 Séparateur à vaste marge et qualité d'un noyau

2.1 Cadre général du SVM

Nous considérons dans cet article le cadre de la classification supervisée à deux classes. Formellement, étant donné n exemples $x_1, ..., x_n$ décrits dans un espace à d dimensions noté X et le vecteur correspondant des réponses (ou classes, étiquettes) $y = \{y_1, ..., y_n\}$ avec $y_i = \{-1, +1\}$, l'objectif de la classification supervisée est trouver une fonction f telle qu'une

fonction dite *fonction de perte* $L(y, f(x))$ soit minimale. Généralement, cette dernière est choisie comme la fonction des moindres carrés $L(y_i, f(x_i)) = (y_i - f(x_i))^2$ ou la fonction *hinge loss*, $L(y_i, f(x_i)) = max(0, 1 - y_i f(x_i))$. Un spectre très large de méthodes est dédié à la résolution de ce problème. Parmi celles-ci, le SVM fait partie des plus performantes. Il consiste à chercher la fonction f prédéfinie comme $f(x) = signe(\langle w, \varphi(x)\rangle + b)$ en trouvant la solution optimale au problème suivant $(P1)$:

$$\begin{array}{c} \min\limits_{w\in\varphi(x), b\in\Re, \xi\in\Re^n} 0.5\langle w, w\rangle + C\sum\limits_{i=1}^{n}\xi_i \\ s.c \quad y_i(\langle w, \varphi(x_i)\rangle + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{array}$$

où $C > 0$ est le paramètre de régularisation et $\varphi(x)$ est obtenue à partir du *mapping* $\varphi : X \rightarrow \Im$. $\Im$ est classiquement dénomé *espace des caractéristiques* et est un espace de Hilbert muni du produit scalaire $\langle ., .\rangle$ (Schölkopf et Smola, 2001). Il est à noter que $\Im$ peut être de dimension infinie, c'est pourquoi l'obtention de la solution du programme $(P1)$ passe généralement par sa forme duale $(P2)$:

$$\begin{array}{c} \min\limits_{\alpha\in\Re^n} 0.5\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\alpha_i\alpha_j y_i y_j \tilde{K}(x_i, x_j) - \sum\limits_{i=1}^{n}\alpha_i \\ s.c \quad 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{n}\alpha_i y_i = 0 \end{array}$$

$\tilde{K}$ représente le noyau défini comme $\tilde{K}(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j)\rangle$. Le passage par la forme duale permet de réécrire la fonction de décision f qui devient :

$$f(x) = signe\left(\sum_{i=1}^{n} y_i\alpha_i\tilde{K}(x, x_i) + b\right)$$

Cette écriture utilise uniquement le noyau $\tilde{K}$ sans avoir besoin de calculer explicitement $\varphi(x)$. On parle alors d'*astuce du noyau*. Sa principale utilité est de pouvoir plonger les exemples dans des espaces de grande dimension en calculant uniquement des produits scalaires. Cela produit un gain d'espace mémoire et de rapidité de calcul considérable. En contrepartie, le modèle perd toute interprétabilité puisque la fonction de décision ne dépend plus des variables initiales. Signalons enfin que le noyau doit impérativement être un produit scalaire valide dans $\Im$. Pour cela, il doit respecter les *conditions de Mercer* stipulant qu'un $\tilde{K}$ symétrique est un produit scalaire valide si et seulement si sa matrice de Gram K définie par $K_{i,j} = \tilde{K}(x_i, x_j)$ est toujours semi-définie positive.

2.2 Qualité d'un noyau

Dans ce contexte, le succès du SVM dépend complètement du noyau utilisé. Une pratique largement répandue consiste à faire apprendre le SVM à partir d'un noyau souvent choisi au hasard en l'absence de connaissances préalables, puis de modifier petit à petit ses paramètres pour finalement sélectionner celui donnant les meilleurs résultats. Partant du constat que la

complexité algorithmique d'un SVM varie entre $O(n^2)$ et $O(n^3)$ (sans compter la construction de la matrice noyau qui a un coût de $O(n^2)$), l'évaluation d'un nombre important de paramétrages peut rapidement devenir un lourd handicap. De cette problématique est né le besoin de pouvoir évaluer un noyau en amont de l'apprentissage. Pour répondre à ce besoin, (Cristianini et al., 2002) ont créé un indicateur de qualité d'un noyau appelé *Kernel Target Alignment* (KTA) défini sur un échantillon S comme suit :

