Journal of Machine Learning Research 6 (2005) 817–850 Submitted 2/05; Revised 5/05; Published 5/05

Multiclass Classification with Multi-Prototype Support Vector Machines

Fabio Aiolli AIOLLI@MATH.UNIPD.IT
Alessandro Sperduti SPERDUTI@MATH.UNIPD.IT
Dip. di Matematica Pura e Applicata
Università di Padova
Via G. Belzoni 7
35131 Padova, Italy

Editor: Yoram Singer

Abstract

Winner-take-all multiclass classifiers are built on the top of a set of prototypes each representing one of the available classes. A pattern is then classified with the label associated to the most 'similar' prototype. Recent proposal of SVM extensions to multiclass can be considered instances of the same strategy with one prototype per class.

The multi-prototype SVM proposed in this paper extends multiclass SVM to multiple prototypes per class. It allows to combine several vectors in a principled way to obtain large margin decision functions. For this problem, we give a compact constrained quadratic formulation and we propose a greedy optimization algorithm able to find locally optimal solutions for the non convex objective function.

This algorithm proceeds by reducing the overall problem into a series of simpler convex problems. For the solution of these reduced problems an efficient optimization algorithm is proposed. A number of pattern selection strategies are then discussed to speed-up the optimization process. In addition, given the combinatorial nature of the overall problem, stochastic search strategies are suggested to escape from local minima which are not globally optimal.

Finally, we report experiments on a number of datasets. The performance obtained using few simple linear prototypes is comparable to that obtained by state-of-the-art kernel-based methods but with a significant reduction (of one or two orders) in response time.

Keywords: multiclass classification, multi-prototype support vector machines, kernel machines, stochastic search optimization, large margin classifiers

1. Introduction

In multiclass classification, given a set of labelled examples with labels selected from a finite set, an inductive procedure builds a function that (hopefully) is able to map unseen instances to their appropriate classes. In this work, we exclusively focus on the *single-label* version of the multiclass classification problem in which instances are associated with *exactly one* element of the label set. However, throughout this paper, we will refer to this problem simply as multiclass problem. Binary classification can be considered a particular instance of the multiclass setting where the cardinality of the label set is two.

Multiclass classifiers are often based on the *winner-take-all* (WTA) rule. WTA based classifiers define a set of prototypes, each associated with one of the available classes from a set $\mathcal{Y}$. A scoring function $f : \mathcal{X} \times \mathcal{M} \to \mathbb{R}$ is then defined, measuring the similarity of an element in $\mathcal{X}$ with prototypes defined in a space $\mathcal{M}$. For simplicity, in the following, we assume $\mathcal{M} \equiv \mathcal{X}$. When new instances are presented in input, the label that is returned is the one associated with the most 'similar' prototype:

$$H(\mathbf{x}) = \mathcal{C}\left(\arg\max_{r\in\Omega} f(\mathbf{x}, M_r)\right) \tag{1}$$

where Ω is the set of prototype indexes, the M_r's are the prototypes and $\mathcal{C} : \Omega \to \mathcal{Y}$ the function returning the class associated to a given prototype. An equivalent definition can also be given in terms of the minimization of a distance or loss (these cases are often referred to as *distance-based* and *loss-based* decoding respectively).

1.1 Motivations and Related Work

Several well-known methods for binary classification, including neural networks (Rumelhart et al., 1986), decision trees (Quinlan, 1993), k-NN (see for example (Mitchell, 1997)), can be naturally extended to the multiclass domain and can be viewed as instances of the WTA strategy. Another class of methods for multiclass classification are the so called *prototype based methods*, one of the most relevant of which is the *learning vector quantization* (LVQ) algorithm (Kohonen et al., 1996). Although different versions of the LVQ algorithm exist, in the more general case these algorithms quantize input patterns into codeword vectors c_i and use these vectors for 1-NN classification. Several codewords may correspond to a single class. In the simplest case, also known as LVQ1, at each step of the codewords learning, for each input pattern $\mathbf{x}_i$, the algorithm finds the element c_k closest to $\mathbf{x}_i$. If that codeword is associated to a class which is the same as the class of the pattern, then c_k is updated by $c_k \leftarrow c_k + \eta(t)(\mathbf{x}_i - c_k)$ thus making the prototype get closer to the pattern, otherwise it is updated by $c_k \leftarrow c_k - \eta(t)(\mathbf{x}_i - c_k)$ thus making the prototype farther away. Other more complicated versions exist. For example, in the LVQ2.1, let y be the class of the pattern, at each step the closest codeword of class $c \neq y$ and the closest codeword of class y are updated simultaneously. Moreover, the update is done only if the pattern under consideration falls in a "window" which is defined around the midplane between the selected codewords.

