

Single-Pass Distributed Learning of Multi-Class SVMs using Core-Sets

Stefano Lodi ^a

Ricardo Ñanculef ^{a,b}

Claudio Sartori ^a

Abstract

We explore a technique to learn Support Vector Models (SVMs) when training data is partitioned among several data sources. The basic idea is to consider SVMs which can be reduced to Minimal Enclosing Ball (MEB) problems in an feature space. Computation of such SVMs can be efficiently achieved by finding a core-set for the image of the data in the feature space. Our main result is that the union of local core-sets provides a close approximation to a global core-set from which the SVM can be recovered. The method requires hence a single pass through each source of data in order to compute local core-sets and then to recover the SVM from its union. Extensive simulations in small and large datasets are presented in order to evaluate its classification accuracy, transmission efficiency and global complexity, comparing its results with a widely used single-pass heuristic to learn standard SVMs.

1 Introduction

Motivation. Advances in computing, communication technologies, storage methods and user interaction paradigms have resulted in many large scale and decentralized environments in which both data and computation are distributed through a communication network. Certainly, data mining tools play a fundamental role for engineering applications on such domains. Fraud detection in electronic commerce, content annotation on social networks and surveillance sensor networks are examples on which automatized knowledge extraction and monitoring is required. Energy, time, cost, security and privacy concerns prevent however the full centralization of data in a single computation node, as required by many traditional data mining algorithms. Methods providing the same or almost the same performance of such centralized solution but dealing with the domain constraints like synchronization, communication overhead and scalability are hence of primary interest in

practice and have oriented recent research efforts, ranging from general frameworks [32, 3, 5] to model specific methods [14, 27, 2].

Research Focus. Support Vector Models (SVMs) [17], are currently one of the most effective techniques to approach classification and other data analysis problems, improving more traditional techniques like decision trees and neural networks in several applications. In classification, SVMs take as input a set of examples for which the correct categories are known and output a decision function used to determine the categories of new objects. Traditional [26] and even modern methods [11] [19] to obtain SVMs require that the set of examples be completely available in a common place to access them an arbitrary number of times in order to converge to an optimal decision function. A straightforward implementation of these algorithms in a distributed environment would require hence a prohibitive amount of communication overhead and synchronization. Recent works have considered this problem proposing approximate and exact algorithms to efficiently learn an SVM in distributed environments [24, 20, 13, 12, 15, 6, 14]. In the next sections we provide a more detailed discussion about related work. Most current approaches are however variants of a widely used heuristic to efficiently learn SVMs from distributed data: compute local SVMs, extract the support sets (subsets of data on which the SVM decision functions are defined), compute the union of the support sets and build a global SVM from this subset of examples in a central node. The goal of this paper is to explore an alternative single-pass method based on a recently proposed equivalence between SVMs and Minimal Enclosing Ball (MEB) problems from which important improvements on training efficiency has been reported [30, 29] for large-scale datasets. We directly focus on multi-class problems exploring two methods to extend binary SVMs to the multi-category setting which preserve the equivalence between the model and MEBs.

Skeleton of the Proposal. Algorithms to compute SVMs based on the MEB equivalence are based on the greedy computation of a core-set, a typically small

^aDepartment of Electronics, Computer Science and Systems, University of Bologna, Italy

^bDepartment of Informatics, Federico Santa María University, Chile

subset of the data which provides the same MEB as the full dataset. Our algorithm proceeds by computing local core-sets at each data source and then computing an SVM from the union of these core-sets. For simplicity we implement this second step in a coordinator node which receives all the intermediate results from the nodes in the network and recovers the final model. More general topologies to hierarchically implement this step can be considered. For the computation of the core-sets we use the approximate algorithm of Badoiu and Clarkson [4]. This method relaxes the enclosing-ball property using a given tolerance ϵ . We show that this parameter allows the distributed algorithm to control the local computation times and communication overhead.

Model Instantiation. The only requirement to apply the method described above is that the reduction of the SVM to a MEB problem holds. Since the SVM is designed for binary classification, we explore two methods to support multiple classes: the one-versus-one decomposition heuristic (OVO) based on the computation of several binary SVMs [22, 18] and a new direct implementation which works highly better in practice than [1], a previously proposed direct implementation which supports the MEB reduction.

Experiments and Results. Experimental simulations in 3 small, 2 medium and 5 large scale datasets are provided. Results confirm that the distributed solution is highly effective in terms of prediction accuracy when compared with (1) a SVM trained on the complete dataset and (2) a SVM trained using the support vector heuristic. We show that the method provides at the same time the possibility to control both the computational complexity and the amount of data transmitted from nodes as a function of the tolerance parameter ϵ . The accuracy actually does not depend on the number of nodes as we show by providing simulations in which the number of nodes is increased from 10 to 1000.

Organization. Section 2 introduces SVMs. Section 3 concerns MEBs and core-sets. In Sections 4 and 5 we formulate the distributed solution and our modified multi-class method. In Section 6 we discuss related work. Section 7 provides the experimental methodology and results. Section 8 summarizes the conclusions.

2 Support Vector Models for Classification

Given a set of data $S = \{\mathbf{x}_i\}$ with $\mathbf{x}_i \in \mathcal{X}$, $i \in I \subset \mathbb{N}$ where each instance is associated with a given category in the set $L = \{c_1, c_2, \dots, c_K\}$, a classification problem consists in inferring from S a prediction mechanism $f : \mathcal{X} \rightarrow C$, termed hypothesis, to associate new objects \mathbf{x} with the correct category. SVMs build such hypothesis by optimizing a functional which balances error on the training examples and sparsity or simplicity

of the solution [17]. Under regularity conditions, this simple principle has theoretical guarantees of good performance on new decision instances [31].

Since in realistic problems the configuration of the data can be highly non-linear, SVMs build a linear model not in the original space \mathcal{X} but in a high-dimensional dot-product space $Z = \phi(\mathcal{X})$, named the feature space, where the decision function can be linearly represented. This space is related with the \mathcal{X} by means of a function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ called *the kernel* which computes the dot products $\mathbf{z}_i^T \mathbf{z}_j$ in Z directly from the input space $k(\mathbf{x}_i, \mathbf{x}_j)$ avoiding the explicit computation of the mapping ϕ . The reader may refer to [26] for details.

2.1 Binary L2-SVMs In a binary classification problem, examples $\{\mathbf{x}_i\}$ can be labeled as $y_i = +1$ or $y_i = -1$ according to whether they belong to a class c_1 or c_2 respectively. With this encoding, SVMs proceed by building a hyperplane which is aimed to represent the boundary between the classes. In the feature space $Z = \phi(\mathcal{X})$, this separating hyperplane takes the form $f(\mathbf{z}_i) = \mathbf{w}^T \mathbf{z}_i + b = 0$. Points labelled as $+1$ should be at the positive side of the hyperplane $f(\mathbf{z}_i) > 0$ and the rest at the negative side. Future instances are hence classified as c_1 if $f(\mathbf{z}) > 0$ and c_2 otherwise.

The so called *L2-SVM* chooses the separating hyperplane $f(\mathbf{z})$ by solving the following quadratic program:

$$(2.1) \quad \min(\mathbf{w}, b, \rho, \xi): \frac{1}{2} (\|\mathbf{w}\|^2 + b^2 + C \sum_i \xi_i^2) - \rho$$

$$\text{st: } y_i f(\mathbf{z}_i) \geq \rho - \xi_i \quad \forall i \in I$$

If $y_i f(\mathbf{z}_i) > \rho > 0$, the example \mathbf{z}_i is correctly classified. Variable ρ , called *the margin*, is hence a measure of classification confidence and the *slacks* ξ_i a measure of the amount of confidence violation. Variable ρ is thus maximized in the objective function and slacks penalized using a hyper-parameter C . The term $\|\mathbf{w}\|^2 + b^2$, on the other hand, encourages sparsity or simplicity of the solution.

The main difference with the so called L1-SVM, which corresponds to the original SVM formulation [8], is that slacks are penalized using the l_2 -norm instead of the l_1 -norm. Previous research shows that both models obtain comparable performance in practice [34] [21] [30]. However, the L2 implementation supports a convenient reduction that has been previously exploited to build efficient training algorithms [30, 29] and that here will be used to support distributed training.

