UNIVERSITY
OF TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

FAST LOCAL SUPPORT VECTOR MACHINES FOR LARGE DATASETS

Nicola Segata and Enrico Blanzieri

November 2008

Technical Report # DISI-08-063

Fast Local Support Vector Machines for Large Datasets

Nicola Segata
DISI, University of Trento
segata@disi.unitn.it

Enrico Blanzieri
DISI, University of Trento
blanzier@disi.unitn.it

Abstract

Local SVM is a classification method that combines instance-based learning and statistical machine learning. It builds an SVM on the feature space neighborhood of the query point in the training set and uses it to predict its class. There is both empirical and theoretical evidence that Local SVM can improve over SVM and kNN in terms of classification accuracy, but the computational cost of the method permits the application only on small datasets. Here we propose FastLSVM, a classifier based on Local SVM that decreases the number of SVMs that must be built in order to be suitable for large datasets. FastLSVM precomputes a set of local SVMs in the training set and assigns to each model all the points lying in the central neighborhood of the k points on which it is trained. The prediction is performed applying to the query point the model corresponding to its nearest neighbor in the training set. The empirical evaluation we provide points out that FastLSVM is a good approximation of Local SVM and its computational performances on big datasets (a large artificial problem with 100000 samples and on a very large real problem with more than 500000 samples) dramatically ameliorate SVM ones improving also the generalization accuracies.

1 Introduction

The direct integration of k-nearest neighbors (kNN) with support vector machines (SVM) has been proposed in [1]. The algorithm is called kNNSVM, and it builds a maximal margin classifier on the neighborhood of a test sample in the feature space induced by a kernel function. Theoretically, it permits better generalization power since it can have, for some values of k, a lower radius/margin bound with respect to SVM as shown in [2]. It has been successfully applied for remote sensing tasks in [1] and on 13 small benchmark datasets [3], confirming the potentialities of this approach. kNNSVM can be seen as a method of integrating locality in kernel methods (compatible with the traditional strategy of using local kernel functions [4]) and as a modification of the SVM approach in order to obtain a local learning algorithm [5] able to locally adjust the capacity of the training systems.

The main drawback of the original idea of Local SVM concerns the computational performances. The prediction phase is in fact very slow since for each query point it is necessary to train a specific SVM before performing the classification, in addition to the selection of its k-nearest neighbors on which the local SVM is trained. In [6] it has been independently proposed a similar method in which however the distance function for the kNN operations is performed in the input space and it is approximated with a “crude” distance metric in order to improve the computational performances.

In this work we developed a fast local support vector machine classifier, called FastLSVM, introducing various modifications to the Local SVM approach in order to make it scalable and thus suitable for large datasets. Differently from [6] we maintain the feature space metric for the nearest neighbor operations and we do not adopt any approximation on the distance function and thus on the neighborhood selection. We aim, in fact, to be as close as possible to the original formulation of kNNSVM in order to maintain its theoretical and empirical advantages over SVM. Moreover, our intuition is that, in general, as the number of samples in the training size increases, also the positive effect of locality on classification accuracy increases. Roughly speaking, the idea is to precompute a set of local SVMs covering (with redundancy) all the training set and to apply to a query point the model to which its nearest neighbor in the training set has been assigned. The training time complexity analysis reveals that the approach is asymptotically faster than the state-of-the-art accurate SVM solvers and the training of the local models can be very easily paralyzed. Notice that the issue of scalability for the local SVM approach is particularly appealing also because our intuition is that locality can play a more crucial role as the problem becomes larger and larger and the ideal decision function is complex and highly non-linear.

Multiple approaches have been proposed in order to overcome SVM computational limitation for large and very large datasets approximating the traditional approach; some of the more recent are those based on minimum enclosing ball approximated algorithms [7], on using editing or clustering techniques to select the more informative samples [8], on training SVM between clusters of different class nearest to the query point [9] and on parallel algorithms for training phase [10, 11]. It is important to underline, however, that what we are proposing here is not a method to *approximate* SVM in order to enhance performances. Our main purpose is to make kNNSVM, which has been shown to be more accurate of SVM for small datasets, suitable for large scale problems. Indirectly, since the method is asymptotically faster than SVM, it can be seen as an alternative to SVM for large datasets on which traditional SVM algorithms cannot be applied.

An attempt to computationally unburden the Local SVM approach of [6] has been proposed in [12] where the idea is to train multiple SVMs on clusters retrieved with a k-means based algorithm; however, differently from this work the method does not follow directly the idea of kNNSVM, it can build only linear models, the clustering considers together training and testing sets, the neighborhood is retrieved only in input space and the testing point can lie in very peripheral regions of the local models. Moreover the clusters have problems of class balancing and their dimensions cannot be controlled thus not assuring the SVM optimization to be small enough. The computational performances (only empirically tested on a small dataset) are in fact much worse than SVM (although better than their local approach) and seems to decrease asymptotically much faster than SVM.

