Algorithmes rapides de boosting de SVM

Thanh-Nghi Do*, Jean-Daniel Fekete*, François Poulet**

*Equipe Aviz, INRIA Futurs, LRI
Bât.490, Université Paris Sud 91405 Orsay Cedex
{dtnghi@lri.fr | Jean-Daniel.Fekete@inria.fr}
{http://www.lri.fr/~dtnghi | http://www.lri.fr/~fekete}
**IRISA TexMex, Université de Rennes I
Campus de Beaulieu, 35042 Rennes Cedex
francois.poulet@irisa.fr
http://www.irisa.fr/texmex/people/poulet/index_fr.php

Résumé. Les algorithmes de boosting de Newton Support Vector Machine (NSVM), Proximal Support Vector Machine (PSVM) et Least-Squares Support Vector Machine (LS-SVM) que nous présentons visent à la classification de très grands ensembles de données sur des machines standard. Nous présentons une extension des algorithmes de NSVM, PSVM et LS-SVM, pour construire des algorithmes de boosting. A cette fin, nous avons utilisé un terme de régularisation de Tikhonov et le théorème Sherman-Morrison-Woodbury pour adapter ces algorithmes au traitement d'ensembles de données ayant un grand nombre de dimensions. Nous les avons ensuite étendus par construction d'algorithmes de boosting de NSVM, PSVM et LS-SVM afin de traiter des données ayant simultanément un grand nombre d'individus et de dimensions. Les performances des algorithmes sont évaluées sur des grands ensembles de données de l'UCI comme Adult, KDDCup 1999, Forest Covertype, Reuters-21578 et RCV1-binary sur une machine standard (PC-P4, 2,4 GHz, 1024 Mo RAM).

1 Introduction

Les algorithmes de Séparateurs à Vaste Marge proposés par (Vapnik, 1995) et les méthodes de noyaux permettent de construire des modèles précis et deviennent des outils de classification de données de plus en plus populaires. On peut trouver de nombreuses applications des SVM (réf. http://www.clopinet.com/isabelle/Projects/SVM/applist.html) comme la reconnaissance de visages, la catégorisation de textes ou la bioinformatique. Cependant, les SVM demandent la résolution d'un programme quadratique dont le coût de calcul est au moins d'une complexité égale au carré du nombre d'individus de l'ensemble d'apprentissage et la quantité de mémoire nécessaire les rend impossible à utiliser sur de grands ensembles de données à l'heure actuelle (Lyman et al., 2003). Il y a besoin de permettre le passage à l'échelle des SVM pour traiter de grands ensembles de données sur des machines standard. Une heuristique possible pour améliorer l'apprentissage des SVM est de décomposer le programme quadratique en une série de plus petits problèmes (Boser et al, 1992), (Chang et al, 2003), (Osuna et al, 1997), (Platt, 1999). Au niveau de la mise en œuvre,

les méthodes d'apprentissage incrémental (Cauwenberghs et al, 2001), (Do et Poulet, 2006), (Do et Poulet, 2003), (Fung et Mangasarian, 2002), (Poulet et Do, 2004), (Syed et al, 1999) permettent de traiter de grands ensembles de données par mise à jour des solutions partielles en chargeant consécutivement les sous-ensembles d'apprentissage en mémoire sans avoir à charger l'ensemble total. Les algorithmes parallèles et distribués (Do et Poulet, 2006), (Poulet et Do, 2004) utilisent des machines en réseaux pour améliorer le temps d'exécution de l'apprentissage. Les algorithmes d'apprentissage actif (Do et Poulet, 2005), (Tong et Koller, 2000) choisissent un sous-ensemble d'individus (ensemble actif) représentatif pour la construction du modèle.

Dans cet article, nous continuons à développer nos algorithmes de boosting de PSVM et LS-SVM (Do et Poulet, 2004) (Do et Poulet, 2007) et (Do & Fekete, 2007) pour classifier simultanément un grand nombre d'individus et de dimensions. Nous présentons une classe d'algorithmes de boosting se basant sur les NSVM (Mangasarian, 2001), PSVM (Fung et Mangasarian, 2001) et LS-SVM (Suykens et Vandewalle, 1999) pour la classification de grands ensembles de données sur des machines standard. Les algorithmes de NSVM, PSVM et LS-SVM obtiennent la solution par résolution d'un système d'équations linéaires au lieu du programme quadratique. Ils permettent de classifier beaucoup plus rapidement des ensembles ayant de très grands nombres d'individus. Puis nous les avons reformulés en utilisant un terme de régularisation de Tikhonov et le théorème de Sherman-Morrison-Woodbury (Golub et Van Loan, 1996) pour traiter des ensembles de données ayant un très nombre de dimensions. Enfin nous avons construit les algorithmes de boosting, Adaboost (Freund et Schapire, 1995), Arc-x4 (Breiman, 1998) de NSVM, PSVM et LS-SVM pour la classification d'ensembles de données ayant simultanément un grand nombre d'individus et de dimensions. Les performances des algorithmes sont évaluées sur de grands ensembles de l'UCI (Blake et Merz, 1998), comme Adult, KDDCup 1999, Forest Covertype, Reuters-21578 et de RCV1-binary. Les résultats sont comparés avec ceux obtenus avec LibSVM (Chang et Lin, 2003), Perf-SVM (Joachims, 2006) et CB-SVM (Yu et al., 2003).

