Classification d'images à grande échelle avec des SVM

Thanh-Nghi Doan[1], Thanh-Nghi Do[2], François Poulet[3,1]

1. IRISA, Campus de Beaulieu, 263 Avenue du Général Leclerc
35042 Rennes Cedex, France
thanh-nghi.doan@irisa.fr

2. College of Information Technology, Can Tho University
1 Ly Tu Trong street, Can Tho city, Vietnam
dtnghi@cit.ctu.edu.vn

3. Université de Rennes I, Campus de Beaulieu
263 avenue du Général Leclerc 35042 Rennes Cedex, France
francois.poulet@irisa.fr

RÉSUMÉ. Nous présentons des améliorations de l'algorithme de Power Mean SVM (PmSVM) pour la classification d'ensembles d'images tels qu'ImageNet (14 millions d'images et 21 000 classes) qui rendent l'apprentissage beaucoup plus complexe (temps et coût mémoire). Les SVM ne sachant traiter que deux classes, les approches les plus utilisées sont un contre un ou un contre le reste. Avec les grands ensembles de données, l'approche un contre le reste est préférée pour des raisons de coût, mais implique des problèmes de déséquilibre. Pour le déséquilibre, nous proposons un algorithme de bagging équilibré de SVM, parallélisé pour obtenir les résultats dans un temps raisonnable. Sur les 1 000 plus grandes classes d'ImageNet (ILSVRC 2010) il est 286 fois plus rapide que le PmSVM original et 1 434 fois plus rapide que LIBLINEAR avec une amélioration relative de plus de 20 % de la précision par rapport à ce dernier.

ABSTRACT. We present some improvements of Power Mean SVM (PmSVM) for large scale image classification, with large number of images and classes. Usual SVM algorithms can only deal with two classes, for more than two classes, one can use one against one or one against all approaches. With large datasets, one against all is the most used because of computational cost but this implies very imbalanced classes. To deal with these imbalanced data, we present a parallel bagging of SVM algorithms to get results within reasonable time. For the classification of the 1000 largest classes of ImageNet dataset (900000 images, 12.5GB) our algorithm is 286 times faster than original PmSVM and 1434 times faster than state-of-the-art algorithms like LIBLINEAR with a relative increase of accuracy more than 20% compared to the latter one.

MOTS-CLÉS : catégorisation d'images, passage à l'échelle, algorithmes parallèles de SVM, échantillonnage, HPC.

KEYWORDS: large scale visual classification, high performance computing, sampling strategy, parallel support vector machines.

DOI:10.3166/TS.31.39-56

Extended abstract

We present some improvements of the Power Mean SVM algorithm (PmSVM) for large scale image classification, with large number of images and classes. Usual SVM algorithms can only deal with two classes, for k > 2 classes, one can use one against one (train k(k-1)/2 classifiers) or one against all (train k classifiers) approaches. With large datasets, one against all is the most used because of computational cost but this implies very imbalanced classes with large number of classes. With 1000 classes, the one against one approach has to train one million classifiers and in one against all approach, the minority class is only 0.1% of the whole dataset. The class prediction is then performed with a majority vote. To deal with these very imbalanced data, we present balanced bagging of SVM algorithms. It uses under sampling of the majority class to separate the ith class from the rest (this already seed-up the training task too). To get results within reasonable time the algorithms are parallelized on several machines / cores. We perform the training task of the k classifiers in a parallel way. In order to test the efficiency of our approach, we have used the following datasets: ImageNet 10 (made of the 10 largest classes of ImageNet data set (14 million images 21000 classes for the whole dataset)), ImageNet 100 (the 100 largest classes) and ILSVRC2010 (the 1000 largest classes). The parallelization task is performed via the use of two classical libraries: OpenMP (Open Multi Processing) and MPI (Message Passing Interface). For the classification of the 1000 largest classes of ImageNet dataset (900000 images, 12.5GB) our algorithm is 286 times faster than original PmSVM and 1434 times faster than state-of-the-art algorithms like LIBLINEAR with a relative increase of accuracy more than 20% compared to the latter one. This means we are able to perform the classification in a few (almost two) minutes while the original Pm-SVM performs in some hours (almost ten) and LIBLINEAR in some days (almost two).

1. Introduction

La plupart des méthodes de classification d'images actuelles comportent trois étapes principales : 1) l'extraction de caractéristiques locales de bas niveau dans l'image, 2) la construction d'un sac de mots visuels et 3) l'apprentissage par un algorithme de classification (Lin *et al.,* 2011). L'évaluation se fait ensuite sur des ensembles de données tels que Caltech 101 (Li *et al.,* 2007), Caltech256 (Griffin *et al.,* 2007) ou Pascal Voc (Everingham *et al.,* 2010) de tailles suffisamment restreinte pour tenir en mémoire vive. Mais l'arrivée de nouveaux ensembles de données tels qu'ImageNet (21 841 classes et 14,2 millions d'images) (Deng *et al.,* 2009) pose de nouveaux challenges. Avec une telle quantité d'images, l'exécution d'un algorithme de classification efficace peut nécessiter plusieurs semaines, voire mois. Les algorithmes d'apprentissage les plus utilisés dans ce cas sont les algorithmes de SVM linéaires, pour leur rapidité, mais les SVM linéaires donnent de moins bons résultats que les non linéaires dans le cas de la classification d'images. Wu (2012) a proposé l'algorithme de Power Mean SVM (PmSVM) qui donne de meilleurs résultats que Liblinear et d'autres algorithmes à noyaux additifs. Cependant sur un ensemble de données tel que ILSVRC 2010 il nécessite beaucoup de temps

(presqu'une journée) pour effectuer l'apprentissage des différents cas binaires (classification un contre un ou un contre le reste). Nous proposons deux améliorations de cet algorithme :

1) un bagging équilibré de PmSVM, pour éviter le déséquilibre entre les classes dans le cas un contre tous et qui permet par ailleurs de ne pas effectuer l'apprentissage sur la totalité des données tout en garantissant une convergence rapide vers la solution optimale ;

2) la parallélisation de l'apprentissage de tous les cas binaires pour obtenir un résultat dans un temps raisonnable (de l'ordre de quelques minutes pour ILSVRC 2010).

