Incremental Parallel Support Vector Machines for Classifying Large-Scale Multi-class Image Datasets

Thanh-Nghi Do[1,2](✉) and Minh-Thu Tran-Nguyen[1]

[1] College of Information Technology, Can Tho University, Can Tho 92100, Vietnam
{dtnghi,tnmthu}@cit.ctu.edu.vn
[2] UMI UMMISCO 209 (IRD/UPMC), Can Tho, Vietnam

Abstract. In this paper, we propose an incremental parallel support vector machines (SVM) training with stochastic gradient descent (SGD) for dealing with the very large number of images and large-scale multi-class on standard personal computers (PCs). The two-class SVM-SGD algorithm is extended in several ways to develop the new incremental parallel multi-class SVM-SGD in large-scale classifications. We propose the balanced batch SGD of SVM (BBatch-SVM-SGD) for trainning two-class classifiers used in the one-versus-all strategy of the multi-class problems and the incremental training process of classifiers in parallel way on multi-core computers. The numerical test results on ImageNet datasets show that our algorithm is efficient compared to the state-of-the-art linear SVM classifiers in terms of training time, correctness and memory requirements.

Keywords: Large-scale multi-class image classification · Incremental training · Support vector machines (SVM) · Stochastic gradient descent (SGD)

1 Introduction

The classification of images is one of the most important research topics in computer vision and machine learning. The purpose of 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 the main steps as follows: extracting features and building code-book, training classifiers. The popular systems of image classification (first publications [1,2]) for representing images use the Scale-Invariant Feature Transform method (SIFT [3,4]), the Bag-of-visual-Words representation model (BoW). The SIFT algorithm is to detect and describe local features in images which are invariant to image scale, rotation and also robust to changes in illumination, noise, occlusion. And then, k-means algorithm [5] 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 SVM algorithms [6] are suited for dealing with very-high-dimensional datasets. In spite of the accurate classification models, SVMs are not favorable to handle the challenge of large datasets. SVM solutions are obtained from quadratic programming, so that the computational cost [7] is at least square of the number of training datapoints and the memory requirement making SVM impractical.

The effective heuristics to improve SVM learning task are to divide the original QP into series of small problems [7,8]. Incremental learning [9–12] try to update solutions in growing training set. The other techniques include parallel and distributed learning on PC network [13,14], on graphics processing units [15] or choosing active set [16–18] for learning, ensemble-based [19], local learning [20–22], using the stochastic gradient descent for large scale linear SVM solvers [23–27]. However, these proposed algorithms are difficult to deal with large-scale multi-class image datasets on PCs (e.g. Caltech with 101 classes [28], Caltech with 256 classes [29] having hundreds of classes, and ImageNet dataset [30] with more than 14 million images in 21,841 classes). It yields huge classification challenges of very-high-dimensional and large-scale multi-class image datasets. For scaling-up the training in practice, the data is first transformed by a nonlinear mapping induced by a particular kernel and then the efficient linear classifiers are trained on the mapping space [30]. Furthermore, the comparative study in [31] shows that the training of a linear SVM is about 600 times faster than the training of a non-linear one with the same accuracy.

This challenge motivates us to study an efficient incremental linear SVM training for dealing with the very large number of images and large-scale multi-class on standard personal computers (PCs). We propose the extensions of the stochastic gradient descend (SGD [23,24]) for two-class SVM to develop the new incremental parallel multi-class SVM-SGD for efficiently classifying large image datasets into many classes. Our contributions include:

1. the balanced batch stochastic gradient descend of support vector machine (BBatch-SVM-SGD) for very large number of classes,
2. the incremental training process of classifiers in parallel way on multi-core computers.

The numerical test results on ImageNet datasets [30] show that our algorithm is efficient compared to the state-of-the-art linear SVM classifiers in terms of training time, correctness and memory requirements.

The remainder of this paper is organized as follows. Section 2 briefly presents the SGD algorithm for two-class SVM problems. Section 3 describes how to extend the two-class SVM-SGD to develop the new incremental parallel multi-class SVM-SGD for efficiently classifying large image datasets into many classes. Section 4 presents evaluation results, before the conclusions and future work in Sect. 5.

2 Stochastic Gradient Descent for Binary Classification of Support Vector Machines

2.1 Support Vector Machines for Binary Classification

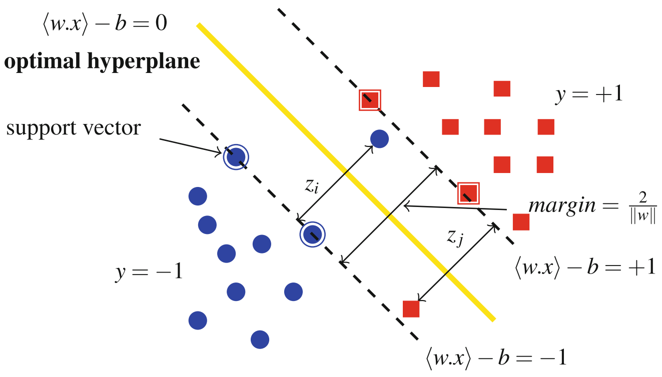
Let us consider a linear binary classification task, as depicted in Fig. 1. The dataset D consists of m datapoints $\{x_1, x_2, \dots, x_m\}$ in the n-dimensional input space R^n, having corresponding labels $\{y_1, y_2, \dots, y_m\}$ being ± 1. For this classification problem, the SVM algorithms [6] try to find the best separating plane (denoted by the normal vector $w \in R^n$ and the scalar $b \in R$), i.e. furthest from both class $+1$ and class -1. It is accomplished through the maximization of the margin (or the distance) between the supporting planes for each class ($x.w - b = +1$ for class $+1$, $x.w - b = -1$ for class -1). The margin between these supporting planes is $2/\|w\|$ (where $\|w\|$ is the 2-norm of the vector w). Any point x_i falling on the wrong side of its supporting plane is considered to be an error, its error distance denoted by $z_i \geq 0$. Therefore, SVM has to simultaneously maximize the margin and minimize the error. The standard SVM pursues these goals with the quadratic programming (1).

$$\begin{aligned} \min\ \Psi(w,b,z) &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} z_i \\ s.t.: y_i(w.x_i - b) + z_i &\geq 1 \\ z_i &\geq 0 \end{aligned} \quad (1)$$

where the positive constant C is used to tune errors and margin size.