$$A(S, K, y^t y) = \frac{\langle K, y^t y \rangle_F}{\sqrt{\langle K, K \rangle_F \langle y^t y, y^t y \rangle_F}} = \frac{\sum_{i,j=1}^{n} y_i y_j K(x_i, x_j)}{n \sqrt{\sum_{i,j=1}^{n} K(x_i, x_j)^2}}$$

où $\langle ., . \rangle_F$ représente le produit scalaire de Frobenius. La matrice $y^t y$ est habituellement appelée *matrice idéale* ou *matrice cible*. Chaque cellule (i, j) de $y^t y$ a une valeur de 1 si x_i et x_j appartiennent à la même classe, -1 sinon. KTA a une valeur comprise entre -1 et 1 et est d'autant plus élevé que le noyau est considéré comme performant. Vu autrement, KTA calcule simplement la somme des "similarités" induites par le noyau entre individus de même classe et y soustrait la somme des similarités des individus de classes différentes. Le dénominateur permet de normaliser cette valeur entre -1 et 1. Une caractéristique intéressante de KTA est son relatif faible coût algorithmique.

Au delà de la simple évaluation, le critère KTA peut aussi être utilisé pour apprendre directement le noyau. Le *Multiple Kernel Learning* est sans doute le principe le plus approprié dans cette optique. Il consiste à supposer que la somme de plusieurs noyaux peut conduire à un noyau plus performant que la sélection du meilleur noyau individuellement. Plus techniquement, on désire trouver un vecteur $\mu = (\mu_1, ..., \mu_p), \mu_i \geq 0, \forall i$ tel que le noyau $K'(x_i, x_j) = \sum_{t=1}^{p} \mu_t K_t(x_i, x_j)$ ait un alignement maximal avec la matrice idéale $y^t y$. La contrainte de positivité des coefficients μ_i est essentielle pour assurer que K' soit semi-définie positive. En l'absence d'informations préalables, les noyaux initiaux K_t, ainsi que leur nombre sont choisis plus ou moins au hasard.

L'intérêt de la combinaison linéaire est double. Tout d'abord, la mise en forme de la fonction objective sous cette forme permet le recours à un arsenal important de techniques d'optimisation. Mais plus que par l'aspect pratique, le choix de l'agrégation de noyaux par combinaison linéaire se justifie grâce à une possible réécriture de KTA. En effet, l'alignement de la somme de deux matrices noyaux K_1 et K_2 sur un échantillon S s'écrit :

$$A(S, K_1 + K_2, y^t y) = \frac{\|K_1\|_F}{\|K_1 + K_2\|_F} A(S, K_1, y^t y) + \frac{\|K_2\|_F}{\|K_1 + K_2\|_F} A(S, K_2, y^t y)$$

On voit alors clairement que combiner deux noyaux (et par extension p noyaux) peut être avantageux. La situation est particulièrement intéressante si les deux noyaux ont un alignement individuel élevé et s'ils sont indépendants (la notion d'indépendance renvoie ici au produit de Frobenius entre les deux matrices K_1 et K_2). En effet, dans le cas de deux noyaux identiques, nous avons $\|K_1\|_F + \|K_2\|_F = \|K_1 + K_2\|_F$, et par conséquent $A(K_1 + K_2, y^t y) = A(K_1, y^t y) = A(K_2, y^t y)$.

Dans la section suivante, nous montrons que le paradigme des forêts aléatoires (Breiman, 2001) possède de fortes connexions avec celui de la maximisation de KTA et peut par conséquent être utilisé pour construire des noyaux performants.