After introducing Lagrange multipliers $\alpha_i \geq 0$ it can be shown that the latter problem is equivalent to

solve

$$(2.2) \quad \begin{aligned} \min(\alpha) : & \sum_{i,j \in I} \alpha_i \alpha_j \mathbf{K}_{ij} \\ \text{st: } & 0 \leq \alpha_i, \sum_i \alpha_i = 1 \end{aligned}$$

where $K_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}$ and $k(\mathbf{x}_i, \mathbf{x}_j)$ implements the dot-product $\mathbf{z}_i^T \mathbf{z}_j$. The relation between the first problem (primal) and the last one is given by $\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i$ and $b = \sum_i y_i \alpha_i$. Training examples for which $\alpha_i \neq 0$ are called the *support vectors*.

2.2 Multi-Class Extensions L1 and L2 SVMs were originally formulated for binary classification problems. In a multi-class problem, examples $\{\mathbf{x}_i\}$ belong to a set of K categories $c \in \{c_k; k \in L\}$ with $K \leq 2$ and hence the two “codes” $+1$ and -1 used to denote the two sides of a separating hyperplane are no longer enough to implement a decision function.

There are two types of extensions to build multi-class SVMs. One corresponds to use several binary classifiers, separately trained and joined into a multi-category decision function. For example in one-versus-the-rest (**OVR**, [31]), where a different binary SVM is used to separate each class from the rest; one-versus-one (**OVO**, [22]) where one binary SVM is used to separate each possible pair of classes; and **DDAG** [26] where one-versus-one classifiers are trained and then organized in a directed acyclic graph decision structure. Previous experiments in the context of SVMs show however that **OVO** frequently obtains a better performance both in terms of accuracy and training time [18].

Another type of extension consists in reformulating the quadratic program underlying SVMs to directly address the multiple classes in a single optimization problem. For the standard formulation of the SVM (L1-SVMs) examples of this approach are [9], [23] and [25].

Up to our knowledge the only proposal of this nature directly addressing the multi-class extension of L2-SVMs is [1]. This extension preserves the reduction to a minimal-enclosing-ball problem characteristic of the binary L2-SVM, which is the key requirement of our algorithm. The formulation associates each class $c_k, k \in L$ of the problem with a K -dimensional vector code \mathbf{t}_k and looks for a projector \mathbf{W} operating on the feature space $Z = \phi(\mathcal{X})$ which should allow to recover the correct code for a given input \mathbf{z} . Denote by \mathbf{t}_k the vector associated to a class $k \in L$ and $\hat{\mathbf{t}}_k$ the K -dimensional vector recovered by the projector $\hat{\mathbf{t}}_k = \mathbf{W}^T \mathbf{z} + \mathbf{b}$. An easy and convenient way to account for the discrepancy between both vectors is by the dot product. If the codes are normalized to have the same norm, the greater the dot product the greater the match between the class predicted by the model and the true class. In the soft-margin implementation we can tolerate

errors for some examples and these errors are tradeoff, as usual, against the sparseness of the solution measured in terms of a norm which now applies on the operator \mathbf{W} instead of the vector \mathbf{w} . The formulation takes hence the form

$$(2.3) \quad \begin{aligned} \min(\alpha) : & \frac{1}{2} (\|\mathbf{W}\|^2 + \|\mathbf{b}\|^2 + C \cdot \sum_i \xi_i^2) - \rho \\ \text{st: } & \mathbf{y}_i^T (\mathbf{W}^T \mathbf{z} + \mathbf{b}) \geq \rho - \xi_i \quad \xi_i \geq 0 \quad \forall i \in I \end{aligned}$$

Several selections are possible for the norm $\|\mathbf{W}\|^2$. A common choice is the so called *Frobenius norm* $\|\mathbf{W}^T \mathbf{W}\|^2 = \text{trace}(\mathbf{W}^T \mathbf{W})$. With this choice the dual of the optimization problem obtained after introducing Lagrange multipliers is

$$(2.4) \quad \begin{aligned} \min(\alpha) : & \sum_{i,j \in I} \alpha_i \alpha_j \mathbf{K}_{ij} \\ \text{st: } & 0 \leq \alpha_i, \sum_i \alpha_i = 1 \end{aligned}$$

where $\mathbf{K}_{ij} = \mathbf{y}_i^T \mathbf{y}_j k(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{y}_i^T \mathbf{y}_j + \frac{\delta_{ij}}{C}$ implements the feature dot-products $\mathbf{z}_i^T \mathbf{z}_j$. In the following this method will be called *DL2* (direct L2 extension). Note that in this formulation the selection of the codes used to represent the classes is arbitrary. No criteria to select them are provided. In practice, the vectors employed are the following

$$(2.5) \quad \mathbf{t}_k(j) = \begin{cases} \sqrt{\frac{K-1}{K}} & \text{if the class of } \mathbf{x}_i \text{ is } j \\ \sqrt{\frac{1}{K(K-1)}} & \text{otherwise} \end{cases}$$

where $\mathbf{t}_k(j)$ denotes the j -th dimension of the code \mathbf{t}_k . The primal solutions \mathbf{W}, \mathbf{b} are obtained from the Lagrangian multipliers as $\mathbf{W} = \sum_i \alpha_i \mathbf{y}_i \mathbf{z}_i^T$ and $\mathbf{b} = \sum_i \alpha_i \mathbf{y}_i$. The decision mechanism for new inputs $\mathbf{z} = \phi(\mathbf{x})$ asks for the code which is more similar to the code recovered by the operator \mathbf{W} that is $\arg \max_{k \in L} \mathbf{t}_k^T (\mathbf{W}^T \mathbf{z} + \mathbf{b})$.

3 L2-SVMs and its Reduction to MEBs

Here we show that the SVMs described in the previous section supports a reduction to a minimal-enclosing-ball (MEB) problem under some conditions on the kernel function. Then we show how to efficiently solve a MEB problem and how the complexity of the algorithm can be controlled using a single parameter.

3.1 Definition of a MEB Problem Let \tilde{Z} a space equipped with an dot product $\tilde{\mathbf{z}}_1^T \tilde{\mathbf{z}}_2$ and the corresponding norm $\|\tilde{\mathbf{z}}\|^2 = \tilde{\mathbf{z}}^T \tilde{\mathbf{z}}$. We define the ball $\mathcal{B}(\mathbf{c}, R)$ of center $\mathbf{c} \in \tilde{Z}$ and radius R in \mathbb{R} as the subset of points $\tilde{\mathbf{z}} \in \tilde{Z}$ for which $\|\tilde{\mathbf{z}} - \mathbf{c}\|^2 \leq R^2$. The minimal-enclosing-ball (MEB, [33]) of a set of points $S = \{\tilde{\mathbf{z}}_i : i \in I\}$ in Z is in turn the ball $\mathcal{B}^*(S, \mathbf{c}^*, R^*)$ of smallest radius

that contains S , that is, the solution to the following optimization problem

$$(3.6) \quad \begin{aligned} \min(\mathbf{R}, \mathbf{c}) : R^2 \\ \text{st: } \|\mathbf{z} - \mathbf{c}\|^2 \leq R^2 \quad \forall \tilde{\mathbf{z}} \in S \end{aligned}$$

After introducing Lagrange multipliers we obtain from the optimality conditions the following dual problem

$$(3.7) \quad \begin{aligned} \min(\alpha) : \sum_{i,j \in I} \alpha_i \alpha_j \tilde{\mathbf{z}}_i^T \tilde{\mathbf{z}}_j - \sum_{i \in I} \alpha_i \tilde{\mathbf{z}}_i^T \tilde{\mathbf{z}}_i \\ \text{st: } 0 \leq \alpha_i, \sum_i \alpha_i = 1 \quad \forall i \in I \end{aligned}$$