The plane (w, b) is obtained by solving the quadratic programming (1). Then, the classification of a new datapoint x based on the plane is:

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



$\langle w.x\rangle - b = 0$
optimal hyperplane
support vector
$y = +1$
z_i
$margin = \frac{2}{\|w\|}$
z_j
$y = -1$
$\langle w.x\rangle - b = +1$
$\langle w.x\rangle - b = -1$

Fig. 1. Linear separation of the datapoints into two classes

SVM can use some kernel functions (e.g. a polynomial function of degree d or a Radial Basis Function) for dealing with non-linear classification tasks. More details about SVM and others kernel-based learning methods can be found in [32].

The study in [7] illustrated that the computational cost requirements of the SVM solutions in (1) are at least $O(m^2)$ (where m is the number of training datapoints), making standard SVM intractable for large datasets.

2.2 Stochastic Gradient Descent for Binary Classification of SVM

We can reformulate the SVM problem in quadratic programming (1) in an unconstraint problem. We can ignore the bias b without generality loss. The constraints $y_i(w.x_i) + z_i \geq 1$ in (1) are rewritten as follows:

$$z_i \geq 1 - y_i(w.x_i) \tag{3}$$

The constraints (3) and $z_i \geq 0$ are rewritten by the hinge loss function:

$$z_i = max\{0, 1 - y_i(w.x_i)\} = L(w, [x_i, y_i]) \tag{4}$$

Substituting for $z_i = L(w, [x_i, y_i])$ from the constraint in terms of w into the objective function Ψ of the quadratic programming (1) yields an unconstrained problem (5):

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

And then, [23,24] proposed the stochastic gradient descent method to solve the unconstrained problem (5). The stochastic gradient descent for SVM (denoted by SVM-SGD) updates w on T epochs with a learning rate η. For each epoch t, the SVM-SGD uses a single randomly received datapoint (x_i, y_i) to compute the sub-gradient $\nabla_t\Psi(w, [x_i, y_i])$ and update w_{t+1} as follows:

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

$$\nabla_t L(w, [x_t, y_t]) = \nabla_t max\{0, 1 - y_t(w.x_t)\} = \begin{cases} -y_t x_t & if\ y_t(w.x_t) < 1 \\ 0 & otherwise \end{cases} \tag{7}$$

The SVM-SGD using the update rule (6) is described in Algorithm 1.

As mentioned in [23,24], the SVM-SGD algorithm quickly converges to the optimal solution due to the fact that the unconstrained problem (5) is convex optimization problems on very large datasets. The algorithmic complexity of SVM-SGD is linear with the number of datapoints. An example of its effectiveness is given with the classification into two classes of 780 000 datapoints in 470000-dimensional input space in 2 s on a PC and the test accuracy is similar to standard SVM.