Le paragraphe 2 présente brièvement les algorithmes de NSVM, PSVM et LS-SVM, le paragraphe 3 décrit les extensions pour la classification d'ensembles de données ayant un grand nombre de dimensions. Dans le paragraphe 4 nous présentons les algorithmes de boosting pour le traitement de très grands ensembles de données (simultanément en nombre d'individus et de dimensions) puis quelques résultats dans le paragraphe 5 avant la conclusion et les travaux futurs.

Quelques notations sont utilisées dans cet article. Tous les vecteurs sont représentés par des matrices colonne. Le produit scalaire de deux vecteurs x et y est noté $x.y$. La norme d'un vecteur v est $||v||$. La matrice inverse de X est notée par X^{-1}. e est un vecteur colonne de 1. I représente la matrice identité.

2 Algorithmes de NSVM, PSVM et LS-SVM

On considère une tâche de classification binaire linéaire avec m individus x_i $(i=1..m)$ en n dimensions, representés par la matrice $A[mxn]$ et leurs classes (+1 ou –1) stockées dans la matrice diagonale $D[mxm]$. L'algorithme de SVM cherche le meilleur hyperplan (w, b) de séparation des données. Il se ramène à d'une part maximiser la marge qui est la distance entre les plans supports $(2/||w||)$ des deux classes (le plan support de la classe +1 [resp. –1] sépare tous les individus de la classe +1 [resp. –1] des autres) et d'autre part minimiser les

erreurs (les individus du mauvais côté de leur plan support). Les distances au plan des erreurs sont notées par les variables $z_i \geq 0$ $(i=1..m)$. Si l'individu x_k est du bon côté de son plan support, alors z_k est égal à 0. L'algorithme de SVM revient au programme quadratique (1) :

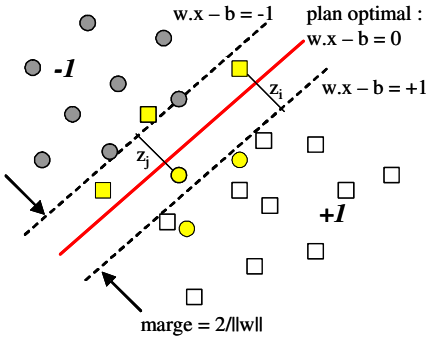

FIG. 1 – *SVM linéaire pour la classification des données en deux classes*

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

où une constante $c > 0$ est utilisée pour contrôler la marge et les erreurs.

Le plan optimal (w, b) est obtenu par la résolution du programme quadratique (1) dont la mise en œuvre est coûteuse en temps et mémoire vive.

2.1 Algorithme de NSVM

L'algorithme de generalized SVM (GSVM) (Mangasarian, 1998) modifie l'algorithme de SVM (1) en :

- maximisant la marge par $(1/2)\ \|w, b\|^2$
- minimisant les erreurs par $(c/2)\ \|z\|^2$

En substituant z par w et b dans la fonction objectif Ψ du programme quadratique (1), on obtient un problème d'optimisation non contrainte (2) :

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

où $[e - D(Aw - eb)]_+$ est obtenu en remplaçant les valeurs négatives par zéro.

En notant $X = (w_1\ w_2 \ldots w_n\ b)^T$ et $E = (A \quad -e)$, on peut réécrire le problème (2) en :

$$\min\ \Psi(X) = (1/2)\ X^T X + (c/2)\|(e - DEX)_+\|^2 \tag{3}$$

(Mangasarian, 2001) a proposé d'utiliser la méthode itérative de Newton pour résoudre efficacement le problème d'optimisation (3). Le principe de la méthode de Newton est de minimiser successivement les approximations au second ordre de la fonction objectif Ψ en se basant sur un développement de Taylor au second ordre au voisinage de X_v. Le problème

d'optimisation (3) se base sur la dérivée première $\Psi'(X)$ de $\Psi(X)$ (4) et ensuite le Hessien (5) (dérivée seconde $\Psi''(X)$) de $\Psi(X)$).

$$\Psi'(X) = c(-DE)^T[(e - DEX)_+] + X \quad (4)$$

$$\Psi''(X) = c(-DE)^T diag[(e - DEX)*](-DE) + I \quad (5)$$

Avec $diag[(e - DEX)*]$ matrice diagonale $(m)x(m)$ dont le i-ième élément diagonal est un sous gradient de $(e - DEX)_+$.