Les performances de notre algorithme sont évaluées sur l'ensemble de données ILSVRC 2010 qui comporte les 1 000 plus grandes classes de l'ensemble de données ImageNet soit environ 900 000 images représentant 12,5 Go de données.

La suite de l'article est organisée de la manière suivante : la section 2 présente un bref état de l'art de la classification d'images à grande échelle, la section 3 présente l'algorithme de PmSVM avec ses améliorations dans la section 4. La section 5 présente les résultats numériques avant la conclusion et les travaux futurs.

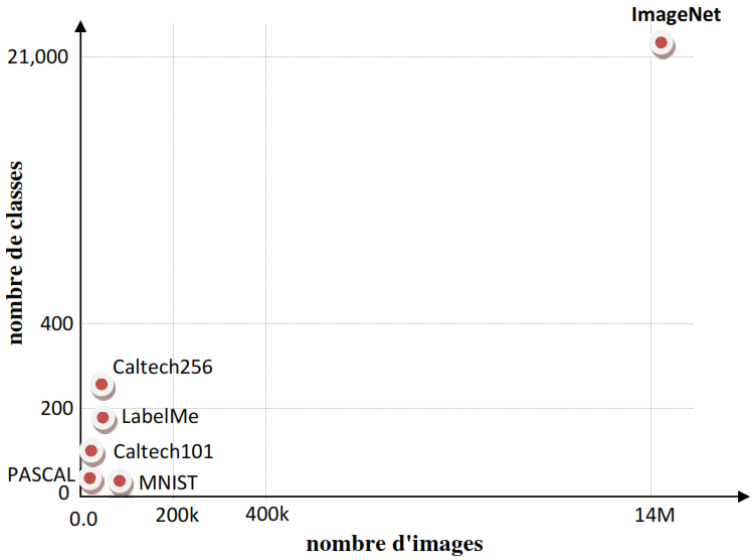

Figure 1. Comparaison d'ImageNet et des autres ensembles de données d'images

2. État de l'art

De nombreux travaux sur la classification d'images ont été présentés en utilisant des descripteurs locaux de bas niveau (comme les SIFT, Scale Invariant Feature Transform (Lowe, 1999), les SURFs (Bay *et al.*, 2006) ou les GLOHs (Mikolajczyk

et Schmid, 2005)), le modèle de sac de mots visuels (Csurka *et al.,* 2004) (bag of words ou BoW) et l'apprentissage à l'aide de séparateurs à vaste marge (Support Vector Machine ou SVM) (Vapnik, 1995). Des approches plus récentes apportent des améliorations tangibles (Griffin et Perona, 2008) en utilisant la structure hiérarchique de l'ensemble de données (utilisation de classes de classes). Cependant dans le cas d'ensembles de données comme ImageNet (Deng *et al.,* 2009) avec plusieurs milliers de classes et millions d'images, ces approches deviennent très difficiles à mettre en œuvre. Les deux principaux problèmes sont le temps de calcul et la place mémoire nécessaire (comme détaillé dans la section 4).

(Fergus *et al.,* 2009) se sont intéressés à la classification de 126 classes d'images étiquetées manuellement, (Wang *et al.,* 2009). ont traités 315 classes, (Li *et al.* 2009) ont effectué la classification de 500 classes de paysages avec plus de deux millions d'images. Sur un petit sous-ensemble de 10 classes, ils ont amélioré les résultats de la classification en portant la taille du sac de mots à 80 000. La disponibilité d'ensembles de données de plus en plus volumineux comme ImageNet conduit à essayer d'obtenir des stratégies pour améliorer la précision des algorithmes de classification en évitant l'utilisation devenue trop coûteuse des noyaux non linéaires. Les travaux récents dans cet axe (Deng *et al.,* 2010 ; Lin *et al.,* 2011 ; Perronnin *et al.,* 2010 ; Sánchez et Perronnin, 2011) effectuent une transformation des données initiales dans un espace de dimension augmentée par une fonction de noyau non linéaire, puis utilisent un algorithme de SVM linéaire dans l'espace ainsi obtenu. Leur argument est que la classification avec un SVM à noyau non linéaire dans l'espace original est équivalente à la classification avec un SVM linéaire dans l'espace de dimension augmentée. Lin *et al.,* (2011) utilisent des descripteurs locaux qui sont obtenus en utilisant le Local Coordinate Coding (Yu *et al.,* 2009) ou le Super-vector Coding (Zhou *et al.,* 2010), avec l'utilisation de pyramides spatiales l'espace obtenu est de dimension 262 000. Pour l'apprentissage, les auteurs proposent une méthode d'Averaging Stochastic Gradient Descend (ASGD) qui nécessite 4 jours d'apprentissage (un contre tous) sur les 1 000 classes de ILSVRC 1 000 sur trois machines à 8 cœurs. Sanchez et Perronin (2011) montrent que plus la taille de l'ensemble de données est importante et plus le nombre de dimensions sera important pour obtenir une bonne précision. Pour obtenir de bons résultats sur l'ensemble de données ILSVRC-2010 les auteurs utilisent des pyramides spatiales et obtiennent environ 524000 dimensions, les données sont ensuite compressées en utilisant le Product Quantizer (Jégou *et al.,* 2011) avant l'étape d'apprentissage qui nécessite un jour et demi sur une machine munie de 16 cœurs.

À l'opposé de ces méthodes basées sur des SVM linéaires, des articles récents ont montré que les noyaux linéaires donnent des résultats moins bons en termes de précision que les noyaux non linéaires et qu'en particulier les noyaux additifs donnent de meilleurs résultats que les noyaux basés sur un produit scalaire (Maji *et al.,* 2008 ; Vedaldi et Zisserman, 2012 ; Wu, 2010 ; Maji *et al.,* 2013). Bien entendu ce type d'approche a un coût beaucoup plus élevé en ce qui concerne le temps d'exécution qui peut être jusqu'à plus de 1 000 fois plus important. Des progrès récents (Griffin et Perona, 2008) ont permis aux noyaux additifs d'être plus performants que les noyaux linéaires à la fois en ce qui concerne le temps

d'exécution et la précision. Leur algorithme de PmSVM est trois fois plus rapide que Liblinear et améliore la précision de 4.6 % à 7.1 % par rapport aux meilleurs algorithmes, mais il nécessite encore 18 h de calcul sur un seul cœur sur ILSVRC2010.