When the direct extension of a binary method into a multiclass one is not possible, a general strategy to build multiclass classifiers based on a set of binary classifiers is always possible, the so called *error correcting output coding* (ECOC) strategy, originally proposed by Dietterich and Bakiri in (Dietterich and Bakiri, 1995). Basically, this method codifies each class of the multiclass problem as a fixed size binary string and then solves one different binary problem for each bit of the string. Given a new instance, the class whose associated string is most 'similar' to the output of the binary classifiers on that instance is returned as output. Extensions to codes with values in $\{-1, 0, +1\}$ (Allwein et al., 2000) and continuous codes (Crammer and Singer, 2000) have been recently proposed.

Recently, large margin kernel-based methods have shown state-of-the-art performance in a wide range of applications. They search for a large margin linear discriminant model in a typically very high dimensional space, the *feature space*, where examples are implicitly mapped via a function $\mathbf{x} \mapsto \phi(\mathbf{x})$. Since kernel-based algorithms use only dot products in this space, it is possible to resort

to the 'kernel trick' when dot products can be computed efficiently by means of a kernel function $k(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ defined in terms of the original patterns. Examples of kernel functions are the polynomial kernel

$$k(\mathbf{x},\mathbf{y}) = (\langle \mathbf{x},\mathbf{y} \rangle + u)^d, u \geq 0, d \in \mathbb{N}$$

of which the linear case is just an instance ($d = 1$) and the radial basis function (RBF) kernel

$$k(\mathbf{x},\mathbf{y}) = \exp(-\lambda ||\mathbf{x}-\mathbf{y}||^2), \lambda \geq 0.$$

Kernel machines, and the SVM in particular, has been initially devised for the binary setting. However, extensions to the multiclass case have been promptly proposed (e.g. Vapnik, 1998; Weston and Watkins, 1999; Guermeur et al., 2000; Crammer and Singer, 2000).

The discriminant functions generated by general kernel-based methods are implicitly defined in terms of a subset of the training patterns, the so called *support vectors*, on the basis of a linear combination of kernel products $f(\mathbf{x}) = \sum_{i \in SV} \alpha_i k(\mathbf{x}_i, \mathbf{x})$. In the particular case of the kernel function being linear, this sum can be simplified in a single dot product. When this is not the case, the implicit form allows to elegantly deal with non linear decision functions obtained by using non linear kernels. In this last case, the efficiency with respect to the time spent for classifying new vectors tends to be low when the number of support vectors is large. This has motivated some recent works, briefly discussed in the following, whose aim was at building kernel-based machines with a minimal number of support vectors.

The *relevance vector machine* (RVM) in (Tipping, 2001) is a model used for regression and classification exploiting a probabilistic Bayesian learning framework. It introduces a prior over the weights of the model and a set of hyperparameters associated to them. The form of the RVM prediction is the same as the one used for SVM. Sparsity is obtained because the posterior distributions of many of the weights become sharply peaked around the zero. Other interesting advantages of the RVM are that it produces probabilistic predictions and that it can be applied to general functions and not only to kernel functions satisfying the Mercer's condition. The *minimal kernel classifier* (MKC) in (Fung et al., 2002) is another model theoretically justified by linear programming perturbation and a bound on the leave-one-out error. This model uses a particular loss function measuring both the presence and the magnitude of an error. Finally, quite different approaches are those in (Schölkopf et al., 1999; Downs et al., 2001) that try to reduce the number of support vectors after the classifiers have been constructed.

