Parallel Learning of Local SVM Algorithms for Classifying Large Datasets

Thanh-Nghi Do[1,2](✉) and François Poulet[3]

[1] College of Information Technology, Can Tho University, Can Tho 92100, Vietnam
dtnghi@cit.ctu.edu.vn
[2] UMI UMMISCO 209, IRD/UPMC, Paris, France
[3] University of Rennes I - IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
francois.poulet@irisa.fr

Abstract. We propose new parallel learning algorithms of local support vector machines (local SVMs) for effectively non-linear classification of large datasets. The algorithms of local SVMs perform the training task of large datasets with two main steps. The first one is to partition the full dataset into k subsets of data, and then the second one is to learn non-linear SVMs from k subsets to locally classify them in parallel way on multi-core computers. The k local SVMs algorithm (kSVM) uses $k$means clustering algorithm to partition the data into k clusters, then constructs in parallel non-linear SVM models to classify data clusters locally. The decision tree with labeling support vector machines (tSVM) uses C4.5 decision tree algorithm to split the full dataset into terminal-nodes, and then it learns in parallel local SVM models for classifying impurity terminal-nodes with mixture of labels. The krSVM algorithm is to train random ensemble of kSVM. The numerical test results on 4 datasets from UCI repository, 3 benchmarks of handwritten letters recognition and a color image collection of one-thousand small objects show that our proposed algorithms of local SVMs (kSVM, tSVM, krSVM) are efficient compared to the standard SVM (LibSVM) in terms of training time and accuracy for dealing with large datasets.

Keywords: Support vector machines · Local support vector machines · Large-scale non-linear classification

1 Introduction

In recent years, the size of data stored electronically is constantly growing due to the increasing number of internet users, more and more mobile access to internet. For example, there are 1.04 billion daily active users on Facebook. 600,000 h of video are uploaded every day on YouTube while 46,000 years are viewed at the same time. And they are not the most visited web sites (Amazon.com and Yahoo! are on top of them). As almost all the mobile phone can take photos, it is estimated that at least 2 trillions photos will be shared on various web sites this year! There are 310 millions people who use Twitter and 600 millions who use Weibo (the "Chinese Twitter"). So there are more and more Internet users generating more and more data. Furthermore the size of the data stored is increasing too, photos have always higher resolution, videos are available in HD with sound in Dolby 5.1, text messages are replaced by voice messages... With such a huge amount of data, it is a big priority to have classification algorithms able to help one in order to find what he is looking for. That is why we present parallel learning algorithms of local Support Vector Machines (SVM) for the classification of very large datasets.

The SVM algorithm proposed by [1] and kernel-based methods have shown practical relevance for classification, regression and novelty detection. Successful applications of SVMs have been reported for various fields like face identification, text categorization and bioinformatics [2]. They become increasingly popular data analysis tools. In spite of the prominent properties of SVM, they are not favorable to deal with the challenge of large datasets. SVM solutions are obtained from quadratic programming (QP), so that the computational cost of a SVM approach [3] is at least square of the number of training datapoints and the memory requirement makes SVM impractical. There is a need to scale up learning algorithms to handle massive datasets on personal computers (PCs).

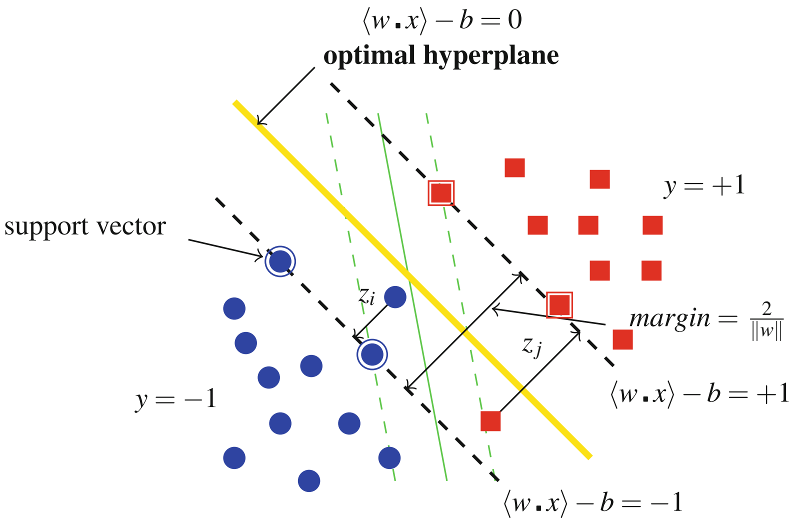
In this extended version of [4,5], we propose new parallel learning algorithms of local SVMs for effective non-linear classification of large datasets. Instead of building a global SVM model, as done by the classical algorithm which is very difficult to deal with large datasets, our algorithms of local SVMs construct an ensemble of local ones that are easily trained by the standard SVM algorithms. The algorithms of local SVMs perform the training task with two main steps. Firstly, the algorithms of local SVMs partition the full dataset into k subsets of data, then the algorithms learn non-linear SVMs from k subsets to locally classify them in the parallel way on multi-core computers. The k local SVMs algorithm (kSVM) uses $k$means clustering algorithm [6] to partition the data into k clusters, and then it constructs in parallel non-linear SVM models to locally classify data clusters. The decision tree with labelling support vector machines (tSVM) uses C4.5 decision tree algorithm [7] to split the full dataset into terminal-nodes, and then it learns in parallel local SVM models for classifying impurity terminal-nodes with mixture of labels. The krSVM algorithm is to train random ensemble of kSVM. The experimental results on 4 datasets from UCI repository [8], 3 benchmarks of handwritten letters recognition [9], MNIST [10,11] and a color image collection of one-thousand small objects, ALOI [12] show that our proposed algorithms (kSVM, tSVM, krSVM) are faster than the standard SVM (LibSVM [13]) in the non-linear classification of large datasets while maintaining high classification accuracy.

The paper is organized as follows. Section 2 briefly introduces the SVM algorithm. Section 3 presents our proposed parallel algorithm of random local SVM for the non-linear classification of large datasets. Section 4 shows the experimental results. Section 5 discusses about related works. We then conclude in Sect. 6.

2 Support Vector Machines

2.1 Support Vector Machines for Binary Classification Problems

Let us consider a linear binary classification task, as depicted in Fig. 1, with m datapoints x_i $(i = 1, \ldots, m)$ in the n-dimensional input space R^n, having corresponding labels $y_i = \pm 1$. For this problem, the SVM algorithms [1] 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 can simply maximize the distance or the margin between the supporting planes for each class ($x.w - b = +1$ for class $+1$, $x.w - b = -1$ for class -1)[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, denoted by z_i ($z_i \geq 0$)[2]. Therefore, SVM has to simultaneously maximize the margin and minimize the error. The standard SVMs pursue these goals with the quadratic programming (1).



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

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

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

[1] The inner dot product of two vectors x and y is denoted by $x.y$.

[2] If the point x_i falling on the right side of its supporting plane then its corresponding $z_i = 0$.

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 function of a new datapoint x based on the plane is:

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

With the mathematical programming concept of duality, the primal quadratic programming problem in (1) is reformulated in the Lagrangian dual one (3) as follows:

$$\begin{aligned} &min\ \Phi(\alpha) = (1/2)\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K\langle x_i, x_j\rangle - \sum_{i=1}^{m}\alpha_i \\ &s.t. \begin{cases} \sum_{i=1}^{m} y_i\alpha_i = 0 \\ 0 \le \alpha_i \le C \quad \forall i = 1, 2, ..., m \end{cases} \end{aligned} \tag{3}$$

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

The support vectors (for which the Lagrange multipliers $\alpha_i > 0$) are given by the solution of the quadratic program (3), and then, the separating surface and the scalar b are determined by the support vectors. The classification of a new data point x based on the SVM model is as follows:

$$predict(x) = sign(\sum_{i=1}^{\#SV} y_i\alpha_i K\langle x, x_i\rangle - b) \tag{4}$$

Variations on SVM algorithms use different classification functions [14]. No algorithmic changes are required from the usual kernel function $K\langle x_i, x_j\rangle$ as a linear inner product, $K\langle x_i, x_j\rangle = \langle x_i.x_j\rangle$ other than the modification of the kernel function evaluation. We can get different support vector classification models. There are two other popular non-linear kernel functions as follows:

- a polynomial function of degree d: $K\langle x_i, x_j\rangle = (\langle x_i.x_j\rangle + 1)^d$
- a RBF (Radial Basis Function): $K\langle x_i, x_j\rangle = e^{-\gamma\|x_i - x_j\|^2}$

SVMs are accurate models with practical relevance for classification, regression and novelty detection. Successful applications of SVMs have been reported for such various fields including facial recognition, text categorization and bioinformatics [2].