Algorithm 1. SVM-SGD algorithm for binary classification

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

1  begin
2      init $w_1 = 0$
3      for $t \leftarrow 1$ to $T$ do
4          randomly pick a datapoint $[x_t, y_t]$ from training dataset $D$
5          set $\eta_t = \frac{1}{\lambda t}$
6          if $(y_i(w_t.x_i) < 1)$ then
7              $w_{t+1} = w_t - \eta_t(\lambda w_t - y_i x_i)$
8          else
9              $w_{t+1} = w_t - \eta_t \lambda w_t$
10         end
11     end
12     return $w_{t+1}$
13 end
```

3 Incremental Parallel SVM-SGD for Large-Scale Multi-class

The original SVM algorithms are only able to deal with two-class problems. There are several extensions of a two-class SVM solver for multi-class (c classes, $c \geq 3$) 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 problem in an optimization problem [33–35]. The second one is to decompose multi-class into a series of binary SVMs, including one-versus-all [6], one-versus-one [36], Decision Directed Acyclic Graph and hierarchical methods for multi-class SVM [37–39] (hierarchically partitioning the data into two subsets).

In practice, the most popular methods are One-Versus-All (ref. LIBLINEAR [40]), One-Versus-One (ref. LibSVM [41]) and are due to their simplicity. The One-Versus-All strategy builds c different binary SVM models where the i^{th} one separates the i^{th} class from the rest, illustrated in Fig. 2. The One-Versus-One strategy constructs $c(c1)/2$ binary SVM models for all the binary pairwise combinations of the c classes, illustrated in Fig. 3. The class is then predicted with the largest distance vote.

When dealing with very large number of classes, e.g. $c = 1,000$ classes, the one-versus-one strategy is too expensive because it needs training $499,500$ of binary classifiers and using them in the classification (compared to $1,000$ binary models learned by the one-versus-all strategy). Therefore, the one-versus-all strategy is suited for this case. And then, our multi-class SVM-SGD algorithm also use the one-versus-all approach to train independently c binary classifiers.

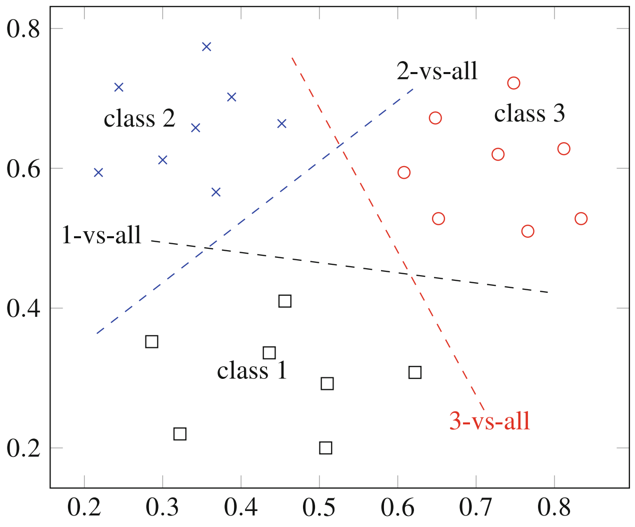


Fig. 2. Multi-class SVM (One-Versus-All)

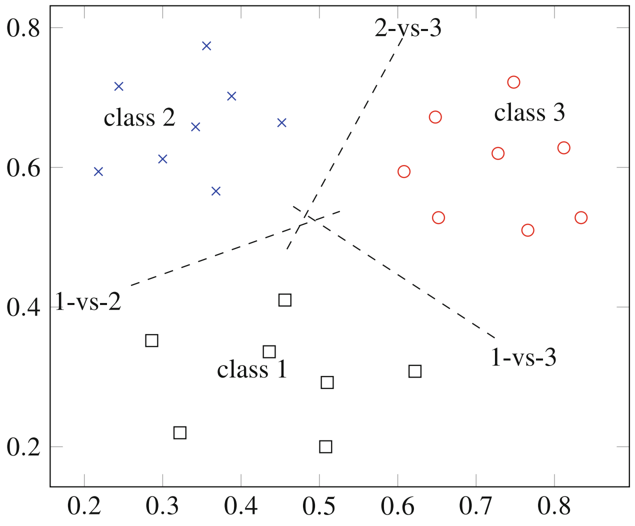


Fig. 3. Multi-class SVM (One-Versus-One)

Therefore, the multi-class SVM-SGD algorithm using one-versus-all leads to the two problems:

1. the SVM-SGD algorithm deals with the imbalanced datasets for building binary classifiers,

2. the SVM-SGD algorithm also takes very long time to train very large number of binary classifiers in sequential mode using a single processor,
3. furthermore, loading the whole large training dataset into memory requires very large memory capacity.

Due to these problems, we propose three ways for creating the new incremental parallel multi-class SVM-SGD algorithm (denoted by Incr-Par-MC-SVM-SGD) being able to handle the very large number of datapoints and large-scale multi-class on standard personal computers (PCs) in the high speed. The first one is to build balanced batch binary classifiers with under-sampling strategy. The second one is to parallelize the training task of all binary classifiers with several multi-core machines. The last one is the incremental learning of parallel multi-class SVM-SGD that avoids loading the whole large training dataset into memory.

3.1 Balanced Batch of Binary SVM-SGD Classifier

In the one-versus-all approach, the learning task of binary SVM-SGD classifier is try to separate the i^{th} class (positive class) from the $c-1$ others classes (negative class). For very large number of classes, this leads to the extreme unbalance between the positive and the negative class. The problem of binary SVM-SGD comes from **line 4** of Algorithm 1. Let us consider a classification problem with $1,000$ classes, the probability for a positive datapoint sampled is very small (about 0.001) compared with the large chance for a negative datapoint sampled (e.g. 0.999). And then, the binary SVM-SGD classifier focuses mostly on the errors produced by the negative datapoints. Therefore, the binary SVM-SGD classifier has difficulty to separate the positive class from the negative class, well-known as the class imbalance problems.

The survey papers [42–44] present the solutions for dealing with the imbalanced data. At the data level, the algorithms change the class distribution, including over-sampling the minority class [45] or under-sampling the majority class [46,47]. The algorithmic approaches [48–50] are to re-balance the error rate by weighting each type of error with the corresponding cost.

Our balanced batch of binary SVM-SGD (denoted by BBatch-SVM-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 0.1% in the $1,000$ classes classification problem), and then over-sampling the minority class is very expensive. We propose the BBatch-SVM-SGD algorithm using under-sampling the majority class (negative class). The training dataset D consists of the positive class D_+ ($|D_+|$ is the cardinality of the positive class) and the negative class D_- ($|D_-|$ is the cardinality of the positive 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. The balanced batch (denoted by BB) includes a datapoint randomly sampling from the positive class D_+ and $k\sqrt{\frac{|D_-|}{|D_+|}}$ datapoints sampling without replacement from the negative class D_-. As illustrated by

[51,52], SGD with a such mini-batch setting can asymptotically achieve optimal speed-up with the average sub-gradients for updating the predictor. Therefore, the updating rule (**lines 6–10** of Algorithm 1) uses the average hinge loss on datapoints in the balanced batch BB and then the classifier is the tail averaged $\bar{w}_t$ on all w_t. The BBatch-SVM-SGD in Algorithm 2 is to separate the i^{th} class (positive class) from the rest (negative class).

Algorithm 2. Training balanced batch of binary SVM-SGD classifier used in the one-versus-all approach of large-scale multi-class SVM

```
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 $BB_t$ by sampling without replacement $D'_-$
           from dataset $D_-$ (with $|D'_-| = k\sqrt{\frac{|D_-|}{|D_+|}}$) and a datapoint from dataset
           $D_+$
5          set $\eta_t = \frac{1}{\lambda t}$
6          $BB_k = \{[x_i, y_i] \in BB_t : y_i(w_t.x_i) < 1\}$
7          $w_{t+1} = (1 - \eta_t\lambda)w_t + \frac{\eta_t}{|BB_t|}\sum_{[x_i,y_i]\in BB_k} y_i x_i$
8      end
9      return $\bar{w}_t = \frac{2}{T}\sum_{t=\lfloor\frac{T}{2}\rfloor+1}^{T} w_t$
10 end
```

3.2 Parallel Training of BBatch-SVM-SGD

Although BBatch-SVM-SGD classifies very large dataset with high speed, but it does not take the benefits of high performance computing. Furthermore, BBatch-SVM-SGD independently trains c binary classifiers for c classes in multi-class SVM. This is a nice property for parallel learning. The main idea is to learn c binary classifiers in parallel to speedup training tasks of multi-class SVM-SGD. The simplest development of parallel BBatch-SVM-SGD described in Algorithm 3 is based on the shared memory multiprocessing programming model OpenMP on multi-core computers.

The parallel training algorithm of multi-class SVM-SGD (denoted by Par-MC-SVM-SGD) uses Algorithms 2 and 3 for handling large-scale multi-class datasets.

Algorithm 3. Parallel training of BBatch-SVM-SGD in the one-versus-all approach of large-scale multi-class SVM