3.2 Equivalence between L2-SVMs and MEB Problems

Now, suppose we are computing the minimal-enclosing-ball in feature space $\tilde{Z} = \phi(\tilde{X})$ which has been induced from \mathcal{X} by a mapping function $\tilde{\phi} : \mathcal{X} \rightarrow \tilde{Z}$ and suppose we can compute dot products in \tilde{Z} directly from \mathcal{X} using a kernel function $\tilde{k}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) = \tilde{\mathbf{z}}_i^T \tilde{\mathbf{z}}_j$. Additionally suppose that the kernel is normalized, i.e., $\forall \mathbf{x} \in \mathcal{X}, \tilde{k}(\mathbf{x}, \mathbf{x}) = \Delta$ with $\Delta \in \mathbb{R}$ a constant. Problem (3.7) is hence equivalent to solve the following quadratic program

$$(3.8) \quad \begin{aligned} \min(\alpha) : \sum_{i,j \in I} \alpha_i \alpha_j \tilde{\mathbf{K}}_{ij} \\ \text{st: } 0 \leq \alpha_i, \sum_i \alpha_i = 1 \quad \forall i \in I \end{aligned}$$

where $\tilde{\mathbf{K}}_{ij} = \tilde{k}(\mathbf{x}_i, \mathbf{x}_j)$. This problem coincides with the binary L2-SVM problem (2.2) and its multi-class implementation (2.4) if we set $\tilde{k}(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}$ in the binary case, and $\tilde{k} = \mathbf{y}_i^T \mathbf{y}_j k(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{y}_i^T \mathbf{y}_j + \frac{\delta_{ij}}{C}$ in the multi-category case. The key requirement of the latter equivalence is the normalization constraint on \tilde{k} . Note however that both in the binary and the multi-category case, \tilde{k} is constant when the kernel used by the SVMs k is, i.e., $\tilde{k}(\mathbf{x}_i, \mathbf{x}_i) = \Delta$. This is a property satisfied by any kernel of the form $k(\mathbf{x}_i, \mathbf{x}_j) = \tilde{k}(\|\mathbf{x}_i - \mathbf{x}_j\|^2)$, for example the gaussian or RBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, which is the most commonly used in practice.

Thus, we can train L2-SVMs by solving a MEB problem in which the kernel \tilde{k} implementing its geometry depends on the kernel, the hyper-parameter C and the codes used to represent the classes by the SVM.

3.3 Core-sets and the Badoiu-Clarkson algorithm

A great appeal of the equivalence between L2-SVMs and MEB problems is that Badoiu and Clarkson [4] have recently proposed efficient algorithms to approximate the solution to this classic and hard problem of computational geometry. These algorithms works under the concept of core-sets and ϵ -approximations to the minimal-enclosing-ball of a set of points. A set $\mathcal{C}_S \subset S$ will be called a *core-set* of S if the minimal-enclosing-ball computed over \mathcal{C}_S is equivalent to the

minimal-enclosing-ball considering all the points in S . A ball $\mathcal{B}(\mathbf{c}, R)$ is said a ϵ -approximation to the minimal-enclosing-ball $\mathcal{B}^*(S, \mathbf{c}^*, R^*)$ of S if $R \leq R^*$ and it contains S up to precision ϵ , that is, $S \subset \mathcal{B}(\mathbf{c}, (1 + \epsilon)R)$. Consequently, a set $\mathcal{C}_{S,\epsilon}$ is called a ϵ -core-set if the minimal-enclosing-ball of $\mathcal{C}_{S,\epsilon}$ is a ϵ -approximation to $\mathcal{B}^*(S, \mathbf{c}^*, R^*)$. Now, the algorithms of Badoiu and Clarkson [4] are greedy methods to find a ϵ -core-set of S . Here we present the most usual version of the algorithm [4]

Algorithm 1 Badoiu-Clarkson Algorithm

- 1: Initialize the core-set $\mathcal{C}_{S,\epsilon}$.
 - 2: Compute the minimal-enclosing-ball $\mathcal{B}(\mathcal{C}, \mathbf{c}, R)$ of the core-set $\mathcal{C}_{S,\epsilon}$.
 - 3: **while** A point $\mathbf{z} \in S$ out of the ball $\mathcal{B}(\mathbf{c}, (1 + \epsilon)R)$ exists **do**
 - 4: Include \mathbf{z} in $\mathcal{C}_{S,\epsilon}$.
 - 5: Compute the minimal-enclosing-ball $\mathcal{B}(\mathcal{C}, \mathbf{c}, R)$ of the core-set $\mathcal{C}_{S,\epsilon}$.
 - 6: **end while**
-

It can be shown [4] that this algorithm obtains the desired ϵ approximation in no more than $O(\frac{1}{\epsilon})$ iterations. On the other hand, each iteration adds only one point to the core-set and thus the final size of the core-set is also $O(\frac{1}{\epsilon})$. By controlling ϵ we can hence tradeoff accuracy with convergence time but also with the size of the subset of data summarizing the overall set.

4 Distributed Computation of L2-SVMs

We now focus on the problem of learning an SVM when the training examples are distributed, i.e., fragmented among a set of p remote sources. Let $\mathcal{N} = \{N_1, \dots, N_p\}$ the set of nodes connected one another via an underlying communication network and let $S_j = \{\mathbf{z}_i = \phi(\mathbf{x}_i); i \in I_j\}$ the set of data hosted by N_j . Our task is to efficiently reproduce/approximate the SVM that would be extracted by training on the full data, that is, the union of the datasets $S = S_1 \cup S_2 \cup \dots \cup S_p$. Our approach is based on the extraction of summaries \mathcal{C}_j of each local set S_j which put together will be enough to recover the global SVM. For this purpose we identify a special node C called *coordinator* which will integrate the results of the local tasks but as mentioned then more general ways to organize this second step can be designed.

4.1 Generic Formulation The key idea of our method is to cast an SVM as a MEB problem in a feature space $\tilde{Z} = \phi(\tilde{X})$ where the training examples are embedded via a mapping $\tilde{\phi}$. Hence, we first formulate an algorithm to compute the MEB of the images \tilde{S} of

S in \tilde{Z} when S is decomposed in a collection of subsets S_j . Then we will instantiate the solution for classifiers supporting the reduction to MEB problems.

The algorithm is based on the idea of computing core-sets \mathcal{C}_j for each set $\tilde{S}_j = \tilde{\phi}(S_j)$ and taking its union $\mathcal{C} = \bigcup_j \mathcal{C}_j$ as an approximation to a core-set for $\tilde{S} = \bigcup_j \tilde{S}_j$. The generic procedure is depicted as algorithm (2). In a first step the algorithm extracts a core-set for each subset \tilde{S}_i . In the second step the MEB of the union of the core-sets is computed.

Algorithm 2 Distributed Computation of the MEB of $\tilde{S} = \phi(S)$

Require: A partition of the set S in a collection of subsets S_j and the desired precision ϵ .

- 1: **for** Each subset S_j , $j = 1, \dots, p$ **do**
 - 2: Compute a ϵ -core-set \mathcal{C}_j for $\tilde{S}_j = \phi(S_j)$.
 - 3: Send the core-set to the coordinator.
 - 4: **end for**
 - 5: Join the core-sets $\mathcal{C} = \mathcal{C}_1 \cup \dots \cup \mathcal{C}_p$.
 - 6: Compute the minimal enclosing ball of \mathcal{C} . This is the minimal enclosing ball of \tilde{S} .
-

For the computation of the core-sets at each remote node we use the Badoiu and Clarkson algorithm [4] described in the previous section. In a distributed application the amount of data which is transmitted over the network should be as small as possible. The fulfillment of this requirement depends on the size of the core-sets obtained remotely. As we commented in the previous section, both the size of the core-set and the time incurred until convergence depends monotonically on ϵ , the parameter used by this algorithm to relax the enclosing-ball property. This parameter should hence determine the tradeoff between accuracy and complexity in the distributed solution, the latter measured as local computation times until convergence and communication overhead. Our simulations will show that this property is corroborated in practice.

