Parallel Multiclass Logistic Regression for Classifying Large Scale Image Datasets

Thanh-Nghi Do[1] and François Poulet[2]

[1] College of Information Technology
Can Tho University, 92100-Cantho, Vietnam
dtnghi@cit.ctu.edu.vn
[2] University of Rennes I - IRISA
Campus de Beaulieu, 35042 Rennes Cedex, France
francois.poulet@irisa.fr

Abstract. The new parallel multiclass logistic regression algorithm (PAR-MC-LR) aims 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 trainning 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.

Keywords: Logistic regression (LR), Stochastic gradient descent (SGD), Multiclass, Parallel algorithm, Large scale image classification.

1 Introduction

The image classification is to automatically assign predefined categories to images. Its applications include handwriting character recognition, zip code recognition for postal mail sorting, numeric entries in forms filled up by hand, fingerprint recognition, face recognition, auto-tagging images and so on. The image classification task involves two main steps as follows: extracting features and building codebook, training classifiers. Recently, the local image features and bag-of-words (BoW) models are used at the first step of state-of-the-art image classification. The most popular local image features are scalable invariant feature transform descriptors - SIFT [1], speeded up robust Features - SURF [2] and dense SIFT - DSIFT [3]. These feature extraction methods are locally based on the appearance of the object at particular interest points, invariant to image scale, rotation and also robust to changes in illumination, noise, occlusion. And then, k-means algorithm [4] performs the clustering task on descriptors to form visual words from the local descriptors. The representation of the image for classification is the bag-of-words is constructed from the counting of the occurrence of words in a histogram like fashion. The step of the feature extraction and the BoW representation leads to datasets with very large number of dimensions (e.g. thousands of dimensions). The support vector machine algorithms [5] are suited for dealing with very-high-dimensional datasets.

H.A. Le Thi et al. (eds.), *Advanced Computational Methods for Knowledge Engineering*,
Advances in Intelligent Systems and Computing 358, DOI: 10.1007/978-3-319-17996-4_23

However there are also many classes in the images classification (e.g. Caltech with 101 classes [6], Caltech with 256 classes [7] having hundreds of classes) and ImageNet dataset [8] with more than 14 million images in 21,841 classes. It makes the complexity of image classification become very hard. This challenge motivates us to study an efficient algorithm in both computation time and classification accuracy. In this paper, we propose the extensions of the stochastic gradient descend (SGD [9], [10]) to develop the new parallel multiclass SGD of logistic regression (PAR-MC-LR) for efficiently classifying large image datasets into many classes. Our contributions include:

1. the balanced batch stochastic gradient descend of logistic regression (BBatch-LR-SGD) for very large number of classes,
2. the parallel training process of classifiers with several multi-core computers.

The numerical test results on ImageNet datasets [8] show that our PAR-MC-LR algorithm is efficient compared to the state-of-the-art linear classifiers.

The remainder of this paper is organized as follows. Section 2 briefly presents the stochastic gradient descend algorithm of logistic regression (LR-SGD) for two-class problems. Section 3 describes how to extend the LR-SGD algorithm for dealing very large number of classes. Section 4 presents evaluation results, before the conclusions and future work.

2 Logistic Regression for Two-Class Problems

Let us consider a linear binary classification task with m datapoints x_i ($i = 1, \ldots, m$) in the n-dimensional input space R^n, having corresponding labels $y_i = \pm 1$. Logistic regression (LR) tries to learn classification models (i.e. parameter vector $w \in R^n$) to maximize the likelihood of the data. The probability of a datapoint being drawn from the positive class is:

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

And then, the probability of a datapoint being drawn from the negative class is:

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

The probabilities in (1) and (2) are rewritten in:

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

And then the log likelihood of data is as follow:

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

The regularized LR method aims to use Tikhonov regularization in a trade-off with maximizing likelihood and the over-confident generalization. The regularization strategy is to impose a penalty on the magnitude of the parameter values w. The LR algorithm simultaneously tries to maximize the log likelihood of the data and minimize the L_2 norm of the parameter vector w. And then, it yields an unconstrained problem (5):