```
input  : $D$ the training dataset with $c$ classes
output: SVM-SGD model
1 Learning:
2 #pragma omp parallel for
3 for $c_i \leftarrow 1$ to $c$ do                    /* class $c_i$ */
4     training BBatch-SVM-SGD($c_i - vs - all$)
5 end
```

3.3 Incremental Training of Multi-class SVM-SGD in Parallel

The Par-MC-SVM-SGD algorithm needs loading whole dataset in the memory to train the classification models. For very large-scale multi-class datasets such as ImageNet [30] with more than 14 million images and $21,841$ classes, the Par-MC-SVM-SGD algorithm requires at least 350 GB RAM. Any classification algorithm has some difficulties to deal with the challenge of large datasets.

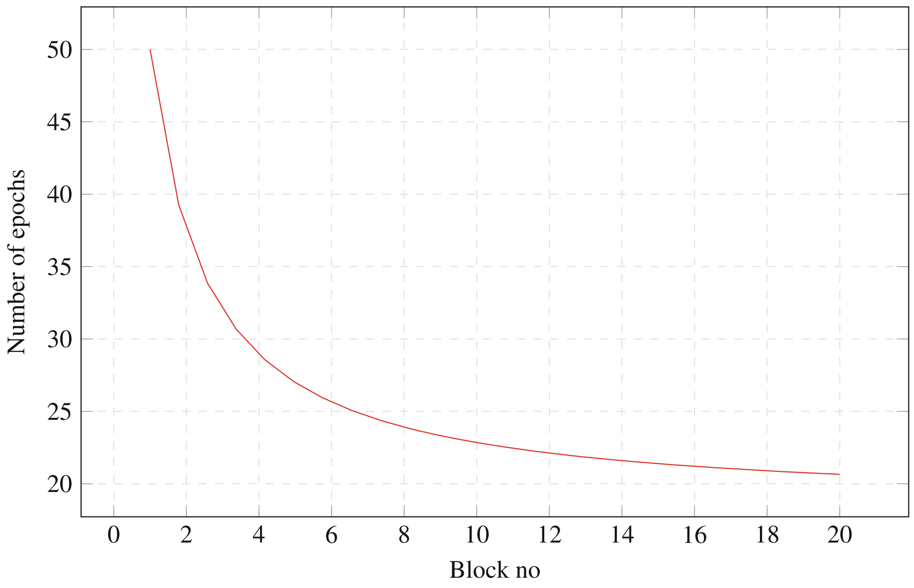


Fig. 4. Number of epochs for MC-SVM-SGD training at block D_b

The incremental training of Par-MC-SVM-SGD (called Incr-Par-MC-SVM-SGD) avoids loading the whole large dataset in main memory: only subsets of the data are considered at any one time and update the solution in growing training set. Let us consider a very large dataset D decomposed into B small blocks of rows, $\{D_1, D_2, \ldots, D_B\}$. For beginning, the Incr-Par-MC-SVM-SGD loads D_1 to learn a multi-class model *mc-svm-sgd*$_1$ with the Par-MC-SVM-SGD. At step b, the Incr-Par-MC-SVM-SGD uses D_b and datapoints in D_{b-1} near from separating boundary (ref. to *mc-svm-sgd*$_{b-1}$) to train a multi-class model *mc-svm-sgd*$_b$ with the Par-MC-SVM-SGD. We remark that the Incr-Par-MC-SVM-SGD aims at updating the previous model *mc-svm-sgd*$_{b-1}$ in growing training set. Hence, the Incr-Par-MC-SVM-SGD trains the model at the next step with the number of epochs getting decreased (for example in Fig. 4). The last model *mc-svm-sgd*$_B$ is the final multi-class classifier.

Algorithm 4. Incremental training of Par-MC-SVM-SGD