Another implementation decision concerns the second stage of the algorithm which needs to compute a MEB for the union of the local core-sets. To make the presentation and the simulations simpler, in this work we have implemented this step in single coordinator node that receives the partial results of each node. As depicted in Figure (1), a tree structure as used in [14] to hierarchically filter support vector sets is straightforward to implement. At each internal node a partial union of core-sets can be carried out, sending the results to the next layer to repeat the process. This architecture provides advantages if the internal nodes can work in parallel and they are sufficiently reliable to avoid synchronization delays. Our method can also be embedded

in a completely decentralized peer-to-peer topology as used in [24] replacing the support vector sets, computed at each node, by core-sets.

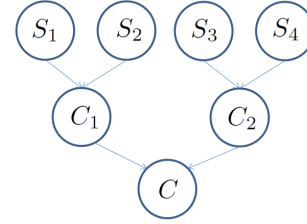


Figure 1: An alternative structure to hierarchically integrate results of local computations.

4.2 Instantiation for the OVO Approach From the previous section we have that training a binary L2-SVM on a dataset S is equivalent to build a minimal-enclosing-ball of S if $\phi(\mathbf{x})^T \phi(\mathbf{x})$ is implemented using the kernel $\tilde{k}(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}$. The OVO procedure to obtain a multi-category SVM works by combining one binary SVM for each pair of classes. An instantiation of procedure (2) would hence consist in computing core-sets for the subset of examples belonging to each pair of classes, then joining them in a coordinator node and finally recovering the binary model for this pair. However, since each class participates in K models, core-sets for each pair of classes can be highly redundant overloading the network unnecessarily. Thus, we proceed as in algorithm (3), joining the core-sets at each node before sending the results to the coordinator node.

Algorithm 3 Distributed Algorithm for One-versus-One L2-SVMs

- 1: **for** Each Remote Node $i = 1, \dots, p$ **do**
 - 2: **for** Each Class $l = 1, \dots, K - 1$ **do**
 - 3: **for** Each Class $m = l + 1, \dots, K$ **do**
 - 4: Let S_i^{ml} the subset of S_i corresponding to classes l and m .
 - 5: Label S_i^{ml} using the standard binary codes $+1$ and -1 for class l and m respectively.
 - 6: Compute a core-set \mathcal{C}_i^{ml} of S_i^{ml} using the kernel $\tilde{k}(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}$.
 - 7: **end for**
 - 8: **end for**
 - 9: Take the union of the core-sets inferred for each pair of classes $\mathcal{C}_i = \mathcal{C}_i^{ml} \cup \dots \cup \mathcal{C}_i^{ml}$.
 - 10: Send the core-set \mathcal{C}_i to the coordinator.
 - 11: **end for**
 - 12: Join the core-sets $\mathcal{C} = \mathcal{C}_1 \cup \dots \cup \mathcal{C}_p$.
 - 13: Compute a standard OVO solution over the reduced set \mathcal{C} taking binary L2-SVMs as base classifiers.
-

4.3 Instantiation for a Direct Method In contrast to the OVO decomposition heuristic, a direct implementation is defined by a single optimization which coincides with a MEB problem just by using the kernel $\tilde{k} = \mathbf{y}_i^T \mathbf{y}_j k(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{y}_i^T \mathbf{y}_j + \frac{\delta_{ij}}{C}$. The use of algorithm (2) is hence straightforward and consists in computing any dot product $\tilde{\phi}(\mathbf{x}_i)^T \tilde{\phi}(\mathbf{x}_i)^T = \tilde{k}(\mathbf{x}_i, \mathbf{x}_i)$ using this kernel. The instantiation is depicted as algorithm (4).

Algorithm 4 Distributed Algorithm for a Direct Multi-Class L2-SVM

- 1: **for** Each Remote Node $j = 1, \dots, p$ **do**
 - 2: Label each example $\mathbf{x}_i \in S_j$ with the code \mathbf{t}_k assigned to the class of \mathbf{x}_i and let \mathbf{y}_i such label.
 - 3: Compute a core-set \mathcal{C}_i of S_i using the kernel $\tilde{k}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{y}_i^T \mathbf{y}_j k(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{y}_i^T \mathbf{y}_j + \frac{\delta_{ij}}{C}$.
 - 4: Send the core-set to the coordinator.
 - 5: **end for**
 - 6: Join the core-sets $\mathcal{C}_S = \mathcal{C}_1 \cup \dots \cup \mathcal{C}_p$
 - 7: Compute the minimal-enclosing-ball of \mathcal{C}_S using the same kernel \tilde{k} used by the remote nodes.
 - 8: From the minimal-enclosing-ball solution recover the variables required to implement the decision function.
-

5 A New Encoding for Direct L2-SVMs

A critical decision in order that the single-objective implementation of the L2-SVM may work is to select an appropriate set of encoding vectors $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_K$ used to represent the classes. There are no theoretical reasons to prefer the encoding originally proposed in [1] and, as we will see in the experiments section, we are not able to obtain good classification results using this method. The problem is that the constraints of the optimization problem (2.3) encourage but do not guarantee correct classification. If a point \mathbf{z} belongs to the class k the projection $\mathbf{W}^T \mathbf{z} + b$ can exhibit a positive dot product with \mathbf{t}_k as required by the constraints but lower than the dot product with the code \mathbf{t}_j associated to wrong class j .

A way to circumvent this problem is to select a set of codes such that a greater projection on one code automatically implies a lower projection on the rest of the codes. For binary classification problems, the codes used by SVMs are optimal since a positive “projection” on $+1$ means automatically a negative projection on the opposite code -1 . For K -dimensional codes we hence propose to select the codes maximizing the projections between different pairs, that is

$$(5.9) \quad \min(\mathbf{t}_1, \dots, \mathbf{t}_K) : \sum_{k, j \neq k} \mathbf{t}_i^T \mathbf{t}_j$$

$$\text{st: } \|\mathbf{t}_k\|^2 = c^2 \quad \forall i \in I$$

where c^2 is a normalizing constant. Geometrically this means that we choose a set of vectors maximizing the angles of separation between them. By the KKT conditions, it is enough that the set of codes satisfy $\sum_k \mathbf{t}_k = 0$ and $\|\mathbf{t}_k\|^2 = c^2$ (the problem is under-constrained and it is clear for example that any permutation of a solution is also a solution). We propose the use of the following solution

$$(5.10) \quad \mathbf{t}_k(j) = \begin{cases} \frac{1}{\|c\|} \frac{K-1}{\sqrt{K(K-1)}} & \text{if the class of } \mathbf{x}_i \text{ is } j \\ \frac{1}{\|c\|} \frac{-1}{\sqrt{K(K-1)}} & \text{otherwise} \end{cases}$$

For $c = 1$ these are the codes used in the multi-class implementation of the L1-SVM proposed in [23]. In this case, the property $\sum_k \mathbf{t}_k = 0$ results fundamental to prove the convergence of the classifier error to the Bayes-Risk. For $c^2 = (K-1)$ we obtain the codes used in [25]. In [25], the codes appear from a geometric construction in the dual space and are not directly formulated. In both cases [23] and [25] the models are intended to extend the binary L1-SVM and the quadratic programs are not equivalent to a ball-enclosing problem.

6 Discussion of Related Work

The main question when designing an algorithm for SVMs in distributed environments is if it is possible to rebuild a globally consistent model from summaries of the data contained at each source, avoiding the full exploration of the dataset. Probably, the first work addressing this problem was [28], where SVMs were incrementally trained on a partition of the full dataset. From section (2) we have that an SVM only depends on a subset of the training examples known as support vectors. This set is hence a natural “summary” of data which contains all the information the SVM needs about the class boundaries. In [28], each incremental step retains only the support vectors obtained after training an SVM with the examples currently available. Its experiments show that only small losses can be observed with respect to an SVM trained with the full dataset. [20], [13] and [12] can be considered direct implementations of [28] in real distributed environments. Support vectors locally detected at a given node are transmitted to the next station which repeats the process including the local data. These works can be considered demonstrations of the effectiveness of the support vectors approach in real environments although the experimental setting is quite poor: in [20] only estimations of energy consumption are provided while in [12] and [13] only one synthetic dataset is used for experimentation.