En classification avec un grand nombre de classes à l'aide de SVM, l'approche un contre le reste est très utilisée car simple à mettre en œuvre. Cependant, la séparation de la classe *i* des autres classes pose le problème de données déséquilibrées. Nous avons étudié le bagging (Breiman, 1996) et le bagging équilibré (Hido et Kashima, 2008) en classification de données déséquilibrées. Nous présentons une version parallèle de bagging équilibré de PmSVM pour la classification de très grands ensembles de données d'images.

3. Power mean SVM

Soit un problème de classification linéairement séparable avec *m* points en dimension *n* avec des étiquettes de classes +/-1. Les algorithmes de SVM cherchent alors l'hyperplan de séparation des données le plus éloigné possible de chacune des deux classes.

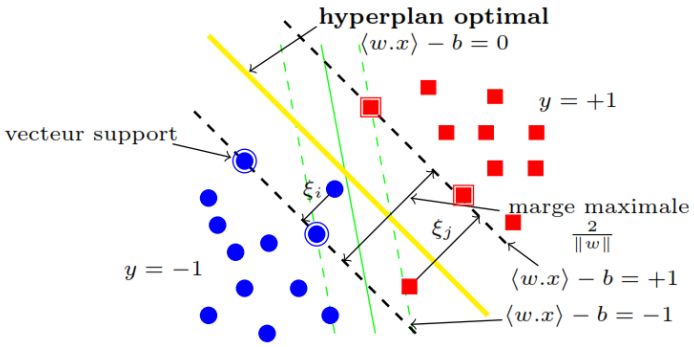

Figure 2. Séparatrice linéaire en deux classes de points

Il essaye à la fois de maximiser la marge (espace entre les deux classes) et de minimiser les erreurs (les point du mauvais côté de la marge). Ceci peut être résolu à l'aide d'un programme quadratique :

$$\min \Psi(w,b,\xi) = \frac{1}{2}\| w \|^2 + C\sum_{i=1}^{m}\xi_i$$
$$avec : y_i(w.x_i - b) + \xi_i \geq 1 \qquad (1)$$

$$\xi_i \geq 0$$

où *C* est une constante positive pour régler l'influence de la marge et des erreurs. Les vecteurs supports (pour lesquels les $\xi_i > 0$) sont obtenus par la résolution du programme quadratique. Ensuite l'appartenance d'un point à une classe ou l'autre sera simplement obtenu par :

$$\text{predict}(x) = \text{sign}(w.x - b) \quad (2)$$

L'équation (1) est équivalente à l'équation suivante dans sa formulation duale :

$$\min 1/2 \sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{i=1}^{m} \alpha_i \quad (3)$$

avec $\sum_{i=1}^{m} y_i \alpha_i = 0$

et $C \geq \alpha_i \geq 0$ (i=1,…,m)

Pour changer de fonction de noyau, il suffit alors simplement de modifier la fonction *K* dans l'équation (3). L'algorithme de PmSVM remplace la fonction de noyau *K* par le *Power Mean Kernel M*, qui est la forme générale de beaucoup de noyaux additifs tels que le χ^2 ou le noyau d'Hellinger.

$$M_p\langle x_i, x_j\rangle = \sum_{d=1}^{n} (x_{id}^p + x_{jd}^p)^{\frac{1}{p}} \quad (4)$$

avec *p* une constante réelle.

L'algorithme de PmSVM utilise une méthode de gradient pour résoudre la tâche d'apprentissage. De plus le calcul du gradient (Yuan et al, 2012) dans l'algorithme peut être approximativement estimé par une régression polynomiale (Wu, 2012), ce qui rend l'algorithme très efficace en ce qui concerne le temps de calcul.

4. Améliorations du Pm-SVM

La plupart des algorithmes de SVM actuels ne savent traiter que le cas de deux classes. Il y a plusieurs méthodes pour permettre le passage à un plus grand nombre de classes, on peut les regrouper en deux grandes familles : la première traite le cas multiclasse comme un problème d'optimisation (Weston et Watkins, 1999 ; Guermeur, 2007), la seconde décompose le cas multiclasse en un ensemble de cas binaires, comme le un contre un (Krebel, 1999), le un contre le reste (Vapnik, 1995) ou le DDAG, Decision Directed Acyclic Graph (Platt *et al.*, 2000).

Les approches un contre un et un contre le reste sont les plus utilisées car elles sont très simples à mettre en œuvre. Soit un ensemble à *k* classes, l'approche un contre le reste construit *k* classifieurs, le i^{e} séparant la classe *i* des autres classes. La stratégie un contre un construit elle k(k-1)/2 classifieurs en utilisant toutes les combinaisons possibles de paires de classes. Dans les deux cas la classe est ensuite prédite par un vote majoritaire. Dans le cas où le nombre de classe est important, l'approche un contre un est trop coûteuse à mettre en œuvre, c'est donc l'approche un contre le reste qui est préférée, même si elle nécessite un long temps d'apprentissage. C'est pourquoi nous proposons deux améliorations de l'algorithme de PmSVM : la première consiste à utiliser un bagging de PmSVM en échantillonnant de manière équilibrée les individus et la seconde est d'effectuer ces calculs en parallèle.