```
input :
        training data $D = \{D_1, D_2, \ldots, D_B\}$
        positive constant $\lambda > 0$
        number of epochs $T$
output:
        MC-SVM-SGD model
1 begin
2     init mc-svm-sgd$_1$ = Par-MC-SVM-SGD($D_1, \lambda, T$)
3     for $b \leftarrow 2$ to $B$ do
4         training sample $S_b$ includes $D_b$ and datapoints in $D_{b-1}$ near from
          separating boundary (ref. to mc-svm-sgd$_{b-1}$)
5         mc-svm-sgd$_b$ = Par-MC-SVM-SGD($S_b, \lambda, T[\frac{b+1}{b+2}]^{b-1}$)
6     end
7     return mc-svm-sgd$_B$
8 end
```

4 Evaluation

In order to evaluate the performance of the new incremental parallel multi-class SVM-SGD (Incr-Par-MC-SVM-SGD) algorithm for classifying large amounts of images into many classes, we have implemented the Incr-Par-MC-SVM-SGD and the Par-MC-SVM-SGD (batch version loading whole dataset in the memory) in C/C++ using the SGD library [24]. We are interested in two recent algorithms, LIBLINEAR (a library for large linear classification [40], the parallel version on multi-core computers) and OCAS (an optimized cutting plane algorithm for SVM [53]) because they are well-known as highly efficient standard linear SVM. Our comparison is reported in terms of correctness, training time and memory requirements obtained by Incr-Par-MC-SVM-SGD, Par-MC-SVM-SGD, LIBLINEAR and OCAS.

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 Incr-Par-MC-SVM-SGD 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 [30], 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 [54] 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 [54]. We end up with 2.6 GB of training data.

ImageNet 100. This dataset consists of the 100 largest classes from ImageNet [30], 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 [30], including 1.2 million images ($\sim$ 126 GB) for training, 50 thousand images ($\sim$ 5.3 GB) for validation and 150 thousand images ($\sim$ 16 GB) for testing. We use BoW feature set provided by [30] and the method reported in [55] 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 Parameters

The positive constant $C = 1,000,000$ (a trade-off between the margin size and the errors in learning SVM algorithms, the same tuning in [12,26,27]) was used in LIBLINEAR and OCAS to train classification models.

Our Incr-Par-MC-SVM-SGD and Par-MC-SVM-SGD algorithms learn the balanced batch stochastic gradient descend of SVM (BBatch-SVM-SGD) with $T = 50$ epochs and regularization term $\lambda = 0.00002$. Furthermore, the Incr-Par-MC-SVM-SGD loads in the main memory the small blocks of rows (instead of the whole dataset) to learn the classification model in incremental way. The large-scale multi-class dataset should be split into the small enough blocks of rows ($\sim$ 10 % – 20 % of the full datasets for a compromise the classification accuracy and the main memory usage). And then, the block sizes of ImageNet 10, ImageNet 100 and ILSVRC 2010 datasets are set to 2, 000, 15, 000 and 200, 000, respectively.

Due to the PC (Intel(R) Core i7-4790 CPU, 4 cores) used in the experimental setup, we try to vary the number of OpenMP threads (1, 4, 8 threads) for all training tasks.

4.3 Classificaton Results

Firstly, we are interested in the performance comparison in terms of training time, memory usage and accuracy.

Memory Usage. The main memory usage of training algorithms is presented in Table 1 and Fig. 5. As it was expected, the Incr-Par-MC-SVM-SGD uses less memory than other algorithms.

Regarding the comparison of the Incr-Par-MC-SVM-SGD with OCAS, one can see that the gains of main memory requirements ensured by the Incr-Par-MC-SVM-SGD against OCAS are 86.66 %, 81.65 % and 90.04 % for ImageNet 10, ImageNet 100 and ILSVRC 2010, respectively.

The improvements of main memory used by the Incr-Par-MC-SVM-SGD against LIBLINEAR correspond to 89.14 %, 87.05 % and 68.72 % for ImageNet 10, ImageNet 100 and ILSVRC 2010.

In the comparison with the Par-MC-SVM-SGD (batch version loading whole dataset in the memory), the Incr-Par-MC-SVM-SGD saves up 89.14 %, 84.80 % and 69.71 % main memory for ImageNet 10, ImageNet 100 and ILSVRC 2010, respectively.

Table 1. Memory usage (GB) of training algorithms

Dataset	ImageNet 10	ImageNet 100	ILSVRC 2010
OCAS	2.55	7.90	52.90
LIBLINEAR	3.13	11.20	16.90
Par-MC-SVM-SGD	3.13	9.54	17.40
Incr-Par-MC-SVM-SGD	0.34	1.45	5.27