2.2 Support Vector Machines for Multi-class Problems

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, $c \ge 3$) 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 [15,16]. The second one is to decompose multi-class into a series of binary SVMs, including one-versus-all [1], one-versus-one [17], Decision Directed Acyclic Graph [18] and hierarchical methods for multi-class SVM [19–21] (hierarchically partitioning the data into two subsets).

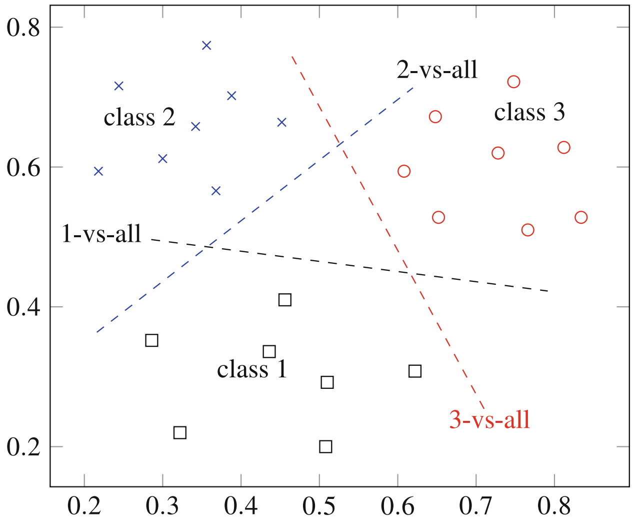


Fig. 2. Multi-class SVM (One-Versus-All)

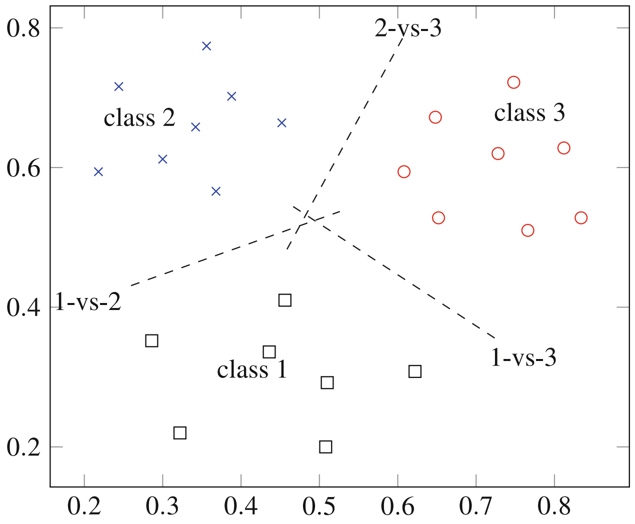


Fig. 3. Multi-class SVM (One-Versus-One)

In practice, the most popular methods are One-Versus-All (ref. LIBLINEAR [22]), One-Versus-One (ref. LibSVM [13]) 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.

3 Parallel Learning Algorithms of Local Support Vector Machines

The study in [3] illustrated that the computational cost requirements of the SVM solutions in (1) or (3) are at least $O(m^2)$ (where m is the number of training datapoints), making standard SVM intractable for large datasets. Learning a global SVM model on the full massive dataset is challenge due to the very high computational cost and the very large memory requirement.

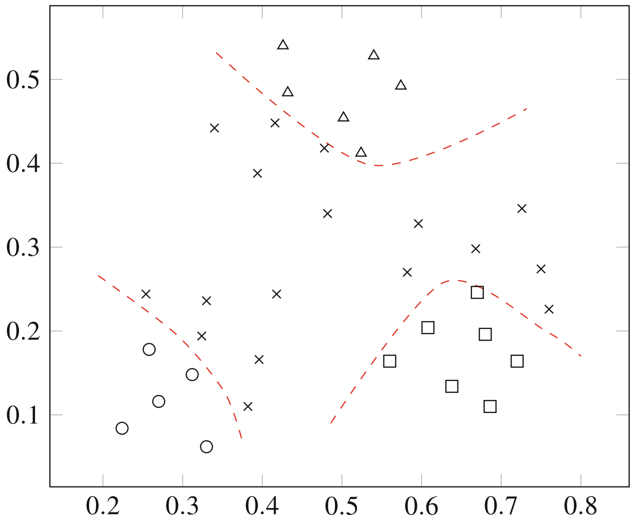
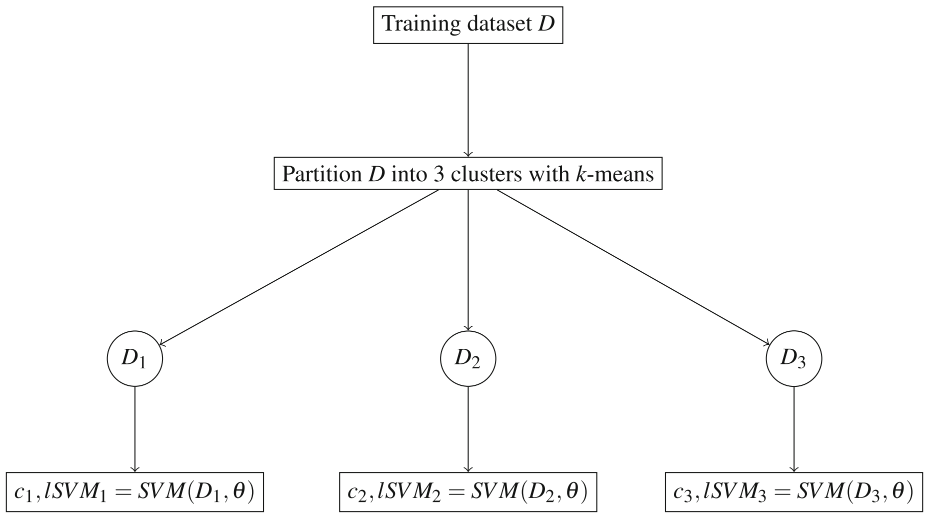


Fig. 4. Global SVM classification model

3.1 *k* Local Support Vector Machines (*k*SVM)

Instead of learning a global SVM model, as done by the classical algorithm which is very difficult to deal with large datasets, our proposed local SVMs algorithms learn local SVMs that are easily trained by the standard SVM algorithms. The local SVMs perform the classification task on large datasets into two main steps. The first one is to partition the full dataset into k subsets of data and then the second one is to learn a non-linear SVM in each subset of data to classify the data locally in parallel way on multi-core computers.

Learning k Local SVMs
Our kSVM algorithm proposed in [4] uses a clustering algorithm to partition the full datasets into k clusters and then it is easy to learn a local non-linear SVM (denoted by $lSVM_i$ with the hyper-parameters θ used to train a non-linear SVM model $lSVM_i$) in each cluster (denoted by D_i) to classify the data locally. Figures 4 and 5 show the comparison between a global SVM model and 3 local SVM models, using a non-linear RBF kernel function with $\gamma = 10$ and a positive


Training dataset D
Partition D into 3 clusters with k-means
D_1
D_2
D_3
$c_1, lSVM_1 = SVM(D_1, \theta)$
$c_2, lSVM_2 = SVM(D_2, \theta)$
$c_3, lSVM_3 = SVM(D_3, \theta)$

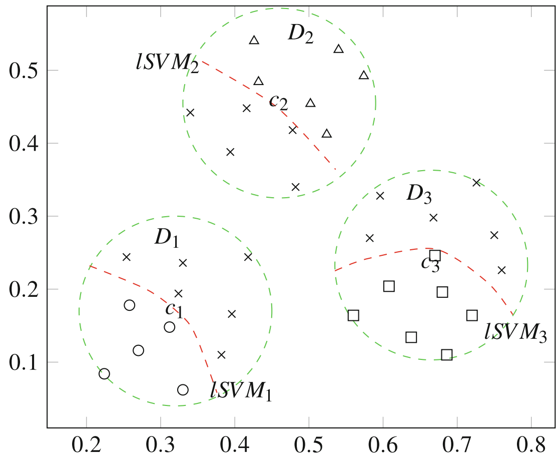


Fig. 5. Three local SVMs classification model (kSVM)

constant $C = 10^6$ (i.e. the hyper-parameters $\theta = \{\gamma, C\}$). In practice, k-means algorithm [6] is the most widely used partitional clustering algorithm because it is simple, easily understandable, and reasonably scalable [23]. Therefore, we propose to use k-means algorithm to partition the full dataset into k clusters and the standard SVM (e.g. LibSVM [13]) to learn k local SVMs.