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

The LR formula (5) uses the logistic loss function $L(w,[x_i,y_i]) = log(1+e^{-y_i(w.x_i)})$. The solution of the unconstrained problem (5) can be also obtained by the stochastic gradient descent method [9], [10]. The stochastic gradient descent for the logistic regression is denoted by (LR-SGD). The LR-SGD updates w on T iterations (epochs) with a learning rate η. For each iteration t, the LR-SGD uses a single randomly datapoint (x_t,y_t) to compute the sub-gradient $\nabla_t \Psi(w,[x_t,y_t])$ and update w_{t+1} as follows:

$$w_{t+1} = w_t - \eta_t \nabla_t \Psi(w,[x_t,y_t]) = w_t - \eta_t(\lambda w_t + \nabla_t L(w,[x_t,y_t])) \tag{6}$$

$$\nabla_t L(w,[x_t,y_t]) = \nabla_t log(1+e^{-y_t(w.x_t)}) = -\frac{y_t x_t}{1+e^{y_t(w.x_t)}} \tag{7}$$

The LR-SGD using the update rule (6) is described in algorithm 1.

Algorithm 1. LR-SGD algorithm

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

begin
  init $w_1 = 0$
  for $t \leftarrow 1$ **to** T **do**
    randomly pick a datapoint $[x_t,y_t]$ from training dataset D
    set $\eta_t = \frac{1}{\lambda t}$
    update $w_{t+1} = w_t - \eta_t(\lambda w_t - \frac{y_t x_t}{1+e^{y_t(w.x_t)}})$
  end
  return w_{t+1}
end

3 Extentions of Logistic Regression to Large Number of Classes

There are several extensions of a binary classification LR to multi-class (k classes, $k \geq 3$) classification tasks. The state-of-the-art multi-class are categorized into two types of approaches. The first one is considering the multi-class case in one optimization problem [11], [12], [13]. The second one is decomposing multi-class into a series of binary classifiers, including one-versus-all [5], one-versus-one [14] and Decision Directed Acyclic Graph [15].

In practice, one-versus-all, one-versus-one are the most popular methods due to their simplicity. Let us consider k classes ($k > 2$). The one-versus-all strategy builds k different classifiers where the i^{th} classifier separates the i^{th} class from the rest. The one-versus-one strategy constructs $k(k-1)/2$ classifiers, using all the binary pairwise combinations of the k classes. The class is then predicted with a majority vote.

When dealing with very large number of classes, e.g. hundreds of classes, the one-versus-one strategy is too expensive because it needs to train many thousands of binary classifiers. Therefore, the one-versus-all strategy becomes popular in this case. And then, our multiclass LR-SGD algorithms also use the one-versus-all approach to train independently k binary classifiers. Therefore, the multiclass LR-SGD algorithms using one-versus-all lead to the two problems:

1. the LR-SGD algorithms deal with the imbalanced datasets for building binary classifiers,
2. the LR-SGD algorithms also take very long time to train very large number of binary classifiers in sequential mode using a single processor.

Due to these problems, we propose two ways for creating the new multiclass LR-SGD algorithms being able to handle very large number of classes in the high speed. The first one is to build balanced classifiers with sampling strategy. The second one is to parallelize the training task of all classifiers with several multi-core machines.

3.1 Balanced Batch of Logistic Regression

In the one-versus-all approach, the learning task of LR-SGD is try to separate the i^{th} class (positive class) from the $k-1$ others classes (negative class). For very large number of classes, e.g. 100 classes, this leads to the extreme unbalance between the positive and the negative class. The problem is well-known as the class imbalance. The problem of LR-SGD comes from **line 4** of algorithm 1. For dealing with hundreds classses, the probability for a positive datapoint sampled is very small (about 0.01) compared with the large chance for a negative datapoint sampled (e.g. 0.99). And then, the LR-SGD concentrates mostly on the errors produced by the nagative datapoints. Therefore, the LR-SGD has difficulty to separate the positive class from the negative class.

As summarized by the review papers of [16], [17], [18] and the very comprehensive papers of [19], [20], solutions to the class imbalance problems were proposed both at the data and algorithmic level. At the data level, these algorithms change the class distribution, including over-sampling the minority class [21] or under-sampling the majority

class [22], [23]. At the algorithmic level, the solution is to re-balance the error rate by weighting each type of error with the corresponding cost.