4.1. Bagging équilibré de PmSVM

Dans le cas d'un grand nombre de classes, c'est l'approche un contre le reste qui est préférée, c'est-à-dire que l'on essaye de séparer la classe *i* des (k-1) autres classes. Si l'on a un grand nombre de classes, par exemple 1 000 cela conduit à un très fort déséquilibre entre la classe *i* et les 999 autres classes, l'une ne représente que 0,1 % de l'autre, donc absorber la classe minoritaire donne tout de même 99,9 % de bonne classification. Comme expliqué dans (Japkowicz, 2000 ; Weiss et Provost, 2003 ; Visa et Ralescu, 2005) ou (Lenca *et al.,* 2008 ; Pham *et al.,* 2008), les solutions peuvent être proposées soit au niveau des données, soit au niveau de l'algorithme. Au niveau des données, on peut choisir de sous-échantillonner la classe majoritaire (Liu *et al.,* 2009 ; Ricamato *et al.,* 2008) ou sur-échantillonner la classe minoritaire (Chawla *et al.,* 2003). Au niveau de l'algorithme on peut choisir d'augmenter le coût de mauvaise classification de la classe minoritaire.

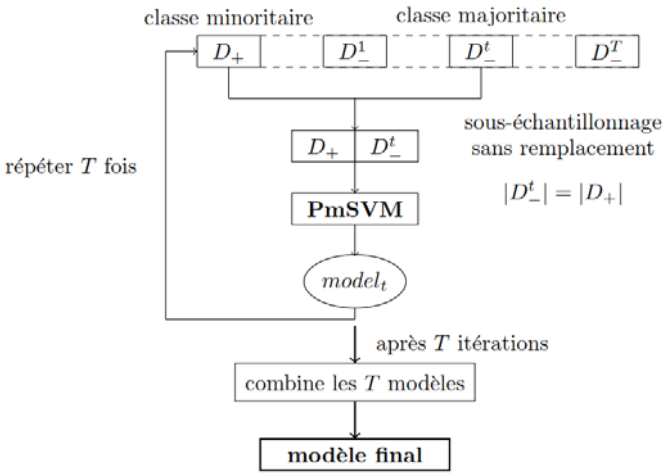

Figure 3. Sous échantillonnage de la classe majoritaire pour l'apprentissage de SVM

Le choix que nous avons fait est de sous-échantillonner la classe majoritaire, ce qui est la solution la moins coûteuse car le nombre de points de l'ensemble d'apprentissage est considérablement réduit par rapport à l'utilisation d'un coût de mauvaise classification sur l'ensemble total des données. Le bagging équilibré va donc échantillonner un nombre de points de la classe majoritaire égal au nombre de points de la classe minoritaire. On obtient donc l'algorithme 1 pour séparer la i^e classe (positive) des (k-1) autres classes (négative).

Algorithme 1. Bagging équilibré de PmSVM

Input : D+ : données de la classe +
D- : données de la classe –
T : nombre de classifieurs
Output : PmSVM model
For i=1 to T do
D'- = sampling (D-)
PmSVM (D+, D'-)
EndFor
Combine T PmSVM models

Dans le cas de séparatrice linéaire, la marge est la distance minimum entre les enveloppes convexes des deux classes. En sous-échantillonnant la classe majoritaire, cette marge peut devenir plus grande. Il est donc moins complexe d'effectuer l'apprentissage sur les sous-ensembles de données que sur l'ensemble de données global. L'algorithme de PmSVM converge rapidement vers la solution (Wu, 2012). Dans la pratique, nos expérimentations ont montré qu'en choisissant $T = \sqrt{(|D-| / |D+|)}$ l'algorithme de bagging équilibré de PmSVM donne de bons résultats dans un temps restreint (quelques minutes) comme nous le détaillons dans la section 5.

Enfin les *T* modèles ne sont pas combinés à l'aide d'un vote majoritaire comme c'est souvent le cas, mais combinés dans un vecteur de poids pour obtenir le modèle final.

4.2. Apprentissage parallèle de PmSVM

Pour améliorer encore la vitesse de calcul de l'algorithme de bagging équilibré de PmSVM, nous allons le paralléliser pour pouvoir bénéficier des architectures actuelles (machines multicœurs ou cluster de machines). Notre algorithme entraine *k* classifieurs dans le cas de *k* classes et ce de manière totalement indépendante, c'est donc un premier niveau de parallélisation. On peut aussi remarquer que les calculs des *T* classifieurs dans l'algorithme 1 sont eux aussi totalement indépendants les uns des autres. C'est un deuxième niveau de parallélisation que nous allons utiliser.

La mise en œuvre de l'algorithme parallèle se fait en utilisant deux modèles courants : MPI (Message Passing Interface (MPI Forum)) et OpenMP (Open Multi Processing (OpenMP Architecture Review Board, 2008)). MPI offre un mécanisme standardisé d'envoi de messages qui permet de distribuer un calcul sur un ensemble

de processus (potentiellement sur des machines différentes). OpenMP quant à lui offre un mécanisme de mémoire partagé pour effectuer des calculs en parallèle sur des processeurs partageant une mémoire commune. MPI est le modèle dominant en parallélisation, cependant dans notre cas, il nécessite le chargement de l'ensemble total de données (environ 25 Go) pour chaque processus MPI. Nous allons donc mettre en œuvre un mécanisme hybride utilisant à la fois MPI et OpenMP comme décrit dans l'algorithme 2.

Le choix du nombre de processus MPI sera bien entendu fonction de l'architecture utilisée (nombre de machines et quantité de mémoire disponible).

Algorithme 2. OpenMP/MPI Bagging de PmSVM

Input : D : ensemble de données (k classes)
nbMPI : nb de proc. MPI
Output : PmSVM model
MPI-proc1
ParallelOMP-For i_1*=1 to* k_1 *do*
PmSVM (classe i_1 *contre le reste)*
EndParallelFor
End-proc1
…
MPI-proc-nbMPI
ParallelOMP-For i_p*=1 to* k_p *do*
PmSVM (classe i_p *contre le reste)*
EndParallelFor
End-proc-nbMPI

5. Résultats expérimentaux

5.1. Ensembles de données

Nous allons utiliser différentes versions de l'ensemble de données ImageNet pour évaluer les performances de notre mise en œuvre en ce qui concerne le temps d'exécution et la précision de l'algorithme de classification.

