Régression logistique pour la classification d'images à grande échelle

Thanh-Nghi Do*, François Poulet**

*Université de Can Tho, Rue 3/2, Ninh Kieu,
Can Tho, Vietnam, dtnghi@cit.ctu.edu.vn
**Université de Rennes I - IRISA
Campus de Beaulieu, 35042 Rennes Cedex, France
francois.poulet@irisa.fr

Résumé. Nous présentons un nouvel algorithme parallèle de régression logistique (PAR-MC-LR) pour la classification d'images à grande échelle. Nous proposons plusieurs extensions de l'algorithme original de régression logistique à deux classes pour en développer une version efficace pour les grands ensembles de données d'images avec plusieurs centaines de classes. Nous présentons un nouvel algorithme LR-BBatch-SGD de descente de gradient stochastique de régression logistique en batch équilibré avec un apprentissage parallèle (approche un contre le reste) multi-classes sur de multiples cœurs. Les résultats expérimentaux sur des ensembles de données d'ImageNet montrent que notre algorithme est efficace comparés aux algorithmes de classification linéaires de l'état de l'art.

1 Introduction

La classification d'images vise à assigner automatiquement une catégorie prédéfinie à une image. Parmi ses nombreuses applications, on peut citer la reconnaissance de caractères manuscrits, la reconnaissance d'empreintes digitales oula reconnaissance de visages. Le nombre d'images stockées dans les différentes bases de données ne cesse de croître, par exemple, Facebook contient aujourd'hui plus de 90 milliards d'images, il a été estimé que les utilisateurs moyens d'appareils photos numériques prendront environ 100 000 photos au long de leur vie. Sur internet 65% du volume des données chargées correspond à des images.

La catégorisation d'images par le contenu est un véritable challenge d'importance aujourd'hui. Les approches les plus performantes de ces dernières années utilisent un modèle de "sac de mots visuels" (Bag of Words (Bow)) construits sur des descripteurs locaux des images. Le "sac de mots visuels" est une adaptation du sac de mots utilisé en catégorisation de textes où l'on a un ensemble de documents, chaque document contenant un ensemble de mots. Le modèle de sac de mots correspond aux nombres d'occurrence des mots dans les documents. Dans le cas du sac de mots visuels, on commence par calculer des descripteurs de bas niveau en des points particuliers de l'image. Les plus populaires sont les SIFTs (Scale Invariant Feature Transform (Lowe, 2004)), les SURFs (Speeded Up Robust Features (Bay et al., 2008) and DSIFTs (Dense SIFTs) (Bosch et al., 2007). Ces méthodes d'extraction de descripteurs

sont basées sur l'apparence des objets en des points particuliers de l'image et sont invariantes aux changements d'échelle, aux rotations, aux occlusions et aux variations de luminosité. Ensuite on utilise une méthode de quantification sur ces descripteurs (par exemple en appliquant un algorithme de clustering comme un k-means (MacQueen, 1967), (chaque cluster est alors considéré comme un mot visuel) pour finalement obtenir la distribution des SIFTs de chaque image dans l'ensemble des clusters obtenus. Le modèle de "sac de mots visuels" conduit à des ensembles de données ayant des très grands nombres de dimensions. Les Support Vector Machine ou Séparateurs à Vaste Marge (SVM) (Vapnik, 1995) sont les plus utilisés pour la classification de tels ensembles de données. Cependant des ensembles d'images tels qu'ImageNet (Deng et al., 2010) avec plus de 14 millions d'images et 21841 classes rendent la tâche de classification très complexe. Ce challenge nous a conduits à développer un nouvel algorithme efficace à la fois en temps d'exécution et précision de la classification. Nous présentons des extensions de l'algorithme de descente de gradient stochastique (SGD) (Shalev-Shwartz et al., 2007), (Bottou et Bousquet, 2008) pour créer un nouvel algorithme parallèle de SGD de régression logistique multi-classes (PAR-MC-LG) pour la classification de grands ensembles de données d'images avec un nombre important de classes. Nos contributions comprennent :

1. un algorithme de descente de gradient stochastique de régression logistique équilibré en batch équilibré(BBatch-LR-SGD), pour la classification d'ensembles de données avec un très grand nombre de classes,
2. un apprentissage en parallèle des classifieurs sur des machines multi-cœurs.