Our balanced batch LR-SGD (denoted by LR-BBatch-SGD) belongs to the first approach. For separating the i^{th} class (positive class) from the rest (negative class), the class prior probabilities in this context are highly unequal (e.g. the distribution of the positive class is 1% in the 100 classes classification problem), and then over-sampling the minority class is very expensive. We propose the LR-BBatch-SGD algorithm using under-sampling the majority class (negative class). Our modification of algorithm 1 is to use a balanced batch (instead of a datapoint at line 4 of algorithm 1) to update the w at epoch t. We also propose to modify the updating rule (**line 6** of algorithm 1) using the skewed costs as follows:

$$w_{t+1} = \begin{cases} w_t - \eta_t \lambda w_t + \eta_t \frac{1}{|D_+|} \frac{y_t x_t}{1+e^{y_t(w.x_t)}} & if\ y_t = +1 \\ w_t - \eta_t \lambda w_t + \eta_t \frac{1}{|D_-|} \frac{y_t x_t}{1+e^{y_t(w.x_t)}} & if\ y_t = -1 \end{cases} \tag{8}$$

where $|D_+|$ is the cardinality of the positive class D_+ and $|D_-|$ is the cardinality of the negative class D_-.

The LR-BBatch-SGD (as shown in algorithm 2) is to separate the i^{th} class (positive class) from the rest (negative class), using under-sampling the negative class (balanced batch) and the skewed costs in (8).

Algorithm 2. LR-BBatch-SGD algorithm

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

1 **begin**
2 init $w_1 = 0$
3 **for** $t \leftarrow 1$ **to** T **do**
4 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_+
5 set $\eta_t = \frac{1}{\lambda t}$
6 **for** $[x_i, y_i]$ *in Bbatch* **do**
7 update w_{t+1} using rule 8
8 **end**
9 **end**
10 return w_{t+1}
11 **end**

3.2 Parallel LR-BBatch-SGD Training

Although LR-BBatch-SGD deals with very large dataset with high speed, but it does not take the benefits of HPC. Furthermore, LR-BBatch-SGD trains independently k binary classifiers for k classes problems. This is a nice property for parallel learning. Our investigation aims at speedup training tasks of multi-class LR-BBatch-SGD with several multi-processor computers. The idea is to learn k binary classifiers in parallel.

The parallel programming is currently based on two major models, Message Passing Interface (MPI) [24] and Open Multiprocessing (OpenMP) [25]. MPI is a standardized and portable message-passing mechanism for distributed memory systems. MPI remains the dominant model (high performance, scalability, and portability) used in high-performance computing today. However, one MPI process loads the whole dataset into memory during learning tasks, thus the parallel algorithm with k MPI processes requires k main memory rooms for storing k datasets, making it wasteful. The simplest development of parallel LR-BBatch-SGD algorithm is based on the shared memory multiprocessing programming model OpenMP that does not require large amount of memory (although the parallel in MPI is more efficient than OpenMP). The parallel learning for LR-BBatch-SGD is described in algorithm 3.

Algorithm 3. Parallel LR-BBatch-SGD training

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

```
1 Learning:
2 #pragma omp parallel for
3 for c_i ← 1 to k do                                   /* class c_i */
4 |  training LR-BBatch-SGD(c_i − vs − all)
5 end
```

4 Evaluation

In order to evaluate the performance of the new parallel multiclass logistic regression algorithm (PAR-MC-LR) for classifying large amounts of images into many classes, we have implemented PAR-MC-LR in C/C++ using the SGD library [10]. Our comparison is reported in terms of correctness and training time. We are interested in two recent algorithms, LIBLINEAR (a library for large linear classification [26]) and OCAS (an optimized cutting plane algorithm for SVM [27]) because they are well-known as highly efficient standard linear SVM.

LIBLINEAR and OCAS use the default parameter value $C = 1000$, our Par-MC-LR is trained the balanced batch stochastic gradient descend of logistic regression using $T = 50$ epochs and regularization term $\lambda = 0.0001$.

All experiments are run on machine Linux Fedora 20, Intel(R) Core i7-4790 CPU, 3.6 GHz, 4 cores and 32 GB main memory.

4.1 Datasets

The PAR-MC-LR algorithm is designed for the large number of images with many classes, so we have evaluated its performance on the three following datasets.

