Un nouvel algorithme de forêts aléatoires d'arbres obliques particulièrement adapté à la classification de données en grandes dimensions

Thanh-Nghi Do[*,****], Stéphane Lallich[**]
Nguyen-Khang Pham[***], Philippe Lenca[*,****]

[*]Institut TELECOM, TELECOM Bretagne
UMR CNRS 3192 LabSTICC, Brest, France
tn.do|philippe.lenca@telecom-bretagne.eu
[**]Université Lyon, Laboratoire ERIC, Lyon 2, Lyon, France
stephane.lallich@univ-lyon2.fr
[***]IRISA, Rennes, France
pnguyenk@irisa.fr
[****]Université Européenne de Bretagne, France

Résumé. L'algorithme des forêts aléatoires proposé par Breiman permet d'obtenir de bons résultats en fouille de données comparativement à de nombreuses approches. Cependant, en n'utilisant qu'un seul attribut parmi un sous-ensemble d'attributs tiré aléatoirement pour séparer les individus à chaque niveau de l'arbre, cet algorithme perd de l'information. Ceci est particulièrement pénalisant avec les ensembles de données en grandes dimensions où il peut exister de nombreuses dépendances entre attributs. Nous présentons un nouvel algorithme de forêts aléatoires d'arbres obliques obtenus par des séparateurs à vaste marge (SVM). La comparaison des performances de notre algorithme avec celles de l'algorithme de forêts aléatoires des arbres de décision C4.5 et de l'algorithme SVM montre un avantage significatif de notre proposition.

1 Introduction

Les performances d'un classifieur dépendent de différents facteurs. Parmi ces derniers notons les paramètres nécessaires à son initialisation (par exemple le nombre de classes –ou clusters– pour un algorithme du type k-means), et les données utilisées pour construire le classifieur (par exemple la construction des ensembles d'apprentissage, de test et de validation). Afin d'atténuer l'influence des différents choix possibles et de compenser les limites des différents classifieurs pouvant être utilisés, la combinaison de classifieurs (ou encore méthode ensembliste) a retenu l'attention des chercheurs en apprentissage automatique depuis fort longtemps.

Les méthodes ensemblistes cherchent notamment à réduire la variance (erreur due à la variabilité des résultats en fonction de l'échantillon d'apprentissage) et/ou le biais (erreur de précision non dépendante de l'échantillon d'apprentissage) des algorithmes d'apprentissage (voir

par exemple Buntine (1992) et Wolpert (1992)). Parmi les différentes techniques proposées, on citera le Boosting (Freund et Schapire, 1995) qui permet de réduire de façon simultanée le biais et la variance, et le Bagging (Breiman, 1996) qui permet de réduire de façon notable la variance sans trop dégrader le biais.

Parmi les méthodes issues du Bagging, les forêts aléatoires (Breiman, 2001) sont certainement l'une des méthodes les plus efficaces pour l'apprentissage d'arbres de décision. La méthode des forêts aléatoires consiste à créer une famille d'arbres non élagués (où à chaque nœud la séparation des individus est réalisée à partir d'un sous-ensemble des attributs prédictifs choisi aléatoirement dans l'ensemble des attributs) à partir d'échantillon bootstrap (échantillon construit par tirage aléatoire avec remise à partir de l'ensemble d'apprentissage). Les performances d'une forêt d'arbres dépendent de la qualité des individus la composant et de leur indépendance. Les forêts aléatoires sont fondées sur des arbres non élagués afin de réduire l'erreur de biais. En outre, le processus aléatoire permet d'assurer une faible corrélation entre les arbres. Elles fournissent de bonnes performances comparativement aux algorithmes de référence en apprentissage supervisé, AdaBoost (Freund et Schapire, 1995) et les SVMs (Vapnik, 1995). La technique d'apprentissage est rapide, robuste au bruit et ne sur-apprend pas (Breiman, 2001) contrairement à AdaBoost (Dietterich, 2000).

Cependant, lors de la construction des arbres, un seul attribut est utilisé à chaque nœud pour séparer les individus. L'information liée à d'éventuelles dépendances entre attributs n'est donc pas utilisée. Cela peut être pénalisant avec des données en très grandes dimensions où mécaniquement il peut exister de nombreuses dépendances entre les variables. C'est notamment le cas lorsque le nombre de variables est grand au regard du nombre d'individus.

Pour tenir compte de cette dépendance, nous proposons d'utiliser l'algorithme de proximal SVM (PSVM) de Fung et Mangasarian (2001) comme heuristique de séparation des individus lors de la construction de chaque arbre. Nous obtenons ainsi des arbres de décision obliques où chaque nœud est multi-varié et utilise l'information de dépendance entre les attributs.