- répéter
  - calculer la dérivée première $\Psi'(X_i)$ (4)
  - calculer le Hessien $\Psi''(X_i)$ (5)
  - $X_{i+1} = X_i - [\Psi''(X_i)]^{-1} \Psi'(X_i)$ (6)
  - $i = i + 1$
- jusqu'à ($\Psi'(X_{i-1}) \cong 0$)
- obtenir les n coordonnées de w et le scalaire b à partir de la solution X_i

TAB 1 – *Algorithme de Newton-SVM*

L'algorithme itératif de Newton pour SVM (tableau 1) converge vers la solution après un relativement faible nombre (de 5 à 8) d'itérations. Il nécessite les résolutions d'un système linéaire (6) à $n+1$ inconnues au lieu du programme quadratique (1), donc si le nombre de dimensions de l'ensemble de données est inférieur à 10^5, l'algorithme de NSVM est capable de classifier rapidement un très grand nombre d'individus sur un PC (P4, 2,4 GHz, 512 Mo RAM). L'algorithme NSVM classifie un million de points en dimension 20 en 20 secondes sur un PC.

2.2 Algorithme de PSVM

L'algorithme de proximal SVM (PSVM) (Fung et Mangasarian, 2001) modifie aussi l'algorithme de SVM (1) en :

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

En substituant z dans la fonction objectif Ψ du programme quadratique (1), nous obtenons alors (7) :

$$min\ \Psi(w, b) = (1/2)||w, b||^2 + (c/2)||e - D(Aw - eb)||^2 \quad (7)$$

Pour ce problème d'optimisation (7), nous calculons les dérivées partielles en w et b. Cela nous donne le système linéaire de $(n+1)$ inconnues $(w_1, w_2, \ldots, w_n, b)$ suivant :

$$(w_1\ w_2 \ldots w_n\ b)^T = (I/c + E^T E)^{-1} E^T De \quad (8)$$

L'algorithme de PSVM ne demande que la résolution d'un système linéaire (8) à $n+1$ inconnues au lieu du programme quadratique (1), donc il est capable de traiter un très grand nombre d'individus en un temps restreint sur une machine standard. Par exemple, la classification d'un million de points en dimension 20 est effectuée en 13 secondes sur un PC (P4, 2,4GHz, 512Mo RAM).

2.3 Algorithme de LS-SVM

L'algorithme de LS-SVM proposé par (Suykens et Vandewalle, 1999) modifie également l'algorithme de SVM (1) en :
- maximisant la marge par $(1/2)\ ||w||^2$
- minimisant les erreurs par $(c/2)\ ||z||^2$ sous la contrainte : $D(Aw - eb) + z = e$

En substituant z dans la fonction objectif Ψ du programme quadratique (1), nous obtenons alors (9) :

$$min\ \Psi(w, b) = (1/2)||w||^2 + (c/2)||e - D(Aw - eb)||^2 \quad (9)$$

Pour ce problème d'optimisation (9), nous calculons également les dérivées partielles en w et b. Cela nous donne le système linéaire de $(n+1)$ inconnues $(w_1, w_2, \ldots, w_n, b)$ suivant :

$$(w_1\ w_2\ \ldots\ w_n\ b)^T = (I°/c + E^TE)^{-1}E^TDe \quad (10)$$

où $I°$ est la matrice diagonale identité dont le dernier élément est 0.

L'algorithme de LS-SVM a la même complexité que celle du PSVM, il ne demande que la résolution d'un système linéaire (10) à $n+1$ inconnues au lieu du programme quadratique (1), donc il est capable de traiter un très grand nombre d'individus en un temps restreint sur une machine standard.

3 Extensions pour un grand nombre de dimensions

Certaines applications comme la bioinformatique ou la fouille de textes nécessitent de traiter des données ayant un nombre très important de dimensions et un nombre d'individus plus réduit. Dans ce cas la matrice de taille $(n+1)x(n+1)$ est trop importante et la résolution du système à $(n+1)$ inconnues nécessite un temps de calcul élevé. Pour adapter l'algorithme à ce type de données nous avons appliqué le théorème de Sherman-Morrison-Woodbury (11) aux systèmes d'équations des algorithmes de NSVM, PSVM et LS-SVM.