3 Forêts aléatoires et espace induit

3.1 Principe des forêts aléatoires

L'algorithme des forêts aléatoires est un des apprenants les plus connus et les plus performants mis au point à ce jour. La méthode s'est même révélée comme la plus performante d'une série de classifieurs dans une récente étude à très grande échelle (Caruana et al., 2008). La procédure générique des forêts aléatoires consiste à répéter M fois de façon indépendante le processus suivant :

- Echantillonner l'ensemble d'apprentissage initial S pour obtenir S' ;
- Induire un arbre de décision sur S' en recherchant pour chaque noeud le meilleur éclatement à partir d'un sous-ensemble aléatoire de tous les éclatements possibles ;
- Développer l'arbre jusqu'à l'obtention des feuilles les plus pures possibles (pas d'élagage)

La prédiction de la classe d'un nouvel exemple x' s'effectue en lui attribuant la classe la plus fréquemment votée par les M arbres. Les forêts aléatoires introduisent une double randomization, par l'échantillonnage d'une part, et d'autre part par une recherche non exhaustive du meilleur éclatement à chaque noeud de l'arbre. De nombreuses techniques différentes de forêts ont été mises au point, chacune gérant de façon différente l'introduction de processus aléatoires dans la procédure. Dans (Breiman, 2001), l'auteur pratique un échantillonnage par le biais du *bootstrap* et recherche le meilleur éclatement en ne testant que sur un nombre restreint d' des d variables initiales. (Geurts et al., 2006) n'utilisent pas d'échantillonnage mais augmentent le degré de randomization en ne sélectionnant qu'un point de coupe au hasard sur chacune des d' variables préalablement sélectionnées. D'après les auteurs, en adaptant correctement d', on obtient un gain de temps de calcul important sans détérioration de performance par rapport aux forêts de Breiman. Un cas particulier des forêts aléatoires est le Bagging (Breiman, 1996) qui consiste à échantillonner selon le *bootstrap* sans introduire de randomization dans le recherche du meilleur éclatement.

L'utilisation de mécanismes aléatoires s'explique par le fait qu'une forêt admet l'existence d'une borne théorique de l'erreur en généralisation (EG). En effet, d'après (Breiman, 2001), $EG \leq \rho(1-s^2)/s^2$, où ρ représente la corrélation moyenne entre les prédictions des classifieurs et s est la *force de prédiction* moyenne, autrement dit, une fonction de la moyenne du taux de bon classement. Nous renvoyons le lecteur intéressé à (Breiman, 2001) pour plus d'informations sur ces deux paramètres. Cette borne de l'erreur est indépendante de la façon dont on injecte de l'aléatoire pour construire la forêt. On remarque aisément que d'après cette écriture, la situation idéale consiste à ce que les classifieurs soient non corrélés et aient individuellement la plus faible erreur possible. Ainsi, les divers algorithmes de forêts aléatoires ont été spécifiquement mis au point pour tenter de trouver un bon compromis entre pertinence individuelle des classifieurs et faible corrélation.

Nous avons vu dans la section précédente que le critère KTA suggère que la combinaison de noyaux indépendants et performants individuellement est une stratégie intéressante pour construire un noyau global plus performant. Nous pouvons remarquer d'emblée la connexion

entre le cadre décrit par les forêts aléatoires et celui décrit par KTA. En conséquence, si nous trouvons un moyen de construire un noyau représentant un arbre, nous pourrions alors représenter la forêt comme une somme de noyaux individuellement pertinents et peu corrélés (paradigme idéal selon KTA), puis tenter d'appliquer des algorithmes de décision tels que les SVM dans l'espace généré par l'ensemble d'arbres. Dans la section suivante, nous montrons comment dériver un noyau à partir d'un arbre, puis à partir d'un ensemble d'arbres. Nous verrons de plus que ce dernier a des propriétés intéressantes et nous paraît donc bien adapté pour l'apprentissage.

3.2 Construire un noyau à partir d'une forêt d'arbres

La particularité d'un arbre est de pouvoir découper l'espace en plusieurs zones dénommées feuilles. A la fin du développement, chaque exemple appartient à une et une seule feuille. Un exemple x_i peut alors être représenté par un vecteur de taille l où l est le nombre de feuilles de l'arbre. La j^{eme} composante de ce vecteur vaut 1 si x_i appartient à la feuille j, 0 sinon. Dans ce cas, un vecteur représentant un exemple ne contient qu'une seule valeur égale à 1 et $l-1$ égales à 0. On peut voir cette représentation comme un nouvel espace de description de dimension l pour les exemples. Il est alors aisé de remarquer que le produit scalaire $\langle x_i, x_j \rangle$ de deux individus x_i et x_j dans ce nouvel espace vaut 1 si x_i et x_j sont dans la même feuille, 0 sinon. Ainsi, le noyau $K(x_i, x_j) = 1$ si x_i et x_j sont dans la même feuille, 0 sinon, est bien le noyau de l'espace induit par l'arbre et K est bien semi-définie positive.