The present paper discusses a subset of the topics of a more general and empirically tested work[1]. In the rest of the introduction we briefly review the main topics necessary to understand the FastLSVM approach discussed in Section 2. Section 3 details the experiments we conducted before drawing some conclusions and discussing further extensions in Section 4.

[1]The larger work is going to be submitted to a machine learning journal.

1.1 The k-nearest neighbors classifier

Let assume to have a classification problem with samples (x_i, y_i) with $i = 1, \dots, n$, $x_i \in \mathbb{R}^p$ and $y_i \in \{+1, -1\}$. Given a point x', it is possible to order the entire set of training samples X with respect to x'. This corresponds to define a function $r_{x'} : \{1, \dots, n\} \to \{1, \dots, n\}$ that reorders the indexes of the n training points as follows:

$$\begin{cases} r_{x'}(1) = \operatorname*{argmin}_{i=1,\dots,n} \|x_i - x'\| \\ r_{x'}(j) = \operatorname*{argmin}_{i=1,\dots,n} \|x_i - x'\| & \begin{array}{l} i \neq r_{x'}(1), \dots, r_{x'}(j-1) \\ \text{for } j = 2, \dots, n \end{array} \end{cases}$$

In this way, $x_{r_{x'}(j)}$ is the point of the set X in the j-th position in terms of distance from x', namely the j-th nearest neighbor, $\|x_{r_{x'}(j)} - x'\|$ is its distance from x' and $y_{r_{x'}(j)}$ is its class with $y_{r_{x'}(j)} \in \{-1, 1\}$. In other terms: $j < k \Rightarrow \|x_{r_{x'}(j)} - x'\| \leq \|x_{r_{x'}(k)} - x'\|$. With this definition, the majority decision rule of kNN for binary classification is defined by $kNN(x) = \text{sign}(\sum_{i=1}^{k} y_{r_x(i)})$.

1.2 Support vector machines

SVMs [13] are classifiers based on statistical learning theory [14]. The decision rule is $SVM(x) = \text{sign}(\langle w, \Phi(x)\rangle_{\mathcal{H}} + b)$ where $\Phi(x) : \mathbb{R}^p \to \mathcal{H}$ is a mapping in a transformed Hilbert feature space $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. The parameters $w \in \mathcal{H}$ and $b \in \mathbb{R}$ are such that they minimize an upper bound on the expected risk while minimizing the empirical risk. The empirical risk is controlled through the set of constraints $y_i(\langle w, \Phi(x_i)\rangle_{\mathcal{H}} + b \geq 1 - \xi_i$ with $\xi_i \geq 0$, $i = 1, \dots, n$, where $y_i \in \{-1, +1\}$ is the class label of the i-th nearest training sample. The presence of the slack variables ξ_i's allows some misclassification on the training set. Reformulating such an optimization problem with Lagrange multipliers α_i $(i = 1, \dots, n)$, and introducing a positive definite kernel (PD) function[2] $K(\cdot, \cdot)$ that substitutes the scalar product in the feature space $\langle \Phi(x_i), \Phi(x)\rangle_{\mathcal{H}}$, the decision rule can be expressed as $SVM(x) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b)$. PD kernels avoids the explicit definition of $\mathcal{H}$ and Φ [15]; the most popular are the linear (LIN) kernel $\langle x, x'\rangle$, the radial basis function (RBF) kernel $\exp \|x - x'\|^2/\sigma$, and the inhomogeneous polynomial (IPOL) kernel $(\langle x, x'\rangle + 1)^d$. SVM has been shown to have important generalization properties and nice bounds on the VC dimension [14]. In particular we refer to the following theorem.

Theorem 1 (Vapnik [14] p.139) *The expectation of the probability of test error for a maximal sep-*

[2]We refer to kernels functions with K and to the number of nearest neighbors with k.

arating hyperplane is bounded by: $EP_{error} \leq E\left\{\min\left(\frac{sv}{n}, \frac{1}{n}\left[\frac{R^2}{\Delta^2}\right], \frac{p}{n}\right)\right\}$*, where n is the cardinality of the training set, sv is the number of support vectors, R is the radius of the sphere containing all the samples, $\Delta = 1/|w|$ is the margin, and p is the dimensionality of the input space.*

Theorem 1 states that the maximal separating hyperplane can generalize well as the expectation on the margin is large, since a large margin minimizes R^2/Δ^2. Computationally, SVM takes $O(n^2)$ time for computing the kernel values, $O(n^3)$ time for solving the problem and $O(n^2)$ space for storing the kernel values as discussed in [7, 16]. Some implementation adopting decomposition techniques have a scaling time factor approaching $f_n \cdot n^2$ where f_n is the number of features.

