Large Scale Visual Classification with Many Classes

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

[1] IRISA,
[2] Université de Rennes 1, Campus de Beaulieu, 35042 Rennes Cedex, France
{thanh-nghi.doan,francois.poulet}@irisa.fr
[3] Institut Telecom, Telecom Bretagne UMR CNRS 6285 Lab-STICC, Brest, France
Université européenne de Bretagne, France, Can Tho University, Vietnam
tn.do@telecom-bretagne.eu

Abstract. The usual frameworks for visual classification involve three steps: extracting features, building codebook and encoding features, and training classifiers. The current release of ImageNet dataset [1] with more than 14M images and 21K classes makes the problem of visual classification become more difficult to deal with. One of the most difficult tasks is to train a fast and accurate classifier. In this paper, we address this challenge by extending the state-of-the-art large scale classifier Power Mean SVM (PmSVM) proposed by Jianxin Wu [2] in two ways: (1) The first one is to build the balanced bagging classifiers with under-sampling strategy. Our algorithm avoids training on full data and the training process of PmSVM rapidly converges to the optimal solution, (2) The second one is to parallelize the training process of all classifiers with multi-core computers. We have developed the parallel versions of PmSVM based on high performance computing models. The evaluation on 1000 classes of ImageNet (ILSVRC 1000 [3]) shows that our approach is 90 times faster than the original implementation of PmSVM and 240 times faster than the state-of-the-art linear classifier (LIBLINEAR [4]).

Keywords: Large Scale Visual Classification, High Performance Computing, Sampling Strategy, Parallel Support Vector Machines.

1 Introduction

Visual classification is one of the important research topics in the area of computer vision and machine learning. Low-level local features and bag-of-words model (BoW) are the core of state-of-the-art visual classification systems. The usual frameworks for visual classification involve three steps: 1) extracting features, 2) building codebook and encoding features, and 3) training classifiers. All these frameworks were evaluated on small datasets, e.g. Caltech 101 [5], Caltech 256 [6], and PASCAL VOC [7] that can fit into desktop memory. In step 3, most researchers choose either linear or non-linear SVM classifiers that can be trained in a few minutes.

However, ImageNet with very large number of classes poses more challenges in training classifiers. ImageNet is much larger in scale and diversity than the other benchmark datasets. The current release ImageNet has grown a big step in terms of the number of images and the number of classes, as shown in Fig. 1 - it has 21,841 classes with more than 1000 images for each class on average.

P. Perner (Ed.): MLDM 2013, LNAI 7988, pp. 629–643, 2013.

With millions of images, training an accurate classifier may take weeks or even years [8], [9]. The recent works in large scale learning classifiers converge on building linear SVM classifiers. We are able to train linear classifiers (e.g. LIBLINEAR) in order of seconds, even with millions of training examples. However, in the context of visual classification, linear classifier is inferior in terms of accuracy, compared to non-linear classifiers [10], [11], [2]. Wu [2] proposes Power Mean SVM classifier that outperforms LIBLINEAR and other additive kernel classifiers in terms of training time and classification accuracy. Nevertheless, the current version of PmSVM does not take into account the benefits of high performance computing (HPC). On ILSVRC 1000, it takes very long time to train all binary classifiers. Therefore, it motivates us to study how to speed-up PmSVM for large scale visual classification. In this paper, we have developed the extended versions of PmSVM in two ways:

1. Propose a balanced bagging algorithm for training binary classifiers. Our algorithm avoids training on full data and the training process of PmSVM rapidly converges to the optimal solution.

2. Parallelize the training process of all binary classifiers based on HPC models. In the training step of classifiers, we apply our balanced bagging algorithm to achieve the best performance.

Our approach is evaluated on the 10 and 100 largest classes of ImageNet and ILSVRC 1000. The result shows that our approach is 90 times faster than the original implementation of PmSVM and 240 times faster than LIBLINEAR without (or very few) compromising classification accuracy.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work on large scale visual classification. Section 3 introduces Power Mean SVM. In section 4, we present its improvement for large number of classes and describe how to speed-up the training process of PmSVM by using our balanced bagging algorithm and take into account the benefits of HPC. Section 5 presents numerical results before the conclusion and future work.