Dans le cas où M arbres différents ont été induits, il est également aisé d'en dériver un noyau. Nous pouvons considérer cette fois qu'un exemple peut être représenté par un vecteur $T = (T_{11}, ..., T_{ij}, ..., T_{Ml_M})$, où T_{ij} représente la j^{eme} feuille de l'arbre i. T est par conséquent de dimension $D = \sum_{i=1}^{M} l_i$ où l_i est le nombre de feuilles du i^{eme} arbre. Dans ce cas, la matrice des produits scalaires de l'ensemble K_{ens} s'écrit : $K_{ens}(x, x') = \sum_{i=1}^{M} K_i(x, x')$, où K_i est la matrice des produits scalaires du i^{eme} arbre. K_{ens} est donc semi-définie positive et correspond au nombre de fois où les exemples x et x' se sont retrouvés dans la même feuille. (Breiman, 2001) avait déjà évoqué l'intérêt d'une matrice presque équivalente pour l'analyse de similarité. La matrice en question est équivalente à K_{ens} mais est en plus normalisée par le nombre d'arbres M pour ramener chaque valeur entre 0 et 1.

Notons la forte connexion entre la construction du noyau par les forêts et l'écriture de KTA dans le cas de la somme de noyaux. En introduisant une part d'aléatoire dans l'algorithme des forêts, l'objectif est d'augmenter l'indépendance entre les arbres (et donc les noyaux issus de ces arbres) tout en assurant que chaque arbre soit assez pertinent (et ait donc un noyau avec un alignement assez élevé). Dans son article fondateur, (Breiman, 2001) développait les arbres jusqu'à l'obtention de feuilles pures. Cela signifie que les noeuds terminaux des arbres ne contenaient que des individus de même classe. Empiriquement, (Breiman, 2001) a remarqué que cette façon de procéder donnait les meilleurs résultats. Dans l'hypothèse où l'obtention de feuilles pures est possible, c'est à dire qu'il n'existe pas d'exemples de classes distinctes ayant toutes les variables avec les mêmes valeurs, nous pouvons alors démontrer que l'espace induit par l'ensemble de M arbres admet l'existence d'un séparateur linéaire dès que $M = 1$. Nous démontrons cela pour $M = 1$:

Proposition 1 *(Arbre parfait et séparabilité linéaire). Un arbre développé sur un échantillon S jusqu'à avoir uniquement des feuilles pures induit un espace dans lequel il existe au moins un séparateur linéaire entre les individus de classes distinctes de S.*

Preuve L'espace généré par l'arbre est de dimension l où l est égal au nombre de feuilles de l'arbre. Les feuilles étant pures, les exemples sont alors représentés par l points, chacun représentant un ensemble d'exemples de même classe, dans un espace de dimension l. Une fonction linéaire dans un espace de dimension p ayant une VC-dimension de $p + 1$, il est donc toujours possible de dichotomiser les exemples de classes distinctes •

Il est maintenant facile de démontrer l'existence de la séparabilité pour $M \geq 1$:

Proposition 2 *(Ensemble d'arbres et séparabilité linéaire). Un ensemble d'arbres dont l'un au moins est développé jusqu'à avoir uniquement des feuilles pures sur un échantillon S induit un espace dans lequel il existe au moins un séparateur linéaire entre les individus de classes distinctes de S.*

Preuve D'après la Proposition 1, on sait qu'il suffit d'un arbre à feuilles pures pour induire un espace dans lequel on peut séparer linéairement les individus de classes distinctes. Ainsi, l'ajout d'arbres est équivalent à ajouter des dimensions à l'espace induit par l'arbre à feuille pure. Par conséquent, il est toujours possible de séparer linéairement les exemples de classes distinctes dans ce nouvel espace •