Training Time. Table 2 and Fig. 6 present the training time of algorithms for ImageNet 10 (the small multi-class dataset). The Incr-Par-MC-SVM-SGD with 8 OpenMP threads is 82.05 times faster than OCAS (running on 1 core) and slight faster than LIBLINEAR (with 8 OpenMP threads). As mentioned in Algorithm 4, the Incr-Par-MC-SVM-SGD needs learning the datapoints near from separating boundary more than the Par-MC-SVM-SGD. The Par-MC-SVM-SGD performs 2 times faster than the Incr-Par-MC-SVM-SGD.

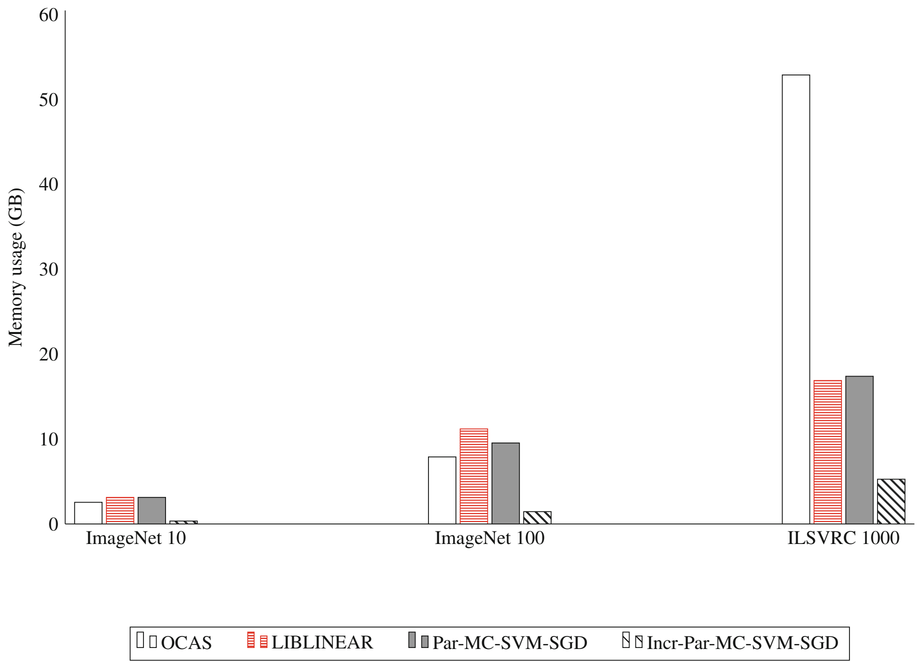


Fig. 5. Memory usage (GB) of training algorithms

The training time of algorithms on ImageNet 100 presented in Table 3 and Fig. 8 show that the Incr-Par-MC-SVM-SGD achieves a significant speed-up in learning process using 8 OpenMP threads. It is 74.62 times faster than OCAS and 2.22 times faster than LIBLINEAR. Once again, the Par-MC-SVM-SGD is 2 times faster than the Incr-Par-MC-SVM-SGD.

ILSVRC 2010 has large amount of images (more than 1 million images) and very large number of classes (1,000 classes). Therefore, OCAS has not finished the learning task in several days. LIBLINEAR with 8 OpenMP threads takes 1,004.00 min to train the classification model for this dataset. Our Incr-Par-MC-SVM-SGD algorithm performs the learning task in 50.35 min with 8

Table 2. Training time (minutes) on ImageNet 10

Algorithm	# OpenMP threads		
	1	4	8
OCAS	106.67		
LIBLINEAR	3.12	1.50	1.48
Par-MC-SVM-SGD	1.85	0.67	0.66
Incr-Par-MC-SVM-SGD	3.86	1.37	1.30

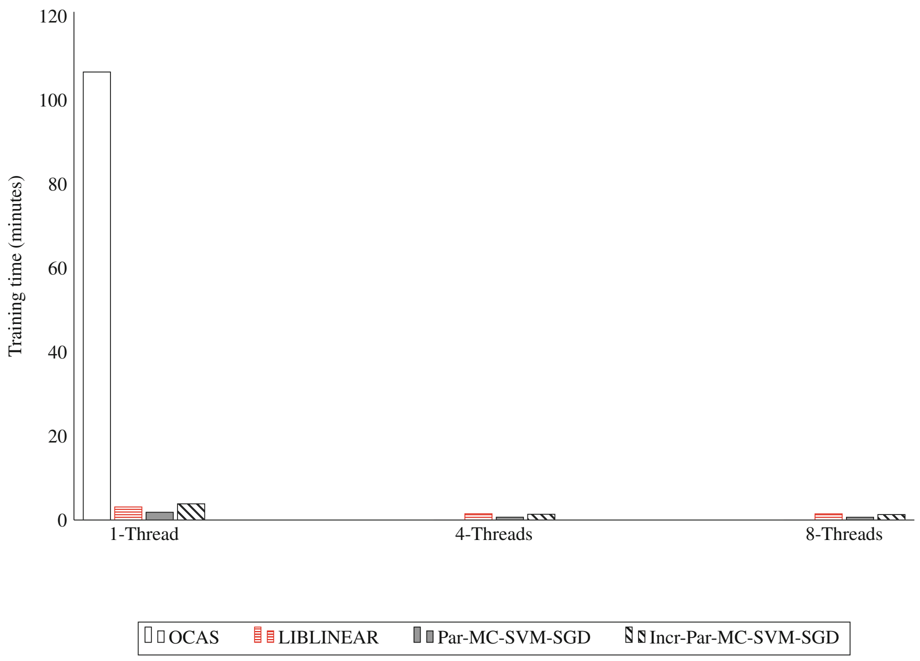


Fig. 6. Training time (minutes) on ImageNet 10

Table 3. Training time (minutes) on ImageNet 100

Algorithm	# OpenMP threads		
	1	4	8
OCAS	1016.35		
LIBLINEAR	63.42	30.49	30.18
Par-MC-SVM-SGD	24.66	7.03	6.91
Incr-Par-MC-SVM-SGD	47.09	14.86	13.62

OpenMP threads. This indicates that the Incr-Par-MC-SVM-SGD is 19.94 times faster than LIBLINEAR. The Incr-Par-MC-SVM-SGD needs 5.5 min more than the Par-MC-SVM-SGD for the learning task (Fig. 7).

Due to Intel(R) i7-4790 processor (4 cores), almost parallel algorithms using 8 threads can not improve much the training time against the 4 threads setting (Table 4).

Classification accuracy. The classification results in terms of accuracy presented in Table 5 and Fig. 9 show that the training of the Incr-Par-MC-SVM-SGD in incremental way has very few compromise the correctness, compared to its batch training of the Par-MC-SVM-SGD.

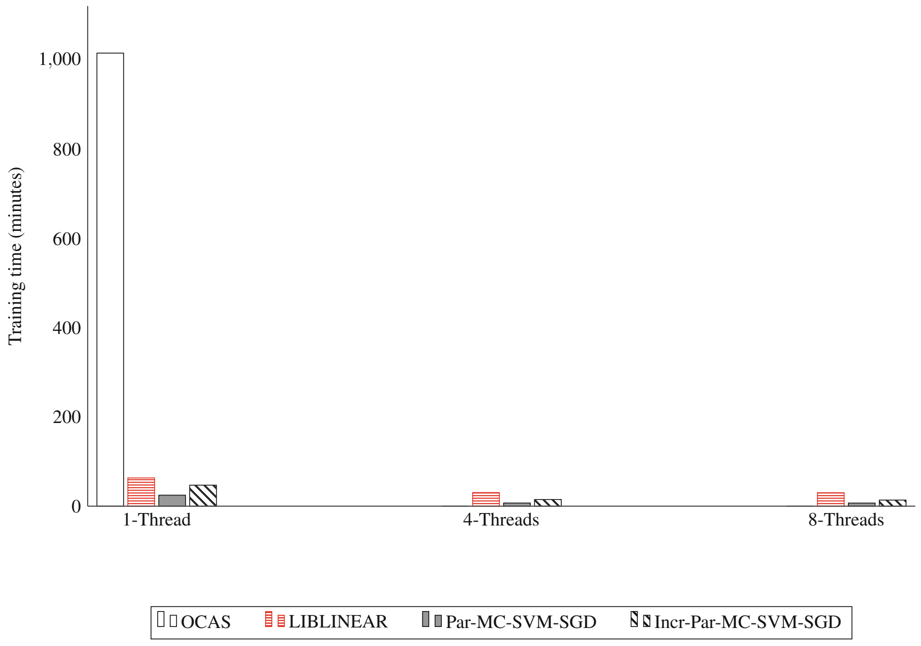


Fig. 7. Training time (minutes) on ImageNet 100

Table 4. Training time (minutes) on ILSVRC 2010

Algorithm	# OpenMP threads		
	1	4	8
OCAS	N/A		
LIBLINEAR	3106.48	1037.00	1004.00
Par-MC-SVM-SGD	188.70	51.79	44.85
Incr-Par-MC-SVM-SGD	206.68	61.98	50.35

The Incr-Par-MC-SVM-SGD is more accurate than OCAS on ImageNet 10 while making more classification mistakes than OCAS on ImageNet 100. The Incr-Par-MC-SVM-SGD also achieves very competitive performances compared to Par-MC-SVM-SGD and LIBLINEAR on ImageNet 10 and 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 competition the state-of-the-art system [30,56] reports an accuracy of approximately 19 % (it is far above random guess, 0.1 %).

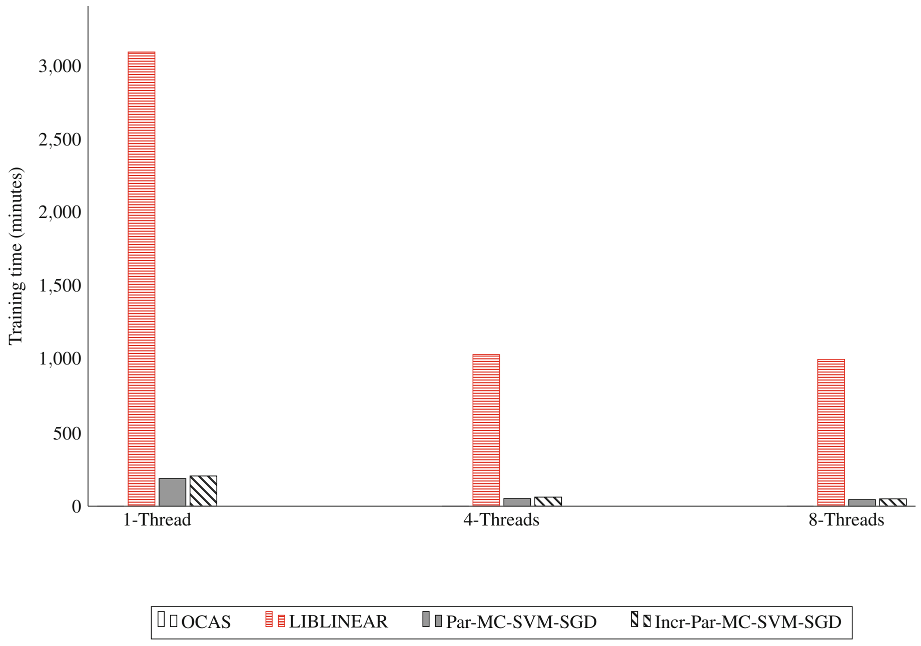


Fig. 8. Training time (minutes) on ILSVRC 2010

Our Incr-Par-MC-SVM-SGD algorithm gives a higher accuracy rate than [30,56] with the same feature set (21.19 % vs. 19 %). The Incr-Par-MC-SVM-SGD holds the rank 2 after the Par-MC-SVM-SGD. Note that the Incr-Par-MC-SVM-SGD learns much faster than LIBLINEAR while maintaining a high correctness rate. These results show that our Incr-Par-MC-SVM-SGD has a great ability to scale-up to full ImageNet dataset.

Table 5. Overall classification accuracy (%)

Dataset	ImageNet 10	ImageNet 100	ILSVRC 2010
OCAS	72.07	52.75	N/A
LIBLINEAR	75.09	54.07	21.11
Par-MC-SVM-SGD	75.33	53.60	21.90
Incr-Par-MC-SVM-SGD	74.16	51.98	21.19

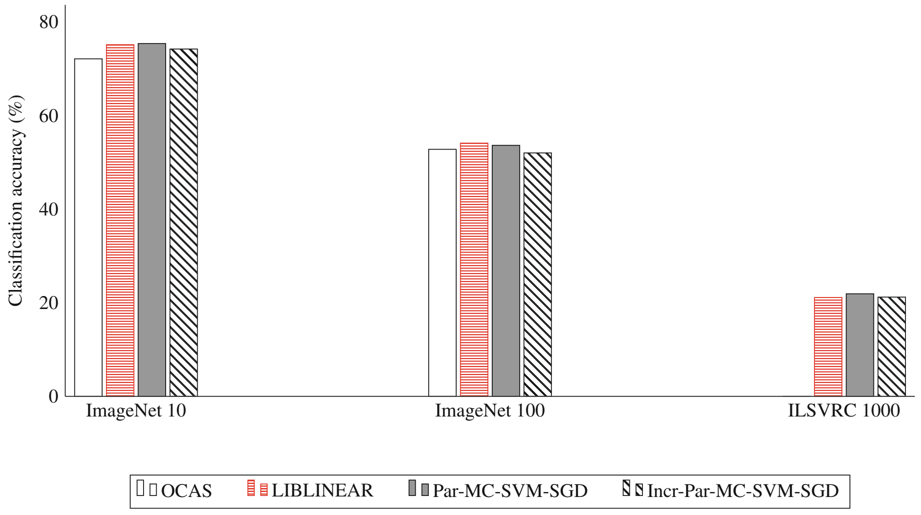


Fig. 9. Overall classification accuracy (%)

5 Conclusion and Future Works

We have presented the new incremental parallel multi-class SVM-SGD that achieves high performances for dealing with large amounts of images and large-scale multi-class on PCs. The balanced batch SGD of SVM (BBatch-SVM-SGD) is proposed for trainning two-class classifiers used in the multi-class problems. The incremental training process of classifiers in parallel way on multi-core computers is also developped for efficiently classifying large image datasets into very large number of classes. Our algorithm is evaluated on the 10, 100 and 1,000 largest classes of ImageNet datasets. The incremental algorithm saves up from 68.72 % to 90.04 % main memory usage while achieving significant low cost in terms of training time without (or very few) compromise the classification accuracy. It is able to handle in high-speed training the dataset larger than the memory capacity of PCs.