Les expérimentations menées sur des ensembles de données issus d'ImageNet montrent que notre algorithme est efficace comparé aux algorithmes de l'état de l'art.

Le reste de cet article est organisé de la manière suivante : la section deux présente brièvement l'algorithme de descente de gradient stochastique de régression logistique (LR-SGD) pour le problème de la classification à deux classes. La section 3 décrit comment étendre l'algorithme au cas de plusieurs classes et le paralléliser. La section 4 présente les résultats des expérimentations avant la conclusion et les travaux futurs.

2 Régression logistique pour la classification de deux classes

Soit un problème de classification à deux classes avec m points x_i ($i = 1..m$) en dimension n et les étiquettes de classe correspondantes $y_i = \pm 1$. La régression logistique (LR) essaye d'apprendre un modèle des données (i.e. un vecteur w de dimension n) qui maximise la vraisemblance. La probabilité d'appartenance d'un point à la classe positive est :

$$p(y_i = +1/x_i) = \frac{1}{1 + e^{-(w.x_i)}} \tag{1}$$

et la probabilité d'appartenance d'un point à la classe négative est :

$$p(y_i = -1/x_i) = 1 - p(y_i = +1/x_i) = 1 - \frac{1}{1 + e^{-(w.x_i)}} = \frac{1}{1 + e^{(w.x_i)}} \tag{2}$$

Les probabilités (1) et (2) peuvent se réécrire sous la forme suivante :

$$p(y_i/x_i) = \frac{1}{1+e^{-y_i(w.x_i)}} \tag{3}$$

Et donc la log-vraisemblance est :

$$log(p(y_i/x_i)) = log(\frac{1}{1+e^{-y_i(w.x_i)}}) = -log(1+e^{-y_i(w.x_i)}) \tag{4}$$

La régression logistique vise à maximiser la log-vraisemblance et la marge, cela conduit au problème :

$$\min\ \Psi(w,[x,y]) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} log(1+e^{-y_i(w.x_i)}) \tag{5}$$

La formule de régression logistique (5) utilise la fonction de perte logistique :

$$(w,[x_i,y_i]) = log(1+e^{-y_i(w.x_i)}). \tag{6}$$

La solution du problème convexe peut aussi être obtenue par une méthode de gradient stochastique (Shalev-Shwartz et al., 2007), (Bottou et Bousquet, 2008). La descente de gradient stochastique pour la régression logistique est notée LR-SGD. L'algorithme LR-SGD effectue le calcul de w en T itérations avec un taux d'apprentissage η. A chaque itération t, il n'utilise qu'un seul point (x_t, y_t) pour calculer le sous-gradient $\nabla_t \Psi(w,[x_t,y_t])$ et mettre à jour w_{t+1} comme suit :

$$w_{t+1} = w_t - \eta_t \nabla_t \Psi(w,[x_t,y_t]) = w_t - \eta_t(\lambda w_t + \nabla_t L(w,[x_t,y_t])) \tag{7}$$

$$\nabla_t L(w,[x_t,y_t]) = \nabla_t log(1+e^{-y_t(w.x_t)}) = -\frac{y_t x_t}{1+e^{y_t(w.x_t)}} \tag{8}$$

L'algorithme LR-SGD utilisant la mise à jour (7) est décrit dans l'algorithme 1.