Remarquons que même si aucun arbre n'est développé jusqu'à l'obtention de feuilles pures, il est néanmoins tout à fait possible que l'espace généré par un ensemble de M arbres contienne un séparateur linéaire. Une condition suffisante serait que tous les couples possibles d'exemples de classes distinctes ne tombent pas dans la même feuille au moins une fois (découle de la Proposition 2 et du principe de VC-dimension) . En pratique, bien d'autres situations sont susceptibles d'aboutir à un espace pouvant séparer linéairement les exemples ayant une étiquette différente. Les forêts aléatoires utilisant un échantillonnage avant la construction des arbres n'entrent pas dans le cadre décrit par les Propositions précedentes puisqu'une partie des exemples ne sera pas bien classée avec certitude. Dans la section suivante, nous étudions la procédure de classement par *vote à la majorité* et montrons que celle-ci est un hyperplan dans l'espace induit par l'ensemble d'arbres. Ce résultat va également nous permettre de démontrer l'existence d'une séparabilité linéaire pour les types de forêts où il existe un échantillonage.

4 Forêt, vote et séparabilité associée

Lorsqu'une nouvelle instance x' se présente, l'algorithme des forêts aléatoires effectue une prédiction en associant à cette instance l'étiquette la plus fréquemment attribuée par les M arbres. Cette procédure simple de classement est communément appelée *vote à la majorité*. Plusieurs travaux ont souligné que d'autres schémas de vote pouvaient apporter des améliorations très significatives au reclassement (Tsymbal et al. (2006); Robnik-Sikojna (2004)). La plupart des autres schéma de vote utilisent un système de pondération donnant un poids à

chaque votant (chaque arbre) proportionnel à sa pertinence. Cette dernière est généralement évaluée grâce à l'ensemble *out of bag* (OOB) de chaque arbre. L'ensemble OOB désigne simplement les exemples qui n'ont pas été sélectionnés par l'échantillonnage lors de la construction du i^{eme} arbre. Par exemple, (Tsymbal et al., 2006) utilise l'ensemble OOB pour estimer le taux de bon classement d'un arbre et lui attribuer un poids en fonction de ce taux. Nous appelons cette dernière technique d'agrégation *vote pondéré*. Nous pouvons facilement montrer la proposition suivante :

Proposition 3 *(Votes d'un ensemble). Les critères de vote à la majorité et de vote pondéré sont des hyperplans dans l'espace induit par l'ensemble d'arbres.*

Preuve Rappelons que dans l'espace induit par un ensemble d'arbres, l'exemple x est représenté par un vecteur $T = (T_{11}, ..., T_{ij}, ..., T_{Ml_M})$, où T_{ij} représente la j^{eme} feuille de l'arbre i. Par simplicité d'écriture, nous omettons l'indice relatif à l'arbre dans la suite de la preuve. Il suffit d'observer que la décision relative au vote à la majorité peut s'écrire $f(x) = signe\left(\sum_{i=1}^{D} \alpha_i T_i\right)$ où $\alpha_i = 1$ si la majorité des exemples de la feuille i (ces exemples vérifient alors $T_i = 1$) appartiennent à la classe $\{1\}$ et -1 sinon. Le vote pondéré consiste simplement à choisir les α_i dans un ensemble potentiellement différent de $\{-1, 1\}$ •

Le fait d'utiliser un hyperplan comme fonction de décision a des garanties intéressantes concernant l'erreur en généralisation (Vapnik, 1996). Ceci donne une autre vision de la robustesse de la décision d'une forêt d'arbres. Une forêt peut ainsi être vue comme un constructeur de noyau pouvant induire un espace dans lequel il existe une séparabilité linéaire entre exemples de classes distinctes. La procédure de vote consiste alors à trouver un hyperplan dans cet espace. Ceci rappelle bien entendu le principe du SVM qui fonctionne sur un mode similaire. Le résultat de la Proposition 3 nous permet également de généraliser l'existence d'une séparabilité même en présence d'un échantillonnage :