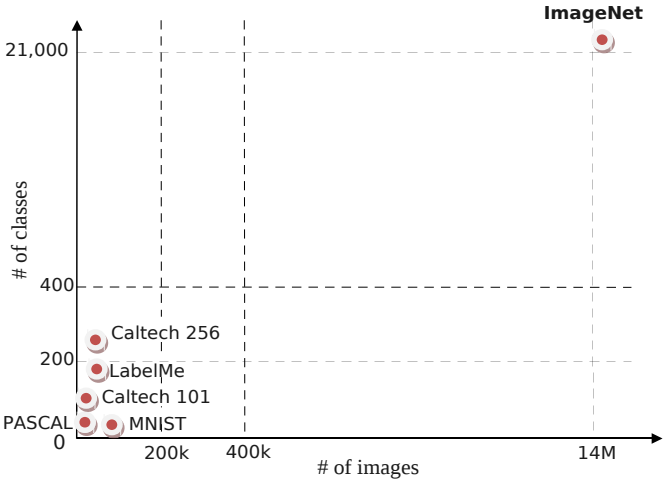

Fig. 1. A comparison of ImageNet with other benchmark datasets

2 Related Work

Many previous works on image classification rely on bag-of-words model (BoW) [12], local feature quantization and support vector machines. These models may be enhanced by multi-scale spatial pyramids [13] on BoWs or histogram of oriented gradient [14] features. Some recent works consider exploiting the hierarchical structure of dataset for image recognition and achieve impressive improvements in accuracy and efficiency [15]. Related to classification is the problem of detection, often treated as repeated one-versus-all classification in sliding windows [7], [16]. In many cases, such localization of objects might be useful for improving classification accuracy performance. However, in the context of large scale visual classification with hundreds or thousands of classes, these common approaches become computationally intractable.

To address this problem, Fergus *et al.* [17] study semi-supervised learning on 126 hand labeled Tiny Images categories, Wang *et al.* [18] show classification experiments on a maximum of 315 categories. Li *et al.* [19] do research with landmark classification on a collection of 500 landmarks and 2 million images. On a small subset of 10 classes, they have improved BoW classification by increasing the visual vocabulary up to 80K visual words. Furthermore, the current released ImageNet makes the complexity of large scale visual classification become a big challenge. To tackle this challenge, many researchers are beginning to study strategies on how to improve the accuracy performance and avoid using high cost non-linear kernel SVMs for training classifiers. The recent prominent works for these strategies are proposed in [8] [9], [20], [21] where the data is first transformed by a nonlinear mapping induced by a particular kernel and then efficient linear classifiers are trained in the resulting space. They argue that the classification accuracy of linear classifiers with high dimensional image representations is similar to low dimensional BoW with non-linear kernel classifiers. In [9], each local descriptor is coded either using Local Coordinate Coding [22] or Supper-vector Coding [23], after performing spatial pyramid pooling the resulting image representation is a vector in approximately 262K dimensions. To train classifiers, they propose a parallel averaging stochastic gradient descent (ASGD) algorithm. With 1000 classes from ILSVRC 1000, it takes 4 days to train 1000 binary SVM classifiers (one-versus-all) for one feature channel on three 8-core computers. Sánchez and Perronin [21] study the impact of high dimensional Fisher vectors on large dataset. They show that the larger the training dataset, the higher the impact of the dimensionality on the classification accuracy. To get the state-of-the-art result on ILSVRC 1000, they use the spatial pyramids to increase the dimensionality of their Fisher vectors to approximately 524K dimensions and then exploit Product Quantizier [24] to compress the data before training classifiers. With this approach, training 1000 SGD SVM classifiers (one-versus-all) for one feature channel takes 1.5 days on a 16-core computer.

In contrast with the approaches using the efficient linear SVMs, some recent works show that in the context of training classifiers for computer vision tasks, linear SVMs are inferior in terms of accuracy, compared to the kernel versions of SVM. In many cases of visual classification, learning SVMs with additive kernels give significantly higher rate in accuracy performance than dot product kernel [10] [11]. The main drawback of these approaches is the high cost of training non-linear kernel classifiers. It may be thousands times higher than linear classifiers. However, some recent solutions

are proposed to solve this limitation [10], [11]. Additive kernel SVMs now use only few times more training time, compared to the state-of-the-art linear SVM classifiers. To design a fast and accurate non-linear kernel classifier for large scale dataset, Wu [2] proposes an efficient algorithm for PmSVM. They show empirically that PmSVM outperforms LIBLINEAR and the state-of-the-art additive kernel classifiers in terms of both training time and classification accuracy. For instance, with 1000 classes from ILSVRC 1000, PmSVM is 3 times faster than LIBLINEAR and 2 times faster than the state-of-the-art additive kernel implementations while getting a significant improvement in classification accuracy from +4.6% to +7.1%. However, the current version of PmSVM does not take the benefits of the modern chip manufacturing. On one core of our computer, it takes more than 18 hours to train 1000 binary classifiers on ILSVRC 1000.