[5] and [6] are probably the first works addressing the problem of distributed SVM estimation with guar-

antees of global consistency and optimality. In this algorithm each node computes the support vectors corresponding to a locally optimal SVM and sends the result to a coordinator node. The coordinator computes the union of all the support vectors and sends them back to each node in the network. This process is repeated until convergence in order to guarantee optimality. Similarly, in [15] an iterative exchange of support vectors over a decentralized structure is used. Each iteration broadcasts the local support vectors and this process is iterated until convergence. These techniques lead hence to a communication overhead which can be infeasible in open distributed environments. In [14] and [24] the topology of the network connecting the nodes is considered in the design of the algorithm. [14] proposes a hierarchical tree structure where leaves correspond to nodes hosting data and the internal nodes are used to join the support vectors filtered by the child nodes and to filter again. The data coming out from the root are sent back to each leaf to repeat the process. Only by iterating this scheme it is possible to guarantee a globally consistent SVM. Going beyond, [24] studied an implementation of the support vector approach on arbitrary strongly connected networks. For our technique, alternative topologies to join the core-sets could lead to important improvements in practice just like [14] and [24] improve [6]. This issue as well as extensions to the peer-to-peer architectures are being currently addressed.

Finally, general frameworks to deal with distributed data mining have been recently proposed [32, 10]. These architectures have been mainly conceived to monitor the global consistency of a function (already known and of a constrained form) that needs to be evaluated on a full dataset which is constantly changing. When an inconsistency is detected, the framework triggers stages of data exchange. An efficient implementation of this step highly depend on the specific function being estimated. We deal in contrast with the problem of estimating an unknown decision function with distributed data, focusing precisely on an efficient procedure to extract data summaries. This method exploits the characteristics of the model being implemented. This algorithm could be embedded in a more general data analysis system like [32] in order to support dynamic datasets and deal with low level implementation issues that are critical in real world applications.

7 Experiments

7.1 Methodology. Datasets: We provide simulations of our procedure for 3 small, 2 medium and 5 large multi-category classification problems: **Iris** (150 examples, 3 classes); **Wine** (178 examples, 6 classes); **Glass** (178 examples, 6 classes); **Pendigits** (7494 examples for

training, 3498 for testing, 10 classes); **Usps** (7291 examples for training, 2007 for testing, 10 classes); **Protein** (17766 examples for training, 6621 for testing, 3 classes); **Letter** (15000 examples for training, 5000 for testing, 26 classes); **Shuttle** (43500 examples for training, 14500 for testing, 7 classes); **Mnist** (60000 examples for training, 10000 for testing, 10 classes) and **Kdd99** (494091 examples, 5 classes). For the Iris, Wine, Glass and Kdd99 problems a 20% of data randomly selected from the dataset is reserved as a test set. These last datasets are publicly available at [16] and the rest at [7].

Simulation: We simulate the distributed environment by partitioning the dataset among fragments on which local computations are carried out. Results of local computations are then joined in order to simulate the work at a centralized node. The statistics collected during the experiments are hence logic measurements of complexity and effectiveness, independent of the physical and computational implementation of the distributed environment. Data fragments are generated randomly. Although other mechanisms to partitionate the data were evaluated (weighted random and based on clustering), the results remain quite the same and cannot be presented due to space constraints.

Measurements: Our main research hypothesis is that the distributed algorithm based on core-sets can obtain the same test accuracy than the centralized version even if the latter works with the full dataset and the first is constrained to locally select which information is useful to build the global model. Since there are no advantages if this result implies sending all the data to the central node we need to monitor how efficient the distributed solution is.

We define the following variables: *test accuracy* as the fraction of the test set correctly classified by the obtained model, *size* of a SVM model as the number of support vectors (subset of examples effectively participating in the decision function), *compression* as the fraction of the training data that the distributed solution needs to transmit to the central node in order to compute the final model and *complexity* as the number of kernel evaluations (inner products in the feature space) carried out by the algorithm. For the distributed solutions we take the sum of the maximum number of kernel evaluations among the nodes (local computations) and the number of kernel evaluations carried out to compute the global model (central node), considering that the local computations are independent and can be done in parallel. As it will be explained soon, taking the sum of kernel evaluations among the nodes gives a similar result. Finally we define the *vector selection efficiency* as the fraction of data transmitted to the central node that effectively takes part in the final model. This

statistic measures the effectiveness of local computation in selecting vectors which are useful for the final classification model. In other words, good vector selection efficiency indicates that the distributed computation has little impact on communication costs.

Statistics: For each measurement we compute a mean and a standard deviation. These are computed among 10 different partitions of the full dataset in order to eliminate the effect of a particular way in which data could be distributed. For the datasets without a predefined test set, 20 random realizations of train/test set are tested leading to a total of 200 trials.

Compared Algorithms: As detailed in previous sections we study two different methods to implement the core-set based method for multi-category SVMs: one based on the OVO decomposition scheme (referred as **OVO-L2**) and a direct implementation (referred as **DL2-AD**). Since a previous multi-category extension of SVMs supporting a MEB reduction has been proposed in [1], we include this method in the experiments with the name **DL2-AS**. We compare these techniques with the support vector heuristic (referred as **OVO-L1**) which has been discussed in the section of related work. In order to measure the effectiveness of these algorithms we need to compare each of them with their *centralized versions*, which consists in putting all the data in a centralized node and computing the global model from the full dataset, avoiding completely the effect of the partition and local computations based on partial knowledge of the problem.

SVM Hyper-parameters: SVMs are trained using a gaussian kernel $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2/\sigma^2)$. Parameter σ and parameter C in the optimization problem are estimated in two different ways: for datasets of less than 10000 examples a standard cross-validation process is carried out. For large datasets σ^2 is estimated as the averaged squared distance among training examples and C is determined using a validation set (30% of the training-set). Parameter selection is considered correct since the accuracies obtained for centralized training of $L1$ and $L2$ SVMs correspond (within quite small differences) to results reported recently in the literature.

7.2 Results. Prediction Accuracy: Figure (2, top panel) presents the test-accuracy of the different distributed algorithms and their centralized implementations. For this experiment, the number of nodes to simulate the distributed scenario was fixed to 5 for the datasets with less than 10000 training examples and 10 for the larger problems (soon we will evaluate the techniques by increasing the number of nodes up to 1000.) In these and the following figures error-bars of 1 and 2 standard deviations are depicted around the mean rep-

resented as a colored bar. For our method, a precision of 10^{-6} was used for the computation of the MEBs.