Proposition 4 *(Ensemble d'arbres et séparabilité linéaire). Soit un ensemble de M arbres construits chacun en utilisant une proportion p des n exemples de l'échantillon S sélectionnés aléatoirement. Sous l'hypothèse que tous les arbres soient construits jusqu'à avoir uniquement des feuilles pures, alors la probabilité θ qu'il existe un séparateur linéaire dans l'espace induit par l'ensemble d'arbres est minorée*

$$\theta \geq \left(\sum_{i>M/2}^{M} \binom{M}{i} p^i (1-p)^{M-i} \right)^n$$

Preuve Si les arbres sont développés jusqu'à contenir uniquement des feuilles pures, alors la probabilité qu'un exemple x_i choisi au hasard dans l'échantillon S soit bien classé par l'arbre A_j est minorée par la probabilité qu'il soit tiré au sort par A_j et est donc supérieure ou égale à p. Lorsque M arbres sont construits, la probabilité qu'un exemple x_i soit bien classé par la règle du *vote à la majorité* est supérieure à la probabilité P que x_i soit tiré au sort strictement plus de $M/2$ fois ce qui correspond à la probabilité qu'une loi binomiale $B(M, p)$ soit strictement supérieure à $M/2$. Ainsi, $P = \left(\sum_{i>M/2}^{M} \binom{M}{i} p^i (1-p)^{M-i}\right)$ et la probabilité pour

que les n exemples soient bien classés est minorée par $\theta = \left(\sum_{i>M/2}^{M} \binom{M}{i} p^i (1-p)^{M-i}\right)^n$. Le vote à la majorité étant un hyperplan dans l'espace induit par l'ensemble d'arbres, il est donc possible de séparer linéairement les exemples de classes distinctes avec une probabilité au moins égale à θ.

Corollaire 1.4 Sous les hypothèses de la Proposition 4, de $p > 0.5$ et de n fini alors $\theta \to 1$ lorsque $M \to \infty$.

Preuve D'après (Condorcet, 1785), $\left(\sum_{i>M/2}^{M} \binom{M}{i} p^i (1-p)^{M-i}\right)$ tend vers 1 lorsque M tend vers l'infini et $p > 0.5$. Ainsi, dans le cas n fini, nous avons bien le corollaire.

Le corollaire précedent montre donc qu'il est toujours possible de générer M arbres induisant un espace dans lequel il existe une séparabilité linéaire entre exemples de classes distinctes, pourvu que l'échantillonnage en amont de chaque arbre sélectionne plus de la moitié des exemples de S et que M soit suffisamment grand. Notons que la probabilité θ est une borne pessimiste de la probabilité réelle. Dans la réalité, la séparabilité devrait exister avec relativement moins d'arbres que suggérés par la Proposition 4 dans la mesure où les arbres classent toujours quelques exemples de l'ensemble OOB correctement et qu'il n'y a objectivement aucune raison de penser qu'il soit nécessaire que le vote à la majorité classe correctement tous les individus pour que la séparabilité linéaire existe. Le cas particulier d'un échantillonnage *bootstrap* n'est pas explicitement traité. On remarquera simplement que dans ce cas, la proportion d'individus p distincts tirés au sort vaut approximativement 0.63. On peut donc légitimement s'attendre à ce que la Propostion 4 se vérifie.

Comme nous l'avons vu, le critère du vote à la majorité et du vote pondéré sont deux possibilités de classement à partir d'un ensemble d'arbres. Ils peuvent également être vu comme des hyperplans dans l'espace induit par l'ensemble. Le vote pondéré offre parfois des performances plus intéressantes que le vote à la majorité en accordant des poids plus important à certains arbres. Devant toutes ces constatations, il nous paraît légitime d'utiliser un SVM dans l'espace induit par l'ensemble d'arbres. L'algorithme consiste alors à résoudre le programme $(P2)$ avec le noyau défini en section 3.2. On peut voir cet algorithme comme la recherche d'une pondération optimale des règles de décision issues des arbres, l'optimalité objective étant le compromis entre maximisation de la marge et du taux de bon classement. Il est important de noter que le noyau utilisé correspondant au produit scalaire des individus dans l'espace généré par les indicatrices des règles, les poids calculés par le SVM sont implicitement attribués aux feuilles (et donc aux règles réprésentées par ces feuilles) et non aux arbres comme c'est le cas du *vote pondéré*.

Finalement, utiliser les forêts d'arbres pour générer un noyau spécifiquement dédié au SVM est une pratique intéressante puisque le paradigme des forêts aléatoires est intimement lié à celui de la recherche d'un bon noyau. Parallèlement, la question de la pondération optimale des règles issues d'une forêt peut être résolue via un SVM. Ainsi, les deux problématiques ont une part de symétrie. Dans la section suivante, nous réalisons plusieurs expérimentations associant SVM et noyaux issus des forêts.