3 Régression logistique pour un grand nombre de classes

Nous présentons plusieurs extensions du cas de régression logistique avec deux classes au cas multi-classes (k classes, avec $k \geq 3$). Les approches de l'état de l'art peuvent être classées en deux catégories. La première considère le cas multi-classes comme un problème d'optimisation (Ben-Akiva et Lerman, 1985), (Weston et Watkins, 1999), (Guermeur, 2007) et la seconde décompose le problème multi-classes en une série de problèmes à deux classes incluant les approches un contre le reste (Vapnik, 1995), un contre un (Kreßel, 1999) et graphe de décision acyclique orienté (Platt et al., 2000).

Dans la pratique, les approches un contre un et un contre le reste sont les plus utilisées car simples à mettre en œuvre. Soit un problème à k classes ($k > 2$). La stratégie un contre le reste

Algorithme 1 : Algorithme LR-SGD

input : training dataset D
positive constant $\lambda > 0$
number of epochs T
output : hyperplane w

```
1 begin
2     init w_1 = 0
3     for t ← 1 to T do
4         randomly pick a datapoint [x_t, y_t] from training dataset D
5         set η_t = 1/(λt)
6         update w_{t+1} = w_t − η_t(λw_t − y_t x_t / (1+e^{y_t (w·x_t)}))
7     end
8     return w_{t+1}
9 end
```

construit k classifieurs où le i^eclassifieur sépare la classe i du reste. La stratégie un contre un quant à elle construit $k(k-1)/2$ classifieurs, en utilisant toutes les paires possibles de classes. Dans les deux cas, la classe est ensuite prédite par un vote majoritaire.

Lorsque l'on traite des ensembles ayant un grand nombre de classes (e.g. plusieurs centaines) la stratégie un contre un peut devenir trop coûteuse à mettre en œuvre car il faut calculer plusieurs milliers de classifieurs. La stratégie un contre le reste est alors préférée dans ce cas. Donc notre algorithme multi-classes LR-SGD va aussi utiliser cette stratégie un contre le reste en effectuant k apprentissages. Par conséquence, l'algorithme LR-SGD avec apprentissage un contre le reste conduit aux problèmes suivants :

1. l'algorithme doit faire face à des classes très déséquilibrées pour construire ses classifieurs binaires,
2. l'algorithme est très coûteux en temps d'exécution pour effectuer l'apprentissage en mode séquentiel.

Nous proposons deux améliorations pour la création d'un nouvel algorithme LR-SGD capable de traiter un très nombre de classes dans un temps raisonnable. La première est la construction des classifieurs équilibrés par une méthode d'échantillonnage de la classe majoritaire et la seconde est la parallélisation du mécanisme d'apprentissage sur des machines multi-cœurs.

3.1 Batch équilibré de régression logistique

Dans l'approche un contre le reste, la tâche d'apprentissage de l'algorithme LR-SGD est d'essayer de séparer la i^e classe (positive) des $k-1$ autres classes (classe négative). Lorsque l'on a des grands nombres de classes (e.g. 100 classes), cela conduit à un très fort déséquilibre entre la classe positive et la classe négative. Le problème de l'algorithme LR-SGD vient de la **ligne 4** de l'algorithme 1. Pour traiter des centaines de classes, la probabilité d'avoir un point positif est très faible (par exemple 1%) par rapport à la très forte probabilité d'avoir

un point de la classe négative (par exemple 99%). L'algorithme LR-SGD se concentre alors essentiellement sur les erreurs de la classe négative et a ainsi des difficultés à séparer la classe positive de la classe négative.

Différentes solutions (aussi bien au niveau des données, qu'au niveau de l'algorithme) au déséquilibre des classes ont été présentées (Japkowicz, 2000), (Weiss et Provost, 2003), (Visa et Ralescu, 2005) ou encore (Lenca et al., 2008), (Pham et al., 2008). Au niveau des données, ces algorithmes modifient la distribution des classes, soit par sur-échantillonnage de la classe minoritaire (Chawla et al., 2003) ou sous-échantillonnage de la classe majoritaire (Liu et al., 2009), (Ricamato et al., 2008). Au niveau de l'algorithme, une solution est de modifier l'équilibre des classes en jouant sur les coûts de mauvaise classification de l'une ou des deux classes.