Furthermore, the kSVM algorithm learns independently k local models from k clusters. This is a nice property for parallel learning. The parallel kSVM does take into account the benefits of high performance computing, e.g. multi-core computers or grids. The simplest development of the parallel kSVM algorithm is based on the shared memory multiprocessing programming model OpenMP [24] on multi-core computers. The parallel training of kSVM is described in Algorithm 1.

Algorithm 1. Local SVM algorithm (kSVM)

```
input :
        training dataset $D$
        number of local models $k$
        hyper-parameter of RBF kernel function $\gamma$
        $C$ for tuning margin and errors of SVMs
output:
        $k$ local support vector machines models
1  begin
2      /*$k$-means performs the data clustering on $D$;*/
3      creating $k$ clusters denoted by $D_1, D_2, \ldots, D_k$ and
4      their corresponding centers $c_1, c_2, \ldots, c_k$
5      #pragma omp parallel for
6      for $i \leftarrow 1$ to $k$ do
7          /*learning a local SVM model from $D_i$;*/
8          $lSVM_i = SVM(D_i, \gamma, C)$
9      end
10     return $kSVM-model = \{(c_1, lSVM_1), (c_2, lSVM_2), \ldots, (c_k, lSVM_k)\}$
11 end
```

Prediction of a New Individual Using kSVM Model

The $kSVM-model = \{(c_1, lSVM_1), (c_2, lSVM_2), \ldots, (c_k, lSVM_k)\}$ is used to predict the class of a new individual x as follows. The first step is to find the closest cluster based on the distance between x and the cluster centers:

$$c_{NN} = \arg\min_{c} \; distance(x, c) \tag{5}$$

And then, the class of x is predicted by the local SVM model $lSVM_{NN}$ (corresponding to c_{NN}):

$$predict(x, kSVMmodel) = predict(x, lSVM_{NN}) \tag{6}$$

Performance Analysis of kSVM

Algorithmic Complexity. Let us now examine the complexity of building k local SVM models with the parallel kSVM algorithm. The full dataset with m individuals is partitioned into k balanced clusters (the cluster size is about $\frac{m}{k}$). The training complexity of a local SVM is $O((\frac{m}{k})^2)$. Therefore, the complexity of parallel training k local SVM models on a P-core processor is $O(\frac{k}{P}(\frac{m}{k})^2) = O(\frac{m^2}{kP})$. This complexity analysis illustrates that parallel learning k local SVM models in the kSVM algorithm [3] is kP times faster than building a global SVM model (the complexity is at least $O(m^2)$).

Generalization Capacity. Let us turn back to Theorem 5.2 proposed by Vapnik [25].

Theorem 5.2 ([25] p. 139). If training sets containing m examples are separated by the maximal margin hyperplanes, the expectation (over training sets) of the probability of test error is bounded by the expectation of the minimum of three values: the ratio $\frac{sv}{m}$, where sv is the number of support vectors, the ratio $\frac{1}{m}\frac{R^2}{\Delta}$, where R is the radius of the sphere containing the data and Δ is the value of the margin, and the ratio $\frac{n}{m}$, where n is the dimensionality of the input space:

$$EP_{error} \leq E\left\{min\left(\frac{sv}{m}, \frac{1}{m}\left[\frac{R^2}{\Delta}\right], \frac{n}{m}\right)\right\} \tag{7}$$

Theorem 5.2 illustrates that the maximal margin hyperplane found by the minimization of $\left[\frac{R^2}{\Delta}\right]$ can generalize well. It means that the generalization ability of the large margin hyperplane is high.

In the kSVM, the full dataset with m datapoints is partitioned into k clusters (the cluster size m_k is about $\frac{m}{k}$). Here, the index notation k is used to present m in the context of the cluster (subset). And then the expectation of the probability of test error for a local SVM model (learnt from a cluster) is bounded by:

$$EP_{error} \leq E\left\{min\left(\frac{sv_k}{m_k}, \frac{1}{m_k}\left[\frac{R_k^2}{\Delta_k}\right], \frac{n}{m_k}\right)\right\} \tag{8}$$

The generalization analysis starts with the comparison between the margin size of the global SVM model for the full dataset and the local SVM model learnt from a cluster illustrated in Theorem 1.

Theorem 1. Given a dataset with m datapoints $X = \{x_1, x_2, \ldots, x_m\}$ in the n-dimensional input space R^n, having corresponding labels $Y = \{y_1, y_2, \ldots, y_m\}$ being ± 1, a maximal margin Δ_X hyperplane is to separate furthest from both class $+1$ and class -1, there exists a maximal margin Δ_{X_k} hyperplane for separating a subset of m_k datapoints $X_k \subset X$ into two classes so that the inequality $\Delta_{X_k} \geq \Delta_X$ holds.

[3] It must be noted that the complexity of the kSVM approach does not include the k-means clustering used to partition the full dataset. But this step requires insignificant time compared with the quadratic programming solution.

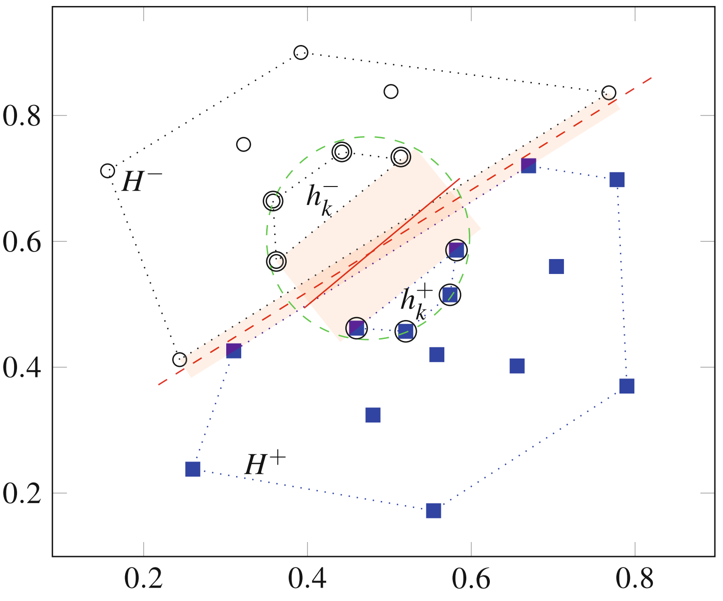


Fig. 6. The comparison of the maximal margin hyperplane of the global SVM and the local SVM

Proof. We remark that the maximal margin $\varDelta_X$ hyperplane can be seen as the minimum distance between two convex hulls, $H+$ of the positive class P and $H-$ of the negative class N (the farthest distance between the two classes, illustrated in Fig. 6). For subset $X_k \subset X$ containing the subset of the positive class $P_k \subset P$ and the subset of the negative class $N_k \subset N$, it leads to the reduced convex hull h_k+ of $H+$ for the positive class and the reduced convex hull h_k- of $H-$ for the negative class. And then the minimum distance between h_k+ and h_k- can not be smaller than between $H+$ and $H-$. It means that the maximal margin $\varDelta_{X_k}$ hyperplane for X_k is larger than the maximal margin $\varDelta_X$ one for fullset X.

The classification performance of a local SVM model in the kSVM is studied in term $\frac{1}{m_k}\left[\frac{R_k^2}{\varDelta_k}\right]$ of Eq. 8. In the comparison with the global SVM constructed for the full dataset X, a local SVM model using a subset $X_k \subset X$ of m_k datapoints can guarantee the classification performance because there exists a compromise between the locality (the subset size, i.e. $R_k \leq R$ and $m_k \leq m$) and the generalized capacity (the margin size, i.e. consequence of Theorem 1 $\varDelta_{X_k} \geq \varDelta_X$).

The generalization analysis illustrates that the parameter k is used in the kSVM to give a trade-off between the generalisation capacity and the computational cost. In [26–28], Vapnik and his colleague also point out the trade-off between the capacity of the local learning system and the number of available individuals. In the context of k local SVM models, this can be understood as follows:

– If k is large then the kSVM algorithm reduces significant training time (the complexity of kSVM is $O(\frac{m^2}{k})$). And then, the size of a cluster is small; the locality is extreme with a very low generalisation capacity.
– If k is small then the kSVM algorithm reduces insignificant training time. However, the size of a cluster is large; it improves the generalisation capacity.