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^TA^{-1}U)^{-1}V^TA^{-1} \quad (11)$$

Pour l'algorithme de NSVM, on note : $Q = diag[\sqrt{c(e - DEX_i)^*}\ ]$ et $P = Q(-DE)$, l'inversion de la matrice $\psi''(X_i)$ dans la formule (6) est re-écrite par (12) :

$$[\psi''(X_i)]^{-1} = (I + P^TP)^{-1} \quad (12)$$

Ensuite nous appliquons la formule de Sherman-Morrison-Woodbury (11) dans la partie droite de la formule (12), nous obtenons (13) :

$$[\psi''(X_i)]^{-1} = (I + P^TP)^{-1} = I - P^T(I + PP^T)^{-1}\ P \quad (13)$$

L'inversion de la matrice $(I + P^TP)$ de taille $(n+1)x(n+1)$ est ramenée à inverser la matrice $(I + PP^T)$ (de taille mxm). Cette nouvelle formulation permet à l'algorithme de NSVM de traiter facilement des ensembles de données ayant un nombre de dimensions très important.

Pour l'algorithme PSVM, nous appliquons directement le théorème de Sherman-Morrison-Woodbury (11) dans $(I/c + E^TE)^{-1}$, nous obtenons (14) :

$$(w_1\ w_2\ \ldots\ w_n\ b)^T = (I/c + E^TE)^{-1}\ E^TDe = cE^T[De - (I/c + EE^T)^{-1}\ EE^TDe] \quad (14)$$

Ce nouveau système (14) nécessite l'inversion de la matrice $(I/c + EE^T)$ (de taille *mxm*) au lieu de la matrice $(I/c + E^TE)$ de taille (n+1)x(n+1) dans le système (8). Cette nouvelle formulation de l'algorithme de PSVM peut classifier facilement des ensembles de données ayant un nombre de dimensions très important.

Pour adapter l'algorithme LS-SVM à un grand nombre de dimensions on applique le théorème de Sherman-Morrison-Woodbury dans $(I°/c + E^TE)^{-1}$ du système d'équations (10). Mais ce faisant nous avions une matrice singulière à inverser ($I°$). Nous avons donc ajouté un terme de régularisation de Tikhonov, ce qui est la méthode la plus couramment utilisée pour résoudre ce genre de problème. Avec le terme de Tikhonov ($\delta > 0$) ajouté à $(I°/c + E^TE)^{-1}$ nous obtenons alors (15) :

$$(I/c + \delta I + E^TE)^{-1} = (H + E^TE)^{-1} \quad (15)$$

où H représente la matrice $(n+1)x(n+1)$ diagonale dont le $(n+1)$ème terme est δ et les autres termes valent $(1/c)+\delta$.

Ensuite nous pouvons utiliser le théorème de Sherman-Morrison-Woodbury (11) dans $(H + E^TE)^{-1}$, nous obtenons (16) :

$$(w_1\ w_2\ \ldots\ w_n\ b)^T = (H + E^TE)^{-1}\ E^TDe = H^{-1}E^T[I - (I + EH^{-1}E^T)^{-1}EH^{-1}E^T]De \quad (16)$$

Ce nouveau système (16) fait aussi l'inversion de la matrice $(I + EH^{-1}E^T)$ (de taille mxm) au lieu de la matrice $(I°/c+E^TE)$ de taille $(n+1)x(n+1)$ dans le système (10). Cette nouvelle formulation permet à l'algorithme de LS-SVM de classifier facilement un ensemble de données ayant nombre de dimensions très important.

4 Boosting de NSVM, PSVM et LS-SVM

Nous avons décrit les versions de NSVM, PSVM et LS-SVM permettant de traiter des ensembles de données ayant soit un grand nombre d'individus, soit un grand nombre de dimensions. Si l'ensemble de données a simultanément un très grand nombre d'individus et de dimensions, il ne peut être traité par aucune des versions parce que les algorithmes de NSVM, PSVM et LS-SVM demandent l'inversion d'une matrice de taille soit le carré du nombre d'individus, soit le carré du nombre de dimensions. Ceci nécessite alors énormément de mémoire vive et de temps d'exécution. Pour remédier à ce problème, nous proposons de construire des algorithmes de boosting de NSVM, PSVM et LS-SVM.