5 Expérimentations

Nous comparons les taux de bon reclassement de trois techniques d'apprentissage sur 12 jeux de données de l'UCI (tableau 1). Les trois techniques testées sont le *Multiple Kernel Learning* (MKL), le *vote à la majorité* d'une forêt aléatoire et le SVM construit avec un noyau issu de la même forêt que celle utilisée pour le *vote à la majorité*. Les résultats du MKL sont issus de (Varma et Babu, 2009), ce qui explique que 7 bases de données n'aient pas de résultat pour cette méthode. Le même protocole de test que dans (Varma et Babu, 2009) a été employé pour évaluer les trois algorithmes. Nous avons partitionné chaque jeu de données en deux parties (70 pourcents des individus pour l'apprentissage et les 30 pourcents restants pour le test). Ce processus a été répété 20 fois et la moyenne des taux de bons reclassement a été calculée. La forêt a été construite en utilisant 300 arbres développés chacun sur un échantillon *bootstrap* et jusqu'à l'obtention des feuilles les plus pures possible. Le meilleur éclatement a été recherché en restreignant la recherche sur un nombre de variables égal à l'entier le plus proche de la racine du nombre total d'attributs. Enfin, concernant le SVM, quatre valeurs possibles du paramètre de régularisation C ont été utilisées : $C = 1$, $C = 10$, $C = 100$ et $C = 10000$. L'ensemble d'apprentissage a été partitionné en 2 parties (70 pourcents pour apprendre et 30 pourcents pour la validation). Nous avons alors retenu le paramétrage donnant les meilleurs résultats sur l'échantillon de validation.

Jeu de données	n	Attributs	MKL	Vote	SVM + Noyau Forêt
Sonar	208	60	0,823	0,846	**0,894***
Liver	345	6	0,727	0,725	**0,736**
Parkinsons	195	22	**0,926**	0,923	**0,926**
Pima	768	8	0,772	0,754	**0,809***
Ionosphere	351	34	0,93	**0,94**	**0,94**
German	1000	20	-	0,752	**0,777**
Wdbc	569	30	-	0,963	**0,972**
Australian	690	14	-	0,803	**0,855***
Heart	270	13	-	0,807	**0,829**
Spambase	4601	57	-	0,969	**0,982***
Vote	435	16	-	**0,960**	0,958
Chess	3196	36	-	0,994	**0,996**

TAB. 1 – *Proportions de bon classement selon les trois méthodes d'apprentissage*

Le tableau 1 montre plusieurs choses intéressantes. Tout d'abord, le *vote à la majorité* de la forêt aléatoire a une performance équivalente à celle du MKL. Aucune différence significative (test de Student au seuil de 0.05) n'a pu être trouvé entre les deux méthodes. La performance issue du couplage SVM-forêt est très intéressante. En effet, elle est significativement meilleure que le MKL sur 2 des 5 jeux de données communs (une différence significative est marquée par un '*'). Comparativement à la forêt, le SVM-forêt est significativement meilleur 4 fois sur 12. Notons que sur les jeux où la différence n'est pas significative, le duo SVM-forêt est néanmoins le plus performant des trois algorithmes, excepté sur le jeu Vote ou la forêt est meilleure (mais l'écart est non significatif).

Il est important de souligner qu'un apprentissage concernant le nombre optimal de variables à sélectionner pour évaluer les éclatements pourrait améliorer les résultats issus de la forêt et du SVM-forêt. Parallèlement, cela montre que calibrer une forêt est une tâche simple. Un autre avantage de l'utilisation des forêts pour la construction de noyaux est la rapidité de calcul. Cela laisse ainsi plus de temps pour la recherche du paramètre C bien que dans nos expérimentations, les différences entre les diverses valeurs n'aient pas été trop importantes. Remarquons de plus que l'algorithme des forêts est parallèlisable, peut prendre en compte des attributs catégoriels et éventuellement des valeurs manquantes (mais cette problématique n'est pas abordée ici).