ImageNet 10
This dataset contains the 10 largest classes from ImageNet [8], including 24,807 images with size 2.4 GB). In each class, we sample 90% images for training and 10% images for testing (with random guess 10%). First, we construct BoW of every image using dense SIFT descriptor (extracting SIFT on a dense grid of locations at a fixed scale and orientation) and 5,000 codewords. Then, we use feature mapping from [28] to get the high-dimensional image representation in 15,000 dimensions. This feature mapping has been proven to give a good image classification performance with linear classifiers [28]. We end up with 2.6 GB of training data.

ImageNet 100
This dataset contains the 100 largest classes from ImageNet [8], including 183,116 images with size 23.6 GB. In each class, we sample 50% images for training and 50% images for testing (with random guess 1%). We also construct BoW of every image using dense SIFT descriptor and 5,000 codewords. For feature mapping, we use the same method as we do with ImageNet 10. The final size of training data is 8 GB.

ILSVRC 2010
This dataset contains 1,000 classes from ImageNet [8], including 1.2M images ($\sim$ 126 GB) for training, 50 K images ($\sim$ 5.3 GB) for validation and 150 K images ($\sim$ 16 GB) for testing. We use BoW feature set provided by [8] and the method reported in [29] to encode every image as a vector in 21,000 dimensions. We take roughly 900 images per class for training dataset, so the total training images is 887,816 and the training data size is about 12.5 GB. All testing samples are used to test SVM models. Note that the random guess performance of this dataset is 0.1%.

4.2 Classificaton Results

Firstly, we are interested in the performance comparison in terms of training time. We try to vary the number of OpenMP threads (1, 4, 8 threads) for all training tasks of our parallel algorithm PAR-MC-LR. Due to the PC (Intel(R) Core i7-4790 CPU, 4 cores) used in the experimental setup, the PAR-MC-LR is fastest by setting 8 OpenMP threads.

Table 1. Training time (minutes) on ImageNet 10

	# OpenMP threads		
Algorithm	1	4	8
OCAS	106.67		
LIBLINEAR	2.02		
PAR-MC-LR	2.24	0.80	0.76

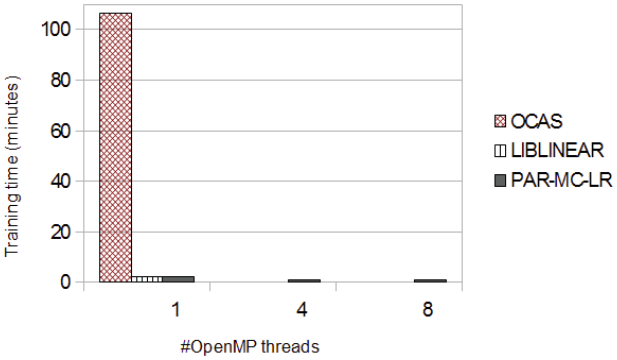

Fig. 1. Training time (minutes) on ImageNet-10

Table 2. Training time (minutes) on ImageNet 100

	# OpenMP threads		
Algorithm	1	4	8
OCAS	1016.35		
LIBLINEAR	30.41		
Par-MC-LR	23.40	8.09	6.55

For the small multi-class dataset as ImageNet 10, the training time of algorithms in table 1 and figure 1 show that our PAR-MC-LR with 8 OpenMP threads is 139.7 times faster than OCAS and 2.64 times faster than LIBLINEAR.

Table 2 and figure 2 present the training time on ImageNet 100 with large number of classes. Once again, the PAR-MC-LR achieves a significant speed-up in learning process using 8 OpenMP threads. It is 155.1 times faster than OCAS and 4.64 times faster than LIBLINEAR.

Table 3. Training time (minutes) on ILSVRC 2010.

	# OpenMP threads		
Algorithm	1	4	8
OCAS	N/A		
LIBLINEAR	3106.48		
Par-MC-LR	153.58	40.59	37.91

ILSVRC 2010 has large amount of images (more than 1 million images) and very large number of classes (1000 classes). Therefore, OCAS has not finished the learning task in several days. LIBLINEAR takes 3,106.48 minutes (about 2 days and 4 hours)