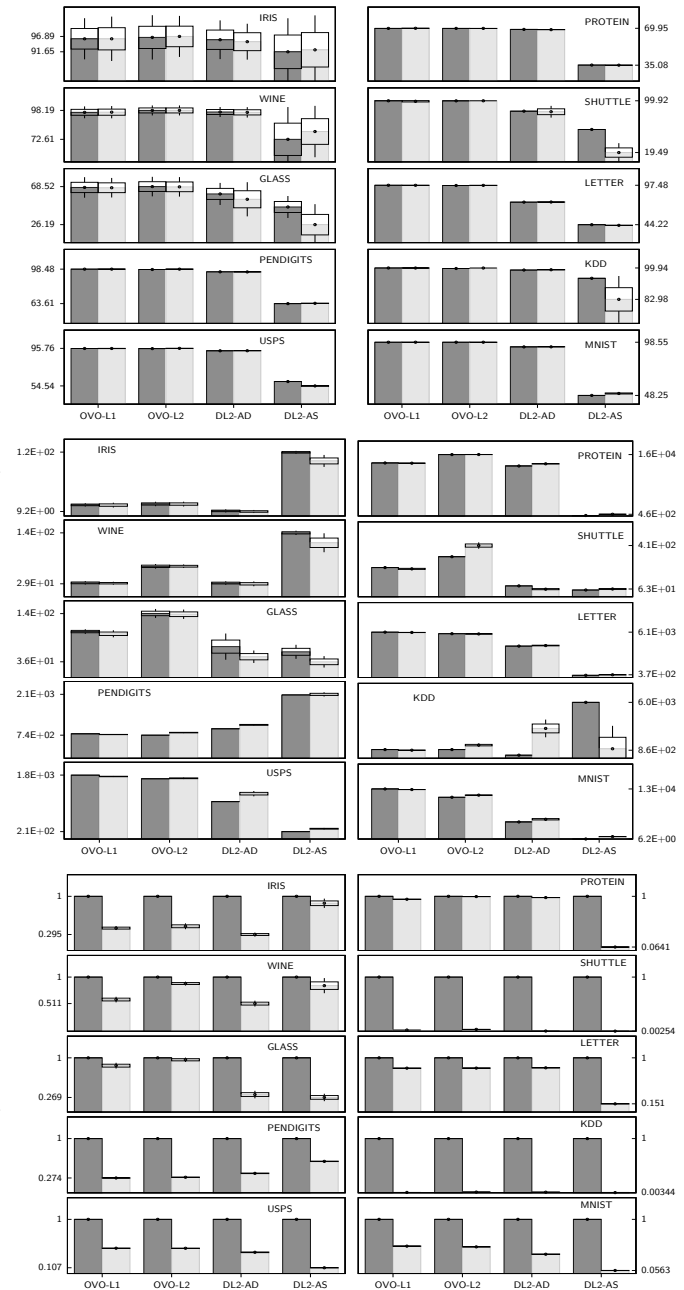


Figure 2: Test accuracy(top), Model-size (mid panel and compression (bottom) for each algorithm (see x -axis). Centralized training in dark-grey and distributed light-grey.

The figure shows that our technique and the support-vector heuristic, both based on the OVO approach to handle multiple classes, are highly accurate obtaining the same accuracy than the full centralized versions. Our direct implementation (DL2-AD) is less accurate than the OVO implementation of the algo-

rithm but it is highly more accurate than the previously proposed method to handle multi-class that supports a MEB reduction (DL2-AS) [1]. Even if the direct implementation offers a lower performance in terms of classification accuracy, it approximates well its centralized version, suggesting that the core-sets strategy is still working (the union of the local core-sets provides a good approximation to a global core set) and improvements in the centralized formulation could be translated to the distributed scenario.

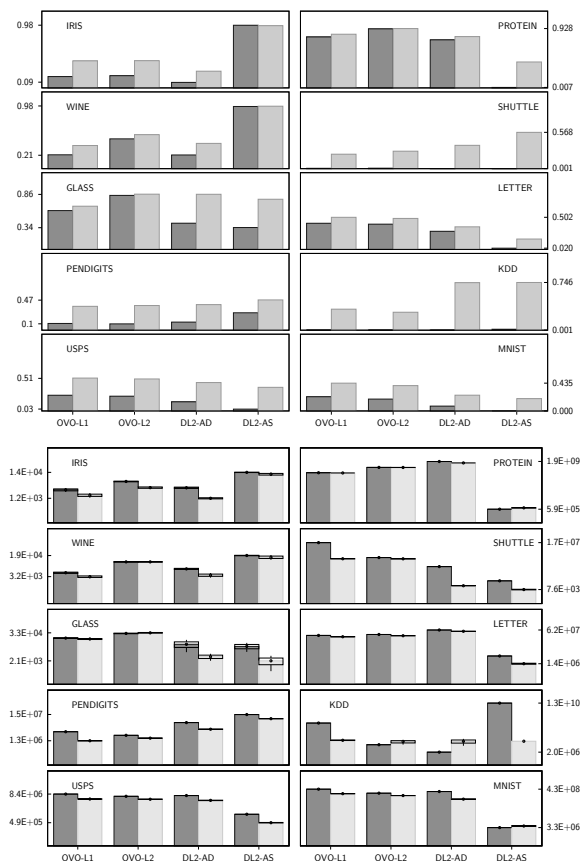


Figure 3: Selection efficiency (top) and complexity (bottom) for each algorithm (see x -axis). Centralized training in dark-grey and distributed light grey.

Complexity, Compression and Efficiency To evaluate the cost at which the distributed solutions can reproduce the centralized performance, we present in Figures (2) and (3) the size, compression, data selection efficiency and complexity as defined previously. We find that our technique implemented with the OVO approach obtains models of quite comparable size with respect to the centralized version, suggesting that the relevant data from nodes is correctly detected and send to the central node. A similar outcome holds for the support vector heuristic working with the OVO approach which results in turn comparable to our method. The

direct implementations obtains smaller models but as we can see from figure (2) at the price of a lower accuracy. The fraction of data that the distributed method needs to put in the central node (compression) tends to be very small for the large datasets. Note that for a fully centralized solution this measurement is always 1 since this approach puts all the data on a central node. Selection efficiency in the bottom-left panel shows that the cases in which a large fraction of data is imported comply to a real importance of these data to build the final model. Finally the bottom-right panel shows that the complexity of the distributed solutions can be lower than the centralized versions. If we measure complexity as the sum of kernel evaluations among the nodes (instead of the maximum among them) the statistics are similar (we omit the figure for space constraints). This suggests that most of the work is done in the central node.

Effect of Parameter ϵ . Figures (4), (5) and (6) show the behavior of the SVM models implemented by the core-sets method againsts the value of parameter ϵ . Parameter ϵ controls the degree of relaxation of the enclosing-property of MEBs and hence explicitly controls the quality of the core-sets obtained from remote nodes (data fragments). As expected, the classification accuracy of the model increases monotonically as ϵ becomes smaller, as well as the complexity and compression measurements. Previous experiments were carried out using a default $\epsilon = 10^{-1}$, however the results show that the value determining the optimal tradeoff between accuracy and complexity, if problem-dependent, can be as coarse as 10^{-3} . For the larger datasets (KDD99 and Shuttle), for example, the complexity is significantly reduced by using a coarser value of ϵ (compared to the default value used in our previous experiments) which suggests interesting lines of future research. The support vector heuristic cannot exploit this property because it does not explore the data in an incremental way as our solution does.

Scalability. In order to determine the robustness of the proposed methods to the degree of distribution of the data we conduct an experiment by increasing the number of nodes from 10 to 1000. For this purpose we use the datasets KDD99 and Mnist, restricting also the analysis to the models which have shown the highest accuracy: our method based on cores-sets and the support vector method, both using the OVO approach to handle multiple classes. The figure shows that the accuracy is highly independent of the number of nodes and hence the distributed solutions are able to reproduce the result of a fully centralized training even with a high degree of fragmentation.