It leads to set k so that the cluster size is large enough (e.g. 200 proposed by [27]).

3.2 Decision Tree Using Labeling Support Vector Machines (tSVM)

Learning tSVM Model
Our proposed tSVM algorithm uses a decision tree algorithm (e.g. C4.5 [7]) to split the full datasets into k partitions. The recursive splitting process of C4.5 algorithm can be early stopped if the number of individuals of the partition is less than the parameter value $minobj$ (the minimum number of individuals that must exist in a node in order for a split to be tried). For the purity terminal-node in which almost all individuals have the same class, the majority label is assigned without any training local SVM models. For the impurity terminal-nodes with mixture of labels, it is to train non-linear SVM models to locally classify these impurity terminal-nodes in the parallel way on multi-core processors (described in Algorithm 2). Figure 7 shows the tSVM model for classifying the same dataset in Fig. 4, using $minobj = 7$ for early stopping the splitting process of C4.5 and a non-linear RBF kernel function with $\gamma = 10$ and a positive constant $C = 10^6$.

Prediction of a New Individual Using tSVM Model
The classification of a new individual x pushes x down the tree t of the tSVM model from the root to the terminal-node. If x arrives at the purity terminal-node then the class of x is the plurality label of this terminal-node; else (x arrives at the impurity terminal-node) the class of x is predicted by the local SVM model learnt from this terminal-node.

Performance Analysis of tSVM
Suppose that the full dataset with m individuals is split into k balanced terminal-nodes (the terminal-node size is about $\frac{m}{k}$; in other words $minobj = \frac{m}{k}$). The training complexity of a SVM for a terminal-node is $O(minobj^2)$. Therefore, the algorithmic complexity of parallel tSVM on a P-core processor is $O(\frac{k}{P}minobj^2) = O(\frac{k}{P}\frac{m}{k}minobj) = O(\frac{m}{P}minobj)$. This complexity analysis [4] illustrates that the speed-up of parallel learning tSVM on a global SVM is $P\frac{m}{minobj}$ times.

tSVM uses the parameter $minobj$ to give a trade-off between the generalisation capacity and the computational cost.

[4] It must be noted that the algorithmic complexity of the tSVM approach does not include the C4.5 algorithm used to split the full dataset because this step requires insignificant time compared with the quadratic programming solution.

Algorithm 2. Local SVM algorithm (tSVM)

```
input :
        training dataset $D$
        minimum number of individuals $minobj$ for a split to be tried
        hyper-parameter of RBF kernel function $\gamma$
        $C$ for tuning margin and errors of SVMs
output:
        $t$SVM model
1  begin
2      /*Decision tree algorithm partitions the full dataset $D$;*/
3      learning a tree $t$ for splitting the full dataset $D$ into $k$ partitions
4      denoted by $D_1, D_2, \ldots, D_k$ (using the early stopped parameter $minobj$)
5      #pragma omp parallel for
6      for $i \leftarrow 1$ to $k$ do
7          if $D_i$ is impurity then
8              /*learning a local SVM model from the impurity terminal-node
               $D_i$;*/
9              $lSVM_i = SVM(D_i, \gamma, C)$
10         else
11             $lSVM_i$ is assigned the plurality label in $D_i$ without any training
12         end
13     end
14     return $tSVM - model$ = tree $t$ and $\{lSVM_1, lSVM_2, \ldots, lSVM_k\}$
15 end
```

If *minobj* is small then the tSVM algorithm significantly reduces training time. And then, the size of a partition is small, the splitting process of C4.5 gives almost purity partitions. tSVM becomes a decision tree using the plurality labeling rules at terminal-nodes without any training local SVMs. The locality is extreme with a very low generalisation capacity.

If *minobj* is large then the tSVM algorithm reduces insignificant training time. However, the size of a terminal-node is large; It improves the capacity. tSVM is a global SVM when *minobj* reaches m.

It leads to set *minobj* large enough (e.g. 200).

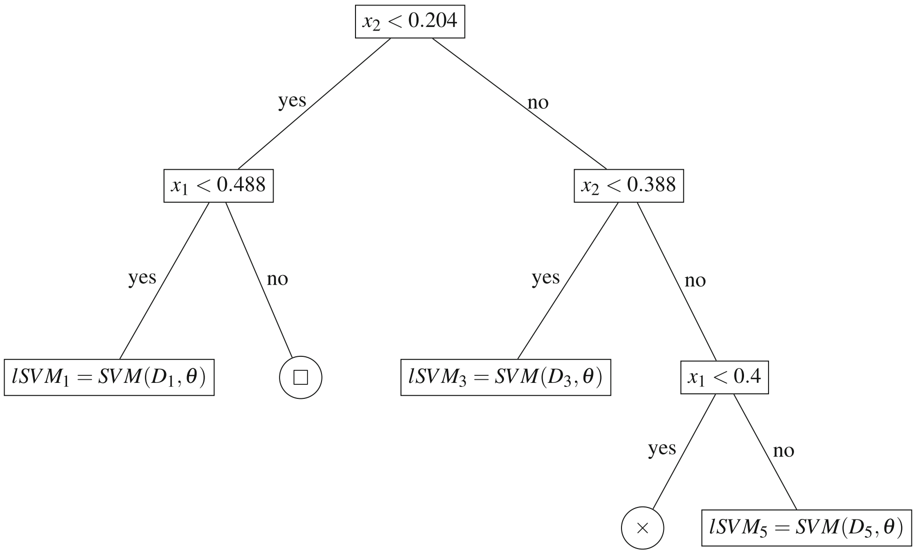
3.3 Ensemble of Random Local SVM Models (*kr*SVM)

The analysis of the trade-off between the generalisation capacity and the computational cost illustrates that the local SVM models tries to speed up the training time of the global SVM model while reducing the generalisation capacity. Due to this problem, we propose to construct the ensemble of random local SVM models to improve the generalisation capacity of the local one. The main idea is based on the random forests proposed by Breiman [29]. The randomization is used for controlling high diversity between local SVM models [5]. It leads to the

[5] Two classifiers are diverse if they make different errors on new data points [30].

improvement of the generalisation capacity of the single one local SVM model. The ensemble of random local SVM (krSVM described in Algorithm 3) creates a collection of T random local SVMs (kSVM described in Algorithm 1) from bootstrap samples (sampling with replacement from the original dataset) using a randomly chosen subset of attributes. Furthermore, the krSVM constructs independently T random local SVM models (kSVM). It allows parallelizing the learning task with OpenMP [24] on multi-core computers. Thus, the complexity of parallel learning krSVM on a P-core processor is $O(T\frac{m^2}{kP})$.

The prediction class of a new individual x is the plurality class of the classification results obtained by T kSVM models.


$x_2 < 0.204$
yes
no
$x_1 < 0.488$
$x_2 < 0.388$
yes
no
yes
no
$lSVM_1 = SVM(D_1, \theta)$
□
$lSVM_3 = SVM(D_3, \theta)$
$x_1 < 0.4$
yes
no
×
$lSVM_5 = SVM(D_5, \theta)$

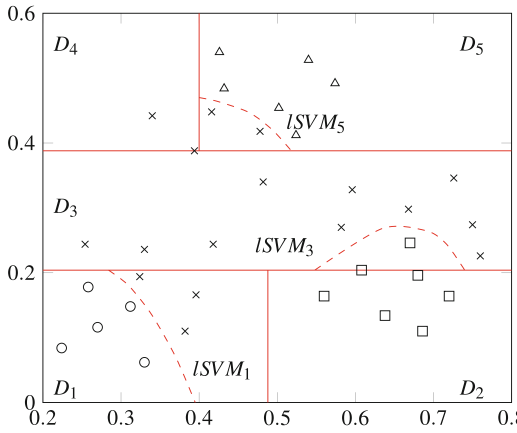


Fig. 7. Decision tree using SVM classifiers at the terminal-nodes (tSVM)

4 Evaluation

We are interested in the performance evaluation of three new parallel learning algorithms of local SVMs for classifying large datasets. Therefore, we conduct the numerical test results to assert their classification performance in terms of correctness and training time, compared to the highly efficient standard SVM library, LibSVM [13].