L'algorithme Adaboost développé par (Freund et Schapire, 1995) dans les années 90 est une méthode mettant en œuvre un ensemble de classifieurs basiques. L'idée principale est de répéter l'apprentissage d'un classifieur basique à plusieurs reprises. A chaque étape de boosting, le classifieur basique se concentre sur les individus mal classifiés la fois précédente. Enfin, on combine les modèles obtenus en un seul. L'algorithme Adaboost est simple à implémenter et donne de bons résultats en pratique. (Reyzin et Schapire, 2006) a démontré la cohérence d'Adaboost.

L'idée est d'utiliser les algorithmes de NSVM, PSVM et LS-SVM comme classifieurs basiques. A chaque étape de boosting, Adaboost se base sur la construction des NSVM, PSVM, LS-SVM avec un échantillon de l'ensemble de données (de taille très inférieure à l'ensemble total) créé à partir des poids assignés aux individus mal classifiés la fois précédente. Dans le cas où les données ont un très grand nombre d'individus, Adaboost utilise les algorithmes de NSVM (6), PSVM (8) et LS-SVM (10) sur des échantillons de données à chaque étape de boosting. Dans le cas où l'ensemble de données a un très grand nombre de dimensions, ou simultanément un grand nombre d'individus et de dimensions, Adaboost utilise les algorithmes de NSVM (13), PSVM (14) ou LS-SVM (16) décrits dans le paragraphe 3. L'algorithme Adaboost de NSVM, PSVM et LS-SVM est présenté dans le tableau 2.

Entrée :
- m individus d'apprentissage : $\{(x_i, y_i)\}_{i=1,m}$ où $x_i \in R^n$ et $y_i \in \{1, -1\}$
- le nombre d'itérations T

Apprentissage :
- initialisation des poids de m individus $W_1(i) = 1/m$
- faire une boucle de T étapes de boosting

pour t de 1 à T faire
- créer un échantillon S_t d'individus en se basant sur les poids W_t
- créer le modèle h_t à partir d'un échantillon S_t

$h_t = \{P|LS|N\}\text{-}SVM(S_t)$
- calculer l'erreur apparente de h_t

$$\varepsilon_t = \sum_{i=1}^{m} h_t(x_i) \neq y_i \, W_t(i)$$
- choisir le coefficient

$\alpha_t = (1/2) ln[(1-\varepsilon_t)/\varepsilon_t]$
- calculer le facteur de la normalisation

$fac_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$
- mettre à jour les poids de m individus

si $(y_i = h_t(x_i))$ alors

$W_{t+1}(i) = W_t(i) \, exp(-\alpha_t) / fac_t$

sinon

$W_{t+1}(i) = W_t(i) \, exp(\alpha_t) / fac_t$

finpour
- retourner les h_t et α_t $(t = 1,T)$

Classification d'un nouvel individu x basée sur : $H(x) = signe\left(\sum_{i=1}^{T} \alpha_i h_i(x)\right)$

TAB. 2 – *Algorithme Adaboost de NSVM, PSVM et LS-SVM*

Remarquons que l'utilisation des algorithmes de NSVM, PSVM et LS-SVM dans l'algorithme Adaboost est très intéressante parce que ces algorithmes traitent beaucoup plus rapidement les sous-ensembles (échantillons) de données qu'avec la résolution d'un programme quadratique d'un SVM standard.

Nous proposons également d'utiliser l'algorithme Arc-x4 (Breiman, 1998) avec les algorithmes de NSVM, PSVM et LS-SVM de manière similaire à l'algorithme Adaboost. Le principe de l'algorithme Arc-x4 est semblable à Adaboost. Il utilise le mécanisme de vote majoritaire au lieu d'un vote avec des poids et il a également une manière différente pour prendre un échantillon en se concentrant sur les erreurs de toutes les étapes précédentes. L'algorithme Arc-x4 de NSVM, PSVM et LS-SVM donne de très bonnes performances en classification de grands ensembles de données.

5 Quelques résultats

Nous avons développé le programme en C/C++ sous Linux en utilisant la librairie Lapack++ (réf. http://math.nist.gov/lapack++/) pour bénéficier de bonnes performances en calcul matriciel. Nous avons aussi développé des algorithmes spécifiques pour les matrices creuses. Nous allons en présenter une évaluation en prenant en compte le taux de précision et le temps d'apprentissage. Nous avons sélectionné 4 grands ensembles de données de l'UCI (Blake et Merz, 1998) et un ensemble de données textuelles RCV1-binary (réf. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets). Ces ensembles sont décrits dans le tableau 3.