On compare les performances (en termes de précision, rappel, mesure F1 et taux global de précision) des forêts aléatoires d'arbres obliques avec celles des forêts aléatoires d'arbres C4.5 (construits à partir de l'algorithme C4.5 de Quinlan (1993)) et celles des SVMs (LibSVM de Chang et Lin (2001)). Nos tests réalisés sur 25 bases de données (10 classiques de l'UCI (Asuncion et Newman, 2007) et du projet Statlog (Michie et al., 1994), et 15 de grandes dimensions (Jinyan et Huiqing, 2002)) montrent l'intérêt de notre proposition. Nous obtenons un gain significatif pour les grandes dimensions et de meilleurs résultats, quoique non significatifs, pour les bases classiques.

L'article est organisé de la façon suivante. En section 2 nous introduisons brièvement les forêts aléatoires ainsi que les forêts aléatoires d'arbres de décision obliques pour la classification. La section 3 présente les expérimentations menées et discute les résultats. Enfin nous concluons en section 4.

2 Forêts aléatoires d'arbres obliques

Ho (1995) a proposé une méthode ensembliste qui sélectionne au hasard un sous-ensemble d'attributs dans la construction des arbres. Breiman (1996) a proposé le Bagging d'arbres de décision qui construit un ensemble d'arbres à partir de différents échantillons bootstrap tirés avec remise depuis l'ensemble d'apprentissage. La prédiction se base sur l'agrégation des

arbres obtenus, par un vote majoritaire pour la classification ou par la moyenne pour la régression. Par la suite, Amit et Geman (2001) ont proposé de tirer aléatoirement un sous-ensemble d'attributs parmi lesquels l'algorithme recherche la meilleure coupe lors de la construction des arbres. Ces approches ont été étendues et reformulées par Breiman (2001) pour créer l'algorithme des forêts aléatoires.

2.1 Forêts aléatoires

L'algorithme de forêts aléatoires (Breiman, 2001) crée un ensemble d'arbres de décision afin de réduire l'erreur de biais et d'assurer une faible corrélation entre les arbres. Les arbres sont construits sans élagage pour assurer un faible biais. Un arbre de la forêt est construit à partir d'un échantillon bootstrap tiré avec remise depuis l'ensemble d'apprentissage et la recherche de la meilleure coupe est fondée sur un sous-ensemble d'attributs tiré aléatoirement. Ces deux dernières propositions ont pour objectif de maintenir à un bas niveau la corrélation entre les arbres afin d'assurer leur diversité.

Considèrons une tâche de classification de m individus $x_i(i = 1, m)$ et n attributs. Un arbre de décision (noté DT) dans la forêt de k arbres (noté RF= $\{DT_i\}_{i=1,k}$) est créé de la façon suivante :

- tirage avec remise depuis l'ensemble d'apprentissage d'un échantillon bootstrap (noté $B_i(i = 1, k)$) qui est utilisé pour la construction de l'arbre ;
- recherche d'une meilleure coupe pour chaque nœud de décision à partir d'un sous-ensemble aléatoire de n' attributs ($n' < n$, e.g. $n' = \sqrt{n}$) ;
- construction de l'arbre le plus profond possible (sans élagage).

Pour prédire l'étiquette d'un nouvel individu, on utilise un vote majoritaire des arbres de la forêt dans le cas de la classification ou la moyenne des prédictions des arbres dans le cas d'une régression. Breiman a aussi proposé d'utiliser les individus en dehors de l'échantillon bootstrap (environ $36,8$ %) pour estimer les attributs importants et l'erreur de la forêt une fois l'arbre courant construit. L'algorithme de forêts aléatoires donne de bons résultats. Par ailleurs, il est rapide et robuste face aux données bruitées. Breiman (2001) a étendu les forêts aléatoires au cas de l'apprentissage non-supervisé.

Récemment, Robnik-Sikonja (2004) a proposé des améliorations possibles portant notamment sur le contrôle de la corrélation entre les arbres de la forêt. Il a également proposé d'utiliser un vote pondéré.

Geurts et al. (2006) ont proposé les arbres extrêmement aléatoires (Extra-Trees). Chaque arbre est construit à partir de l'ensemble d'apprentissage au lieu d'un échantillon bootstrap. Une coupe est effectuée au hasard sur un attribut choisi aléatoirement dans l'ensemble d'attributs sans tenir compte des classes. Ainsi les Extra-Trees sont très rapides pour l'apprentissage et réduisent significativement la corrélation entre les arbres de la forêt (Geurts et al., 2006). Les Extra-Trees sont proches de l'algorithme PERT proposé par Cutler et Guohua (2001).

2.2 Arbres obliques

La construction d'un arbre dans les algorithmes de forêts aléatoires n'utilise qu'un seul attribut parmi un sous-ensemble d'attributs tiré aléatoirement pour séparer les individus à chaque niveau de l'arbre (par exemple dans (Breiman et al., 1984), (Quinlan, 1993), (Ho, 1995), (Amit et Geman, 2001), (Cutler et Guohua, 2001), (Robnik-Sikonja, 2004) et (Geurts et al., 2006)).