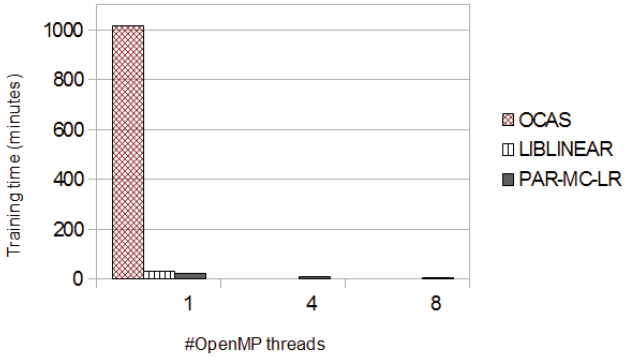

Fig. 2. Training time (minutes) on ImageNet-100

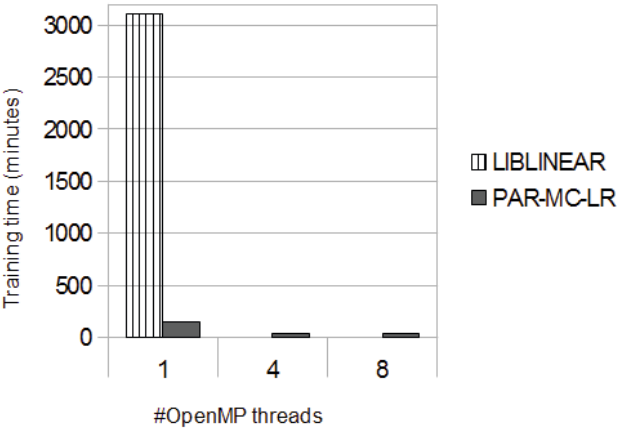

Fig. 3. Training time (minutes) on ImageNet-1000

to train the classification model for this dataset. Our PAR-MC-LR algorithm with 8 OpenMP threads performs the learning task in 37.59 minutes. This indicates that the PAR-MC-LR is 81.95 times faster than LIBLINEAR.

The classification results in terms of accuracy are presented in table 4 and figure 4. On the small datasets ImageNet 10 and medium dataset ImageNet 100, The Par-MC-LR outperforms OCAS in the classification accuracy. The PAR-MC-LR achieves very competitive performances compared to LIBLINEAR. It is more accurate than LIBLINEAR on ImageNet 10 while making more classification mistakes than LIBLINEAR on ImageNet 100.

ILSVRC 2010 is a large dataset (with more than 1 million images and 1,000 classes). Thus, it is very difficult for many state-of-the-art algorithms to obtain a high rate in classification performance. In particular, with the feature set provided by ILSVRC 2010

Table 4. Overall classification accuracy (%)

Dataset	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

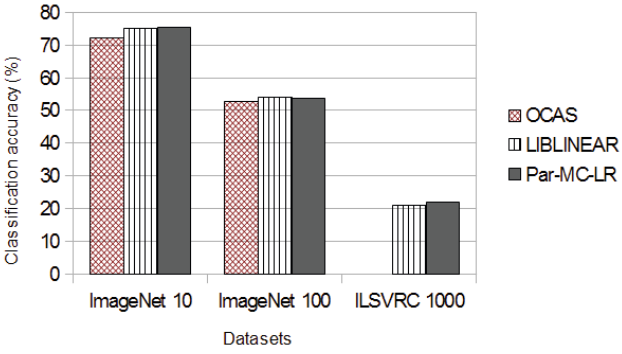

Fig. 4. Overall classification accuracy (%)

competition the state-of-the-art system [8,30] reports an accuracy of approximately 19 % (it is far above random guess, 0.1 %). And now our PAR-MC-LR algorithm gives a higher accuracy rate than [8,30] with the same feature set (21.90 % vs. 19 %). The relative improvement is more than 15 %. Moreover, the PAR-MC-LR outperforms LIBLINEAR (+0.79 %, the relative improvement is more than 3.7 %). Note that the PAR-MC-LR learns much faster than LIBLINEAR while yielding a higher correctness rate. These results show that our PAR-MC-LR has a great ability to scale-up to full ImageNet dataset.

5 Conclusion and Future Works

We have presented the new parallel multiclass logistic regression algorithm that achieves high performances for classifying large amounts of images intomany classes. The balanced batch stochastic gradient descend of logistic regression is proposed for trainning two-class classifiers used in the multiclass problems. The parallel multiclass algorithm is also developped for efficiently classifying large image datasets into very large number of classes on multi-core computers. Our algorithm is evaluated on the 10, 100 largest classes of ImageNet and ILSVRC 2010 dataset. By setting the number of OpenMP threads to 8 on PC (Intel Core i7-4790 CPU, 3.6 GHz), our algorithm achieves a significant speedup in training time without (or very few) compromise the classification accuracy. It is a roadmap towards very large scale visual classification.