	Classes	Individus	Dimensions	Protocole de test
Adult	2	48842	14	32561 Trn - 16281 Tst
Forest cover type	7	581012	54	10-fold
KDD cup 1999	5	5 209 458	41	4 898 429 Trn - 311 029 Tst
Reuters-21578	135	10789	29406	7770 Trn - 3019 Tst
RCV1-binary	2	697 641	47 236	20 242 Trn - 677 399 Tst

TAB. 3 – *Description des ensembles de données.*

Nous avons utilisé les algorithmes de boosting de NSVM (AdaNSVM, Arcx4NSVM), PSVM (AdaPSVM, Arcx4PSVM) et LS-SVM (AdaLSSVM, Arcx4LSSVM) pour effectuer la classification sur un PC (Pentium IV, 2,4GHz et 1024Mo RAM). Les résultats sont comparés avec ceux obtenus avec l'algorithme SVM standard LibSVM (Chang et Lin, 2003) et deux nouvelles versions de SVM, Perf-SVM (Joachims, 2006) et CB-SVM (Yu et al., 2003). Nous mettons dans les tableaux les meilleurs résultats en caractères gras et les deuxièmes en souligné. Nous ne faisons que comparer les premiers résultats expérimentaux des algorithmes de boosting de NSVM, LS-SVM avec LibSVM parce que l'algorithme de PSVM est très semblable à l'algorithme de LS-SVM.

Les ensembles de données Reuters-21578 et RCV1-binary sont utilisés pour évaluer les performances en classification d'ensembles ayant simultanément un grand nombre d'individus et dimensions. Nous avons utilisé Bow (réf. http://www.cs.cmu.edu-/~mccallum/bow) en prétraitement des données Reuters-21578. Chaque document est vu comme un vecteur de mots, nous avons obtenu 29406 mots (dimensions) sans sélection de dimensions. Nous avons effectué la classification des 10 classes les plus nombreuses. Cet ensemble de données ayant plus de deux classes nous avons utilisé l'approche "un contre le reste". Les résultats sont présentés dans le tableau 4 avec la moyenne de la précision et du rappel (breakeven point) pour les 10 plus grandes catégories.

On remarque que nos algorithmes donnent toujours des résultats un peu meilleurs en ce

qui concerne le taux de précision (jusqu'à 5,32% d'amélioration) au détriment, dans certains cas, de la rapidité de calcul.

	Précision (%)				
	AdaLSSVM	ArcLSSVM	AdaNSVM	ArcNSVM	LibSVM
Earn	**98,41**	98,32	98,32	98,36	98,02
Acq	**96,57**	96,22	95,66	96,44	95,66
Money-fx	79,49	80,90	80,36	**81,04**	75,72
Grain	90,75	90,73	**91,07**	90,01	89,33
Crude	**89,83**	88,98	88,89	89,58	86,62
Trade	78,18	76,39	**79,74**	79,05	77,46
Interest	78,32	78,16	78,59	**80,15**	75,57
Ship	84,60	83,72	85,63	**86,25**	83,00
Wheat	86,38	88,12	85,74	**86,94**	85,58
Corn	88,97	**89,12**	88,14	88,05	88,99

TAB. 4 – *Résultats en précision sur 10 catégories de l'ensemble de données Reuters-21578*

	Temps (s)				
	AdaLSSVM	ArcLSSVM	AdaNSVM	ArcNSVM	LibSVM
Earn	8,42	**7,24**	12,22	13,15	9,40
Acq	8,47	6,85	**5,81**	15,74	10,78
Money-fx	7,68	9,30	11,29	26,52	**7,42**
Grain	9,86	**4,74**	12,67	12,99	5,05
Crude	8,05	9,25	16,85	17,38	**5,70**
Trade	8,01	**3,02**	11,81	10,94	5,73
Interest	13,33	11,66	**6,02**	11,62	8,69
Ship	14,21	16,80	16,07	25,13	**5,36**
Wheat	14,10	**2,41**	9,56	10,40	3,54
Corn	**6,28**	6,62	19,48	13,88	**3,79**

TAB. 5 – *Temps de la classification des 10 catégories de l'ensemble Reuters-21578*

Pour l'ensemble de données RCV-1, nous avons utilisé le même prétraitement que (Chang et Lin, 2003), CCAT et ECAT formant la classe positive et GCAT et MCAT la négative (en enlevant les instances apparaissant simultanément dans les classes positive et négative). Les résultats montrent que nos algorithmes permettent une petite amélioration de la précision tout en étant très significativement plus rapide (jusqu'à 19 fois plus rapide). Ce facteur est encore plus significatif sur l'ensemble de données Adult (200 fois plus rapide pour une précision équivalente).