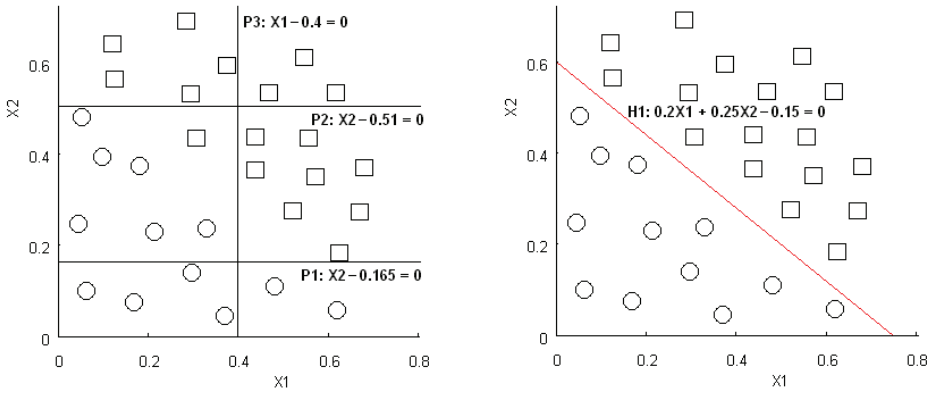

FIG. 1 – *Coupe selon un seul attribut (à gauche) et selon deux attributs (à droite)*

La dépendance éventuelle entre attributs n'est ainsi pas prise en considération, entraînant une perte d'information. L'exemple sur la figure 1 montre qu'il n'existe aucune coupe perpendiculaire aux axes permettant de séparer totalement les individus en une seule fois. Mais la coupe oblique H_1 (combinaison de deux attributs) classifie parfaitement les individus en deux classes.

Pour remédier à ce problème il a été proposé de construire des arbres multivariés qui effectuent des coupes obliques pour séparer les individus. La complexité de la construction d'un arbre oblique est NP-difficile (Heath, 1992). Breiman et al. (1984) ont proposé une méthode de construction d'arbres obliques qui combine linéairement les attributs en utilisant des coefficients générés aléatoirement dans l'intervalle $[-1, 1]$. Cependant cette approche ne peut traiter qu'un petit nombre d'attributs à cause de l'explosion combinatoire.

Murthy et al. (1993) ont proposé OC1, un système d'induction d'arbres obliques qui fait une recherche locale (méthode heuristique) pour trouver une bonne coupe oblique (hyperplan).

Wu et al. (1999) ont étendu l'algorithme OC1 en modifiant profondément la recherche d'une coupe oblique. Ils ont proposé d'utiliser les SVMs (hyperplan optimal) plus robustes que l'algorithme OC1. Cette approche améliore la performance de l'algorithme OC1. Cependant, l'utilisation d'un SVM standard se ramène à résoudre un programme quadratique dont la complexité est au moins le carré du nombre d'individus.

2.3 Forêts aléatoires d'arbres obliques

Notre algorithme de forêts aléatoires (noté RF-ODT) construit une collection d'arbres obliques de la même manière que celle proposée par Breiman (2001). La différence est qu'un arbre aléatoire oblique (ODT) dans la forêt ($\{ODT_i\}_{i=1,k}$) utilise un hyperplan pour effectuer les coupes obliques. Nous proposons d'utiliser l'algorithme de proximal SVM (Fung et Mangasarian, 2001) pour faire les coupes à partir d'un sous-ensemble aléatoire d'attributs parce que l'algorithme PSVM est très rapide pour l'apprentissage en comparaison des SVMs standards (voir par exemple (Do et Poulet, 2006)).

Considérons une tâche de classification binaire linéaire (voir la figure 2), avec m individus $x_i (i = 1, m)$ en n dimensions (attributs), représentés par la matrice $A[m\text{x}n]$ et leurs classes

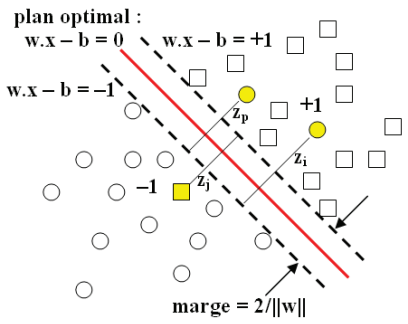

FIG. 2 – *Séparation linéaire d'individus en deux classes*

$y_i = \pm 1$, représentées par la matrice diagonale $D[mxm]$ de ± 1 ($D[i,i] = 1$ si x_i appartient à la classe +1 et $D[i,i] = -1$ si x_i est en classe -1). La recherche du meilleur hyperplan (noté par un vecteur w et un scalaire b) d'un algorithme de SVM se ramène simultanément à maximiser la marge des deux classes (la distance entre les plans supports) et à minimiser les erreurs (les individus du mauvais côté de leur plan support). L'algorithme SVM revient à résoudre un programme quadratique (1) :

$$\begin{aligned} \min \psi(w,b,z) &= (1/2)\|w\|^2 + cz \\ avec : D(Aw - eb) + z &\geq e \\ z &\geq 0 \end{aligned} \tag{1}$$

où e est un vecteur colonne composé de 1, et $c > 0$ une constante positive utilisée pour contrôler la marge et les erreurs, z le vecteur de variables positives de relâchement de contraintes. La classification d'un nouvel individu x est déterminée par $predict(x) = signe(w.x - b)$.