Algorithm 3. Ensemble of random local SVM (krSVM)

```
input :
        training dataset $D$
        number of $k$SVM models $T$
        $rdims$ random attributes used in the $k$SVM model
        $k$ local models in the $k$SVM model
        hyper-parameter of RBF kernel function $\gamma$
        $C$ for tuning margin and errors of SVMs
output:
        $T$ $k$SVM models
1 begin
2     #pragma omp parallel for
3     for $t \leftarrow 1$ to $T$ do
4         Sampling a bootstrap $D_t$ (train set) from $D$ using $rdims$ random
          attributes)
5         $kSVM_t = kSVM(D_t, k, \gamma, C)$
6     end
7     return $krSVM - model = \{kSVM_1, kSVM_2, \dots, kSVM_T\}$
8 end
```

4.1 Software Programs

In order to evaluate the classification effectiveness of our local parallel learning algorithms of local SVMs, denoted by kSVM (k local SVMs), tSVM (tree local SVMs), krSVM (ensemble of k random local SVMs), we have implemented them in C/C++, OpenMP [24], using the Automatically Tuned Linear Algebra Software (ATLAS [31]), the free source code of decision tree algorithm C4.5 [7] and LibSVM [13].

The comparative studies are reported in terms of accuracy and training time obtained by our proposed algorithms of local SVMs (kSVM, tSVM and krSVM) and LibSVM (standard global SVM).

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

4.2 Datasets

Experiments are conducted with the 4 datasets collected from UCI repository [8] and the 3 benchmarks of handwritten letters recognition, including USPS [9], MNIST [10], a new benchmark for handwritten character recognition [11] and a color image collection of one-thousand small objects, ALOI [12]. Table 1 presents the description of datasets. The evaluation protocols are illustrated in the last column of Table 1. Datasets are already divided into training set (Trn) and testing set (Tst). We used the training data to build the SVM models. Then, we classified the testing set using the resulting models.

Table 1. Description of datasets

ID	Dataset	Individuals	Attributes	Classes	Evaluation protocol
1	Opt. Rec. of Handwritten Digits	5620	64	10	3832 Trn - 1797 Tst
2	Letter	20000	16	26	13334 Trn - 6666 Tst
3	Isolet	7797	617	26	6238 Trn - 1559 Tst
4	USPS Handwritten Digit	9298	256	10	7291 Trn - 2007 Tst
5	A New Benchmark for HCR	40133	3136	36	36000 Trn - 4133 Tst
6	MNIST	70000	784	10	60000 Trn - 10000 Tst
7	ALOI	108000	128	1000	72000 Trn - 36000 Tst
8	Forest Cover Types	581012	54	7	400000 Trn - 181012 Tst

The three last datasets (MNIST, ALOI, Forest Cover Types) in Table 1 yield large-scale non-linear SVM classification problems. Typically, the training time of a non-linear SVM for Forest Cover Types is at least 23 days [32–36].

4.3 Tuning Parameters

We propose to use RBF kernel type in local SVMs and SVM models because it is general and efficient [37]. We also tried to tune the hyper-parameter γ of RBF kernel (RBF kernel of two individuals x_i, x_j, $K[i,j] = exp(-\gamma\|x_i - x_j\|^2)$) and the cost C (a trade-off between the margin size and the errors) to obtain a good accuracy.

For the parameter k local models (number of clusters) of kSVM and krSVM, we propose to set k so that each cluster has about 1000 individuals. The idea gives a trade-off between the generalization capacity [28] and the computational cost.

For the parameter *minobj* in tSVM (the minimum number of individuals that must exist in a node in order for a split to be attempted) is set equal to 1000.

Furthermore, our krSVM uses 20 random k local SVM models with the number of random (*rdims*) attributes being one half full set.

Table 2 presents the hyper-parameters of kSVM, tSVM, krSVM and LibSVM in the classification.

Table 2. Hyper-parameters of kSVM, tSVM, krSVM and LibSVM

ID	Datasets	γ	C	k	$minobj$
1	Opt. Rec. of Handwritten Digits	0.0001	100000	10	1000
2	Letter	0.0001	100000	30	1000
3	Isolet	0.0001	100000	10	1000
4	USPS Handwritten Digit	0.0001	100000	10	1000
5	A New Benchmark for HCR	0.001	100000	50	1000
6	MNIST	0.05	100000	100	1000
7	ALOI	0.01	100000	100	1000
8	Forest Cover Types	0.0001	100000	500	1000

4.4 Classification Results

The classification results of kSVM, tSVM, krSVM and LibSVM on the 8 datasets are given in Tables 3 and 4 and Figs. 8, 9, 10, 11 and 12.

As it was expected, our three algorithms of local SVMs (kSVM, tSVM, krSVM) outperform LibSVM in terms of training time. The average training time kSVM, tSVM, krSVM, LibSVM are 44.63 s, 54.85 s, 75.57 s, at least 4646.98 s, respectively. kSVM is the fastest learning algorithm. tSVM holds the rank 2. krSVM is about 1.7 times slower than kSVM. Training time of LibSVM is at least 61 times longer than krSVM.

Table 3. Training time (s)

ID	Datasets	Training time(s)			
		LibSVM	krSVM	kSVM	tSVM
1	Opt. Rec. of Handwritten Digits	0.58	0.54	0.21	0.12
2	Letter	2.87	1.94	0.5	0.42
3	Isolet	8.37	7.14	2.94	3.98
4	USPS Handwritten Digit	5.88	5.32	3.82	4.62
5	A New Benchmark for HCR	107.07	91.72	35.7	95.37
6	MNIST	1531.06	82.26	45.5	124.48
7	ALOI	2400	142.28	44.68	30
8	Forest Cover Types	NA ($>$ 33120)	273.36	223.7	179.84
Average		NA ($>$ 4646.98)	75.57	44.63125	54.85375

For the 5 first small datasets, the classification results (presented in Tables 3 and 4 and Fig. 8) show that the improvement of local SVM algorithms against LibSVM is slight.

For the 3 last large datasets, the classification performances in Tables 3 and 4 and Figs. 9, 10 and 11 demonstrate that the speed-up in learning of local SVM algorithms on LibSVM is significant.

Table 4. Classification results in terms of accuracy (%)

ID	Datasets	Classification accuracy(%)			
		LibSVM	*kr*SVM	*k*SVM	*t*SVM
1	Opt. Rec. of Handwritten Digits	98.33	97.61	97.05	96.99
2	Letter	97.40	97.16	96.14	95.65
3	Isolet	96.47	96.15	95.44	95.38
4	USPS Handwritten Digit	96.86	96.46	95.86	95.02
5	A New Benchmark for HCR	95.14	94.77	92.98	92.72
6	MNIST	98.37	98.71	98.11	98.24
7	ALOI	95.16	95.09	93.33	93.17
8	Forest Cover Types	NA	97.07	97.06	96.73
Average		96.82	96.63	95.75	95.49

For MNIST dataset, *kr*SVM is 18.61 times faster than LibSVM while maintaining a slight correctness improvement (0.34%). *k*SVM and *t*SVM perform the classification with very competitive correctness compared to LibSVM but their training time are 33 times and 12 times faster than LibSVM.

ALOI dataset has very large number of classes (1000 classes). LibSVM uses the One-Versus-One strategy, therefore it requires 499500 binary SVM models for training and testing. And then, LibSVM classifies this dataset in 2400 s with 95.13% accuracy. *kr*SVM performs the classification task in 142.28 s with 95.09% accuracy. The speed-up of *kr*SVM on LibSVM is about 16 times without loss of the test correctness. *k*SVM is about 53 times faster than LibSVM with loss of 1.83% accuracy. *t*SVM is about 80 times faster than LibSVM with loss of 1.99% accuracy.

Typically, Forest cover type dataset is well-known as a difficult dataset for non-linear SVM [32,33]; LibSVM ran for 23 days without any result. *kr*SVM performs this non-linear classification in 273.36 s with 97.07% accuracy. *k*SVM takes 223.7 s of the training time with the same accuracy as *kr*SVM. *t*SVM achieves an accuracy of 96.73 in 179.84 s training time.

The classification results in terms of test correctness (presented in Table 4 and Fig. 12) show that our algorithms of local SVMs give very competitive correctness compared to LibSVM. *k*SVM, *t*SVM, *kr*SVM and LibSVM achieve the average accuracy of 95.75%, 95.49%, 96.63%, 96.82%, respectively. The superiority of LibSVM on *kr*SVM corresponds to 0.19%. The classification results show that *kr*SVM has an improvement of 0.88% compared to *k*SVM and of 1.14% compared to *t*SVM.