6 Conclusion et perspectives

L'article a montré que les forêts avec une composante aléatoire étaient particulièrement efficaces pour construire un noyau adapté à l'apprentissage supervisé par SVM. Ceci est dû au fait que le cadre théorique motivant leur constuction présente de fortes similarités avec celui de l'apprentissage optimal de noyaux. Nous avons vu comment dériver un noyau à partir d'un ensemble d'arbres et montré que sous des conditions peu exigeantes, l'espace induit par l'ensemble d'arbres admettait l'existence d'un séparateur linéaire entre exemples de classe distincte. L'utilisation d'un SVM dans cet espace produit des résultats supérieurs au MKL et au vote à la majorité de l'ensemble. Parallèlement, le SVM peut être vu comme une fonction d'agrégation des règles induites par la forêt aléatoire.

Plusieurs perspectives découlent de ce travail. Tout d'abord, nous souhaitons analyser théoriquement et empiriquement d'autres façons de construire un noyau à partir d'un ensemble d'arbres. Nous pouvons par exemple prendre en compte la topologie de l'arbre en calculant la similarité entre deux individus par le biais du noeud commun le plus proche. Nous souhaitons également étudier la possibilité de construire des noyaux représentant une infinité d'arbres. L'article montre que l'étude théorique des méthodes ensemblistes doit rester de premier plan et rappelle ce qui était déjà connu, à savoir qu'il est souvent plus judicieux de faire coopérer des méthodes a priori concurrentes plutôt que de les opposer.

Références

Bach, F., G. Lanckriet, et M. Jordan (2004). Multiple kernel learning, conic duality and the smo algorithm. *Proceedings of the 21st International Conference on Machine Learning*, 775–782.

Baram., Y. (2005). Learning by kernel polarization. *Neural Computation 17*, 1264–1275.

Breiman, L. (1996). Bagging predictors. *Machine Learning 24*, 123–140.

Breiman, L. (2001). Random forests. *Machine Learning 45*, 5–32.

Caruana, R., N. Karampatziakis, et A. Yessenalina (2008). An empirical evaluation of supervised learning in high dimensions. *Proceedings of the 25th International Conference on Machine Learning 307*, 96–103.

Condorcet, J. (1785). Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. *Imprimerie Royale*.

Cristianini, N., J. Kandola, A. Elisseef, et J. Shawe-Taylor (2002). On kernel-target alignment. *Advances in Neural Information Processing Systems*, 367–373.

Geurts, P., D. Ernst, et L.Wehenkel (2006). Extremely randomized trees. *Machine Learning 63*, 3–42.

Lanckriet, G., N. Cristianini, L. Ghaoui, P. Bartlett, et M. Jordan (2004). Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Reasearch 5*, 27–72.

Markowska-Kaczmar, U. et P. Kubacki (2005). Support vector machines in handwritten digits classification. *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, 406–411.

Nguyen, C. et T. Ho (2008). An efficient kernel matrix evaluation measure. *Pattern Recognition 41*, 3366–3372.

Pavlidis, P., I. Wapinski, et W. Noble (2004). Support vector machine classification on the web. *Bioinformatics 20*, 586–587.

Polat, K. et S. Günes (2007). Breast cancer diagnosis using least square support vector machine. *Digital Signal Processing*, 694–701.

Rakotomamonjy, A., F. Bach, Y. Grandvalet, et S. Canu (2008). Simplemkl. *Journal of Machine Learning Research 9*, 2491–2521.

Robnik-Sikojna, M. (2004). Improving random forests. *Proceedings of the 15th European Conference on Machine Learning 3201*, 359–370.

Schölkopf, B. et A. Smola (2001). *Learning with Kernels : Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA, USA: MIT Press.

Sonnenburg, S., G. Rötsch, B. Schölkopf, et G.Schäfer (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research 7*, 1531–1565.

Tsymbal, A., M. Pechenizkiy, et P. Cunningham (2006). Dynamic integration with random forests. *Proceedings of the 17th European Conference on Machine Learning 4212/2006*, 56–68.

Vapnik, V. (1996). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.

Varma, M. et B. Babu (2009). More generality in efficient kernel learning. *ACM International Conference Proceedings Series 382*, 2491–2521.

Summary

We show that an ensemble of decision trees with a randomness component allows the construction of powerful kernels adapted to supervised learning. We study theoretically such kernels ans show that under weak conditions it is possible to find a linear separator in the space induced by these kernels. We also observe that the classical *majority voting* is an hyperplane in the space induced by these last. We demonstrate empirically that combining a Support Vector Machine and ensemble of decision trees produces excellent results.