In the multi-core era, computers with multi-cores or multiprocessors are becoming more and more popular and affordable. So it motivates us to investigate parallel solutions and demonstrate how PmSVM can benefit from modern platforms. Furthermore, in the case of large number of classes, we show that our balanced bagging algorithm is very useful to speed-up the training process of classifiers without (or very few) compromising classification accuracy. Our experiments show very good results and confirm that the balanced bagging algorithm and parallel solutions are very essential for large scale visual classification in terms of training time.

3 Power Mean Support Vector Machines

Let us consider a binary linear classification task with m datapoints in a n-dimensional input space $x_1, x_2, \ldots, x_m$ having corresponding labels $y_i = \pm 1$. SVM classification algorithm [25] aims to find the best separating surface as being furthest from both classes. It can simultaneously maximize the margin between the support planes for each class and minimize the error. This can be accomplished through the quadratic program (1).

$$min_\alpha (1/2) \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K\langle x_i, x_j \rangle - \sum_{i=1}^{m} \alpha_i$$

$$s.t. \begin{cases} \sum_{i=1}^{m} y_i \alpha_i = 0 \\ 0 \leq \alpha_i \leq C \quad \forall i = 1, 2, ..., m \end{cases} \tag{1}$$

where $K\langle x_i, x_j \rangle$ is a kernel function of x_i and x_j, C is a positive constant used to tune the margin and the error.

The support vectors (for which $\alpha_i > 0$) are given by the solution of the quadratic program (1), and then, the separating surface and the scalar b are determined by the support vectors. The classification of a new data point x is based on:

$$sign(\sum_{i=1}^{\#SV} y_i \alpha_i K\langle x, x_i \rangle - b) \tag{2}$$

Variations on SVM algorithms use different classification functions. No algorithmic changes are required from the usual kernel function K as a linear inner product other

than the modification of the kernel evaluation, including a polynomial function of degree d, a RBF (Radial Basis Function) or a sigmoid function. We can get different support vector classification models.

PmSVM proposed by Wu [2] replaces the kernel function $K\langle x_i, x_j\rangle$ in (1) and (2) with the power mean kernel $M\langle x_i, x_j\rangle$ (x_i and $x_j \in R_+^n$), which is well-known as a general form of many additive kernels (e.g. χ^2 kernel, histogram intersection kernel or Hellinger's kernel).

$$M_p\langle x_i, x_j\rangle = \sum_{d=1}^{n} (x_{id}^p + x_{jd}^p)^{\frac{1}{p}} \tag{3}$$

where $p \in R$ is a constant.

PmSVM uses the coordinate descent method [26] for dealing with training tasks. Furthermore, the gradient computation step of the coordinate descent algorithm can be estimated approximately with the polynomial regression with a very low cost [2]. Therefore, PmSVM is very efficient in both training and testing tasks, compared to LIBLINEAR and other additive kernel SVMs.

4 Extensions of PmSVM to Large Number of Classes

Most SVM algorithms are only able to deal with a two-class problem. There are several extensions of a binary classification SVM solver to multi-class (k classes, $k \geq 3$) classification tasks. The state-of-the-art multi-class SVMs are categorized into two types of approaches. The first one is to consider the multi-class case in an optimization problem [27], [28]. The second one is to decompose multi-class into a series of binary SVMs, including one-versus-all [25], one-versus-one [29] and Decision Directed Acyclic Graph [30]. Recently, hierarchical methods for multi-class SVM [31], [32] start from the whole data set, hierarchically divide the data into two subsets until every subset consists of only one class.

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 classifiers. Therefore, the one-versus-all strategy becomes popular in this case. PmSVM algorithm also uses the one-versus-all approach to train independently k binary classifiers. However, the current PmSVM takes very long time to classify very large number of classes.

Due to this problem, we propose two ways for speed-up learning tasks of PmSVM. The first one is to build the balanced bagging classifiers with sampling strategy. The second one is to parallelize the training task of all classifiers with multi-core computers.