La résolution du programme quadratique (1) est coûteuse en temps et en mémoire vive (au moins le carré du nombre d'individus). Pour pouvoir améliorer les performances d'un SVM standard, Fung et Mangasarian (2001) ont proposé l'algorithme de proximal PSVM. L'algorithme modifie la formule (1) du SVM standard en :

- maximisant la marge par $min(1/2)\|w,b\|^2$
- minimisant les erreurs par $min(c/2)\|z\|^2$
- sous la contrainte $D(Aw - eb) + z = e$

En substituant z par w et b dans la fonction objectif ψ, nous obtenons le problème d'optimisation (2) :

$$\min \Psi(w,b) = (1/2)\|w,b\|^2 + (c/2)\|e - D(Aw - eb)\|^2 \tag{2}$$

La configuration pour que la fonction objectif ψ du problème (2) soit minimale est que les dérivées partielles en w et b soient nulles. Cela donne le système linéaire (3) suivant :

$$(w_1,\ w_2,\ \ldots,\ w_n,\ b)^T = (\frac{1}{c}I + E^T E)^{-1} E^T De \tag{3}$$

où $E = [A \quad -e]$, I est la matrice identité.

Les PSVMs ne demandent que la résolution du système linéaire (3) à $(n+1)$ inconnues au lieu du programme quadratique (1). Leur complexité est linéaire en fonction du nombre d'individus, ils sont donc capable de traiter un très grand nombre d'individus en un temps restreint. Les tests empiriques ont montré que les PSVMs donnent de bons taux de précision en comparaison de ceux obtenus par un SVM standard (e.g. LibSVM (Chang et Lin, 2001)) et ils sont surtout beaucoup plus rapides que les SVMs classiques. Nous proposons donc d'utiliser les PSVMs pour effectuer des coupes obliques lors de la construction des arbres. De plus, la recherche d'une coupe oblique se fonde sur un sous-ensemble de n' attributs ($n' < n$, e.g. $n' = \sqrt{n}$), situation où les PSVMs sont très efficaces en temps d'exécution.

Nous utilisons la méthode de Veropoulos et al. (1999) pour régler le problème du déséquilibre (entre les faux positifs et les faux négatifs) dans la construction des arbres. Notre algorithme de forêts aléatoires construit un ensemble d'arbres obliques sans élagage afin d'améliorer la performance des différents arbres de la forêt tout en maintenant une faible corrélation entre ceux-ci.

3 Evaluation

Pour évaluer les performances de notre algorithme, nous comparons les résultats obtenus par celui-ci en classification avec ceux des forêts aléatoires et des SVMs. Nous avons développé notre algorithme de forêts aléatoires d'arbres obliques (RF-ODT) en C/C++. L'algorithme C4.5 de Quinlan (1993) a été modifié pour obtenir l'algorithme de forêts aléatoires des arbres C4.5 (RF-C4.5). Enfin pour les SVMs nous avons utilisé l'algorithme standard LibSVM de Chang et Lin (2001).

Les comparaisons ont été faites à partir de 15 ensembles de données en grandes dimensions décrits dans le tableau 1 et de 10 ensembles de données standard décrits dans le tableau 2. Les 15 ensembles de données en grande dimension proviennent du site Bio-médical de Jinyan et Huiqing (2002) et les 10 ensembles de données standard proviennent des sites de l'UCI (Asuncion et Newman, 2007) et de Statlog (Michie et al., 1994).

Dans les tableaux 1 et 2, les colonnes 2, 3 et 4 indiquent le nom de l'ensemble de données, le nombre d'individus et le nombre d'attributs. Lorsqu'un ensemble de données comportait plus de 2 classes, nous nous sommes ramenés par regroupement de classes au cas de données bi-classes. La colonne 5 indique comment nous avons opéré le regroupement. Par exemple, dans le cas de l'ensemble 12, Subtypes of Acute Lymphoblastic (Hyperdip), la classe Hyperdip est considérée comme la classe positive, le regroupement des autres classes constituant la classe négative.

Le protocole de test est présenté dans la colonne 6. Dans certains cas, l'ensemble est déjà divisé en ensemble d'apprentissage (trn) et ensemble de test (tst). Sinon, nous avons procédé par validation croisée. Le leave-one-out (loo) est utilisé lorsque l'ensemble comporte moins de 300 individus. Autrement, nous avons utilisé la validation croisée à 10 segments avec ré-échantillonnage pour chaque algorithme (10-fold).