1.3 The kNNSVM classifier

The method [1] combines locality and searches for a large margin separating surface by partitioning the entire Hilbert feature space through a set of local maximal margin hyperplanes. In order to classify a given point x', we need first to find its k nearest neighbors in the feature space $\mathcal{H}$ and, then, to search for an optimal separating hyperplane only over these k neighbors. In practice, this means that an SVM is built over the neighborhood of each test point x'. Accordingly, the constraints become:$y_{r_x(i)}\left(\langle w, \Phi(x_{r_x(i)})\rangle + b\right) \geq 1 - \xi_{r_x(i)}$, with $i = 1, \ldots, k$, where $r_{x'} : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ is a function that reorders the indexes of the training points:

$$\begin{cases} r_{x'}(1) = \underset{i=1,\ldots,n}{\operatorname{argmin}} \|\Phi(x_i) - \Phi(x')\|^2 \\ r_{x'}(j) = \underset{i=1,\ldots,n}{\operatorname{argmin}} \|\Phi(x_i) - \Phi(x')\|^2 \\ \qquad\quad i \neq r_{x'}(1), \ldots, r_{x'}(j-1) \quad \text{for } j = 2, \ldots, n \end{cases}$$

In this way, $x_{r_{x'}(j)}$ is the point of the set X in the j-th position in terms of distance from x' and the thus $j < k \Rightarrow \|\Phi(x_{r_{x'}(j)}) - \Phi(x')\| \leq \|\Phi(x_{r_{x'}(k)}) - \Phi(x')\|$. The computation is expressed as $||\Phi(x) - \Phi(x')||^2 = \langle\Phi(x), \Phi(x)\rangle_{\mathcal{H}} + \langle\Phi(x'), \Phi(x')\rangle_{\mathcal{H}} - 2 \cdot \langle\Phi(x), \Phi(x')\rangle_{\mathcal{H}} = K(x, x) + K(x', x') - 2 \cdot K(x, x')$. If the kernel is the RBF kernel or any polynomial kernels with degree 1, the ordering function can be built using the Euclidean metric. For non-linear kernels (other than the RBF kernel) the ordering function can be quite different to that produced using the Euclidean metric.

The decision rule of this method is:

$$k\text{NNSVM}(x) = \operatorname{sign}\left(\sum_{i=1}^{k} \alpha_{r_x(i)} y_{r_x(i)} K(x_{r_x(i)}, x) + b\right) \quad (1)$$

For $k = n$, kNNSVM becomes the usual SVM whereas, for $k = 2$ with LIN or RBF kernels, corresponds to the NN classifier. The method is computationally expensive because, for each test point, it computes the kNN in $\mathcal{H}$, train an SVM and finally perform SVM prediction. Implementing kNN simply sorting the distances, takes $O(n \log n \cdot k^3 \cdot m)$ time for m testing samples.

The bound on the expectation of the probability of test error for kNNSVM becomes $EP_{error} \leq E\{\min(\frac{sv}{k}, \frac{1}{k}[\frac{R^2}{\Delta^2}], \frac{p}{k})\}$. Whereas SVM has the same bound with $k = n$, apparently the three quantities increase due to $k < n$. However, in the case of kNNSVM R^2/Δ^2 decreases because: R (in the local case) is smaller than the radius of the sphere that contains all the training points and the margin Δ increases or at least remains unchanged. The former point is easy to show, while the second point (limited to the case of linear separability) is stated in the following theorem.

Theorem 2 ([2]) *Given a set of n points $X = \{x_i \in \mathbb{R}^p\}$, each associated with a label $y_i \in \{-1, 1\}$, over which is defined a maximal margin separating hyperplane with margin Δ_X, if for an arbitrary subset $X' \subset X$ there exists a maximal margin hyperplane with margin $\Delta_{X'}$ then the inequality $\Delta_{X'} \geq \Delta_X$ holds.*

Proof 1 (Sketch of the proof.) *Observe that for $X' \subset X$ the convex hull of each class is contained in the convex hull of the same class in X. Since the margin can be seen as the minimum distance between the convex hulls of different classes and since given two convex hulls H_1, H_2 the minimum distance between them cannot be lower than the minimum distance between H_1' and H_2 with $H_1' \subseteq H_1$, we have the thesis. For and alternative and rigourous proof see [2].*

As a consequence of Theorem 2, kNNSVM has the potential of improving over both 1NN and SVM for some $2 < k \leq n$ as empirically confirmed in [3] on 13 small-size classification problems.

2 FastLSVM: a Local SVM approach for large datasets

In this section we present FastLSVM, a modified version of Local SVM that allows for the use on large datasets. As a first step, we can generalize the decision rule of Local SVM considering the case in which the local model is trained on a set of points that are the k-nearest neighbors of a point that, in general, is different from the query point. The modified decision function for a query point x and another (possibly different) point t is:

$$k\text{NNSVM}_t(x) = \operatorname{sign}\left(\sum_{i=1}^{k} \alpha_{r_t(i)} y_{r_t(i)} K(x_{r_t(i)}, x) + b\right)$$

where $r_t(i)$ is the kNNSVM ordering function (see above) and $\alpha_{r_t(i)}$ and b come from the training of an SVM on the k-nearest neighbors of t in the feature space. In the following we will refer to $k\text{NNSVM}_t(x)$ as being centered in t and to t as the center of the model. The original decision function of kNNSVM corresponds to the case in which $t = x$, and thus $k\text{NNSVM}_x(x) = k\text{NNSVM}(x)$.