4.1 Balanced Bagging PmSVM

In the one-versus-all approach, the learning task of PmSVM is to 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. 1000 classes, this leads to the extreme imbalance between the positive class and the negative class. The problem is well-known as the class imbalance. As summarized by the review papers of [33], [34], [35] and the very comprehensive papers of [36], [37], 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 [38] or under-sampling the majority class [39], [40]. 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 bagging PmSVM belongs to the first approach (forms of re-sampling). Furthermore, the class prior probabilities in this context are highly unequal (e.g. the distribution of the positive class is 0.1% in the 1000 classes classification problem), and over-sampling the minority class is very expensive. We propose the balanced bagging PmSVM using under-sampling the majority class (negative class).

For separating the i^{th} class (positive class) from the rest (negative class), the balanced bagging PmSVM trains T models as shown in algorithm 1.

Algorithm 1. Balanced bagging PmSVM

input :
- D_p the training data of the positive class
- D_n the training data of the negative class
- T the number of base learners

output:
- *PmSVMmodel*

Learn:

for $k \leftarrow 1$ **to** T **do**
1. The subset D'_n is created by sampling without replacement $|D'_n|$ negative datapoints from D_n (with $|D'_n| = |D_p|$)
2. Build a PmSVM model using the training set (including D_p and D'_n)

end

combine T models (averaging) into the aggregated *PmSVMmodel*

We remark that the margin can be seen as the minimum distance between two convex hulls, H_p of the positive class and H_n of the negative class (the farthest distance between the two classes). Under-sampling the negative class (D'_n) done by balanced bagging provides the reduced convex hull of H_n, called H'_n. And then, the minimum distance between H_p and H'_n is larger than H_p and H_n (full dataset). It is easier to achieve the largest margin than learning on the full dataset. Therefore, the training task of PmSVM is fast to converge to the solution. According to our experiments, by setting $T = \sqrt{\frac{|D_n|}{|D_p|}}$, the balanced bagging PmSVM achieves good results in very fast training speed.

4.2 Parallel PmSVM Training

Although PmSVM and balanced bagging PmSVM deal with very large dataset with high speed, they do not take into account the benefits of HPC, e.g. multi-core

computers or grids. Furthermore, both PmSVM and balanced bagging PmSVM train independently k binary classifiers for k classes problems. This is a nice property for parallel learning. Our investigation aims at speed-up training tasks of multi-class PmSVM, balanced bagging PmSVM with multi-processor computers or grids. The idea is to learn k binary classifiers in parallel.

Algorithm 2. Hybrid MPI/OpenMP parallel PmSVM

input :
  D the training dataset with k classes
  T the number of MPI processes
output:
  $PmSVMmodel$

Learn:
$MPI - PROC_1$
#pragma omp parallel for
for $i_1 \leftarrow 1$ **to** k_1 **do** `/* class` i_1 `*/`
  Build a binary PmSVM model using the training set D to separate the positive class i_1 from the rest.
end
:
$MPI - PROC_T$
#pragma omp parallel for
for $i_T \leftarrow 1$ **to** k_T **do** `/* class` i_T `*/`
  Build a binary PmSVM model using the training set D to separate the positive class i_T from the rest.
end

The parallel programming is currently based on two major models, Message Passing Interface (MPI) [41] and Open Multiprocessing (OpenMP) [42]. 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, a MPI process loads the whole dataset ($\sim$ 25GB) into memory during learning tasks, making it intractable. The simplest development of parallel PmSVM algorithms is based on the shared memory multiprocessing programming model OpenMP. However OpenMP is not guaranteed to make the most efficient computing. Finally, we present a hybrid approach that combines the benefits from both OpenMP and MPI models. The parallel PmSVM algorithm is described in algorithm 2. The number of MPI processes depends on the memory capacity of high performance computing systems.

5 Experiments

In this section we compare the extended versions of PmSVM with the original implementation and LIBLINEAR in terms of training time and classification accuracy. Our

experiments are run on machine Intel(R) Xeon(R), CPU X5650, 2.67GHz, 24 cores, and 144GB main memory.

We have implemented four parallel versions of PmSVM: 1) OpenMP version of PmSVM (omp-PmSVM), 2) balanced bagging version of omp-PmSVM (omp-iPmSVM), 3) hybrid MPI/OpenMP version of PmSVM (mpi-omp-PmSVM), and 4) balanced bagging version of mpi-omp-PmSVM (mpi-omp-iPmSVM).

PmSVM. We set the parameters of PmSVM as follow: $p = -1$ (equivalent to χ^2 kernel) and $C = 0.01$.