Notre algorithme balancé de batch LR-SGD (noté LR-BBatch-SGD) appartient à la première catégorie. Pour séparer la ie classe (positive) du reste (classe négative), les probabilités sont fortement déséquilibrées (par exemple la probabilité d'appartenance à la classe positive n'est que de 1% dans les cas des 100 classes d'ImageNet). Par ailleurs le coût de sur-échantillonnage étant relativement élevé, nous proposons l'algorithme LR-BBatch-SGD utilisant un sous-échantillonnage de la classe majoritaire. Nous avons modifié l'algorithme 1 pour de batch équilibré (au lieu d'un unique point, **ligne 4**) pour effectuer la mise à jour de w à chaque itération t. Nous avons aussi modifié la mise à jour de la manière suivante :

$$w_{t+1} = \begin{cases} w_t - \eta_t \lambda w_t + \eta_t \frac{1}{|D_+|} \frac{y_t x_t}{1+e^{y_t (w.x_t)}} & if\ y_t = +1 \\ w_t - \eta_t \lambda w_t + \eta_t \frac{1}{|D_-|} \frac{y_t x_t}{1+e^{y_t (w.x_t)}} & if\ y_t = -1 \end{cases} \tag{9}$$

avec $|D_+|$ le cardinal de la classe positive et $|D_-|$ le cardinal de la classe négative.

L'algorithme LR-BBatch-SGD (Algorithme 2) sépare la i^e classe (positive) du reste (classe négative) en utilisant un sous-échantillonnage de la classe négative (batch équilibré) et la modification du coût de (9).

3.2 Parallélisation de l'apprentissage LR-BBatch-SGD

Bien que l'algorithme LR-BBatch-SGD puisse traiter de grands ensembles de données dans un temps raisonnable, il ne tire pas avantage des possibilités des architectures multi-cœurs des machines actuelles. De plus l'algorithme LR-BBatch-SGD effectue k apprentissages indépendants pour la classification de k classes, ce qui est propriété intéressante pour la parallélisation. Pour améliorer le temps d'exécution de l'algorithme LR-BBatch-SGD, nous allons effectuer l'apprentissage des k classifieurs binaires en parallèle sur plusieurs machines multi-cœurs.

La programmation parallèle est principalement basée sur deux modèles, MPI (Message Passing Interface (MPI-Forum.)) et OpenMP (Open Multiprocessing (OpenMP Architecture Review Board, 2008)). MPI est un système standardisé basé sur un mécanisme d'envois de messages pour les systèmes à mémoire distribué. C'est actuellement le système le plus utilisé pour la parallélisation. Cependant son défaut est de nécessiter tout l'ensemble de données en mémoire pour effectuer l'apprentissage. Cela signifie que si l'on utilise MPI pour la classification de k classes en parallèle, on aura besoin de k fois la quantité de mémoire pour charger les k ensembles de données, ce qui le rend inutilisable dans ce contexte. Nous allons utiliser

Algorithme 2 : Algorithme LR-BBatch-SGD

input : training data of the positive class D_+
training data of the negative class D_-
positive constant $\lambda > 0$
number of epochs T
output : hyperplane w

begin
  init $w_1 = 0$
  for $t \leftarrow 1$ **to** T **do**
    creating a balanced batch $Bbatch$ by sampling without replacement D'_- from dataset D_- (with $|D'_-|$ = $sqrt\frac{|D_-|}{|D_+|}$) and a datapoint from dataset D_+
    set $\eta_t = \frac{1}{\lambda t}$
    for $[x_i, y_i]$ *in* $Bbatch$ **do**
      update w_{t+1} using rule 9
    end
  end
  return w_{t+1}
end

OpenMP qui lui n'a pas besoin d'autant de mémoire même si la parallélisation est moins optimisée que celle de MPI. L'apprentissage en parallèle de l'algorithme LR-BBatch-SGD est décrit dans l'algorithme 3.