5 Discussion on Related Works

Our proposal is related to large-scale SVM learning algorithms. The improvements of SVM training on very large datasets include effective heuristic methods

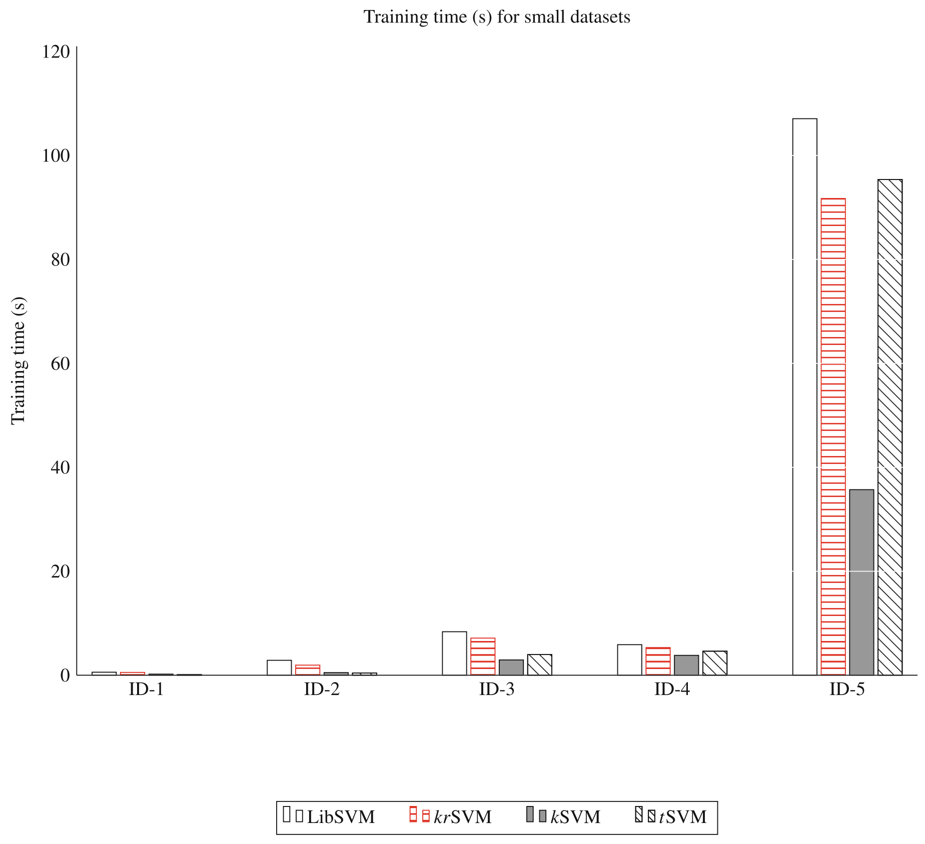


Fig. 8. Comparison of training time on small datasets

in the decomposition of the original quadratic programming into series of small problems [3,38,39] and [13].

Mangasarian and his colleagues proposed to modify SVM problems to obtain new formulas, including Lagrangian SVM [40], proximal SVM [41], Newton SVM [42]. The Least Squares SVM proposed by Suykens and Vandewalle [43] changes standard SVM optimization to lead the new efficient SVM solver. And then, these algorithms only require solving a system of linear equations instead of a quadratic programming. This makes training time very short for linear classification tasks. Their extensions proposed by [33,44–47] aim at improving memory performance for massive datasets by incrementally updating solutions in a growing training set without needing to load the entire dataset into memory at once. More recent [48, 49] proposed the stochastic gradient descent methods for dealing with large scale linear SVM solvers. The parallel and distributed algorithms [45,47,50,51] for the linear classification improve learning performance for large datasets by dividing the problem into sub-problems that execute on large numbers of networked PCs, grid computing, multi-core computers. Parallel SVMs proposed by [52] use GPU

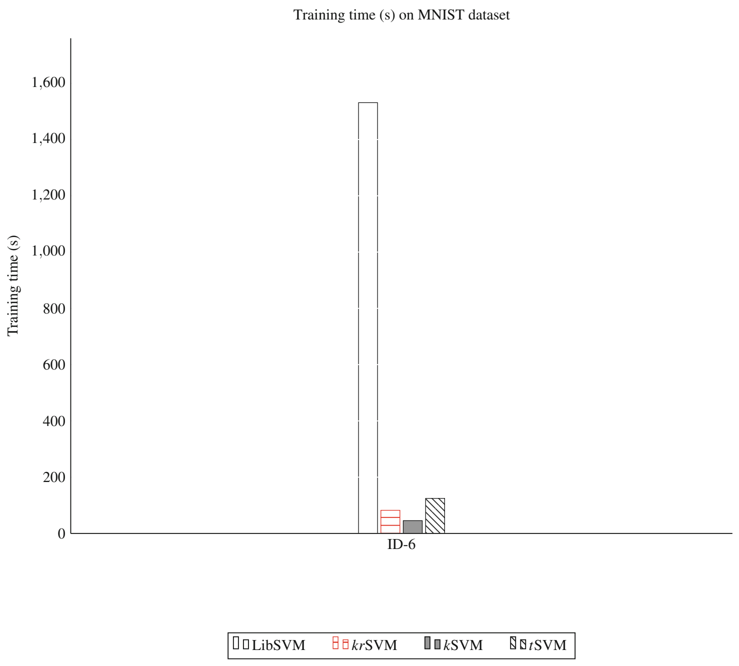


Fig. 9. Comparison of training time on MNIST dataset

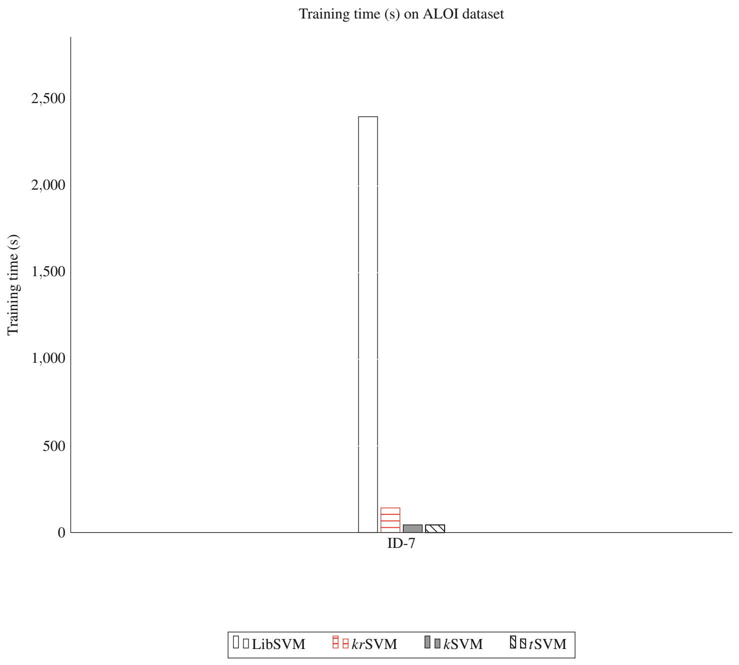


Fig. 10. Comparison of training time on ALOI dataset

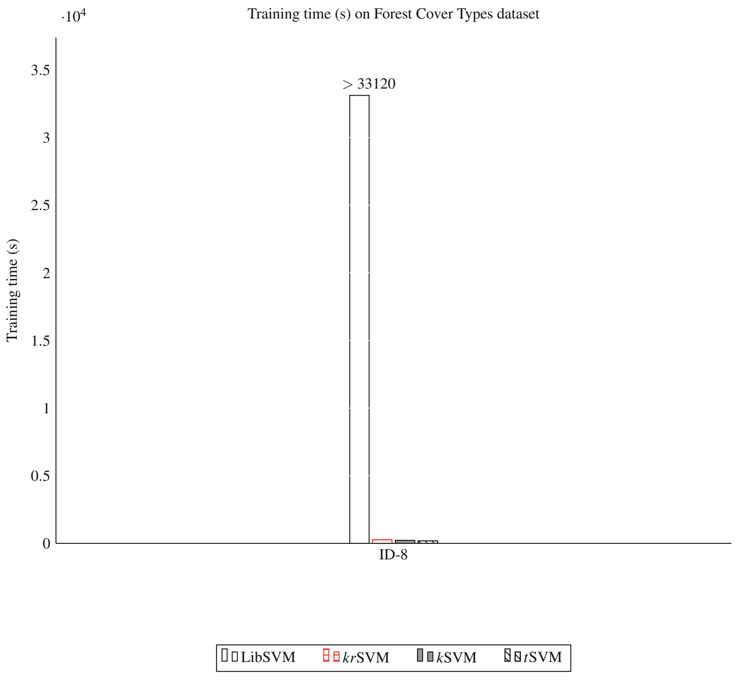


Fig. 11. Comparison of training time on Forest Cover Types dataset

to speed-up training tasks. These algorithms are efficient for linear classification tasks.