TAB. 1 – *Description des ensembles de données en grandes dimensions*

ID	Ensemble	#Individus	#Attributs	Classes	Protocole de test
1	Colon Tumor	62	2000	tumor, normal	loo
2	ALL-AML-Leukemia	72	7129	ALL, AML	trn-tst
3	*MLL-Leukemia	72	12582	MLL, rest	trn-tst
4	Breast Cancer	97	24481	relapse, non-relapse	trn-tst
5	Duke Breast Cancer	42	7129	cancer, normal	loo
6	Prostate Cancer	136	12600	cancer, normal	trn-tst
7	Lung Cancer	181	12533	cancer, normal	trn-tst
8	Central Nervous System	60	7129	positive, negative	loo
9	Translation Initiation Site	13375	927	positive, negative	10-fold
10	Ovarian Cancer	253	15154	cancer, normal	loo
11	Diffuse Large B-Cell Lymphoma	47	4026	germinal, activated	loo
12	*Subtypes of Acute Lymphoblastic (Hyperdip)	327	12558	Hyperdip, rest	trn-tst
13	*Subtypes of Acute Lymphoblastic (TEL-AML1)	327	12558	TEL-AML1, rest	trn-tst
14	*Subtypes of Acute Lymphoblastic (T-ALL)	327	12558	TEL-ALL, rest	trn-tst
15	*Subtypes of Acute Lymphoblastic (Others)	327	12558	Others, diagnostic groups	trn-tst

TAB. 2 – *Description des ensembles de données standard*

ID	Ensemble	#Individus	#Attributs	Classes	Protocole de test
16	Bupa	345	6	1, 2	10-fold
17	Breast Cancer Wisconsin	569	30	M, B	10-fold
18	Pima	768	8	1, 2	10-fold
19	*Segment	2310	19	1, rest	10-fold
20	Spambase	4601	57	spam, non-spam	10-fold
21	*Opticdigits	5620	64	0, rest	trn-tst
22	*Satimage	6435	36	1, rest	trn-tst
23	*Pendigits	10992	16	9, rest	trn-tst
24	*Letters	20000	16	A, rest	10-fold
25	*Shuttle	58000	9	1, rest	trn-tst

3.1 Résultats sur les ensembles de données en grandes dimensions

Nous avons comparé les performances de RF-ODT avec celles de RF-C4.5 et LibSVM sur les 15 ensembles de données présentant de grandes dimensions (tableau 1). Ces résultats sont présentés dans le tableau 3. Globalement (tableau 4), le recours à RF-ODT améliore significativement le taux global de précision, de 3,6 points par rapport à RF-C4.5 (p-value $= 0,0462$) et de $6,4$ points par rapport à LibSVM (p-value $= 0,0204$). La comparaison ensemble par ensemble avec le test du signe montre que sur les 15 ensembles, RF-ODT l'emporte systématiquement sur RF-C4.5 (9 victoires, 6 égalités, 0 défaite, p-value $= 0,0020$) et n'est battue qu'une fois par LibSVM (10 victoires, 4 égalités, 1 défaite, p-value $= 0,0059$).

Pour avoir une appréciation plus fine de la performance de RF-ODT face à RF-C4.5 et LibSVM, en plus du taux de précision globale, nous avons aussi comparé la précision, le rappel et la mesure F1 (van Rijsbergen, 1979).

En ce qui concerne la comparaison de RF-ODT avec RF-C4.5, on voit que le gain apporté par RF-ODT porte principalement sur le rappel (tableau 6) qui est amélioré de $7,7$ points en moyenne (p-value $= 0,0195$). La comparaison ensemble par ensemble montre que RF-ODT n'est battu qu'une fois par RF-C4.5 (8 victoires, 6 égalités, 1 défaite, p-value $= 0,0195$). Le gain sur la précision (tableau 5) qui est de 3 points, en raison des excellentes performances de RF-ODT sur les ensembles 6 et 8, n'est pas significatif. Le résultat ensemble par ensemble (6 victoires, 6 égalités, 3 défaite, p-value $= 0,0195$) recoupe ces observations. Globalement (tableau 7), la mesure F1 est améliorée en moyenne de $6,3$ points, ce qui est significatif (p-value $= 0,0269$). La comparaison ensemble par ensemble donne toujours l'avantage à RF-ODT (10 fois sur 15), ou une égalité (5 fois sur 15).

La comparaison de RF-ODT avec LibSVM donne le résultat inverse. La supériorité de RF-ODT sur LibSVM est essentiellement due à l'amélioration de la précision. En effet, RF-ODT améliore en moyenne de $8,1$ points la précision de LibSVM (tableau 5), ce qui est très significatif (p-value $= 0,0044$). Sur les 15 ensembles, RF-ODT n'est battu qu'une seule fois par LibSVM (11 victoires, 3 égalités, 1 défaite, p-value $= 0,0032$). RF-ODT améliore en moyenne de $3,4$ points le rappel (tableau 6), ce qui n'est pas tout à fait significatif. Cependant, on remarque que les 6 victoires (en particulier sur les ensembles 1, 4 et 15) sont plus larges que les 3 défaites. Globalement, la mesure F1 est améliorée en moyenne de $6,2$ points (p-value $= 0,0038$), ce qui est très significatif. La comparaison ensemble par ensemble recoupe cette conclusion (12 victoires, 2 égalités, 1 défaite, p-value $= 0,0017$).