2.1 A first approximation of Local SVM.

In the original formulation of kNNSVM, the training of an SVM on the k-nearest neighbors of the query point must be performed in the prediction step. Although this approach is convenient when we have a rather large training set and very few points to classify, it introduces a considerable overhead in the prediction step which is not acceptable in the great majority of classification problems. As a first approximation of Local SVM, we propose to compute and maintain in memory a set of Local SVMs centered on each point of the training set. This unburdens the prediction step in which it is sufficient to select a model for the query point and use it to perform the classification. In particular, we chose to select the precomputed model to classify a point x with the model centered on its nearest point in the training. Formally the classification of a point x with this method is $k\text{NNSVM}_t(x)$ with $t = x_{r_x(1)}$. The set of precomputed local SVMs in the training set with corresponding central points is $\mathcal{S} = \{(t, k\text{NNSVM}_t) \mid t \in X\}$. Notice that in situations where the neighbourhood contains only one class the local model does not find any separation and so considers all the neighbourhood to belong to the predominant class thus simulating the behaviour of the majority rule.

2.2 Introducing the assignment neighborhood

With the previous modification of kNNSVM we made the prediction step much more computationally efficient, but a considerable overhead is added to the training phase. In fact, the training of an SVM for every point of the training set can be slower than the training of a unique global SVM (especially for non small k values), so we introduce another modification of the method which aims to drastically reduce the number of SVMs that need to be precomputed. Theoretically, this can cause a loss in classification accuracy, so we must take care of not reducing too much the number of SVMs and to maintain the more representative ones. The modification is based on assigning to the local model centered in a point c not only c itself but also the first k' (with $k' < k$) nearest neighbors of c. In this way we aim to make a compromise (controlled by k') between the kNNSVM approach, in which the test point is surrounded by the samples used to build the model, and the need of decreasing the total number of SVM trained. The set of points used to select the k-nearest neighbors for the models is defined as follows.

Definition 1 *Given $k' \in \mathbb{N}$, a k'-neighborhood covering set of centers $\mathcal{C}_{k'} \subseteq X$ is a subset of the training set such that the following holds:*

$$\bigcup_{c \in \mathcal{C}_{k'}} \{x_{r_c(i)} \mid i = 1, \ldots, k'\} = X.$$

Definition 1 means that the union of the sets of the k'-nearest neighbors of $\mathcal{C}_{k'}$ corresponds to the whole training set. Theoretically, for a fixed k', the minimization of the number of local SVMs that we need to train can be obtained computing the SVMs centered on the points contained in the *minimal* k'-neighborhood covering set of centers[3] $\mathcal{C}$. However, since the computing of the minimal $\mathcal{C}$ is not a simple and computationally easy task, we choose to select each $c_i \in \mathcal{C}$ as follows:

$$\begin{aligned} & c_i = x_j \in X \\ & \text{with } j = \min\left(z \in \{1, \ldots, n\} \middle| x_z \in X \setminus X_{c_i}\right) \\ & \text{where } X_{c_i} = \bigcup_{l<i} \left\{ x_{r_{c_l}(h)} \mid h = 1, \ldots, k' \right\}. \end{aligned} \tag{2}$$

The idea of this definition is to recursively take as centers those points which are not k'-neighbors of any point that has already been taken as center. So $c_1 = x_1$ corresponds to the first point of X since, being c_1 the first center, the union of the neighbors of the other centers is empty; c_2, instead, is the point with the minimum index taken from the set obtained eliminating from X all the k'-neighbors of c_1. The procedure is repeated until all the training points are removed from X. X must be thought here as a random reordering of the training set. This is done in order to avoid the possibility that a training set in which the points are inserted with a particular spatial strategy affects the spatial distribution of the k'-neighborhood covering centers. The reason why we adopt this non standard clustering method is twofold: from one side we want each cluster to contain exactly k samples in order to be able to derive rigourous complexity bounds, from the other side in this way we are able to select a variable number of samples that are in the central region (at least form a neighborhood viewpoint) of each cluster. Moreover the proposed clustering strategy follows quite naturally from kNNSVM approach.

[3] From now on we simply denote $\mathcal{C}_{k'}$ with $\mathcal{C}$ because we do not discuss here particular values for k'.