In the future, we will provide more empirical test on full ImageNet dataset with $21,000$ classes. We also intend to develop distributed MC-SVM-SGD algorithms for efficiently dealing with large scale multi-class problems on Spark [57].

References

1. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: 9th IEEE International Conference on Computer Vision (ICCV 2003), 14–17, October 2003, Nice, France, pp. 1470–1477 (2003)
2. Li, F., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20–26 June 2005, San Diego, CA, USA, pp. 524–531 (2005)

3. Lowe, D.G.: Object recognition from local scale invariant features. In: Proceedings of the 7th International Conference on Computer Vision, pp. 1150–1157 (1999)
4. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
5. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, vol. 1, pp. 281–297. University of California Press, January 1967
6. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1995)
7. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods Support Vector Learning, pp. 185–208 (1999)
8. Boser, B., Guyon, I., Vapnik, V.: An training algorithm for optimal margin classifiers. In: Proceedings of 5th ACM Annual Workshop on Computational Learning Theory of 5th ACM Annual Workshop on Computational Learning Theory, pp. 144–152. ACM (1992)
9. Syed, N., Liu, H., Sung, K.: Incremental learning with support vector machines. In: Proceedings of the ACM SIGKDD International Conference on KDD. ACM (1999)
10. Do, T.N., Poulet, F.: Incremental SVM and visualization tools for bio-medical data mining. In: Proceedings of the European Workshop on Data Mining and Text Mining for Bioinformatics, pp. 14–19 (2003)
11. Yu, H., Hsieh, C., Chang, K., Lin, C.: Large linear classification when data cannot fit in memory. ACM Trans. Knowl. Discov. Data **5**(4), 23: 1–23: 23 (2012)
12. Doan, T.N., Do, T.N., Poulet, F.: Large scale classifiers for visual classification tasks. Multimedia Tools Appl. **74**(4), 1199–1224 (2015)
13. Poulet, F., Do, T.N.: Mining very large datasets with support vector machine algorithms. In: Camp, O., Filipe, J., Hammoudi, S., Piattini, M. (eds.) Enterprise Information Systems V, pp. 177–184 (2004)
14. Do, T.N., Poulet, F.: Classifying one billion data with a new distributed svm algorithm. In: RIVF, pp. 59–66 (2006)
15. Do, T.N., Nguyen, V.H.: A novel speed-up svm algorithm for massive classification tasks. In: IEEE International Conference on Research, Innovation and Vision for the Future, RIVF 2008, pp. 215–220. IEEE (2008)
16. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. In: proceedings of the 17th International Conference on Machine Learning, pp. 999–1006. ACM (2000)
17. Do, T.N., Poulet, F.: Mining very large datasets with SVM and visualization. In: proceedings of 7th International Conference on Entreprise Information Systems, pp. 127–134 (2005)
18. Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast kernel classifiers with online and active learning. J. Mach. Learn. Res. **6**, 1579–1619 (2005)
19. Do, T.N., Le Thi, H.A.: Massive classification with support vector machines. In: Nguyen, N.T. (ed.) Transactions on Computational Collective Intelligence XVIII. LNCS, vol. 9240, pp. 147–165. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48145-5_8
20. Segata, N., Blanzieri, E.: Fast local support vector machines for large datasets. In: Perner, P. (ed.) MLDM 2009. LNCS (LNAI), vol. 5632, pp. 295–310. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03070-3_22

21. Do, T.-N.: Non-linear classification of massive datasets with a parallel algorithm of local support vector machines. In: Le Thi, H.A., Nguyen, N.T., Do, T.V. (eds.) Advanced Computational Methods for Knowledge Engineering. AISC, vol. 358, pp. 231–241. Springer, Heidelberg (2015)
22. Do, T.-N., Poulet, F.: Random local SVMs for classifying large datasets. In: Dang, T.K., Wagner, R., Küng, J., Thoai, N., Takizawa, M., Neuhold, E. (eds.) FDSE 2015. LNCS, vol. 9446, pp. 3–15. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26135-5_1
23. 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)
24. 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 (2008)
25. 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)
26. Do, T.N.: Parallel multiclass stochastic gradient descent algorithms for classifying million images with very-high-dimensional signatures into thousands classes. Vietnam J. Comput. Sci. **1**(2), 107–115 (2014)