TAB. 3 – *Résultats de la classification sur les ensembles de données en grandes dimensions*

Ensemble	Précision			Rappel			Mesure F1			Précision globale		
ID	LibSVM	RF-C4.5	RF-ODT	LibSVM	RF-C4.5	RF-ODT	LibSVM	RF-C4.5	RF-ODT	LibSVM	RF-C4.5	RF-ODT
1	68.18	76.19	**82.61**	75.00	72.73	**86.36**	71.43	74.42	**84.44**	80.65	82.26	**88.71**
2	**100**	95.24	95.24	95.00	**100**	**100**	97.44	**97.56**	**97.56**	**97.06**	**97.06**	**97.06**
3	75.00	**100**	**100**	**100**	**100**	**100**	85.71	**100**	**100**	93.33	**100**	**100**
4	69.23	83.33	**84.62**	75.00	83.33	**91.67**	72.00	83.33	**88.00**	63.16	78.94	**84.21**
5	85.00	**94.12**	90.00	**94.44**	80.00	90.00	89.47	86.49	**90.00**	**90.48**	88.10	**90.48**
6	73.53	75.76	**100**	**100**	**100**	96.00	84.75	86.21	**97.96**	73.53	76.47	**97.06**
7	88.26	**93.75**	**93.75**	**100**	**100**	**100**	93.75	**96.77**	**96.77**	98.66	**99.33**	**99.33**
8	47.62	45.46	**61.91**	55.56	23.81	**61.91**	51.28	31.25	**61.91**	68.33	63.33	**73.33**
9	83.13	**92.58**	90.78	**84.42**	73.83	79.75	83.77	82.15	**84.91**	92.15	92.30	**93.20**
10	**100**	98.78	**100**	**100**	**100**	**100**	**100**	99.39	**100**	**100**	99.21	**100**
11	91.30	**95.65**	92.00	87.50	91.67	**95.83**	89.36	93.62	**93.88**	89.36	**93.62**	**93.62**
12	95.46	95.24	**100**	**95.46**	90.91	**95.46**	95.46	93.02	**97.67**	98.21	97.32	**99.11**
13	**100**	**100**	**100**	**100**	96.30	96.30	**100**	98.11	98.11	**100**	99.11	99.11
14	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
15	92.59	**100**	**100**	39.68	29.63	**55.56**	55.56	45.71	**71.43**	64.29	83.93	**89.29**

TAB. 4 – *Comparaison des taux globaux de précision*

Taux global de précision	LibSVM	RF-C4.5	RF-ODT	RF-ODT vs LibSVM	RF-ODT vs RF-C4.5
moyenne	87.28	90.07	93.63	**6.35**	**3.57**
écartype	13.65	22.31	21.69	9.41	6.32
ratio de Student				2.61	2.19
p-value				**0.0204**	**0.0462**
résultat de RF-ODT				gain*	gain*
RF-ODT victoire				10	9
RF-ODT égalité				4	6
RF-ODT défaite				1	0
p-value				0.0059	0.0020
résultat de RF-ODT				gain**	gain**

TAB. 5 – *Comparaison des taux de précision*

Précision	LibSVM	RF-C4.5	RF-ODT	RF-ODT vs LibSVM	RF-ODT vs RF-C4.5
moyenne	84.62	89.74	92.73	8.11	2.99
écartype	15.30	14.69	10.38	9.26	7.68
ratio de Student				3.39	1.51
p-value				**0.0044**	0.1540
résultat de RF-ODT				gain**	
RF-ODT victoire				11	6
RF-ODT égalité				3	6
RF-ODT défaite				1	3
p-value				**0.0032**	0.2539
résultat de RF-ODT				gain**	

TAB. 6 – *Comparaison des taux de rappel*

Rappel	LibSVM	RF-C4.5	RF-ODT	RF-ODT vs LibSVM	RF-ODT vs RF-C4.5
moyenne	86.80	83.06	90.17	3.37	7.11
écartype	18.37	24.94	14.13	6.98	11.37
ratio de Student				1.87	2.42
p-value				0.0828	**0.0296**
résultat de RF-ODT					gain*
RF-ODT victoire				6	8
RF-ODT égalité				6	6
RF-ODT défaite				3	1
p-value				0.2539	0.0195
résultat de RF-ODT					gain*