ImageNet 10. Ce sous-ensemble d'ImageNet contient les 10 plus grandes classes de l'ensemble original soit 24 807 images (2.4 Go). Il y a plus de 2 000 images par classe, on va donc en prendre 90 % pour l'apprentissage et 10 % pour le test. Un modèle de sac de mots est construit en utilisant libHIK (Wu *et al.*, 2011) avec des descripteurs SIFT (Lowe, 1999) et une taille de vocabulaire de 1000 mots. La fonction de noyau explicite de (Vedaldi et Zisserman, 2012) est utilisée en prétraitement de l'algorithme d'apprentissage, on obtient alors des vecteurs en dimensions 15 000 avec une taille de l'ensemble d'apprentissage de 2.6 Go de descripteurs.

ImageNet 100. Ce sous-ensemble d'ImageNet contient les 100 plus grandes classes de l'ensemble original soit 183 116 images (23.6 Go). 50 % des images sont utilisées comme ensemble d'apprentissage et 50 % pour le test. Le même prétraitement qu'ImageNet10 est effectué ce qui donne une taille de l'ensemble d'apprentissage de 8 Go de descripteurs.

ILSVRC2010. Ce sous-ensemble d'ImageNet contient les 1 000 plus grandes classes de l'ensemble original soit 1 261 406 images (126 Go), 50 000 (5,3 Go) pour la validation et 150 000 (16 Go) pour l'ensemble de test. Pour pouvoir comparer nos résultats à ceux de (Wu, 2012), nous avons utilisé la même méthode que celle décrite dans leur article pour encoder les données en un vecteur de dimension 21 000 avec le sac de mots de (Berg *et al,* 2010). Cela nous donne finalement un ensemble de 887 816 images et 12,5 Go de descripteurs pour l'ensemble d'apprentissage.

5.2. Résultats de l'apprentissage

Nous allons comparer notre algorithme avec Liblinear (l'une des références en SVM pour les noyaux linéaires), la version originale de PmSVM et nos versions parallèles qui sont au nombre de 4 : deux qui n'utilisent qu'OpenMP (une sans équilibrer les données omp-PmSVM, une avec équilibrage omp-iPmSVM) deux hybrides OpenMP/MPI avec les deux mêmes sous cas mpi-omp-PmSVM et mpi-omp-iPmSVM.

Nous avons effectué deux séries d'expérimentations sur ces ensembles de données, la première sur une machine unique Intel(R) Xeon(R), CPU X5650, 2.67 GHz, 24 cœurs et 144 Go de mémoire vive avec les ensembles de données ImageNet 10, ImageNet 100 et ILSVRC2010 et la seconde sur un ensemble de cinq machines à 16 cœurs Intel Xeon E5645 et 48 Go de mémoire vive, soit un total de 80 cœurs avec les ensembles de données ImageNet 100 et ILSVRC2010. Nous rapportons donc successivement les résultats obtenus de ces deux expérimentations.

Les réglages de paramètres utilisés pour les différents algorithmes sont les suivants :

– PmSVM : p=-1, équivalent à un noyau χ^2 et C=0.01,

– LIBLINEAR : paramètres par défaut, C=1,

– iPmSVM : mêmes paramètres que PmSVM.

Nous avons choisi les mêmes paramètres pour l'algorithme de PmSVM que l'algorithme original pour pouvoir comparer équitablement nos résultats et les paramètres par défaut de Liblinear.

5.2.1. Première expérimentation

Les temps qui sont rapportés sont les temps de calcul, ils ne prennent pas en compte les temps de chargement des ensembles de données en mémoire vive. Plusieurs points doivent être précisés avant de commenter les résultats présentés dans les tableaux 1 à 3 en ce qui concerne les temps d'apprentissage.

Tableau 1. Temps d'apprentissage (mn) ImageNet10

#omp-threads	1	5	10	Précision
LIBLINEAR	2.02			75.09
PmSVM	6.23			73.16
omp-PmSVM	6.23	1.75	0.98	73.16
omp-iPmSVM	4.66	1.34	0.77	72.79
2mpi-omp-PmSVM	3.29	0.78	0.76	73.16
2mpi-omp-iPmSVM	2.44	0.67	0.65	72.79

Tableau 2. Temps d'apprentissage (mn) ImageNet100

#omp-threads	1	5	10	Précision
LIBLINEAR	30.41			54.07
PmSVM	165.45			50.17
omp-PmSVM	165.45	21.12	14.52	50.17
omp-iPmSVM	47.67	12.97	9.09	49.42
2mpi-omp-PmSVM	62.18	10.63	8.85	50.17
2mpi-omp-iPmSVM	21.56	5.68	4.16	49.42

Tableau 3. Temps d'apprentissage (mn) et précision sur ILSVRC2010

#omp-threads	1	5	10	Précision
LIBLINEAR	3106.48			21.11
PmSVM	1132.03			25.64
omp-PmSVM	1132.03	231.00	152.87	25.64
omp-iPmSVM	173.26	39.55	23.63	25.35
2mpi-omp-PmSVM	550.04	119.17	102.81	25.64
2mpi-omp-iPmSVM	72.04	16.69	13.08	25.35

1) les temps mentionnés dans ces tableaux sont les temps utilisateur et non pas les temps CPU. En effet, si un processus met par exemple deux minutes pour s'exécuter sur un cœur d'une machine, s'il est parallélisé sur deux cœurs, il mettra le même total de temps CPU pour effectuer le même calcul, soit deux minutes CPU, mais comme il utilise deux cœurs, l'utilisateur aura le résultat au bout d'une minute au lieu de deux. Donc rapporter le temps CPU n'avait aucun intérêt ici.

2) Ce qui est écrit dans le point 1) (on divise le temps utilisateur par deux si l'on utilise deux cœurs au lieu d'un) n'est vrai que si un cœur supplémentaire est disponible sur la machine en question. Ici la machine qui a servi pour ces expérimentations est partagée entre plusieurs utilisateurs, nous ne sommes pas du tout sûrs de sa disponibilité ce qui explique que les gains diminuent au fur et à mesure que l'on souhaite utiliser de plus en plus de cœurs.