Algorithme 3 : Apprentissage parallèle de LR-BBatch-SGD

input : D the training dataset with k classes
output : LR-SGD model

Learning :
#pragma omp parallel for
for $c_i \leftarrow 1$ **to** k **do** `/* class` c_i `*/`
  training LR-BBatch-SGD($c_i - vs - all$)
end

4 Evaluation

Pour évaluer les performances de notre nouvel algorithme parallèle de régression logistique multi-classes (PAR-MC-LR) pour la classification de grands ensembles d'images en un grand nombre de classes, nous avons mis en ?uvre l'algorithme PAR-MC-LR en C/C++ en utilisant la bibliothèque SGD Library (Bottou et Bousquet, 2008). Nos comparaisons porteront sur la précision et le temps d'apprentissage. Nous avons choisi deux algorithmes récents de classification : LIBLINEAR (A library for large linear classification (Fan et al., 2008))) et OCAS (An

optimized cutting plane algorithm for SVM (Franc et Sonnenburg, 2009)) car ils sont réputés pour être efficaces pour la classification de SVM linéaires.

LIBLINEAR et OCAS sont utilisés avec leurs paramètres par défaut ($C = 1000$), l'algorithme Par-MC-LR effectue l'apprentissage de batch équilibré de descente de gradient stochastique de régression logistique en utilisant 50 itérations et le terme de régularisation $\lambda = 0.0001$. Toutes les expériences sont effectuées sous Linux Fedora 20, Intel(R) Core i7-4790 CPU, 3.6GHz, 4 cœurs et 32 Go de mémoire vive.

4.1 Ensembles de données

L'algorithme PAR-MC-LR a été conçu pour la classification de grands ensembles d'images avec un grand nombre de classes, nous allons l'évaluer sur les trois ensembles de données suivants :

ImageNet10

Cet ensemble de données contient les 10 plus grandes classes de l'ensemble ImageNet, soit 24 807 images et 2,4Go. Pour chaque classe 90% des images servent à l'apprentissage et 10% pour le test. Nous commençons par construire le sac de mots visuels en utilisant pour chaque image les descripteurs DSIFTs (Dense SIFTs) et un vocabulaire de 5000 mots. Ensuite on utilise les featuresmaps de (Vedaldi et Zisserman, 2012) une représentation en haute dimension de l'image (de dimension 15000). Cette méthode a donné de bons résultats pour la classification d'images à l'aide de classifieurs linéaires. Nous obtenons finalement un ensemble de données d'apprentissage de 2,6 Go.

ImageNet100

Cet ensemble de données contient les 100 plus grandes classes de l'ensemble ImageNet, soit 183116 images et 23,6Go. Dans chaque classe nous prenons 50% des images pour l'apprentissage et les 50% restants pour le test. De la même manière nous construisons un sac de mots visuels en utilisant des DSIFTs et un vocabulaire de 5000 mots visuels. Puis nous utilisons les featuresmaps et nous obtenons finalement en entrée de l'algorithme d'apprentissage un ensemble de 8 Go.

ILSVRC2010

Cet ensemble de données contient les 1000 plus grandes classes de l'ensemble ImageNet, soit 1,2 million d'images et 126 Go pour l'apprentissage, 50000 images pour la validation (5,3 Go) et 150000 (16 Go) images pour le test. Nous utilisons le sac de mots visuels fournit par (Deng et al., 2010) et la méthode décrite dans (Wu, 2012) pour encoder chaque image dans un vecteur de dimensions 21000. Nous retenons 900 images par classe pour l'ensemble d'apprentissage donc le nombre total d'images pour l'apprentissage est de 887816 et 12,5 Go. Toutes les données de l'ensemble de test sont utilisées pour tester le modèle de SVM obtenu.