TAB. 7 – *Comparaison des taux de mesure F1*

Mesure F1	LibSVM	RF-C4.5	RF-ODT	RF-ODT vs LibSVM	RF-ODT vs RF-C4.5
moyenne	84.67	84.54	90.84	**6.18**	**6.31**
écartype	15.61	27.15	22.65	6.90	9.88
ratio de Student				3.47	2.47
p-value				**0.0038**	**0.0269**
résultat de RF-ODT				gain**	gain*
RF-ODT victoire				12	10
RF-ODT égalité				2	5
RF-ODT défaite				1	0
p-value				0.0017	0.0010
résultat de RF-ODT				gain**	gain***

3.2 Résultats sur les ensembles de données standard

TAB. 8 – *Comparaison des taux globaux de précision sur les ensembles de données standard*

Taux global de précision	RF-ODT vs RF-C4.5
moyenne	**0.69**
écartype	3.10
ratio de Student	0.70
p-value	**0.5001**
résultat de RF-ODT	non significatif
RF-ODT victoire	8
RF-ODT égalité	0
RF-ODT défaite	2
p-value	0.0547
résultat de RF-ODT	limite de signification

Il est intéressant de compléter la comparaison qui précède par une comparaison de RF-ODT et RF-C4.5 sur des ensembles standard. Nous avons travaillé sur 10 ensembles (tableau 2) qui ont chacun un nombre de variables compris entre 6 et 64 et dont le ratio (nombre de dimensions / nombre d'indidivus) ne dépasse pas 5%. On peut en conclure que RF-ODT est au moins aussi

performant que RF-C4.5 sur les ensembles standard. En effet, le gain moyen de RF-ODT sur RF-C4.5 est de $0,6$ points ce qui n'est pas significatif. Les différences entre RF-ODT et RF-C4.5 sont faibles, à l'exception de l'ensemble Bupa, où RF-ODT l'emporte de 4 points. On remarquera cependant que RF-ODT l'emporte 8 fois sur 10 sur RF-C4.5, ce qui est à la limite de la signification.

En conclusion, les résultats expérimentaux confirment le bien-fondé de notre approche : RF-ODT fait au moins jeu égal avec RF-C4.5 sur les ensembles standard et il surclasse de façon très significative RF-C4.5 sur les ensembles en grandes dimensions, ce qui est le but de cette recherche.

3.3 Temps d'exécution

Avant de comparer les temps d'exécution des différentes forêts aléatoires rappelons la complexité théorique de ces algorithmes. Soit un ensemble de données ayant m individus et n attributs, la construction d'une forêt aléatoire de k arbres de décision C4.5 avec ($n' < n$ attributs tirés aléatoirement) a une complexité en $\mathcal{O}(kqn'mlog(m))$ ($q = 1$ si l'attribut est nominal, $q = 2$ si l'attribut est numérique). En ce qui concerne notre forêt aléatoire d'arbres obliques, sa complexité théorique est $\mathcal{O}(kp(n' + m)n'^2)$ où p est la profondeur moyenne des arbres. En pratique, la taille des arbres obliques est plus compacte que celle des arbres C4.5, un arbre oblique est en général moins profond. De plus le sous-ensemble aléatoire de n' attributs n'est pas très grand (ordre de 10^2). Pour ces raisons, le temps d'exécution des forêts aléatoires d'arbres obliques est plus court que les forêts d'arbres C4.5. Nous avons utilisé un PC (Pentium 2,4 GHz, 1 Go RAM, Linux Mandriva 2008) pour faire les expérimentations. Le temps d'exécution est présenté dans le tableau 9.

TAB. 9 – *Temps d'exécution*

Ensemble	Temps d'exécution (s)		Ensemble	Temps d'exécution (s)	
ID	RF-C4.5	RF-ODT	ID	RF-C4.5	RF-ODT
1	2.96	**0.66**	14	**4.00**	4.30
2	3.96	**3.42**	15	**74.68**	86.20
3	**5.98**	17.97	16	**0.21**	0.28
4	11.38	**6.58**	17	1.87	**0.15**
5	7.41	**6.81**	18	0.93	**0.83**
6	3.09	**2.49**	19	1.87	**0.20**
7	**1.30**	2.50	20	10.08	3.65
8	**3.33**	5.88	21	2.79	**1.07**
9	797.10	**671.00**	22	7.83	**3.74**
10	29.97	**22.99**	23	4.90	**1.35**
11	2.55	**1.73**	24	8.62	**1.56**
12	19.82	**10.37**	25	28.07	**6.56**
13	17.70	**11.96**	Moyenne	42.10	**34.97**

Nous donnons également des résultats pour l'ensemble de données Forest cover type de grande taille (avec les deux plus grandes classes, Spruce-Fire –211840 individus– et Lorgepole-Pine –83301 individus–, et 54 attributs, (Asuncion et Newman, 2007)). Nous avons utilisé 495141 individus pour l'ensemble d'apprentissage et 45141 individus pour le test et construit des forêts aléatoires de 30 arbres. L'apprentissage de l'algorithme de forêts d'arbres obliques met $801,61$ secondes et atteint $99,98\%$ de taux de bon classement. L'algorithme de forêts de C4.5 a $99,57\%$ de bon classement et met $17484,92$ secondes pour l'apprentissage. Notre

algorithme améliore la précision de 0,41% et est 22 fois plus rapide que l'algorithme de forêts d'arbres C4.5.