The approach we propose here gives an alternative method to combine simple predictors together to obtain large margin multiclass classifiers. This can be extremely beneficial for two main reasons. First, adding prototypes can produce higher margin decision functions without dramatically increasing the complexity of the generated model. This can be trivially shown by considering that the single-prototype margin is a lower bound on the margin for multi-prototype since it can be obtained when all the prototypes of the same class coincide. Second, combining several simple models can be advisable when no a priori knowledge is available about the task at hand. In the following, we will study only the linear version of the algorithm without exploring more complex kernel functions, the rationale being that adding linear prototypes in the original space allows to increase the expressiveness of the decision functions without requiring the (computationally expensive) use of kernels. Moreover, linearity makes easier the interpretation of the produced models, which can be useful in some particular tasks, and allows for an easier extension to the on-line set-

ting since the explicit representation for the models can be used.

In Section 2 we give some preliminaries and the notation we adopt along the paper. Then, in Section 3 we derive a convex quadratic formulation for the easier problem of learning one prototype per class. The obtained formulation can be shown to be equivalent, up to a change of variables and constant factors, to the one proposed by Crammer and Singer (2000). When multiple prototypes are introduced in Section 4, the problem becomes not convex in general. However, in Section 5 we will see that once fixed an appropriate set of variables, the reduced problem is convex. Moreover, three alternative methods are given for this optimization problem and heuristics for the "smart" selection of patterns in the optimization process are proposed and compared. Then, in Section 6 we give a greedy procedure to find a locally optimal solution for the overall problem and we propose an efficient stochastic-search based method to improve the quality of the solution. In Section 7 we give theoretical results about the generalization ability of our model. Specifically, we present an upper bound on the leave-one-out error and upper bounds on the generalization error. Finally, the experimental work in Section 8 compares our linear method with state-of-the-art methods, with respect to the complexity of the generated solution and with respect to the generalization error.

This paper substantially extends the material contained in other two conference papers. Namely, (Aiolli and Sperduti, 2003) which contains the basic idea and the theory of the multi-prototype SVM together with preliminary experimental work and (Aiolli and Sperduti, 2002a) which proposes and analyzes selection heuristics for the optimization of multiclass SVM.

2. Preliminaries

Let us start by introducing some definitions and the notation that will be used in this paper. We assume to have a labelled training set $\mathcal{S} = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\}$ of cardinality n, where $\mathbf{x}_i \in \mathcal{X}$ are the examples in a inner-product space $\mathcal{X} \subseteq \mathbb{R}^d$ and $c_i \in \mathcal{Y} = \{1, \ldots, m\}$ the corresponding class or label. To keep the notation clearer, we focus on the linear case where kernels are not used. However, we can easily consider the existence of a feature mapping $\phi : I \to \mathcal{X}$. In this case, it is trivial to extend the derivations we will obtain to non-linear mappings $\mathbf{x} \mapsto \phi(\mathbf{x})$ of possibly non vectorial patterns by substituting dot products $\langle \mathbf{x}, \mathbf{y} \rangle$ with a suited kernel function $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ and the squared 2-norm $||\mathbf{x}||^2$ with $k(\mathbf{x}, \mathbf{x})$ consequently. The kernel matrix $K \in \mathbb{R}^{n \times n}$ is the matrix containing the kernel products of all pairs of examples in the training set, i.e. $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