Differently from the first approximation in which a local SVM is trained for each training sample, in this case we need to train only $|\mathcal{C}|$ SVMs centered on each $c \in \mathcal{C}$ obtaining the following models:

$$k\text{NNSVM}_c(x), \quad \forall c \in \mathcal{C}.$$

Now we have to link the points of the training set with the precomputed SVM models. This is necessary because a point can lie in the k' neighborhood of more than one center. In particular we want to consider the assignments of each training point to a unique model such that it is in the k' neighborhood of the center on which the model is built. Formally this is done with the function $cnt(t) : X \to \mathcal{C}$ that assigns each point in the training set to a center:

$$\begin{aligned} & cnt(x_i) = x_j \in \mathcal{C} \\ & \text{with } j = \min\left(z \in \{1, \ldots, n\} \middle| x_z \in \mathcal{C} \text{ and } x_i \in X_{x_z}\right) \\ & \text{where } X_{x_z} = \left\{x_{r_{x_z}(h)} \;\middle|\; h = 1, \ldots, k'\right\}. \end{aligned} \quad (3)$$

With the *cnt* function, each training point is assigned to the first center whose k'-nearest neighbors set includes the training point itself. The order of the c_i points derives from the randomization of X used for defining $\mathcal{C}$. In this way each training point is univocally assigned to a center and so the decision function of this approximation of Local SVM, called FastLSVM, is simply:

$$\text{FastLSVM}(x) = k\text{NNSVM}_c(x) \quad \text{with } c = cnt(x_{r_x(1)}) \quad (4)$$

Algorithm 1 presents the pseudo-code of FastLSVM implementing the formal definition of Equation 2 for selecting the centers and Equation 3 to assign each training point to a unique corresponding center and thus to the SVM model trained on the center neighborhood.

Although not deeply discussed and experimented here, notice that it is not required that all local models share the same hyperparameters. In fact, it is possible to set different parameters for different local models, being able of better capturing local properties of the data. This can be done with local model selection, e.g. performing cross validation (CV) on the local models or estimating the parameters using local data statistics (as proposed in [7] for RBF kernel, based on distance distribution). In particular setting the σ parameter of RBF kernel locally, leads to a very similar goal of traditional RBF-SVM with variable width that demonstrated good potentialities for classification as shown for example in [17].

Algorithm 1 FastLSVM_TRAIN (training set **x**[], training size **n**, neighborhood size **k**, assignment neighborhood size **k'**)

```
models[] ⇐ null //the set of models
modelPtrs[] ⇐ null //the set pointers to the models
c ⇐ 0 //the counter for the centers of the models
indexes[] ⇐ {1, ..., n} //the indexes for centers selection
Randomize indexes //randomize the indexes
for i ⇐ 1 to n do
    index ⇐ indexes[i] //get the i-th index
    if modelPtrs[index] = null then //if the point has not been assigned to a model...
        localPoints[] ⇐ get ordered kNN of x[i]
        models[c] ⇐ SVMtrain on localPoints[]
        modelPtrs[index] ⇐ models[c]
        for j = 1 to k' do //assign the model also to the first k' < k nearest neighbors of the center
            ind ⇐ get index of localPoints[j]
            if modelPtrs[ind] = null then
                modelPtrs[ind] ⇐ models[c]
            end if
        end for
        c ⇐ c+1
    end if
end for
return models, modelPtrs
```

2.3 Complexity bounds

FastLSVM requires $O(|\mathcal{C}| \cdot n \log n + |\mathcal{C}| \cdot k^3)$ for training, thus overcoming SVM performance $O(n^3)$. Notice that for $k = k' = n$ we have a global SVM computable, as expected, in $O(n \log n + n^3) = O(n^3)$ since $|\mathcal{C}| = 1$. kNNSVM testing is instead slightly slower than SVM: $O(n \cdot k \cdot m)$ against $O(n \cdot m)$. Although not considered in the implemented version, FastLSVM can take great advantages from data-structures supporting nearest neighbors searches [18]. For example, using the recently developed cover tree data-structure [19] allowing kNN searches in $k \log(n)$ with $n \log n$ construction time, FastLSVM can further decrease its training computational complexity to $O(n \log n + |\mathcal{C}| \cdot \log n \cdot k + |\mathcal{C}| \cdot k^3)$ which is much lower than SVM complexity for fixed and non-high values of k. Similarly, for testing, the required time becomes $O(\log n \cdot k \cdot m)$. Another not implemented modification able to reduce computational complexity consists in avoiding the training of local SVMs with samples of one class only. Moreover, FastLSVM can be very easily parallelized differently from SVM for which parallelization, although possible [10, 11], is a rather critical aspect; for FastLSVM is sufficient that, every time the points for a model are retrieved, the training of the local SVM is performed on a different processor. In this way the time complexity of FastLSVM can be further lowered to $O(|\mathcal{C}| \cdot n \log n + |\mathcal{C}| \cdot k^3 / n_{procs})$.

It can be argued that some modern SVM solvers, mainly based on decomposition strategies, scales better than $O(n^3)$; in particular, in [16] the authors state that, generally, the dual problem can be solved accurately in $\Omega(n_f \cdot n^2)$ where n_f is the number of features.

However, asymptomatically, FastLSVM is faster than every SVM solver taking more than $|\mathcal{C}| \cdot n \log n$ for training ($n \log n$ using cover trees).