In the future, we will develop a hybrid MPI/OpenMP parallel logistic regression for efficiently dealing with large scale multiclass problems and also intend to provide more empirical test on full dataset with 21,000 classes of ImageNet.

References

1. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
2. Bay, H., Ess, A., Tuytelaars, T., Gool, L.J.V.: Speeded-up robust features (SURF). Computer Vision and Image Understanding 110(3), 346–359 (2008)
3. Bosch, A., Zisserman, A., Muñoz, X.: Image classification using random forests and ferns. In: International Conference on Computer Vision, pp. 1–8 (2007)
4. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
5. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
6. Li, F.F., Fergus, R., Perona, P.: 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 (2007)
7. Griffin, G., Holub, A., Perona, P.: Caltech-256 Object Category Dataset. Technical Report CNS-TR-2007-001, California Institute of Technology (2007)
8. Deng, J., Berg, A.C., Li, K., Fei-Fei, L.: What does classifying more than 10,000 image categories tell us? In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 71–84. Springer, Heidelberg (2010)
9. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for svm. In: Proceedings of the Twenty-Fourth International Conference Machine Learning, pp. 807–814. ACM (2007)
10. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems, vol. 20, pp. 161–168. NIPS Foundation (2008), `http://books.nips.cc`
11. Ben-Akiva, M., Lerman, S.: Discrete Choice Analysis: Theory and Application to Travel Demand. The MIT Press (1985)
12. Weston, J., Watkins, C.: Support vector machines for multi-class pattern recognition. In: Proceedings of the Seventh European Symposium on Artificial Neural Networks, pp. 219–224 (1999)
13. Guermeur, Y.: Svm multiclasses, théorie et applications (2007)
14. Kreßel, U.: Pairwise classification and support vector machines. In: Advances in Kernel Methods: Support Vector Learning, pp. 255–268 (1999)
15. Platt, J., Cristianini, N., Shawe-Taylor, J.: Large margin dags for multiclass classification. Advances in Neural Information Processing Systems 12, 547–553 (2000)
16. Japkowicz, N. (ed.): AAAI'Workshop on Learning from Imbalanced Data Sets. Number WS-00-05 in AAAI Tech. Report (2000)
17. Weiss, G.M., Provost, F.: Learning when training data are costly: The effect of class distribution on tree induction. Journal of Artificial Intelligence Research 19, 315–354 (2003)
18. Visa, S., Ralescu, A.: Issues in mining imbalanced data sets - A review paper. In: Midwest Artificial Intelligence and Cognitive Science Conf., Dayton, USA, pp. 67–73 (2005)
19. Lenca, P., Lallich, S., Do, T.-N., Pham, N.-K.: A Comparison of Different Off-Centered Entropies to Deal with Class Imbalance for Decision Trees. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 634–643. Springer, Heidelberg (2008)

20. Pham, N.K., Do, T.N., Lenca, P., Lallich, S.: Using local node information in decision trees: coupling a local decision rule with an off-centered. In: International Conference on Data Mining, pp. 117–123. CSREA Press, Las Vegas (2008)
21. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: Improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 107–119. Springer, Heidelberg (2003)
22. Liu, X.Y., Wu, J., Zhou, Z.H.: Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B 39(2), 539–550 (2009)
23. Ricamato, M.T., Marrocco, C., Tortorella, F.: Mcs-based balancing techniques for skewed classes: An empirical comparison. In: ICPR, pp. 1–4 (2008)
24. MPI-Forum: MPI: A message-passing interface standard
25. OpenMP Architecture Review Board: OpenMP application program interface version 3.0 (2008)
26. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: a library for large linear classification. Journal of Machine Learning Research 9(4), 1871–1874 (2008)
27. Franc, V., Sonnenburg, S.: Optimized cutting plane algorithm for large-scale risk minimization. Journal of Machine Learning Research 10, 2157–2192 (2009)
28. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(3), 480–492 (2012)
29. Wu, J.: Power mean svm for large scale visual classification. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2344–2351 (2012)
30. Berg, A., Deng, J., Li, F.F.: Large scale visual recognition challenge 2010. Technical report (2010)