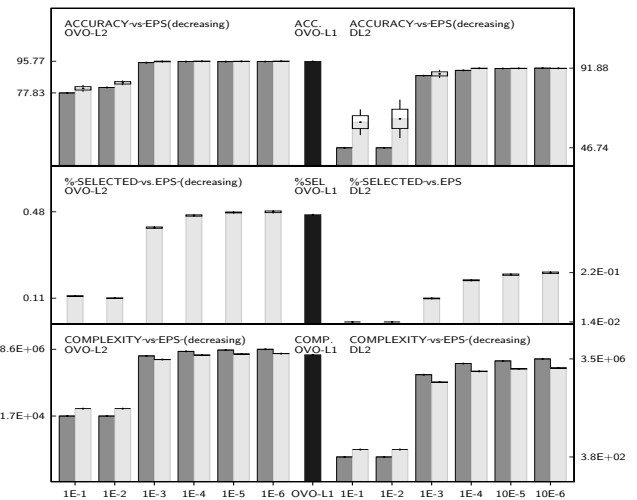
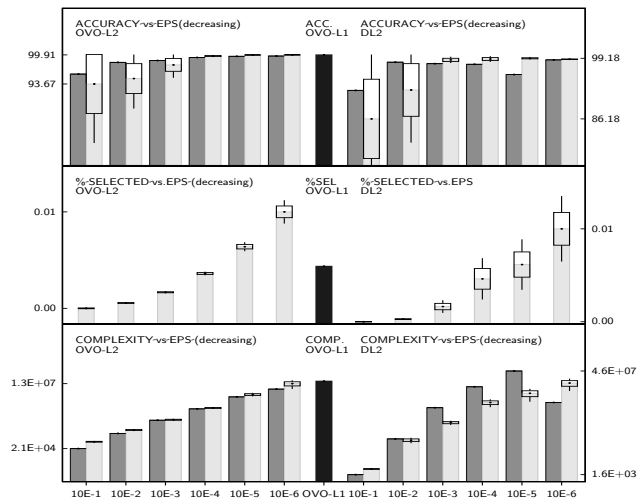
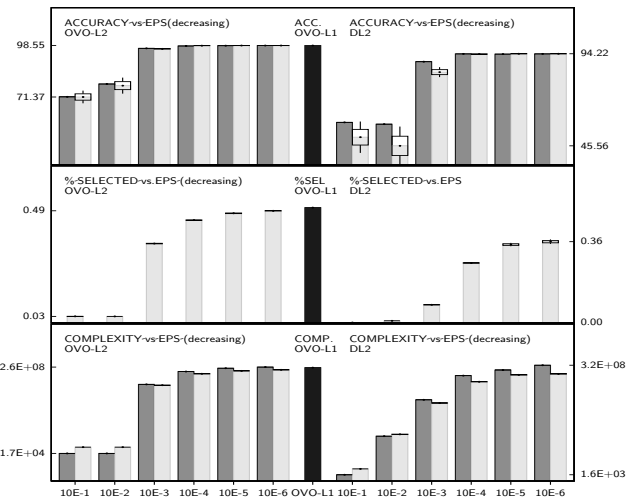
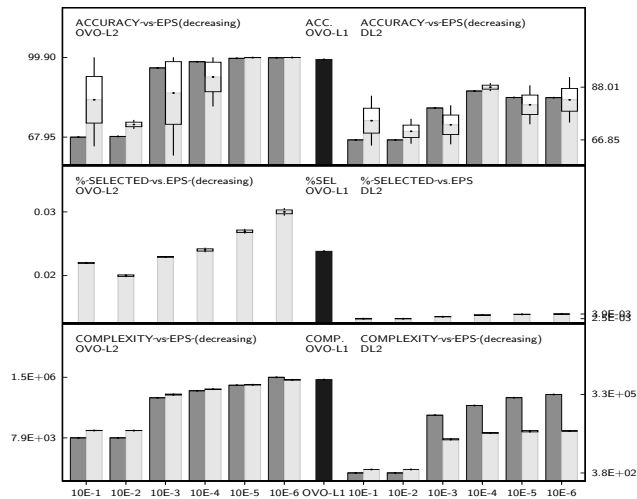
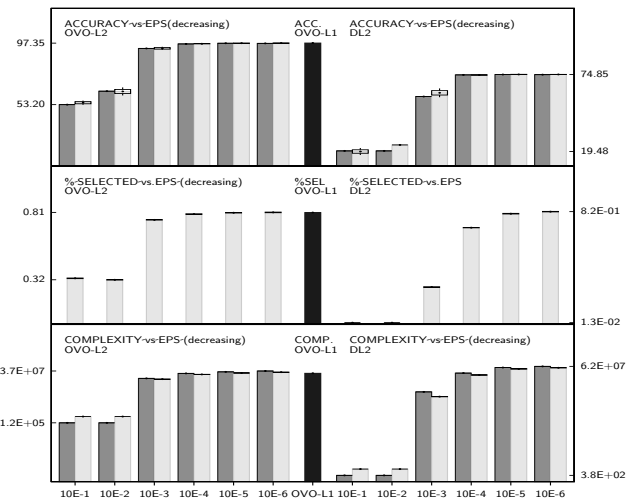
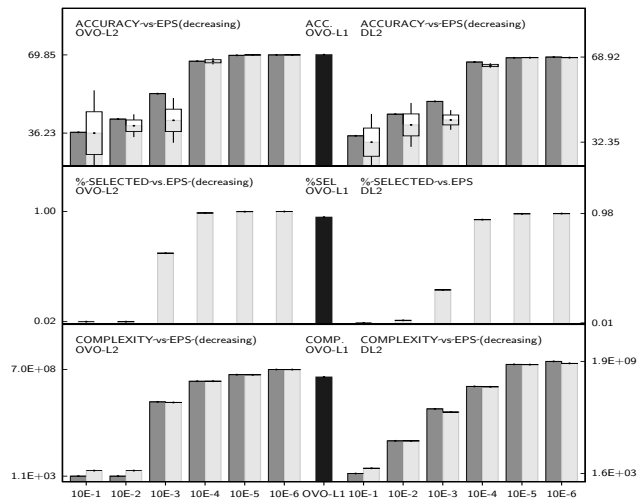


Figure 4: Algorithm's behavior versus ϵ (EPS). From top to bottom: Protein, Shuttle and Kdd99 datasets. Within each panel: OVO-L2 (at left), OVO-L1 (at center) and DL2 (at right). Dark-grey is used for centralized versions of the algorithms and light-grey for distributed ones.

Figure 5: Algorithm's behavior versus ϵ (EPS). From top to bottom: Letter, Mnist and Usps datasets. Within each panel: OVO-L2 (at left), OVO-L1 (at center) and DL2 (at right). Dark-grey is used for centralized versions of the algorithms and light-grey for distributed ones.

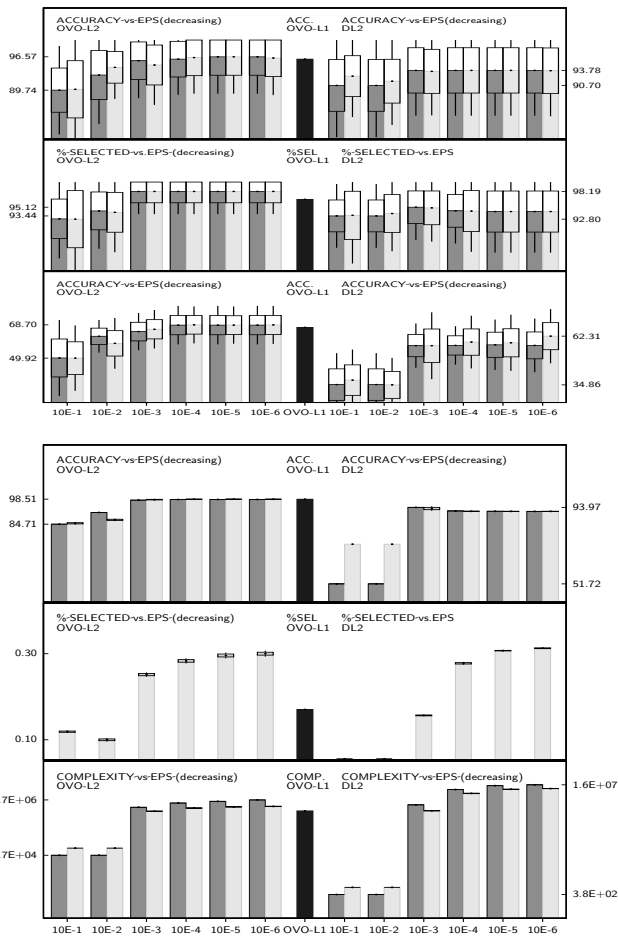


Figure 6: At top: accuracy versus ϵ (EPS) for the small datasets (from top to bottom subpanels: Iris, Wine and Glass). At bottom algorithm's behavior versus ϵ (EPS) for the Pendigits dataset. Within each panel: OVO-L2 (at left), OVO-L1 (at center) and DL2 (at right). Dark-grey is used for centralized training and light-grey for distributed one.

For the datasets considered in this experiment the method based on core-sets exhibits a better scalability in terms of complexity than the support vector heuristic for the standard SVM.