Another advantage of FastLSVM over SVM is the space complexity. Since FastLSVM performs SVM training on small subregions (assuming a reasonable low k), there are no problems of fitting the kernel matrix into main memory. The overall required space is $O(n + k^2)$, i.e. linear in n, that is much lower than SVM space complexity of $O(n^2)$ which forces, for large datasets, the discarding of some kernel values thus increasing SVM time complexity due to the need of recomputing them.

3 Empirical evaluation

In this work we used LibSVM (version 2.85) [20] for SVM enabling shrinking and caching, and our implementation of FastLSVM that uses LibSVM for training and prediction of the local SVMs and a simple *brute-force* implementation of kNN. The experiments are carried out on an AMD Athlon™ 64 X2 Dual Core Processor 5000+, 2600MHz, with 3.56Gb of RAM.

3.1 kNNSVM - FastLSVM comparison

In order to understand if FastLSVM is a good approximation of kNNSVM, we compared the two methods on the 13 small datasets of [3] using the SVM results as references. We present the 10-fold cross validation (CV) accuracies obtained with the three methods using the LIN, RBF, HPOL and IPOL kernels. The model selection is performed internally to each fold minimizing the empirical risk with 10-fold CV choosing $C \in \{1, 5, 10, 25, 50, 75, 100, 150, 300, 500\}$, the σ parameter of the RBF kernel among $\{2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10}\}$ and the degree of the polynomial kernels is bounded to 5. The dimension of the neighborhood for the kNNSVM classifier, i.e. k, is chosen among the first 5 odd natural numbers followed by the ones obtained with a base-2 exponential increment from 9 and the cardinality of the training set, namely in {1, 3, 5, 7, 9, 11, 15, 23, 39, 71, 135, 263, 519, $|training\_set|$}. The k' parameter of FastLSVM is fixed to $1/4 \cdot k$. To assess the statistical significance of the differences between the 10-fold CV of kNNSVM and FastLSVM with respect to SVM we use the two-tailed paired t-test ($\alpha = 0.05$) on the two sets of fold accuracies.

The results are reported in Table 1. We can notice that the generalization accuracies are generally a little worse for FastLSVM than kNNSVM, but the overall advantage over SVM is maintained. In fact FastLSVM demonstrates 12 cases in which the classification accuracies are significantly different (according to the t-test) to the SVM ones, and all 12 cases are in favour of FastLSVM without cases in which it is significantly worse than SVM. In total, there are 7 cases in which the significant improvements of kNNSVM over SVM are not maintained by the FastLSVM algorithm; this can be due to the choice of k', in fact, a lower value of k' guaranties much lower differences between FastLSVM and kNNSVM. However, since our final objective is the application of the approach to large and very large problems on which it is reasonable to hypothesize that locality can assume an even more important role, and since FastLSVM is still better than SVM, the empirical comparison between FastLSVM and kNNSVM on small datasets let us to conclude that FastLSVM is a good approximation of kNNSVM.

Table 1: 10-fold CV accuracies of SVM, kNNSVM and FastLSVM with LIN, RBF, HPOL and IPOL kernels on 13 small datasets. The statistical significative differences of kNNSVM and FastLSVM with respect to SVM (two-tailed paired t-test with $\alpha = 0.05$) are in bold.

dataset	LIN kernel			RBF kernel			HPOL kernel			IPOL kernel		
	SVM	kNN-SVM	Fast-LSVM	SVM	kNN-SVM	Fast-LSVM	SVM	kNN-SVM	Fast-LSVM	SVM	kNN-SVM	Fast-LSVM
iris	0.97	0.96	0.95	0.95	0.96	0.95	0.97	0.96	0.96	0.97	0.97	0.97
wine	0.97	0.98	0.97	0.99	0.99	0.99	0.97	**0.99**	0.98	0.97	**0.99**	0.97
leukemia	0.95	0.93	0.93	0.71	**0.93**	**0.88**	0.95	0.93	0.93	0.95	0.93	0.93
liver	0.68	**0.74**	**0.73**	0.72	0.73	0.72	0.71	**0.74**	**0.74**	0.70	**0.73**	0.72
svmguide2	0.82	**0.86**	0.84	0.84	0.84	0.84	0.82	0.84	0.82	0.83	**0.86**	**0.85**
vehicle	0.80	**0.86**	**0.85**	0.85	0.84	0.85	0.84	**0.86**	**0.86**	0.85	0.85	0.86
vowel	0.84	**1.00**	**0.99**	0.99	1.00	1.00	0.98	**1.00**	**1.00**	0.99	**1.00**	**1.00**
breast	0.97	0.97	0.96	0.97	0.97	0.97	0.97	0.96	0.96	0.97	0.96	0.96
fourclass	0.77	**1.00**	**1.00**	1.00	1.00	1.00	0.81	**1.00**	**1.00**	1.00	1.00	1.00
glass	0.62	**0.69**	**0.69**	0.69	0.67	0.70	0.72	0.72	0.72	0.70	0.71	0.70
heart	0.83	0.82	0.83	0.83	0.82	0.81	0.82	0.82	0.82	0.82	0.82	0.82
ionosphere	0.87	**0.93**	0.88	0.94	0.93	0.94	0.89	**0.93**	0.91	0.91	0.93	0.92
sonar	0.78	**0.88**	0.85	0.89	0.90	0.89	0.88	0.89	0.88	0.88	0.89	0.88

Table 2: Percentage accuracy and computational (in seconds) results for SVM and FastLSVM on the 2SPIRAL dataset. The parameters reported are the one permitting the lowest empirical risk, found with 5-fold CV.