LIBLINEAR. This is the linear SVM from [4] with default parameters ($C = 1$).

iPmSVM. This is the balanced bagging PmSVM with the same SVM parameters as PmSVM.

5.1 Dataset

The parallel versions of PmSVM are designed for large scale datasets, so we have evaluated the performance of our approach on the three following datasets.

ImageNet 10. This dataset contains the 10 largest classes from ImageNet (24,807 images with data size 2.4GB). There are more than 2000 diversified images per class. In each class, we sample 90% images for training and 10% images for testing. First, we construct bag-of-words histogram of every image by using dense SIFT descriptor (extracting SIFT on a dense grid of locations at a fixed scale and orientation), and 5000 codewords. Then, we use feature mapping from [10] to get the high-dimensional image representation in 15,000 dimensions. This feature mapping has been proven to give a good image classification performance [10]. We end up with 2.6GB of training data.

ImageNet 100. This dataset contains the 100 largest classes from ImageNet (183,116 images with data size 23.6GB). In each class, we sample 50% images for training and 50% images for testing. We also construct bag-of-words histogram of every image by using dense SIFT descriptor and 5000 codewords. For feature mapping, we use the same method as we do with ImageNet 10. The final size of training data is 8GB.

ILSVRC 1000. This dataset contains 1000 classes from ImageNet with 1.2M images (126GB) for training, 50K images (5.3GB) for validation and 150K images (16GB) for testing. To compare with the results reported in [2], we use the same method to encode every image as a vector in 21,000 dimensions. We also take ≤ 900 images per class for training dataset. Therefore, the total number of training images is 887,816 and the size of training data is 12.5GB. All testing samples are used to test SVM models.

5.2 Training Time

We have only evaluated the training time of SVM classifiers excluding the time needed to load data from disk. As shown in Fig. 2 and 3, on small and medium datasets as ImageNet 10, ImageNet 100, our four parallel versions show a very good speed-up in training process, compared to the original implementation of PmSVM and LIBLINEAR (Table 1 and 2).

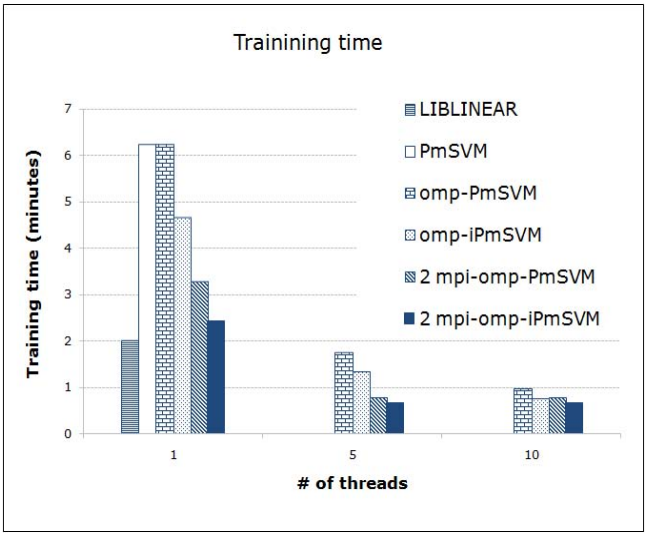

Fig. 2. SVMs training time with respect to the number of threads for ImageNet 10

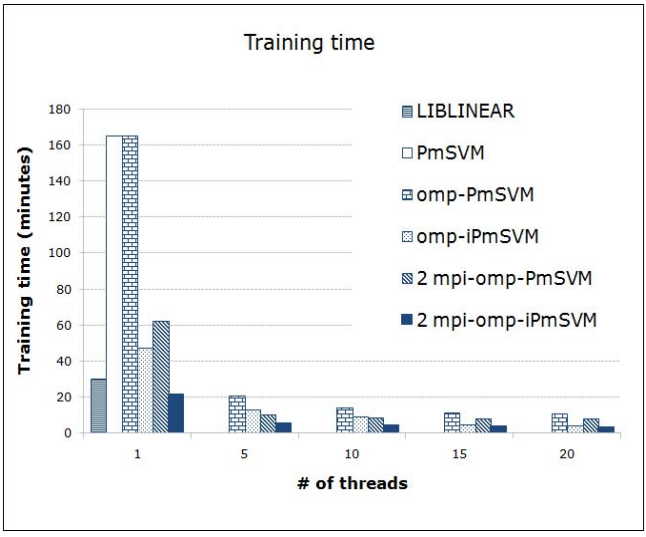

Fig. 3. SVMs training time with respect to the number of threads for ImageNet 100