3) Même si la quantité de mémoire vive de la machine est plutôt confortable a priori, nous ne sommes pas les seuls utilisateurs de la machine et chaque processus MPI nécessite environ 25 Go de mémoire pour s'exécuter, c'est pourquoi nous avons volontairement limité le nombre de processus MPI à 2 (ce qui correspond aux deux dernières lignes des tableaux).

Malgré cela, nous pouvons tirer des conclusions des résultats de ces expérimentations. Tout d'abord intéressons-nous à l'algorithme de bagging équilibré de PmSVM (noté iPmSVM) comparé à l'algorithme original de PmSVM. On constate que sans parallélisation l'algorithme de bagging équilibré de PmSVM est plus rapide que l'algorithme de PmSVM d'un facteur variant de 1,33 (10 classes) à 6,54 (1 000 classes). On retrouve le même type de variation par rapport à Liblinear : notre algorithme est 2,3 fois plus lent sur l'ensemble de données à 10 classes et 18 fois plus rapide sur les 1 000 classes. Ces résultats sont cohérents avec ceux que nous avions déjà obtenus sur du boosting de SVM dans le cadre de la classification de texte (Do *et al.,* 2008) : les différentes étapes du boosting sont trop coûteuses sur les ensembles de données de petites tailles, mais elles deviennent intéressantes dès que la taille augmente suffisamment.

En ce qui concerne la précision de l'algorithme de bagging équilibré de PmSVM par rapport à l'algorithme de PmSVM, le gain en temps de calcul est au détriment de la précision mais dans une proportion qui reste (très) faible : 73,16 %/72,79 % (10 classes) à 25,64 %/25,35 % (1 000 classes). Ici encore la comparaison des algorithmes de PmSVM par rapport à LIBLINEAR montre que ce dernier est plus performant sur les petits ensembles de données (10 et 100 classes) : 75,09 %/73,16 % mais est moins bon sur les grands ensembles de données (1 000 classes) 21,11 %/ 25,64%, ce qui représente une amélioration relative de plus de 20 % par rapport à Liblinear.

Enfin terminons par les résultats de la parallélisation de notre algorithme de bagging équilibré de PmSVM. Si l'on ne prend en compte que la colonne avec 5 threads OpenMP et 2 MPI (soit un total de 10 threads) les gains en vitesse vont d'un facteur 3 (10 classes) à plus de 190 (1 000 classes) par rapport à Liblinear. Si l'on prend en compte le meilleur résultat quel que soit le nombre de threads, on obtient alors un algorithme qui est 240 fois plus rapide que Liblinear et 90 fois plus rapide que la version originale du PmSVM. Comme nous l'avons expliqué ces valeurs sont des minima, les valeurs réelles doivent être supérieures à cela, d'après les variations observées entre les colonnes 5 et 10 threads OpenMP (soit un total de 10 ou 20 threads) des tableaux 1 à 3.

5.2.2. Deuxième expérimentation

La deuxième expérimentation que nous avons menée sur les mêmes ensembles de données utilise un ensemble de 5 machines identiques, chaque machine est équipée de 16 cœurs soit un total de 80 cœurs disponibles pour le calcul et 48 Go de mémoire par machine (pour 16 cœurs). Nous avons donc fait varier le nombre de processus de 1 à 16 soit le nombre maximal de cœurs alloués par machine.

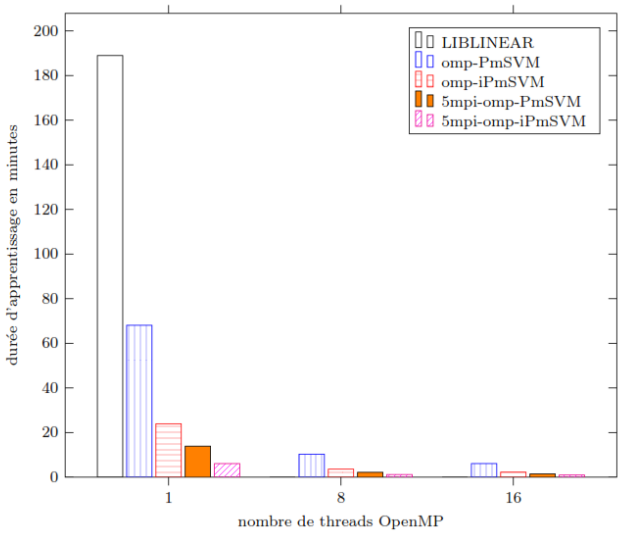

Figure 4. Temps d'apprentissage des SVM en fonction du nombre de threads pour ImageNet 100

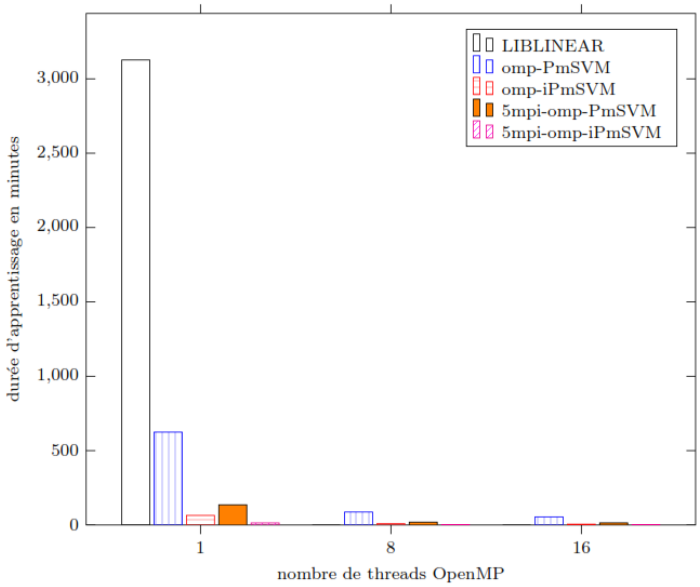

Figure 5. Temps d'apprentissage des SVM en fonction du nombre de threads pour ILSVRC 2010