Pour l'ensemble de données Forest Cover Type, nos algorithmes ont effectué la classification des deux classes les plus nombreuses en moins de 30 secondes, LibSVM n'a donné aucun résultat après 21 jours de calcul. Cependant des travaux récents ont montré que l'algorithme SVM-Perf (Joachims, 2006) effectue la classification de cet ensemble de données en 171 secondes sur un Intel Xeon, 3,6GHz, 2Go RAM. En tenant compte des différences de rapidité des processeurs, nous pouvons estimer raisonnablement que nos algorithmes sont 8 fois plus rapides.

Enfin pour l'ensemble de données KDD-Cup'99, CB-SVM (Yu et al, 2003) a effectué la classification en 4750 secondes sur un P-III, 800MHz, 1Go RAM, avec 90% de précision. Nos algorithmes obtiennent 92% de précision en seulement 180 secondes soit à peu près 10 fois plus rapidement que CB-SVM.

	Précision (%)				
	AdaLSSVM	ArcLSSVM	AdaNSVM	ArcNSVM	LibSVM
RCV1-bin	95,45	95,77	95,57	**95,93**	93,28
Adult	85,08	85,03	85,18	**85,34**	85,29
Forest covertype	77,16	76,69	**77,18**	76,72	N/A
KddCup'99	91,96	**92,73**	92,31	92,65	N/A

TAB. 6 – *Performances en précision des algorithmes sur les grands ensembles de données*

	Temps (s)				
	AdaLSSVM	ArcLSSVM	AdaNSVM	ArcNSVM	LibSVM
RCV1-bin	54,10	**42,21**	48,97	178,57	787,14
Adult	**12,08**	**12,08**	21,00	49,32	2472,59
Forest covertype	29,83	**28,83**	29,52	43,14	N/A
KddCup'99	191	176,62	172,23	**149,92**	N/A

TAB. 7 – *Performances en temps d'exécution sur les grands ensembles de données*

6 Conclusion et perspectives

Nous avons présenté une classe algorithmes de boosting de NSVM, PSVM et LS-SVM pour la classification de grands ensembles de données sur des machines standard. L'idée principale est d'étendre les algorithmes récents de Mangasarian et de Suykens pour en construire des algorithmes de boosting. Nous avons utilisé un terme de régularisation de Tikhonov et le théorème de Sherman-Morrison-Woodbury pour adapter ces algorithmes à traiter des ensembles de données ayant un grand nombre de dimensions. Nous les avons ensuite étendus par construction des algorithmes de boosting, Adaboost et Arc-x4 de NSVM, PSVM, LS-SVM afin de traiter des données ayant simultanément un grand nombre d'individus et de dimensions. Les performances des algorithmes sont évaluées sur les grands ensembles de données de l'UCI comme Adult, KDDCup 1999, Forest Covertype, Reuters-21578 et de RCV1-binary sur une machine standard (P4, 2,4 GHz, 1Go RAM). Les résultats expérimentaux montrent que nos algorithmes de boosting de NSVM, PSVM et LS-SVM sont rapides avec une bonne précision. Ils permettent le passage à l'échelle et obtiennent de bons taux de précision en comparaison avec LibSVM (l'un des algorithmes de SVM les plus efficaces) et deux autres nouveaux algorithmes, Perf-SVM et CB-SVM. Pour des ensembles de données de très grandes tailles (à la fois en nombre de dimensions et d'individus), ils ont également montré une bonne rapidité d'exécution et un bon taux de précision.

Une première extension de ces travaux va consister à étendre ces algorithmes pour en faire des versions parallèles et distribuées sur un ensemble de machines. Cette extension permettra d'améliorer le temps de la tâche d'apprentissage. Une seconde sera de proposer une nouvelle approche pour la classification non linéaire.

Références

Blake, C. and C. Merz (1998). UCI Repository of Machine Learning Databases. http:// www.ics.uci.edu/~mlearn/MLRepository.html.

Boser, B., I. Guyon, and V. Vapnik (1992). An Training Algorithm for Optimal Margin Classifiers. in proc. of 5th ACM Annual Workshop on Computational Learning Theory, Pittsburgh, Pennsylvania, 144-152.

Breiman, L. (1998). Arcing classifiers. *The annals of statistics*, 26(3): 801–849.

Cauwenberghs, G. and T. Poggio (2001). Incremental and Decremental Support Vector Machine Learning. in *Advances in Neural Information Processing Systems*, MIT Press, 13:409-415.

Chang, C-C. and C-J. Lin (2003). LIBSVM -- A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Do, T-N. and J-D. Fekete (2007). Large Scale Classification with Support Vector Machine Algorithms. (to appear) in proc. of IEEE ICMLA'07, Ohio, USA.