4.2 Résultats de la classification

Tout d'abord on s'intéresse aux performances en ce qui concerne le temps d'exécution. Nous avons fait varier le nombre de threads OpenMP (1, 4, 8 threads) pour chaque exécution de notre algorithme PAR-MC-LR. En raison de l'architecture utilisée (Intel(R) Core i7-4790

TAB. 1 – *Temps d'apprentissage (mn) pour ImageNet10*

Algorithme	**1 thread**	**4 threads**	**8 threads**
OCAS	106.67		
LIBLINEAR	3.12	1.50	1.48
PAR-MC-LR	2.24	0.80	0.76

TAB. 2 – *Temps d'apprentissage (mn) pour ImageNet100*

Algorithme	**1 thread**	**4 threads**	**8 threads**
OCAS	1016.35		
LIBLINEAR	63.42	30.49	30.18
Par-MC-LR	23.40	8.09	6.55

CPU, 3.6GHz, 4 cœurs) lors de nos expérimentations, l'algorithme PAR-MC-LR est le plus rapide avec 8 threads, mais pas tellement plus qu'avec 4 threads.

Pour le petit ensemble d'images multi-classes ImageNet10, le temps d'apprentissage des algorithmes dans le tableau 1 montre que notre algorithme PAR-MC-LR avec 8 threads est 140 fois plus rapide que OCAS et 2 fois plus rapide que LIBLINEAR.

Les tableau 2 et figure 1 présentent les temps d'apprentissage pour l'ensemble de données ImageNet100 avec un plus grand nombre de classes. Ici encore, notre algorithme PAR-MC-LR procure un gain significatif pour le temps d'exécution en utilisant 8 threads. Il est 155 fois plus rapide que OCAS et 4,6 fois plus rapide que LIBLINEAR.

L'ensemble de données ILSVRC-2010 possède un très grand nombre d'images (plus d'un million) et un grand nombre de classes (1000). OCAS n'a pas pu achever le calcul de l'apprentissage après plusieurs semaines CPU. LIBLINEAR a eu besoin de 3106 minutes (soit 2 jours et 4 heures) pour l'apprentissage du modèle de classification pour cet ensemble de données (16h45 avec 8 threads). Notre algorithme PAR-MC-LR avec 8 threads n'a, quant à lui, eu besoin que de 38 minutes pour effectuer cette tâche d'apprentissage. Ceci montre que notre algorithme est 26 fois plus rapide que LIBLINEAR.

En ce qui concerne la précision de la classification, les résultats sont présentés dans les tableau 4 et figure 3. Sur le petit ensemble de données ImageNet10 et le moyen ImageNet100, l'algorithme PAR-MC-LR obtient un meilleur taux de bonne précision que OCAS. En ce qui concerne la comparaison avec LIBLINEAR, les performances sont quasiment identiques. Notre algorithme a une meilleure précision sur les ensembles ImageNet10 et ILSVRC2010 et

TAB. 3 – *Temps d'apprentissage (mn) pour ILSVRC-2010*

Algorithme	**1 thread**	**4 threads**	**8 threads**
OCAS	N/A		
LIBLINEAR	3106.48	1037	1004
Par-MC-LR	153.58	40.59	37.91