We consider dot-product based WTA multiclass classifiers having the form

$$H_M(\mathbf{x}) = \mathcal{C}\left(\arg\max_{r \in \Omega} \langle M_r, \mathbf{x} \rangle\right) \tag{2}$$

where Ω is the set of prototype indices and the prototypes are arranged in a matrix $M \in \mathbb{R}^{|\Omega| \times d}$ and $\mathcal{C} : \Omega \to \mathcal{Y}$ the function that, given an index r, returns the class associated to the r-th prototype. We also denote by y_i^r, $1 \leq i \leq n$, $r \in \Omega$, the constant that is equal to 1 if $\mathcal{C}(r) = c_i$ and -1 otherwise. Moreover, for a given example $\mathbf{x}_i$, $\mathcal{P}_i = \{r \in \Omega : y_i^r = 1\}$ is the set of 'positive' prototypes for the example $\mathbf{x}_i$, i.e. the set of prototype indices associated to the class of $\mathbf{x}_i$, while $\mathcal{N}_i = \Omega \setminus \mathcal{P}_i = \{r \in \Omega : y_i^r = -1\}$ is the set of 'negative' prototypes, i.e. the set of prototype indices associated to classes different from the class of $\mathbf{x}_i$. The dot product $f_r(\mathbf{x}) = \langle M_r, \mathbf{x} \rangle$ is referred to as the *similarity score*

(or simply *score*) of the r-th prototype vector for the instance $\mathbf{x}$. Finally, symbols in bold represent vectors and, as particular case, the symbol $\mathbf{0}$ represents the vector with all components set to 0.

3. Single-Prototype Multi-Class SVM

One of the most effective multi-class extension of SVM has been proposed by Crammer and Singer (2000). The resulting classifier is of the same form of Eq. (2) where each class has associated exactly one prototype, i.e. $\Omega \equiv \mathcal{Y}$ and $\forall r \in \Omega,\ \mathcal{C}(r) = r$. The solution is obtained through the minimization of a convex quadratic constrained function. Here, we derive a formulation that, up to a change of variables, can be demonstrated to be equivalent to the one proposed by Crammer and Singer (see Aiolli and Sperduti (2002a)). This will serve to introduce a uniform notation useful for presenting the multi-prototype extension in the following sections.

In multiclass classifiers based on Eq. (2), in order to have a correct classification, the prototype of the correct class is required to have a score greater than the maximum among the scores of the prototypes associated to incorrect classes. The multiclass margin for the example $\mathbf{x}_i$ is then defined by

$$\rho(\mathbf{x}_i, c_i | M) = \langle M_{y_i}, \mathbf{x}_i \rangle - \max_{r \neq y_i} \langle M_r, \mathbf{x}_i \rangle,$$

where y_i such that $\mathcal{C}(y_i) = c_i$, is the index of the prototype associated to the correct label for the example $\mathbf{x}_i$. In the single prototype case, with no loss of generality, we consider a prototype and the associated class indices to be coincident, that is $y_i = c_i$. Thus, a correct classification of the example $\mathbf{x}_i$ with a margin greater or equal to 1 requires the condition

$$\langle M_{y_i}, \mathbf{x}_i \rangle \geq \theta_i + 1 \text{ where } \theta_i = \max_{r \neq y_i} \langle M_r, \mathbf{x}_i \rangle. \tag{3}$$

to be satisfied. Note that, the condition above is implied by the existence of a matrix $\hat{M}$ such that $\forall r \neq y_i,\ \langle \hat{M}_{y_i}, \mathbf{x}_i \rangle > \langle \hat{M}_r, \mathbf{x}_i \rangle$. In fact, the matrix M can always be obtained by an opportune re-scaling of the matrix $\hat{M}$. With these premises, a set of examples is said to be *linearly separable* by a multiclass classifier if there exists a matrix M able to fulfill the above constraints for every pattern in the set.

Unfortunately, the examples in the training set can not always be separated and some examples may violate the margin constraints. We consider these cases by introducing soft margin slack variables $\xi_i \geq 0$, one for each example, such that

$$\xi_i = [\theta_i + 1 - \langle M_{y_i}, \mathbf{x}_i \rangle]_+,$$

where the symbol $[z]_+$ corresponds to the soft-margin loss that is equal to z if $z > 0$ and 0 otherwise. Note that the value ξ_i can also be seen as an upper bound on the binary loss for the example $\mathbf{x}_i$, and consequently its average value over the training set is an upper bound on the empirical error.