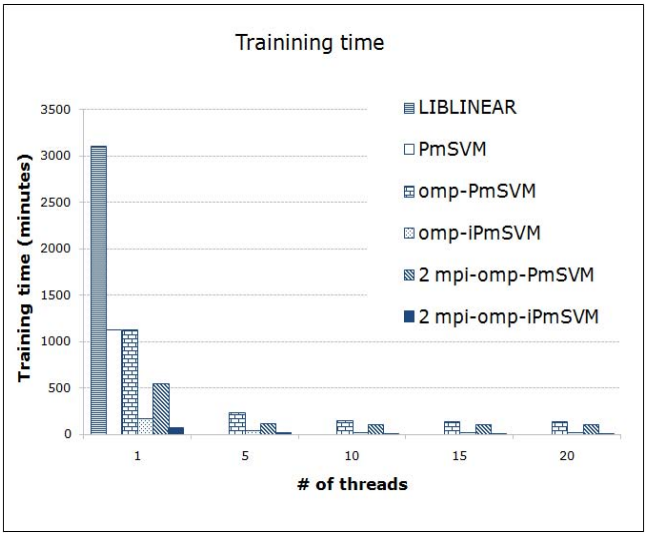

Fig. 4. SVMs training time with respect to the number of threads for ILSVRC 1000

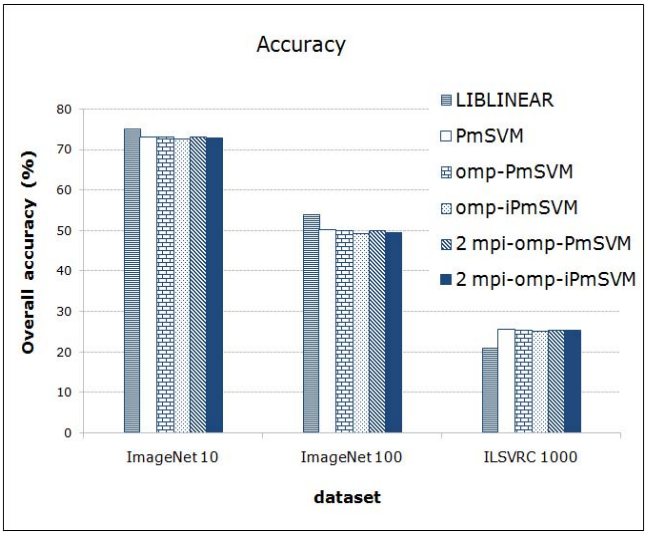

Fig. 5. Overall accuracy of SVM classifiers (%)

Table 1. SVMs training time (minutes) on ImageNet 10

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

Table 2. SVMs training time (minutes) on ImageNet 100

# OpenMP threads	1	5	10	15	20
LIBLINEAR	30.41				
PmSVM	165.45				
omp-PmSVM	165.45	21.12	14.52	11.67	10.92
omp-iPmSVM	47.67	12.97	9.09	5.05	4.44
2 mpi-omp-PmSVM	62.18	10.63	8.85	8.25	8.06
2 mpi-omp-iPmSVM	21.56	5.68	4.16	3.79	**3.58**

Table 3. SVMs training time (minutes) on ILSVRC 1000

# OpenMP threads	1	5	10	15	20
LIBLINEAR	3106.48				
PmSVM	1132.03				
omp-PmSVM	1132.03	231.00	152.87	140.72	135.63
omp-iPmSVM	173.26	39.55	23.63	19.43	18.12
2 mpi-omp-PmSVM	550.04	119.17	102.81	103.62	103.12
2 mpi-omp-iPmSVM	72.07	16.69	**13.08**	13.37	13.25

Table 4. SVMs overall classification accuracy (%)

dataset	ImageNet 10	ImageNet 100	ILSVRC 1000
LIBLINEAR	75.09	54.07	21.11
PmSVM	73.16	50.17	25.64
omp-PmSVM	73.16	50.17	25.64
omp-iPmSVM	72.79	49.42	25.35
2 mpi-omp-PmSVM	73.16	50.17	25.64
2 mpi-omp-iPmSVM	72.79	49.42	25.35

ILSVRC 1000. Our implementations achieve a significant speed-up in training process when performing on large dataset ILSVRC 1000.