Do, T-N. et F. Poulet (2007). Classification de grands ensembles de données avec un nouvel algorithme de SVM. Actes d'EGC'07, RNTI-E-9 – Série Extraction et Gestion des Connaissances, Cépaduès Editions, 2: 739-750.

Do, T-N. and F. Poulet (2006). Classifying one billion data with a new distributed SVM algorithm. in proc. of IEEE RIVF'06, Ho Chi Minh, Vietnam, pp. 59-66.

Do, T-N. and F. Poulet (2004). Towards High Dimensional Data Mining with Boosting of PSVM and Visualization Tools. in proc. of ICEIS'04, 6th Int. Conf. on Entreprise Information Systems, 2: 36-41, Porto, Portugal.

Do, T-N. and F. Poulet (2005). Mining Very Large Datasets with SVM and Visualization. in proc. of ICEIS'05, 7th Int. Conf. on Entreprise Information Systems, 2:127-134, Miami, USA.

Freund, Y. and R. Schapire (1995). A decision-theoretic generalization of on-line learning and an application to boosting. in *EuroCOLT '95*, pp. 23–37.

Fung, G. and O. Mangasarian (2002). Incremental Support Vector Machine Classification. in proc. of the 2nd SIAM Int. Conf. on Data Mining SDM'2002 Arlington, Virginia, USA.

Golub, G. and C. Van Loan (1996). *Matrix Computations*. John Hopkins University Press, Balti-more, Maryland.

Joachims, T. (2006). Training Linear SVMs in Linear Time. in proc. of the *ACM SIGKDD* Int. Conf. on KDD, pp. 217-226.

Lyman, P., H-R. Varian, K. Swearingen, P. Charles, N. Good, L. Jordan, and J. Pal (2003). How much information. http://www.sims.berkeley.edu/research/projects/how-much-info-2003/.

Mangasarian, O. (1998), Generalized Support Vector Machines. Data Mining Institute Technical Report 98-14, Computer Sciences Department, University of Wisconsin, Madison, USA, 1998.

Mangasarian, O. (2001). A finite newton method for classification problems. Data Mining Institute Technical Report 01-11, Computer Sciences Department, University of Wisconsin.

Platt, J. (1999). Fast Training of Support Vector Machines Using Sequential Minimal Optimization. in *Advances in Kernel Methods -- Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola Eds., 185-208.

Poulet, F. and T-N. Do (2004). Mining Very Large Datasets with Support Vector Machine Algorithms. in *Enterprise Information Systems V*, Camp O., Filipe J., Hammoudi S. And Piattini M. Eds., Kluwer Academic Publishers, 177-184.

Reyzin, L. and R.E. Schapire (2006). How boosting the margin can also boost classifier complexity. in proc. of *ICML'06*, the 23rd Int. Conf. on Machine Learning, Pittsburgh, Pennsylvania, USA, pp. 753-760.

Suykens, J. and J. Vandewalle (1999). Least Squares Support Vector Machines Classifiers. in *Neural Processing Letters*, 9(3):293-300.

Syed, N., H. Liu, and K. Sung (1999). Incremental Learning with Support Vector Machines. in proc. of the 6th *ACM SIGKDD* Int. Conf. on KDD'99, San Diego, USA.

Tong, S. and D. Koller (2000). Support Vector Machine Active Learning with Applications to Text Classification. in proc. of *ICML'00*, the 17th Int. Conf. on Machine Learning, 999-1006, Stanford, USA.

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

Yu, H., J. Yang, and J. Han (2003). Classifying large data sets using SVMs with hierarchical clusters. in proc. of the *ACM SIGKDD* Int. Conf. on KDD, pp. 306-315.

Summary

The Boosting of PSVM, LS-SVM, NSVM algorithms aim at classifying very large datasets on standard personal computers (PCs). We extend the PSVM, LS-SVM and NSVM in several ways to efficiently classify large datasets. We developed a row-incremental version for datasets with billions of data points. By adding a Tikhonov regularization term and using the Sherman-Morrison-Woodbury formula, we developed a column-incremental version to process datasets with a small number of data points but very high dimensionality. Finally, by applying boosting including AdaBoost and Arcx4 to these incremental algorithms, we developed classification algorithms for massive, very-high-dimensional datasets. Numerical test results on UCI, RCV1-binary, Reuters-21578, Forest cover type and KDD cup 1999 datasets showed that our algorithms are often significantly faster and/or more accurate than state-of-the-art algorithms LibSVM, SVM-perf and CB-SVM.