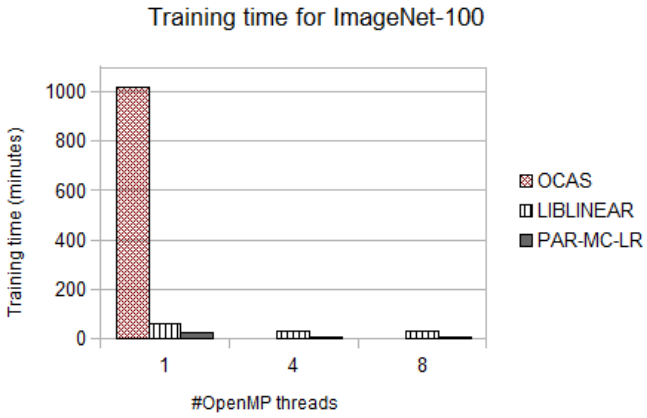

FIG. 1 – *Temps d'apprentissage (mn) pour ImageNet100*

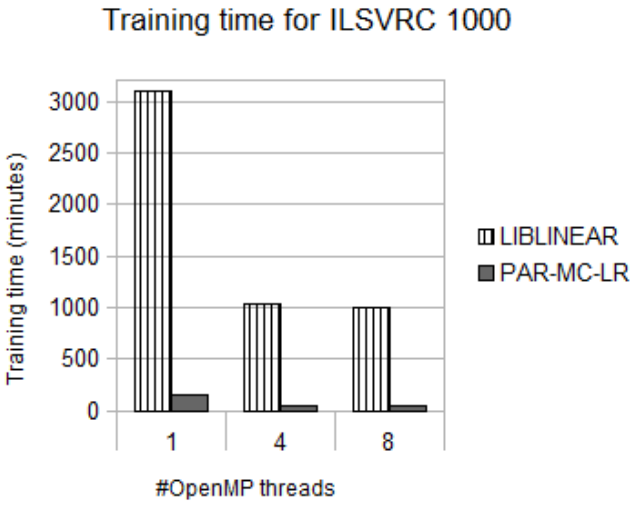

FIG. 2 – *Temps d'apprentissage (mn) pour ILSVRC-2010*

TAB. 4 – *Précision de la classification (%)*

Algorithme	**ImageNet 10**	**ImageNet 100**	**ILSVRC 1000**
OCAS	72.07	52.75	N/A
LIBLINEAR	75.09	54.07	21.11
Par-MC-LR	75.21	52.91	21.90

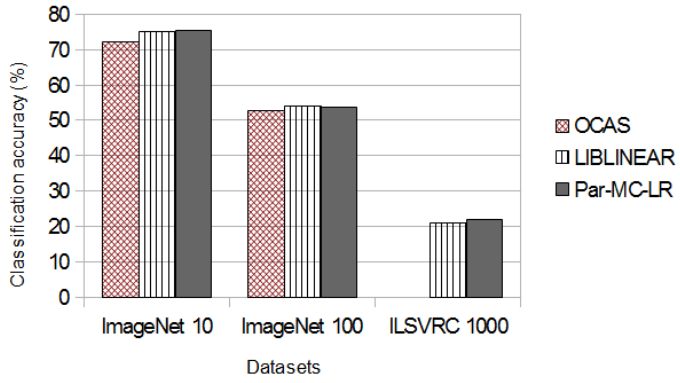

FIG. 3 – *Précision de la classification (%)*

une précision moindre sur ImageNet100.

ILSVRC2010 est un grand ensemble de données avec plus d'un million d'images et 1000 classes. C'est donc un challenge d'obtenir des bons taux de classification pour les algorithmes de l'état de l'art. En particulier avec l'ensemble de données fournit lors de la compétition ILSVRC2010, les meilleurs algorithmes (Deng et al., 2010), (Berg et al., 2010) ont obtenus des taux de précision de l'ordre de 19%. Notre algorithme PAR-MC-LR obtient une meilleure précision (21,9% contre 19%), ce qui représente une amélioration relative de plus de 15%. De plus comparé à LIBLINEAR, on obtient une amélioration relative de plus de 3,7%. On peut remarquer que l'algorithme PAR-MC-LR est beaucoup plus rapide que LIBLINEAR tout en ayant une précision équivalente. Enfin, il est aussi gourmand en mémoire puiqu'il ne nécéssite que 9,4 Go de mémoire contre 16,9 pour LIBLINEAR et 52,9 pour OCAS.

Ces résultats montrent que notre algorithme a de bonnes qualités pour traiter l'intégralité de l'ensemble de données ImageNet. Plus la taille de l'ensemble de données augmente et plus le rapport entre les temps d'apprentissage entre notre algorithme et LIBLINEAR nous sont favorables : on passe d'un facteur de 2 fois plus rapide sur ImageNet10 à 26 fois plus rapide sur ILSVRC2010.