Active SVM learning algorithms proposed by [32,53–55] choose interesting datapoint subsets (active sets) to construct models, instead of using the whole dataset. SVM algorithms [33,56–58] use the boosting strategy [59,60] for the linear classification of very large datasets on standard PCs.

Recently, the review paper [61] provides a comprehensive survey on large-scale linear support vector classification. The parallel algorithms of logistic regression and linear SVM using Spark [62] are proposed in [63]. The distributed Newton algorithm of logistic regression [64] is implemented in the Message Passing Interface (MPI). The parallel dual coordinate descent method for linear classification [65] is implemented in multi-core environments using OpenMP. The incremental and decremental algorithms [66] aim at training linear classification of large data that cannot fit in memory. These algorithms are proposed to efficiently deal large-scale linear classification tasks in a very-high-dimensional input space. But the computational cost of a non-linear SVM approach is still impractical. The work in [67] tries to automatically determine which kernel classifiers perform strictly better than linear for a given data set.

Our proposal is in some aspects related to local SVM learning algorithms. The first approach is to classify data in hierarchical strategy. This kind of training algorithm performs the classification task with two main steps. The first

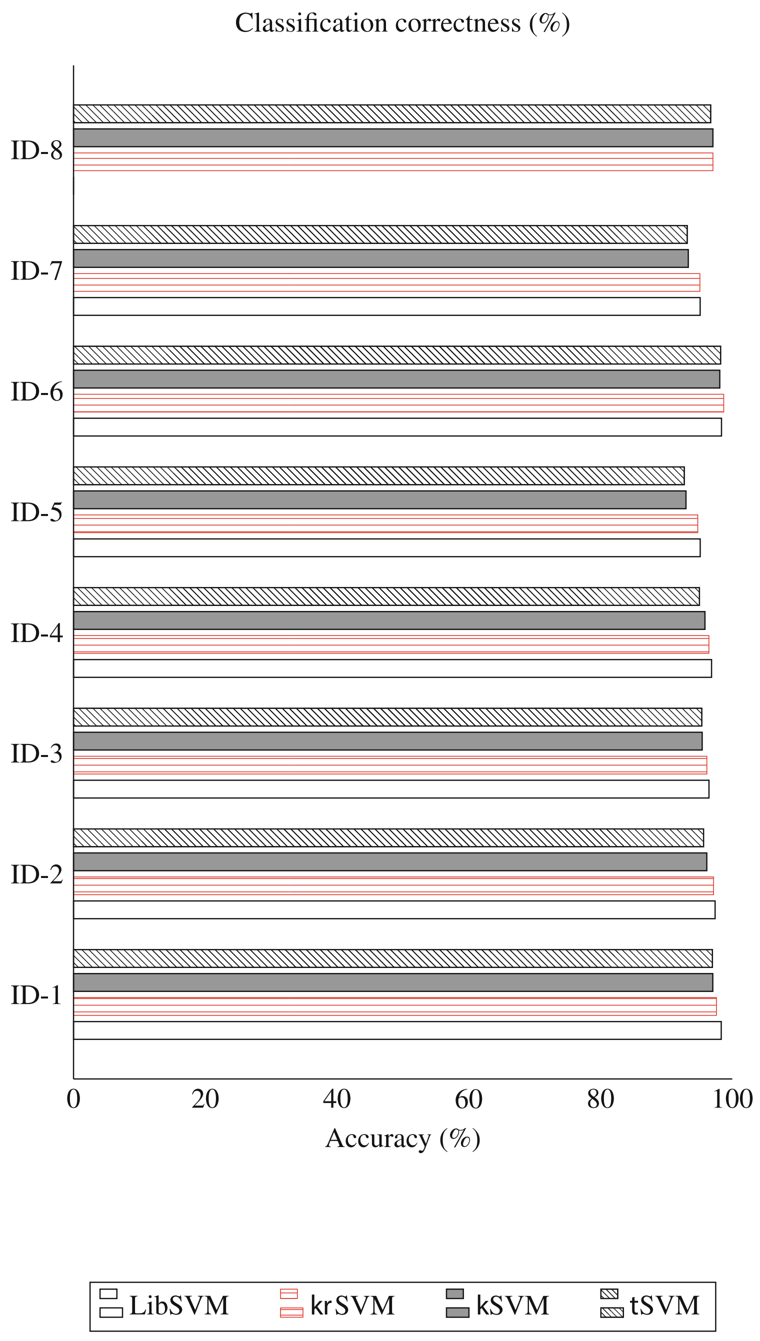


Fig. 12. Comparison of classification results

one is to cluster the full dataset into homogeneous groups (clusters) and then the second one is to learn the local supervised classification models from clusters. The paper [68] proposed to use the expectation-maximization (EM) clustering algorithm [69] for partitioning the training set into k joint clusters (the EM clustering algorithm makes a soft assignment based on the posterior probabilities [70]); for each cluster, a neural network (NN) is learnt to classify the individuals in the cluster. The parallel mixture of SVMs algorithm proposed by [34] constructs local SVM models instead of NN ones in [68]. CSVM [71] uses k-means algorithm [6] to partition the full dataset into k disjoint clusters; then the algorithm learns weighted local linear SVMs from clusters. More recent DTSVM [36,72] uses the decision tree algorithm [7,73] to split the full dataset into disjoint regions (tree leaves) and then the algorithm builds the local SVMs for classifying the individuals in tree leaves. These algorithms aim at speeding up the learning time.

The second approach is to learn local supervised classification models from k nearest neighbors (kNN) of a new testing individual. First local learning algorithm of Bottou and Vapnik [27] find kNN of a test individual; train a neural network with only these k neighborhoods and apply the resulting network to the test individual. k-local hyperplane and convex distance nearest neighbor algorithms are also proposed in [74]. More recent local SVM algorithms aim to use the different methods for kNN retrieval; including SVM-kNN [75] trying different metrics, ALH [76] using the weighted distance and features, FaLK-SVM [35] speeding up the kNN retrieval with the cover tree [77].

A theoretical analysis for such local algorithms discussed in [26] introduces the trade-off between the capacity of learning system and the number of available individuals. The size of the neighborhoods is used as an additional free parameters to control generalisation capacity against locality of local learning algorithms.

6 Conclusion and Future Works

We have presented new parallel learning algorithms of local support vector machines that achieve high performances (in terms of training time and correctness) for the non-linear classification of large datasets. The training task of the local SVMs in kSVM, tSVM and krSVM algorithms is to partition the full data into k subsets of data and then it constructs a non-linear SVM in each subset of data to locally classify them in parallel way. The numerical test results on 4 datasets from UCI repository, 3 benchmarks of handwritten letters recognition and a color image collection of one-thousand small objects showed that our proposed algorithms are efficient in terms of training time and accuracy compared to the standard SVM. An example of their effectiveness is given with the non-linear classification of Forest Cover Types dataset (having 581012 individuals, 54 attributes) into 7 classes, krSVM classifies this dataset in 273.36 s with 97.07% accuracy. kSVM takes 223.7 s of the training time with the same accuracy as krSVM. tSVM achieves an accuracy of 96.73% in 179.84 s training time.

In the near future, we intend to provide more empirical test on large benchmarks and comparisons with other algorithms. A promising future research aims at automatically tuning the hyper-parameters of local SVMs.