27. Do, T.-N., Poulet, F.: Parallel multiclass logistic regression for classifying large scale image datasets. In: Le Thi, H.A., Nguyen, N.T., Do, T.V. (eds.) Advanced Computational Methods for Knowledge Engineering. AISC, vol. 358, pp. 255–266. Springer, Heidelberg (2015)
28. Li, F.F., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. Comput. Vis. Image Underst. **106**(1), 59–70 (2007)
29. Griffin, G., Holub, A., Perona, P.: Caltech-256 Object Category Dataset. Technical Report CNS-TR-2007-001. California Institute of Technology (2007)
30. Deng, J., Berg, A.C., Li, K., Li, F.F.: What does classifying more than 10, 000 image categories tell us? In: European Conference on Computer Vision, pp. 71–84 (2010)
31. Doan, T.-N., Do, T.-N., Poulet, F.: Large scale image classification with many classes, multi-features and very high-dimensional signatures. In: Nguyen, N.T., van Do, T., Thi, H.A. (eds.) ICCSAMA 2013. SCI, vol. 479, pp. 105–116. Springer, Heidelberg (2013)
32. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, New York (2000)
33. Ben-Akiva, M., Lerman, S.: Discrete Choice Analysis: Theory and Application to Travel Demand. The MIT Press, Cambridge (1985)
34. 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)
35. Guermeur, Y.: VC theory of large margin multi-category classifiers. J. Mach. Learn. Res. **8**, 2551–2594 (2007)
36. Kreßel, U.: Pairwise classification and support vector machines, Advances in Kernel Methods: Support Vector Learning, pp. 255–268 (1999)
37. Vural, V., Dy, J.: A hierarchical method for multi-class support vector machines. In: Proceedings of the Twenty-First International Conference on Machine Learning, pp. 831–838 (2004)

38. Benabdeslem, K., Bennani, Y.: Dendogram-based svm for multi-class classification. J. Comput. Inf. Technol. **14**(4), 283–289 (2006)
39. Do, T.N., Lenca, P., Lallich, S.: Classifying many-class high-dimensional fingerprint datasets using random forest of oblique decision trees. Vietnam J. Comput. Sci. **2**(1), 3–12 (2015)
40. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**(4), 1871–1874 (2008)
41. Chang, C.C., Lin, C.J.: LIBSVM : a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(27), 1–27 (2011)
42. Japkowicz, N. (ed.): AAAI'Workshop on Learning from Imbalanced Data Sets. Number WS-00-05 in AAAI Tech Report (2000)
43. Weiss, G.M., Provost, F.: Learning when training data are costly: the effect of class distribution on tree induction. J. Artif. Intell. Res. **19**, 315–354 (2003)
44. Visa, S., Ralescu, A.: Issues in mining imbalanced data sets - a review paper. In: Midwest Artificial Intelligence and Cognitive Science Conference, Dayton, USA, pp. 67–73 (2005)
45. 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)
46. Liu, X.Y., Wu, J., Zhou, Z.H.: Exploratory undersampling for class-imbalance learning. IEEE Trans. Syst. Man Cybern. Part B **39**(2), 539–550 (2009)
47. Ricamato, M.T., Marrocco, C., Tortorella, F.: Mcs-based balancing techniques for skewed classes: an empirical comparison. In: ICPR, pp. 1–4 (2008)
48. Domingos, P.: Metacost: a general method for making classifiers cost sensitive. In: International Conference on Knowledge Discovery and Data Mining, pp. 155–164 (1999)
49. Zhou, Z.H., Liu, X.Y.: On multi-class cost-sensitive learning. In: 21st National Conference on Artificial Intelligence, Boston, MA, USA, pp. 567–572 (2006)
50. Wang, B.X., Japkowicz, N.: Boosting support vector machines for imbalanced data sets. Knowl. Inf. Syst. **25**(1), 1–20 (2010)
51. Cotter, A., Shamir, O., Srebro, N., Sridharan, K.: Better mini-batch algorithms via accelerated gradient methods. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, pp. 1647–1655 (2011)
52. Li, M., Zhang, T., Chen, Y., Smola, A.J.: Efficient mini-batch training for stochastic optimization. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, pp. 661–670 (2014)
53. Franc, V., Sonnenburg, S.: Optimized cutting plane algorithm for large-scale risk minimization. J. Mach. Learn. Res. **10**, 2157–2192 (2009)
54. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 480–492 (2012)
55. 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)
56. Berg, A., Deng, J., Li, F.F.: Large scale visual recognition challenge 2010, Technical report (2010)
57. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, USENIX Association (2010)

Acknowledgement

This work is supported by EUH2020 project-AniAge (691215)