8 Conclusions

We have explored two methods based on the computation of core-sets to train multi-category SVM models when the set of examples is distributed in fragments among a network of nodes. The main contribution has been to demonstrate through a large set of experiments, that the method can reproduce with high fidelity the accuracy of a solution based on the centralization of the full dataset, without complex and costly communication steps among the nodes. The method shows to be as accurate as the widely used support vector heuristic for standard SVMs. SVMs based on core-sets have

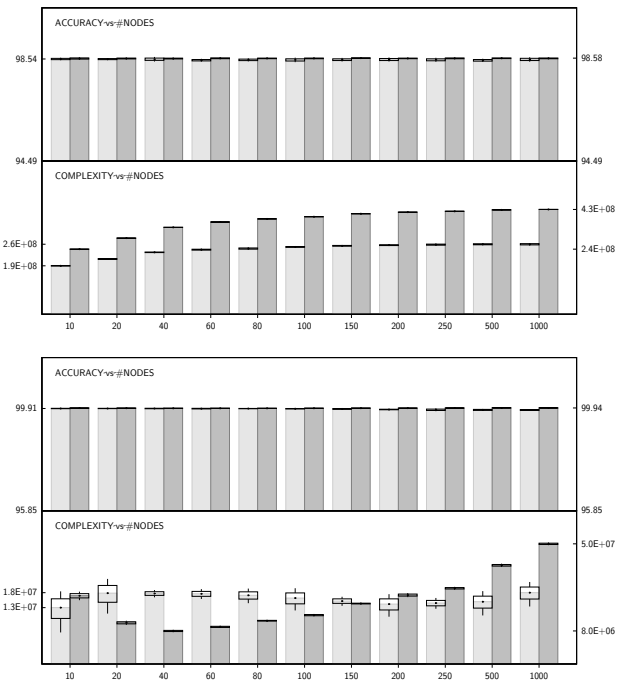


Figure 7: Algorithm's behavior versus Number of Nodes. At top Mnist and at bottom Kdd99. Dark-grey is used for the support vector heuristic (implemented with OVO) and light-grey for the core-sets method (implemented with OVO).

shown however important advantages in large-scale applications [30] [29] which can hence be extended to distributed data-mining problems. The method shows to scale considerably well with the degree of fragmentation of the data, being its accuracy highly robust to the number of nodes in the network.

The accuracy/complexity tradeoff of the final solution can be controlled through a parameter which defines the degree of relaxation with which local and centralized core-sets are estimated. This parameter monotonically determines accuracy, complexity and efficiency in all the problems evaluated in this work. To know in advance the value at which accuracy does not increase significantly but complexity does could report important benefits in real applications. The exact procedure to exploit this advantage of the algorithm based on core-sets is matter of current research.

Complexity of all the distributed solutions we have studied seems to be dominated by the computation carried out at the central node. This suggests that a hierarchical organization of the process through which local results are integrated up to the final model could report benefits of computational performance. This is another direction to which future research efforts are being conducted.

A secondary contribution of this work has been a

new direct implementation of multi-category SVMs supporting a reduction to a minimal-enclosing-ball (MEB) problem. Although the core-sets method exhibits always better prediction accuracy used with the OVO scheme, the direct implementation shows a lower complexity and it is significantly better than the previous direct implementation proposed for MEB based SVMs.

References

- [1] S. Asharaf, M. Murty, and S.K. Shevade, *Multiclass core vector machine*, ICML'07, ACM, 2007, pp. 41–48.
- [2] A. Bar-Or, D. Keren, A. Schuster, and R. Wolff, *Hierarchical decision tree induction in distributed genomic databases*, IEEE Transactions on Knowledge and Data Engineering **17** (2005), no. 8, 1138–1151.
- [3] K. Bhaduri and H. Kargupta, *A scalable local algorithm for distributed multivariate regression*, Statistical Analysis and Data Mining **1** (2008), no. 3, 177–194.
- [4] M. Bădoiu and K. Clarkson, *Optimal core-sets for balls*, Comput. Geom. Theory Appl. **40** (2008), no. 1, 14–22.
- [5] D. Caragea, *Learning classifiers from distributed, semantically heterogeneous, autonomous data sources*, Ph.D. thesis, Iowa State University, 2004.
- [6] D. Caragea, A. Silvescu, and V. Honavar, *Learning support vector machines from distributed data sources*, Proceedings of the 25th AAAI, 2005, pp. 1602–1603.
- [7] C. Chang and C. Lin, *LIBSVM: a library for support vector machines*, 2010, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [8] C. Cortes and V. Vapnik, *Support-vector networks*, Machine Learning **20** (1995), no. 3, 273–297.
- [9] K. Crammer and Y. Singer, *On the algorithmic implementation of multiclass kernel-based vector machines*, JMLR (2001), no. 2, 265–292.
- [10] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta, *Distributed data mining in peer-to-peer networks*, IEEE Internet Computing **10** (2006), no. 4, 18–26.
- [11] R. Fan, P. Chen, and C. Lin, *Working set selection using second order information for training support vector machines*, JMLR **6** (2005), 1889–1918.
- [12] E. Fluori, B. Lonzano, and P. Tsakalides, *Training a svm-based classifier in distributed sensor networks*, Proceedings of the EUSIPCO'06, 2006.
- [13] ———, *Optimal gossip algorithm for distributed consensus svm training in wireless sensor networks*, Proceedings of the DSP'09, 2009.
- [14] H. Graf, E. Cosatto, L. Bottou, I. Durdanovic, and V. Vapnik, *Parallel support vector machines: The cascade svm*, NIPS, MIT Press, 2005, pp. 521–528.
- [15] T. Hazan, A. Man, and A. Shashua, *A parallel decomposition solver for svm: Distributed dual ascend using fenchel duality*, CVPR'08 (2008), 1–8.
- [16] S. Hettich and S. Bay, *The UCI KDD archive*, 2010, <http://kdd.ics.uci.edu>.
- [17] T. Hofmann, B. Schölkopf, and A. Smola, *Kernel methods in machine learning*, Annals of Statistics **36** (2008), no. 3, 1171–1220.
- [18] C. Hsu and C. Lin, *A comparison of methods for multiclass support vector machines*, IEEE Transactions on Neural Networks **13** (2002), no. 2, 415–425.
- [19] S. S. Keerthi, S. K. Shevade, and C. Bhattacharyya, *A fast iterative nearest point algorithm for support vector machine classifier design*, IEEE Transactions on Neural Networks **11** (2000), 124–136.
- [20] S. Kim, M.A. Azim, and J.S. Park, *Privacy preserving support vector machines in wireless sensor networks*, ARES'08, 2008, pp. 1260–1265.
- [21] Y. Koshiba and S. Abe, *Comparison of l1 and l2 support vector machines*, Proceedings of the IJCNN'03, vol. 3, IEEE, 2003, pp. 2054 – 2059.
- [22] U. Kressel, *Pairwise classification and support vector machines*, Advances in kernel methods: support vector learning, MIT Press, 1999, pp. 255–268.
- [23] Y. Lee, Y. Li, and G. Wahba, *Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data*, Journal of the American Statistical Association **99** (2004), no. 465, 67–81.
- [24] Y. Lu, V. Roychowdhury, and L. Vandenberghe, *Distributed parallel support vector machines in strongly connected networks*, IEEE Transactions on Neural Networks **19** (2008), no. 7, 1167–1178.
- [25] R. Nănculef, C. Concha, H. Allende, and C. Moraga, *Ad-svms: A light extension of svms for multicategory classification*, International Journal of Hybrid Intelligent Systems **6** (2009), no. 2, 69–79.
- [26] B. Schölkopf and A. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*, MIT Press, 2001.
- [27] M. Scholz, *On the tractability of rule discovery from distributed data*, ICDM'05, 2005, pp. 761–764.
- [28] N. Syed, H. Liu, and K. Sung, *Incremental learning with support vector machines*, Proceedings of the IJ-CAI'99, 1999.
- [29] I. Tsang, A. Kocsor, and J. Kwok, *Simpler core vector machines with enclosing balls*, ICML '07, ACM, 2007, pp. 911–918.
- [30] I. Tsang, J. Kwok, and P.M. Cheung, *Core vector machines: Fast svm training on very large data sets*, JMLR **6** (2005), 363–392.
- [31] V. Vapnik, *The nature of statistical learning theory*, Springer-Verlag, 1995.
- [32] R. Wolff, K. Bhaduri, and H. Kargupta, *A generic local algorithm for mining data streams in large distributed systems*, IEEE Trans. on Knowl. and Data Eng. **21** (2009), no. 4, 465–478.
- [33] E. Alper Yildirim, *Two algorithms for the minimum enclosing ball problem*, SIAM Journal on Optimization **19** (2008), no. 3, 1368–1391.
- [34] Y. Lee and O. Mangasarian, *Rsvm: reduced support vector machines*, SDM'01, SIAM, 2001, pp. 325–361.