Balanced Bagging PmSVM. As shown in Fig. 4, the balanced bagging version of PmSVM (omp-iPmSVM running with 1 thread) has a very fast convergence speed in training process, it is more than 6 times faster than the original implementation of PmSVM (Table 3).

OpenMP PmSVM. On a multi-core machine, OpenMP version of PmSVM (omp-PmSVM) achieves a significant speed-up in training process with 20 OpenMP threads. As shown in Fig. 4, our implementation is 8 times faster than the original PmSVM and 23 times faster than LIBLINEAR (Table 3). Due to the restriction of our computer (24 cores), we set the maximum number of OpenMP threads to 20. We can set more than 20 OpenMP threads, but according to our observation there is very few significant speed-up in training process because there is no more available core.

OpenMP and Balanced Bagging PmSVM. With balanced bagging algorithm applied to OpenMP version of PmSVM (omp-iPmSVM), we significantly speed-up the training process on this training data. For instance, with the number of OpenMP threads set to 20, omp-iPmSVM is 62 times faster than the original PmSVM and 171 times faster than LIBLINEAR.

MPI/OpenMP PmSVM. As show in Fig. 4, our hybrid MPI/OpenMP version of PmSVM (mpi-omp-PmSVM) achieves a significant speed-up in training process with 2 MPI processes and 10 OpenMP threads. Our implementation is 11 times faster than the original PmSVM and 30 times faster than LIBLINEAR (Table 3). Due to the large size of this training data, each MPI process of PmSVM needs to use $\sim$ 25GB main memory to train classifiers. With the memory restrictions of our computer, we can only evaluate mpi-omp-PmSVM by setting the number of MPI processes to 2. In this case, we vary the number of OpenMP threads running in each MPI process. Again, with 24 cores of our computer we set the maximum number of OpenMP threads to 10.

MPI/OpenMP and Balanced Bagging PmSVM. The most significant parallelization performance of PmSVM we achieve is the combination of MPI/OpenMP and balanced bagging PmSVM (mpi-omp-iPmSVM). As shown in Fig. 4, our implementation achieves a significant performance in training process with 2 MPI processes and 10 OpenMP threads. It is 90 times faster than the original PmSVM and 240 times faster than LIBLINEAR. On ILSVRC 1000, we need only 13 minutes to finish training 1000 binary classifiers, compared to the original PmSVM ($\sim$ 19 hours) and LIBLINEAR ($\sim$ 2 days and 4 hours), as shown in Table 3. This result confirms that our approach has a great ability to scaleup to full ImageNet dataset with more than 21,000 classes.

5.3 Classification Accuracy

As shown in Fig. 5, on the small datasets like ImageNet 10 and ImageNet 100, LIBLINEAR outperforms PmSVM and iPmSVM in terms of classification accuracy.

However, when we perform classification on the dataset with very large number of classes like ILSVRC 1000, the picture of accuracy performance is quite different.

PmSVM and iPmSVM achieve better results than LIBLINEAR (from +4.24% to +4.53%, ie. a relative increase of 20.1%).

Note that iPmSVM runs much faster than PmSVM without (or very few) compromising classification accuracy (Table 4).

6 Conclusion and Future Work

We have developed the extended versions of PmSVM to efficiently deal with large scale datasets with very large number of classes like ImageNet. To speed-up the training process of the binary classifiers, we have extended PmSVM in two ways. The first one is to build the balanced bagging classifiers with under-sampling strategy. Our algorithm avoids training on full training data, so the training process of PmSVM rapidly converges to the optimal solution. The second one is to parallelize the training process of all classifiers with multi-core computers. We have developed the parallel versions of PmSVM based on HPC models (OpenMP, MPI, and hybrid MPI/OpenMP). In each parallel version of PmSVM we apply our balanced bagging algorithm to obtain the best performance in the training process.

We have evaluated the performance in terms of training time on the large scale dataset like ImageNet. On ILSVRC 1000, our implementation is 90 times faster than the original implementation of PmSVM and 240 times faster than LIBLINEAR. To obtain this performance we set the number of threads to 20 on our computer. Therefore, with our approach we can get higher performances by using more resources (CPU cores, computer, etc.). Furthermore, with the balanced bagging approach we significantly speed-up the training process of the classifiers without (or very few) compromising the overall classification accuracy. We need only 13 minutes to train 1000 binary classifiers on ILSVRC 1000. Obviously, this is a roadmap towards very large scale visual classification. However, when the training data is larger, PmSVM requires a large amount of main memory. In the near future, we may study the approach that avoids loading the whole training data to main memory as in [43]. Another possibility could be to compress the training data and handle the compressed data on the fly, as in [21].