5 Conclusion et travaux futurs

Nous avons présenté un nouvel algorithme parallèle de régression logistique multi-classes qui permet la classification efficace de grands ensembles de données d'images avec un grand nombre de classes. Pour cela nous avons utilisé un algorithme de batch équilibré de descente de gradient stochastique de régression logistique pour les apprentissages de deux classes dans le cadre de la classification multi-classes. Ce nouvel algorithme permet notamment la classification de grands ensembles de données d'images sur des architectures multi-cœurs. Les performances de l'algorithme ont été évaluées sur les 10 et 100 plus grandes classes de l'ensemble de données ImageNet et sur l'ensemble de données ILSVRC2010. Notre algorithme présente des gains significatifs en ce qui concerne les temps d'exécution (d'un facteur 2 sur

l'ensemble de données le plus petit à 26 sur le plus grand) tout en ayant une précision similaire. Ces résultats montrent que plus l'ensemble de données à traiter est grand et plus le gain par rapport aux autres algorithmes est important et ce gain serait sans doute encore plus important si nous avions utilisé une machine disposant de 8 cœurs au lieu de 4, on note en effet qu'il n'y a pratiquement pas de différences entre 4 et 8 threads.

Les extensions de ces travaux prévoient une version hybride MPI/OpenMP de l'algorithme parallèle de régression logistique pour classifier efficacement de grands ensembles de données multi-classes et permettre la classification de la totalité de l'ensemble de données ImageNetsur un ensemble de machines multi-cœurs.

Références

Bay, H., A. Ess, T. Tuytelaars, et L. J. V. Gool (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding 110*(3), 346–359.

Ben-Akiva, M. et S. Lerman (1985). *Discrete Choice Analysis : Theory and Application to Travel Demand*. The MIT Press.

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

Bosch, A., A. Zisserman, et X. Muñoz (2007). Image classification using random forests and ferns. In *International Conference on Computer Vision*, pp. 1–8.

Bottou, L. et O. Bousquet (2008). The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, et S. Roweis (Eds.), *Advances in Neural Information Processing Systems*, Volume 20, pp. 161–168. NIPS Foundation (http ://books.nips.cc).

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

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

Fan, R., K. Chang, C. Hsieh, X. Wang, et C. Lin (2008). LIBLINEAR : a library for large linear classification. *Journal of Machine Learning Research 9*(4), 1871–1874.

Franc, V. et S. Sonnenburg (2009). Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research 10*, 2157–2192.

Guermeur, Y. (2007). Svm multiclasses, théorie et applications.

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

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

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

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

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*(2), 91–110.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press 1*, 281–297.

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

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

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

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

Ricamato, M. T., C. Marrocco, et F. Tortorella (2008). Mcs-based balancing techniques for skewed classes : An empirical comparison. In *ICPR*, pp. 1–4.

Shalev-Shwartz, S., Y. Singer, et N. Srebro (2007). Pegasos : Primal estimated sub-gradient solver for svm. In *Proceedings of the Twenty-Fourth International Conference Machine Learning*, pp. 807–814. ACM.

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

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

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

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

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

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

Summary

We present a new parallel multiclass logistic regression algorithm (PAR-MCLR) aiming at classifying a very large number of images with very-high-dimensional signatures into many classes. We extend the two-class logistic regression algorithm (LR) in several ways to develop the new multiclass LR for efficiently classifying large image datasets into hundreds of classes. We propose the balanced batch stochastic gradient descend of logistic regression (BBatch-LR-SGD) for training two-class classifiers used in the one-versus-all strategy of the multiclass problems and the parallel training process of classifiers with several multi-core computers. The numerical test results on ImageNet datasets show that our algorithm is efficient compared to the state-of-the-art linear classifiers.