Tableau 5. Temps de calcul (minutes) des algorithmes sur ImageNet 100

# Open-MP threads	1	8	16
LIBLINEAR	188.97		
omp-PmSVM	68.08	10.32	6.10
omp-iPmSVM	23.94	3.59	2.25
5mpi-omp-PmSVM	13.91	2.18	1.45
5mpi-omp-iPmSVM	6.09	1.18	0.96

Tableau 6. Temps de calcul (minutes) des algorithmes sur ILSVRC2010

# Open-MP threads	1	8	16
LIBLINEAR	3126.78		
omp-PmSVM	624.56	87.22	54.57
omp-iPmSVM	64.41	8.56	5.48
5mpi-omp-PmSVM	135.73	19.13	14.13
5mpi-omp-iPmSVM	13.80	2.38	2.18

La deuxième expérimentation que nous avons effectuée ne nous a pas donnée les résultats escomptés. Si l'augmentation du nombre de processus de 1 à 8 donne bien une accélération de cet ordre pour le temps d'exécution sur ImageNet100 et ILSVRC2010, mais il n'en n'est pas du tout de même lorsque l'on passe à 16 threads par machine. Les temps sont améliorés mais pas autant qu'attendu. Nous n'avons pas encore identifié de manière certaine quelle en était la cause.

Notre algorithme de PmSVM est tout de même 286 fois plus rapide que la version originale de PmSVM et 1 434 fois plus rapide que Liblinear. Il ne faut que 2 minutes pour effectuer l'apprentissage des 1 000 classifieurs de l'ensemble de données ILSVRC2010 (contre 2 jours pour Liblinear et plus de 10 heures pour la version originale de PmSVM).

6. Conclusion et travaux futurs

Nous avons présenté des améliorations de l'algorithme de Power Mean SVM (PmSVM) pour la classification d'ensembles de données de très grandes tailles comme ImageNet. Ces améliorations portent sur deux points : d'une part, dans le cadre de la classification d'ensembles de données comportant un très grand nombre de classes, l'approche la plus souvent utilisée pour des problèmes de coût de calcul est l'approche un contre le reste. Elle conduit inévitablement à des classes très déséquilibrées. Pour y remédier nous avons proposé un algorithme de bagging équilibré de PmSVM qui permet de sous-échantillonner la classe majoritaire et d'améliorer le temps d'exécution (presque) sans perte sur la précision de l'algorithme de classification.

Notre algorithme permet une amélioration relative de 20 % en précision par rapport aux références telles que Liblinear (dans le cas ILSVRC2010). Par ailleurs le traitement d'ensembles de données de données de très grandes tailles est inévitablement coûteux en temps. Nous en avons proposé une version parallèle 286 fois plus rapide que la version originale et 1 434 fois plus rapide que Liblinear (ces valeurs étant des minima à cause des conditions expérimentales). Il ne faut que 2,18 minutes pour effectuer la classification des 1 000 plus grandes classes de l'ensemble de données ImageNet avec une précision similaire aux algorithmes usuels… Ces valeurs peuvent évidemment être encore améliorées en utilisant plus de ressources (plus de machines et ou cœurs).

Cependant la tâche d'apprentissage nécessite presque 30 Go de mémoire vive, il est donc nécessaire de s'intéresser aux approches incrémentales qui ne nécessitent pas le chargement de l'ensemble de données en mémoire vive pour effectuer la classification (Poulet et Pham, 2010). Une autre possibilité serait de compresser l'ensemble d'apprentissage et de le décompresser à la volée (Sanchez et Perronnin, 2011).

Enfin, les algorithmes de SVM ont été utilisés ici pour la catégorisation de grands ensembles de données d'images. Mais leur champ d'application est beaucoup plus vaste que cela et les améliorations que nous avons proposées restent valables quelles que soient les applications qui manipulent des ensembles de données de grandes tailles avec un grand nombre de classes.

Remerciements

Ces travaux de recherche sont partiellement financés par la région Bretagne (France) et le VIED (Vietnam International Education Development).

Bibliographie

Bay H., Tuytelaars T. et Van Gool L. (2006). SURF: Speeded Up Robust Features. *Proc. of 9th European Conference on Computer Vision*, p. 404-017.

Benabdeslem K. et Bennani Y. (2006). Dendogram based svm for multi-class classification. *Journal of Computing and Information Technology,* 14(4):283-289.

Berg A., Deng J., Li F.-F. (2010). *Large scale visual recognition challenge 2010*. Technical Report.

Breiman L. (1996). Bagging predictors. *Machine Learning* 24(2) : 123-140.

Chawla N. V., Lazarevic A., Hall L. O., et Bowyer K. W. (2003). Smoteboost: improving prediction of the minority class in boosting. In the *Principles of Knowledge Discovery in Databases,* p. 107-119.

Cristianini N. et Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Csurka G., Dance C. R., Fan L., Willamowski J., et Bray C. (2004). Visual categorization with bags of keypoints. *Workshop on Statistical Learning in Computer Vision, European Conference on Computer Vision,* p. 1-22.

Dalal N. et Triggs B. (2005). Histograms of oriented gra dients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* p. 886-893. IEEE Computer Society.

Deng J., Berg A. C., Li K., et Li F.-F. (2010). What does classifying more than 10, 000 image categories tell us? *European Conference on Computer Vision,* p. 71-84.

Deng J., Dong W., Socher R., Li L.-J., Li K., et Li F.- F. (2009). Imagenet: A large-scale hierarchical image database. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* p. 248-255.

Do T-N, Nguyen V-H., Poulet F. (2008). Speed up SVM Algorithms for Massive Classification Tasks. *Proc. of Advanced Data Mining and Applications (ADMA'08),* Chengdu, China, C.Tang, C.X.Ling, X.Zhou, N.Cercone, X.i (Eds), Lecture in Computer Science, vol. 5139, Springer-Verlag, Oct.2008, p. 147-157.

Everingham M., Van Gool L., Williams C. K. I., Winn J., et Zisserman A. (2010). The Pascal visual object classes (voc) challenge. *International Journal of Computer Vision,* 88(2):303-338.