Motivated by the *structural risk minimization* (SRM) principle in (Vapnik, 1998; Schölkopf and C. Burges and V. Vapnik, 1995), we search for a matrix M with small norm such to minimize the empirical error over the training set. We use the 2-norm of the matrix M. We thus formulate the problem in a SVM style by requiring a set of small norm prototypes to fulfill the soft constraints given by the classification requirements. Specifically, the single-prototype version of multiclass

SVM (SProtSVM in the following) will result in:

$$\begin{array}{l} \min_{M,\xi,\theta} \frac{1}{2}||M||^2 + C\sum_i \xi_i \\ \text{subject to: } \left\{ \begin{array}{l} \forall i, r \neq y_i,\ \langle M_r, \mathbf{x}_i \rangle \leq \theta_i, \\ \forall i,\ \langle M_{y_i}, \mathbf{x}_i \rangle \geq \theta_i + 1 - \xi_i, \\ \forall i,\ \xi_i \geq 0 \end{array} \right. \end{array} \tag{4}$$

where the parameter C controls the amount of regularization applied to the model.

It can be observed that, at the optimum, θ_i will be set to the maximum value among the negative scores for the instance $\mathbf{x}_i$ (in such a way to minimize the corresponding slack variables) consistently with Eq. (3).

The problem in Eq. (4) is convex and it can be solved in the standard way by resorting to the optimization of the Wolfe dual problem. In this case, the Lagrangian is:

$$\begin{array}{rcl} L(M,\xi,\theta,\alpha,\lambda) & = & \frac{1}{2}||M||^2 + C\sum_i \xi_i + \\ & & \sum_{i,r\neq y_i} \alpha_i^r (\langle M_r, \mathbf{x}_i \rangle - \theta_i) + \\ & & \sum_i \alpha_i^{y_i} (\theta_i + 1 - \xi_i - \langle M_{y_i}, \mathbf{x}_i \rangle) - \\ & & \sum_i \lambda_i \xi_i \\ & = & \frac{1}{2}||M||^2 - \sum_{i,r} y_i^r \alpha_i^r (\langle M_r, \mathbf{x}_i \rangle - \theta_i) + \\ & & \sum_i \alpha_i^{y_i} + \sum_i (C - \alpha_i^{y_i} - \lambda_i)\xi_i, \end{array} \tag{5}$$

subject to the constraints $\alpha_i^r, \lambda_i \geq 0$.

By differentiating the Lagrangian with respect to the primal variables and imposing the optimality conditions we obtain a set of constraints that the variables have to fulfill in order to be an optimal solution:

$$\begin{array}{rcl} \frac{\partial L(M,\xi,\theta,\alpha,\lambda)}{\partial M_r} = 0 & \Leftrightarrow & M_r = \sum_i y_i^r \alpha_i^r \mathbf{x}_i \\ \frac{\partial L(M,\xi,\theta,\alpha,\lambda)}{\partial \xi_i} = 0 & \Leftrightarrow & C - \alpha_i^{y_i} - \lambda_i = 0 \Leftrightarrow \alpha_i^{y_i} \leq C \\ \frac{\partial L(\mathbf{w},\xi,\theta,\alpha,\lambda)}{\partial \theta_i} = 0 & \Leftrightarrow & \alpha_i^{y_i} = \sum_{r\neq y_i} \alpha_i^r \end{array} \tag{6}$$

By using the facts $\alpha_i^{y_i} = \frac{1}{2}\sum_r \alpha_i^r$ and $||M(\alpha)||^2 = \sum_{i,j,r} y_i^r y_j^r \alpha_i^r \alpha_j^r \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, substituting equalities from Eq. (6) into Eq. (5) and omitting constants that do not change the solution, the problem can be restated as:

$$\begin{array}{l} \max_\alpha \sum_{i,r} \alpha_i^r - ||M(\alpha)||^2 \\ \text{subject to: } \left\{ \begin{array}{l} \forall i, r,\ \alpha_i^r \geq 0 \\ \forall i, \alpha_i^{y_i} = \sum_{r\neq y_i} \alpha_i^r \leq C \end{array} \right. \end{array}$$

Notice that, when kernels are used, by the linearity of dot-products, the scoring function for the r-th prototype and a pattern $\mathbf{x}$ can be conveniently reformulated as

$$f_r(\mathbf{x}) = \langle M_r, \phi(\mathbf{x}) \rangle = \sum_{i=1}^{n} y_i^r \alpha_i^r k(\mathbf{x}, \mathbf{x}_i).$$

The next section includes an efficient optimization procedure for the more general multi-prototype setting that includes the single-prototype case as an instance.