Acknowledgements. This work was partially funded by Region Bretagne (France).

References

1. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: Imagenet: A large-scale hierarchical image database. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009)
2. 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)
3. Berg, A., Deng, J., Li, F.F.: Large scale visual recognition challenge 2010. Technical report (2010)
4. Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear svm. In: International Conference on Machine Learning, pp. 408–415 (2008)

5. 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)
6. Griffin, G., Holub, A., Perona, P.: Caltech-256 Object Category Dataset. Technical Report CNS-TR-2007-001, California Institute of Technology (2007)
7. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision 88(2), 303–338 (2010)
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. Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L., Huang, T.S.: Large-scale image classification: Fast feature extraction and svm training. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1689–1696 (2011)
10. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(3), 480–492 (2012)
11. Wu, J.: A fast dual method for HIK SVM learning. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 552–565. Springer, Heidelberg (2010)
12. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: In Workshop on Statistical Learning in Computer Vision, ECCV, pp. 1–22 (2004)
13. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2169–2178 (2006)
14. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 886–893. IEEE Computer Society (2005)
15. Griffin, G., Perona, D.: Learning and using taxonomies for fast visual categorization. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society (2008)
16. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple kernels for object detection. In: IEEE 12th International Conference on Computer Vision, pp. 606–613. IEEE (2009)
17. Fergus, R., Weiss, Y., Torralba, A.: Semi-supervised learning in gigantic image collections. In: Advances in Neural Information Processing Systems, pp. 522–530 (2009)
18. Wang, C., Yan, S., Zhang, H.J.: Large scale natural image classification by sparsity exploration. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 3709–3712. IEEE (2009)
19. Li, Y., Crandall, D.J., Huttenlocher, D.P.: Landmark classification in large-scale image collections. In: IEEE 12th International Conference on Computer Vision, pp. 1957–1964. IEEE (2009)
20. Perronnin, F., Sánchez, J., Liu, Y.: Large-scale image categorization with explicit data embedding. In: CVPR, pp. 2297–2304 (2010)
21. Sánchez, J., Perronnin, F.: High-dimensional signature compression for large-scale image classification. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1665–1672 (2011)
22. Yu, K., Zhang, T., Gong, Y.: Nonlinear learning using local coordinate coding. In: Advances in Neural Information Processing Systems, pp. 2223–2231 (2009)
23. Zhou, X., Yu, K., Zhang, T., Huang, T.S.: Image classification using super-vector coding of local image descriptors. In: European Conference on Computer Vision, pp. 141–154 (2010)
24. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(1), 117–128 (2011)

25. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (1995)
26. Yuan, G.X., Ho, C.H., Lin, C.J.: Recent advances of large-scale linear classification. Proceedings of the IEEE 100(9), 2584–2603 (2012)
27. 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)
28. Guermeur, Y.: Svm multiclasses, théorie et applications (2007)
29. Krebel, U.: Pairwise classification and support vector machines. Advances in Kernel Methods: Support Vector Learning, 255–268 (1999)
30. Platt, J., Cristianini, N., Shawe-Taylor, J.: Large margin dags for multiclass classification. Advances in Neural Information Processing Systems 12, 547–553 (2000)
31. Vural, V., Dy, J.: A hierarchical method for multi-class support vector machines. In: Proceedings of the Twenty-frst International Conference on Machine Learning, pp. 831–838 (2004)
32. Benabdeslem, K., Bennani, Y.: Dendogram-based svm for multi-class classification. Journal of Computing and Information Technology 14(4), 283–289 (2006)
33. Japkowicz, N. (ed.): AAAI'Workshop on Learning from Imbalanced Data Sets. Number WS-00-05 in AAAI Tech. report (2000)
34. 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)
35. 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)
36. 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)
37. 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)
38. 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)
39. 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)
40. Ricamato, M.T., Marrocco, C., Tortorella, F.: Mcs-based balancing techniques for skewed classes: An empirical comparison. In: ICPR, pp. 1–4 (2008)
41. MPI-Forum.: Mpi: A message-passing interface standard
42. OpenMP Architecture Review Board: OpenMP application program interface version 3.0 (2008)
43. Do, T.N., Nguyen, V.H., Poulet, F.: Gpu-based parallel svm algorithm. Journal of Frontiers of Computer Science and Technology 3(4), 368–377 (2009)