Fergus R., Weiss Y., et Torralba A. (2009). Semi- supervised learning in gigantic image collections. *Advances in Neural Information Processing Systems,* p. 522-530.

Griffin G., Holub A., et Perona P. (2007). *Caltech-256 Object Category Dataset*. Technical Report CNS-TR-2007-001, California Institute of Technology.

Griffin G. et Perona D. (2008). Learning and using taxonomies for fast visual categorization. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

Guermeur Y. (2007). *Svm multiclasses, the´orie et applications*. HDR, Université Nancy 1.

Hido S. et Kashima H. (2008). Roughly balanced bagging for imbalanced data. *Proc. of SIAM International Conference on Data Mining,* p. 143-152.

Hsieh C.-J., Chang K.-W., Lin C.-J., Keerthi S. S., et Sundararajan S. (2008). A dual coordinate descent method for large-scale linear SVM. *International Conference on Machine Learning,* p. 408-415.

Japkowicz N., editor (2000). *AAAI'Workshop on Learning from Imbalanced Data Sets*, number WS-00-05 in AAAI Tech Report.

Je´gou H., Douze M., et Schmid C. (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 33(1):117-128.

Krebel U. (1999). Pairwise classification and support vector machines. *Advances in Kernel Methods: Support Vector Learning,* p. 255-268.

Lazebnik S., Schmid C., et Ponce J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* p. 2169-2178.

Lenca P., Lallich S., Do T. N., et Pham N. K. (2008). A comparison of different off-centered entropies to deal with class imbalance for decision trees. *The Pacific Asia Conference on Knowledge Discovery and Data Mining, LNAI* 5012, p. 634-643. Springer-Verlag.

Li F.-F., Fergus R., et Perona P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding,* 106(1):59-70.

Li Y., Crandall D. J., et Huttenlocher D. P. (2009). Landmark classification in large-scale image collections. *IEEE 12th International Conference on Computer Vision,* p. 1957-1964. IEEE.

Lin Y., Lv F., Zhu S., Yang M., Cour T., Yu K., Cao L., et Huang T. S. (2011). Large-scale image classification: Fast feature extraction and svm training. *IEEE Computer Society Conference on Computer Vi- sion and Pattern Recognition,* p. 1689-1696.

Liu X.-Y., Wu J., et Zhou Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics,* Part B, 39(2):539-550.

Lowe D. G. (1999). Object recognition from local scale-invariant features. *Proc. of International Conference on Computer Vision,* vol. 2, p.1150-1157.

Maji S., Berg A. C., et Malik J. (2008). Classification using intersection kernel support vector machines is efficient. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

Maji S., Berg A. C., et Malik J. (2013). Efficient classification for additive kernel SVMs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):66-77.

Mikolajczyk K. et Schmid C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 10(27):1615-1630.

MPI-Forum. Mpi: *A message passing interface standard*.

OpenMP Architecture Review Board (2008). *OpenMP application program interface version 3.0*.

Perronnin F., Sánchez J., et Liu Y. (2010). Large-scale image categorization with explicit data embedding. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* p. 2297-2304.

Pham N. K., Do T. N., Lenca P., et Lallich S. (2008). *Using local node information in decision trees: coupling a local decision rule with an off-centered entropy*. I*nternational Conference on Data Mining,* p. 117-123, Las Vegas, Nevada, USA. CSREA Press.

Platt J., Cristianini N., et Shawe-Taylor J. (2000). Large margin dags for multiclass classification. *Advances in Neural Information Processing Systems,* 12:547-553.

Poulet F., Pham N-K. (2010). *High Dimensional Image Categorization*. *6th International Conference of ADMA'10, Advanced Data Mining and Applications,* Chongqing, China, Springer LNCS 6440, p. 147-157.

Ricamato M. T., Marrocco C., et Tortorella F. (2008). Mcs-based balancing techniques for skewed classes: An empirical comparison. *International Conference on Pattern Recognition,* p. 1-4.

Sánchez J. et Perronnin F. (2011). High-dimensional signature compression for large-scale image classification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* p. 1665-1672.

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

Vedaldi A., Gulshan V., Varma M., et Zisserman A. (2009). Multiple kernels for object detection. *IEEE 12th International Conference on Computer Vision*, p. 606-613.

Vedaldi A. et Zisserman A. (2012). Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 34(3):480-492.

Visa S. et Ralescu A. (2005). Issues in mining imbalanced data sets - A review paper. *Midwest Artificial Intelligence and Cognitive Science Conf.,* p. 67-73, Dayton, USA.

Vural V. et Dy J. (2004). A hierarchical method for multi-class support vector machines. *Proceedings of the twenty-first international conference on Machine learning,* p. 831-838.

Wang C., Yan S., et Zhang H.-J. (2009). Large scale natural image classification by sparsity exploration. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* p. 3709-3712. IEEE.

Weiss G. M. et Provost F. (2003). Learning when train- ing data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315-354.

Weston J. et Watkins C. (1999). Support vector machines for multi-class pattern recognition. *Proceedings of the Seventh European Symposium on Artificial Neural Networks,* p. 219-224.

Wu J. (2010). A fast dual method for hik svm learning. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *European Conference on Computer Vision,* vol. 6312 of Lecture Notes in Computer Science, p. 552-565. Springer.

Wu, J. (2012). Power mean svm for large scale visual classification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* p. 2344-2351.

Wu J., Tan W.-C., et Rehg J. M. (2011). Efficient and effective visual codebook generation using additive kernels. *Journal of Machine Learning Research,* 12:3097-3118.

Yu K., Zhang T., et Gong Y. (2009). Nonlinear learning using local coordinate coding. *Advances in Neural Information Processing Systems,* p. 2223-2231.

Yuan G.-X., Ho C.-H., et Lin C.-J. (2012). Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584-2603.

Zhou X., Yu K., Zhang T., et Huang T. S. (2010). Image classification using super-vector coding of local image descriptors. *European Conference on Computer Vision,* p. 141-154.

Article reçu le 30/09/2013

Accepté le 12/05/2014