4 Conclusion et perspectives

La construction d'arbres de décision dans l'algorithme de forêts aléatoires proposé par Breiman n'utilise à chaque nœud qu'un attribut à partir d'un sous-ensemble d'attributs tiré aléatoirement. Ceci est particulièrement pénalisant avec les ensembles de données en grandes dimensions où il peut exister de nombreuses dépendances entre attributs. C'est notamment le cas lorsque le nombre de variables est grand relativement au nombre d'individus.

Pour remédier à ce problème nous avons proposé un nouvel algorithme d'arbres aléatoires obliques obtenus par des séparateurs à vaste marge du type proximal SVM. La comparaison des performances de notre algorithme avec celles de l'algorithme de forêts aléatoires des arbres de décision C4.5 et celles de l'algorithme de SVM montre un avantage très significatif de notre proposition. Nous envisageons d'étendre ce travail à la classification multi-classes, à la régression et à la détection d'individus atypiques.

Références

Amit, Y. et D. Geman (2001). Shape quantization and recognition with randomized trees. *Machine Learning 45*(1), 5–32.

Asuncion, A. et D. Newman (2007). UCI machine learning repository.

Breiman, L. (1996). Bagging predictors. *Machine Learning 24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Breiman, L., J. H. Friedman, R. A. Olshen, et C. Stone (1984). *Classification and Regression Trees*. Wadsworth International,.

Buntine, W. (1992). Learning classification trees. *Statistics and Computing 2*, 63–73.

Chang, C. C. et C. J. Lin (2001). LIBSVM – A library for support vector machines. Technical report, Department of Computer Science and and Information Engineering, National Taiwan University. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cutler, A. et Z. Guohua (2001). Pert – Perfect random tree ensembles. *Computing Science and Statistics 33*, 490–497.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization. *Machine Learning 40*(2), 139–157.

Do, T. N. et F. Poulet (2006). Classifying one billion data with a new distributed svm algorithm. In *The 4th IEEE International Conference on Computer Science, Research, Innovation and Vision for the Future (RIVF-2006)*, pp. 59–66.

Freund, Y. et R. Schapire (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory : Proceedings of the Second European Conference*, pp. 23–37.

Fung, G. et O. Mangasarian (2001). Proximal support vector classifiers. In *Proceedings KDD-2001 : Knowledge Discovery and Data Mining*, pp. 77–86.

Geurts, P., D. Ernst, et L. Wehenkel (2006). Extremely randomized trees. *Machine Learning 63*(1), 3–42.

Heath, D. (1992). *A Geometric Framework for Machine Learning*. Ph. D. thesis, Johns Hopkins University, Baltimore, Maryland.

Ho, T. K. (1995). Random decision forest. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 278–282.

Jinyan, L. et L. Huiqing (2002). Kent ridge bio-medical data set repository. Technical report. `http://sdmc.lit.org.sg/GEDatasets`.

Michie, D., D. J. Spiegelhalter, et C. C. Taylor (Eds.) (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Murthy, S., S. Kasif, S. Salzberg, et R. Beigel (1993). Oc1 : Randomized induction of oblique decision trees. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 322–327.

Quinlan, J. R. (1993). *C4.5 : Programs for Machine Learning*. San Mateo, CA : Morgan Kaufmann.

Robnik-Sikonja, M. (2004). Improving random forests. In *Proceedings of the Fifth European Conference on Machine Learning*, pp. 359–370.

van Rijsbergen, C. V. (1979). *Information Retrieval*. Butterworth.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Veropoulos, K., C. Campbell, et N. Cristianini (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 55–60.

Wolpert, D. (1992). Stacked generalization. *Neural Networks 5*, 241–259.

Wu, W., K. Bennett, N. Cristianini, et J. Shawe-Taylor (1999). Large margin trees for induction and transduction. In *Proceedings of the Sixth International Conference on Machine Learning*, pp. 474–483.

Summary

The random forests method is fast, robust to noise and does not overfit. However random forests do not have high performance when dealing with very-high-dimensional data in presence of dependency. We here investigate a new approach for supervised classification with a huge number of attributes. We propose a random oblique decision trees method. It consists of randomly choose a subset of predictive attributes and then it uses SVM as a split function. We then compare the effectiveness with classical measures (e.g. precision, recall, F1-measure and accuracy) of random forests of random oblique decision tree with SVMs and random forests of C4.5. The results, obtained on 25 data sets, are promising. In particular our proposal has significant better performance on very-high-dimensional data with better -but not significant- results on lower dimensional data sets.