Method	k	k'	C	σ	valid. acc.	test. acc.	# of SVM	train. time (s)	test. time (s)
RBF-SVM	-	-	2^6	2^{-10}	81.30	81.39	1	6185	392
LIN-FastLSVM	250	62	2^{10}	-	88.37	88.46	3202	222	484
RBF-FastLSVM	1000	250	2^8	2^{-5}	88.48	88.43	853	165	492
IPOL-FastLSVM	500	125	2^{10}	-	88.32	88.41	1657	240	3415

3.2 The 2SPIRAL dataset

The two classes of the 2SPIRAL artificial dataset are defined as follows:

$$\begin{cases} x^{(1)}(\tau) = c \cdot \tau^d \cdot sin(\tau) \\ x^{(2)}(\tau) = c \cdot \tau^d \cdot cos(\tau) \end{cases} \qquad d = 2.5,\ \tau \in [0, 10\pi]$$

using $c = 1/500$ for the first class ($y_i = +1$) and $c = -1/500$ for the second class ($y_i = -1$).

The points are sampled with intervals of $\pi/5000$ on the τ parameter obtaining 50000 points for each class. A Gaussian noise with zero mean and variance proportional to the distance between the point and the nearest internal twist is added on both dimensions. We this procedure we generated three different datasets of 100000 points each for training, validation and testing.

We compare FastLSVM and SVM using LIN, RBF and IPOL (with degree 2) kernels. Since LIN and IPOL kernels can only build linear and quadratic decision functions in the input space, they cannot give satisfactory results for global SVM and thus we do not loose generality in presenting SVM results with the RBF kernel only. For model selection we adopt grid search with 5-fold CV. For both methods, C and σ of RBF kernel are chosen in $\{2^{-10}, 2^{-9}, \ldots, 2^9, 2^{10}\}$. It is possible that values higher than 2^{10} for C and lower than 2^{-10} for σ could give higher validation accuracy results, but the computational overhead of SVM becomes too high to be suitable in practice (e.g. RBF-SVM with $C = 2^{11}$, $\sigma = 2^{-11}$ requires more than 24 hours). For FastLSVM we fix $k' = k/4$ (intuitively a good compromise between accuracy and performance), while k is chosen among {0.25%, 0.5%, 1%, 2%, 4%, 8%, 16%, 32%} of training set size.

Table 2 shows the results obtained for SVM and FastLSVM. We have that RBF-FastLSVM improves over RBF-SVM in test accuracy of 8.65%, LIN-FastLSVM of 8.69% and IPOL-FastLSVM of 8.63%. The improvements on classification accuracies are accomplished by a dramatic increase of computational performances for training phase: while the time needed to compute the global SVM on the training set is more than 100 minutes (6185 seconds), the training of FastLSVM requires no more than 4 minutes. The best prediction time, instead, is achieved by SVM although RBF-FastLSVM and LIN-FastLSVM give comparable performances; the prediction time of IPOL-FastLSVM is, instead, about an order of magnitude higher than RBF-SVM.

3.3 The CoverType dataset

The binary CoverType dataset (retrieved from LibSVM homepage [20]) has 581012 samples with 54 features. We randomly chose 25000 samples for testing, the others for training. Smaller training sets (from 1000 to 500000 samples) are obtained randomly sub-sampling the training data. We apply SVM and FastLSVM with RBF kernel using the same model selection strategy of the 2SPIRAL problem stopping it without giving the best parameters, and thus without performing the classification, if at least one fold of 5 fold CV does not terminate within 6 hours.

Table 3: Percentage accuracies and performances (in seconds) of SVM and FastLSVM on CoverType data.

	SVM			FastLSVM		
n	test acc.	train. time	test. time	test. acc.	train. time	test. time
1000	74.32	0	2	74.77	0	5
2000	76.25	1	3	76.28	0	10
3000	77.83	1	6	77.84	1	14
4000	78.83	3	9	79.34	2	20
5000	80.19	4	10	80.35	10	26
7500	82.36	11	16	82.47	43	42
10000	83.48	34	19	83.64	10	29
15000	85.56	148	32	85.78	15	73
20000	86.45	138	32	86.69	20	97
30000	88.14	588	51	88.25	59	146
40000	89.44	933	64	89.48	78	193
50000	90.22	1814	71	90.32	103	238
75000	91.78	4862	128	91.84	391	361
100000	92.81	7583	152	92.84	439	476
150000	-	-	-	94.07	505	716
200000	-	-	-	94.63	813	952
250000	-	-	-	95.35	1196	1188
300000	-	-	-	95.52	1663	1427
350000	-	-	-	95.83	3573	1663
400000	-	-	-	95.95	2879	1980
450000	-	-	-	96.18	3600	2140
500000	-	-	-	96.36	4431	2370
556000	-	-	-	96.47	5436	2633

The accuracy and performance results at increasing training set sizes are reported in Table 3.