References

1. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
2. Guyon, I.: Web page on SVM applications (1999). http://www.clopinet.com/isabelle/Projects/-SVM/app-list.html
3. 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)
4. 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). doi:10.1007/978-3-319-17996-4_21
5. 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
6. 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, January 1967
7. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
8. Lichman, M.: UCI machine learning repository (2013)
9. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
10. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
11. van der Maaten, L.: A new benchmark dataset for handwritten character recognition (2009). http://homepage.tudelft.nl/19j49/Publications_files/characters.zip
12. Geusebroek, J.M., Burghouts, G.J., Smeulders, A.W.M.: The amsterdam library of object images. Intl. J. Comput. Vis. **61**(1), 103–112 (2005)
13. Chang, C.C., Lin, C.J.: LIBSVM : a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(27), 1–27 (2011)
14. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods. Cambridge University Press, New York (2000)
15. 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)
16. Guermeur, Y.: VC theory of large margin multi-category classifiers. J. Mach. Learn. Res. **8**, 2551–2594 (2007)
17. Kreßel, U.: Pairwise classification and support vector machines. In: Advances in Kernel Methods: Support Vector Learning, pp. 255–268 (1999)

18. Platt, J., Cristianini, N., Shawe-Taylor, J.: Large margin dags for multiclass classification. Adv. Neural Inf. Process. Syst. **12**, 547–553 (2000)
19. 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)
20. Benabdeslem, K., Bennani, Y.: Dendogram-based SVM for multi-class classification. J. Comput. Inf. Technol. **14**(4), 283–289 (2006)
21. 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)
22. 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)
23. Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Philip, S.Y., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowl. Inf. Syst. **14**(1), 1–37 (2007)
24. OpenMP Architecture Review Board: OpenMP Application Program Interface Version 3.0 (2008)
25. Vapnik, V.N.: The Nature of Statistical Learning Theory, 2nd edn. Springer, New York (2000)
26. Vapnik, V.: Principles of risk minimization for learning theory. In: Advances in Neural Information Processing Systems 4, NIPS Conference, Denver, Colorado, USA, December 2–5, 1991, pp. 831–838 (1991)
27. Bottou, L., Vapnik, V.: Local learning algorithms. Neural Comput. **4**(6), 888–900 (1992)
28. Vapnik, V., Bottou, L.: Local algorithms for pattern recognition and dependencies estimation. Neural Comput. **5**(6), 893–909 (1993)
29. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
30. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000). doi:10.1007/3-540-45014-9_1
31. Whaley, R., Dongarra, J.: Automatically tuned linear algebra software. In: Ninth SIAM Conference on Parallel Processing for Scientific Computing, CD-ROM Proceedings (1999)
32. Yu, H., Yang, J., Han, J.: Classifying large data sets using SVMs with hierarchical clusters. In: Proceedings of the ACM SIGKDD International Conference on KDD, pp. 306–315. ACM (2003)
33. Do, T.N., Poulet, F.: Towards high dimensional data mining with boosting of PSVM and visualization tools. In: Proceedings of 6th International Conference on Entreprise Information Systems, pp. 36–41(2004)
34. Collobert, R., Bengio, S., Bengio, Y.: A parallel mixture of SVMs for very large scale problems. Neural Comput. **14**(5), 1105–1114 (2002)
35. Segata, N., Blanzieri, E.: Fast and scalable local kernel machines. J. Learn. Res. **11**, 1883–1926 (2010)
36. Chang, F., Guo, C.Y., Lin, X.R., Lu, C.J.: Tree decomposition for large-scale SVM problems. J. Mach. Learn. Res. **11**, 2935–2972 (2010)
37. Lin, C.: A practical guide to support vector classification (2003)
38. 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)

39. Osuna, E., Freund, R., Girosi, F.: An improved training algorithm for support vector machines. In: Principe, J., Gile, L., Morgan, N., Wilson, E. (eds.) Neural Networks for Signal Processing VII, pp. 276–285 (1997)
40. Mangasarian, O., Musicant, D.: Lagrangian support vector machines. J. Mach. Learn. Res. **1**, 161–177 (2001)
41. Fung, G., Mangasarian, O.: Proximal support vector classifiers. In: Proceedings of the ACM SIGKDD International Conference on KDD, pp. 77–86. ACM (2001)
42. Mangasarian, O.: A finite Newton method for classification problems. Technical report, pp. 01–11. Data Mining Institute, Computer Sciences Department, University of Wisconsin (2001)
43. Suykens, J., Vandewalle, J.: Least squares support vector machines classifiers. Neural Process. Lett. **9**(3), 293–300 (1999)
44. Do, T.N., Poulet, F.: Incremental SVM and visualization tools for bio-medical data mining. In: Proceedings of Workshop on Data Mining and Text Mining in Bioinformatics, pp. 14–19 (2003)
45. Do, T.N., Poulet, F.: Classifying one billion data with a new distributed SVM algorithm. In: Proceedings of 4th IEEE International Conference on Computer Science, Research, Innovation and Vision for the Future, pp. 59–66. IEEE Press (2006)
46. Fung, G., Mangasarian, O.: Incremental support vector machine classification. In: Proceedings of the 2nd SIAM International Conference on Data Mining (2002)
47. 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. Springer, Amsterdam (2004)
48. 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)
49. 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
50. 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)
51. Doan, T., Do, T., Poulet, F.: Large scale classifiers for visual classification tasks. Multimedia Tools Appl. **74**(4), 1199–1224 (2015)
52. Do, T.-N., Nguyen, V.-H., Poulet, F.: Speed up SVM algorithm for massive classification tasks. In: Tang, C., Ling, C.X., Zhou, X., Cercone, N.J., Li, X. (eds.) ADMA 2008. LNCS (LNAI), vol. 5139, pp. 147–157. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88192-6_15
53. 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)
54. Boley, D., Cao, D.: Training support vector machines using adaptive clustering. In: Berry, M.W., Dayal, U., Kamath, C., Skillicorn, D.B. (eds.) Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, 22–24 April, 2004, SIAM, pp. 126–137 (2004)
55. 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)

56. Pavlov, D., Mao, J., Dom, B.: Scaling-up support vector machines using boosting algorithm. In: 15th International Conference on Pattern Recognition, vol. 2, pp. 219–222 (2000)
57. Do, T.N., Le-Thi, H.A.: Classifying large datasets with SVM. In: Proceedings of 4th International Conference on Computational Management Science (2007)
58. Do, T.N., Fekete, J.D.: Large scale classification with support vector machine algorithms. In: Wani, M.A., Kantardzic, M.M., Li, T., Liu, Y., Kurgan, L.A., Ye, J., Ogihara, M., Sagiroglu, S., Chen, X.W., Peterson, L.E., Hafeez, K. (eds.) The Sixth International Conference on Machine Learning and Applications, ICMLA 2007, Cincinnati, Ohio, USA, 13–15 December 2007, pp. 7–12. IEEE Computer Society (2007)
59. Freund, Y., Schapire, R.: A short introduction to boosting. J. Jpn. Soc. Artif. Intell. **14**(5), 771–780 (1999)
60. Breiman, L.: Arcing classifiers. Ann. Stat. **26**(3), 801–849 (1998)
61. Yuan, G., Ho, C., Lin, C.: Recent advances of large-scale linear classification. Proc. IEEE **100**(9), 2584–2603 (2012)
62. 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, HotCloud 2010, Berkeley, CA, USA, p. 10. USENIX Association (2010)
63. Lin, C., Tsai, C., Lee, C., Lin, C.: Large-scale logistic regression and linear support vector machines using spark. In: 2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, 27–30 October, 2014, pp. 519–528 (2014)
64. Zhuang, Y., Chin, W.-S., Juan, Y.-C., Lin, C.-J.: Distributed Newton methods for regularized logistic regression. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015. LNCS (LNAI), vol. 9078, pp. 690–703. Springer, Heidelberg (2015). doi:10.1007/978-3-319-18032-8_54
65. Chiang, W., Lee, M., Lin, C.: Parallel dual coordinate descent method for large-scale linear classification in multi-core environments. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13–17, 2016, pp. 1485–1494 (2016)
66. Tsai, C., Lin, C., Lin, C.: Incremental and decremental training for linear classification. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, NY, USA , 24–27 August, 2014, pp. 343–352 (2014)
67. Huang, H., Lin, C.: Linear and kernel classification: when to use which? In: Proceedings of the SIAM International Conference on Data Mining 2016 (2016)
68. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Comput. **3**(1), 79–87 (1991)
69. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc. Ser. B **39**(1), 1–38 (1977)
70. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
71. Gu, Q., Han, J.: Clustered support vector machines. In: Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, 29 April–1 May, 2013, JMLR Proceedings, vol. 31, pp. 307–315(2013)
72. Chang, F., Liu, C.C.: Decision tree as an accelerator for support vector machines. In: Ding, X. (ed.) Advances in Character Recognition. InTech (2012)
73. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.: Classification and Regression Trees. Wadsworth International, Monterey (1984)

74. Vincent, P., Bengio, Y.: K-local hyperplane and convex distance nearest neighbor algorithms. In: Advances in Neural Information Processing Systems, pp. 985–992. The MIT Press (2001)
75. Zhang, H., Berg, A., Maire, M., Malik, J.: SVM-KNN: discriminative nearest neighbor classification for visual category recognition. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2126–2136 (2006)
76. Yang, T., Kecman, V.: Adaptive local hyperplane classification. Neurocomputing **71**(1315), 3001–3004 (2008)
77. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 97–104. ACM (2006)

Acknowledgement

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