SVM accuracies are lower than FastLSVM ones for every training set size. From a computational viewpoint, FastLSVM can train the model on the whole training set faster than SVM on 100000 samples (less than 1/5 of the data); moreover, starting from $n = 30000$ FastLSVM training is at least one order of magnitude faster than SVM, and the difference is more and more relevant as n increases. It is important to underline that the whole dataset permits a much higher classification accuracy than the random sub-sampled sets, so it is highly desirable to consider all the data. Since we implemented FastLSVM without supporting data-structures for nearest neighbors, we must compute, for all test points, the distances with all the training points, leading to a rather high testing time.

4 Conclusions

Starting from the kNNSVM classifier, we presented FastLSVM, which is scalable for large datasets and maintains the advantages in terms of classification accuracy of the original formulation of Local SVM. Differently from Local SVM, FastLSVM precomputes the local models in the training set trying to minimize the number of SVM that needs to be built assigning the models not only to the central point but to the k' most central samples. Furthermore training of the local model can be very easily parallelized. The prediction is performed applying to the query point the SVM model to which its nearest neighbor in the training set has been assigned. The method demonstrated empirically to be a good approximation of kNNSVM and to substantially overcome SVM both in terms of classification accuracy and training time performance on an artificial large dataset and a real very large dataset. Moreover we discussed some improvements to the method such as kNN supporting data-structures, which can further sensibly increase the training and testing performances of FastLSVM.

References

[1] E. Blanzieri and F. Melgani. An adaptive SVM nearest neighbor classifier for remotely sensed imagery. *IEEE Int Conf on Geoscience and Remote Sensing Symposium (IGARSS-2006)*, pages 3931–3934, 2006.

[2] E. Blanzieri and F. Melgani. Nearest neighbor classification of remote sensing images with the maximal margin principle. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(6):1804–1811, 2008.

[3] N. Segata and E. Blanzieri. Empirical assessment of classification accuracy of Local SVM. Technical Report DISI-08-014, Dip. Ingegneria e Scienza dell'Informazione, University of Trento, 2008.

[4] V.L. Brailovsky, O. Barzilay, and R. Shahave. On global, local, mixed and neighborhood kernels for support vector machines. *Pattern Recognition Letters*, 20(11-13):1183–1190, 1999.

[5] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Comput*, 4(6):888–900, 1992.

[6] H. Zhang, A.C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. *Proc of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:2126–2136, 2006.

[7] I.W. Tsang, J.T. Kwok, and P.M. Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *The Journal of Machine Learning Research*, 6:363–392, 2005.

[8] H. Yu, J. Yang, J. Han, and X. Li. Making SVMs Scalable to Large Data Sets using Hierarchical Cluster Indexing. *Data Mining and Knowledge Discovery*, 11(3):295–321, 2005.

[9] M. Dong and J. Wu. Localized Support Vector Machines for Classification. *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 799–805, 2006.

[10] L. Zanni, T. Serafini, and G. Zanghirati. Parallel Software for Training Large Scale Support Vector Machines on Multiprocessor Systems. *The Journal of Machine Learning Research*, 7:1467–1492, 2006.

[11] J.X. Dong, A. Krzyzak, and C.Y. Suen. Fast SVM training algorithm with decomposition on very large data sets. *IEEE Trans Pattern Anal Mach Intell*, 27(4):603–18, 2005.

[12] H. Cheng, P.N. Tan, and R. Jin. Localized Support Vector Machine and Its Efficient Algorithm. *Proc. SIAM IntlConf. Data Mining*, 2007.

[13] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[14] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[15] B. Schölkopf and A.J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

[16] L. Bottou and C.J. Lin. Support Vector Machine Solvers. *Large-Scale Kernel Machines*, 2007.

[17] Q. Chang, Q. Chen, and X. Wang. Scaling gaussian rbf kernel width to improve svm classification. *Neural Networks and Brain, 2005. ICNN&B '05. International Conference on*, 1:19–22, 13-15 Oct. 2005.

[18] E. Chávez, G. Navarro, R. Baeza-Yates, and J.L. Marroquín. Searching in metric spaces. *ACM Computing Surveys (CSUR)*, 33(3):273–321, 2001.

[19] A. Beygelzimer, S. Kakade, and J. Langford. Cover Trees for Nearest Neighbor. *Proceedings of the 23rd International Conference on Machine learning, Pittsburgh, PA, 2006.*, pages 97–104, 2006.

[20] C.C. Chang and C.J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.