4. Multi-Prototype Multi-Class SVM

The SProtSVM model presented in the previous section is here extended to learn more than one prototypes per class. This is done by generalizing Eq. (3) to multiple prototypes. In this setting, one instance is correctly classified if and only if *at least* one of the prototypes associated to the correct class has a score greater than the maximum of the scores of the prototypes associated to incorrect classes.

A natural extension of the definition for the margin in the multi-prototype case is then

$$\rho(\mathbf{x}_i, c_i|M) = \max_{r\in\mathcal{P}_i}\langle M_r, \mathbf{x}_i\rangle - \max_{r\in\mathcal{N}_i}\langle M_r, \mathbf{x}_i\rangle.$$

and its value will result greater than zero if and only if the example $\mathbf{x}_i$ is correctly classified.

We can now give conditions for a correct classification of an example $\mathbf{x}_i$ with a margin greater or equal to 1 by requiring that:

$$\exists r \in \mathcal{P}_i : \ \langle M_r, \mathbf{x}_i\rangle \geq \theta_i + 1 \text{ and } \theta_i = \max_{r\in\mathcal{N}_i} \langle M_r, \mathbf{x}_i\rangle. \tag{7}$$

To allow for margin violations, for each example $\mathbf{x}_i$, we introduce soft margin slack variables $\xi_i^r \geq 0$, one for each positive prototype, such that

$$\forall r \in \mathcal{P}_i, \ \xi_i^r = [\theta_i + 1 - \langle M_r, \mathbf{x}_i\rangle]_+.$$

Given a pattern $\mathbf{x}_i$, we arrange the soft margin slack variables ξ_i^r in a vector $\xi_i \in \mathbb{R}^{|\mathcal{P}_i|}$. Let us now introduce, for each example $\mathbf{x}_i$, a new vector having a number of components equal to the number of positive prototypes for $\mathbf{x}_i$, $\pi_i \in \{0,1\}^{|\mathcal{P}_i|}$, whose components are all zero except one component that is 1. In the following, we refer to π_i as the *assignment* of the pattern $\mathbf{x}_i$ to the (positive) prototypes. Notice that the dot product $\langle \pi_i, \xi_i\rangle$ is always an upper bound on the binary loss for the example $\mathbf{x}_i$ independently from its assignment and, similarly to the single-prototype case, the average value over the training set represents an upper bound on the empirical error.

Now, we are ready to formulate the general multi-prototype problem by requiring a set of prototypes of small norm and the best assignment for the examples able to fulfill the soft constraints given by the classification requirements. Thus, the MProtSVM formulation can be given as:

$$\begin{array}{l} \min_{M,\xi,\theta,\pi} \frac{1}{2}||M||^2 + C\sum_i \langle \pi_i, \xi_i\rangle \\ \text{subject to:} \left\{ \begin{array}{l} \forall i, r \in \mathcal{N}_i, \ \langle M_r, \mathbf{x}_i\rangle \leq \theta_i, \\ \forall i, r \in \mathcal{P}_i, \ \langle M_r, \mathbf{x}_i\rangle \geq \theta_i + 1 - \xi_i^r, \\ \forall i, r \in \mathcal{P}_i, \xi_i^r \geq 0 \\ \forall i, \pi_i \in \{0,1\}^{|\mathcal{P}_i|}. \end{array} \right. \end{array} \tag{8}$$

Unfortunately, this is a mixed integer problem that is not convex and it is a difficult problem to solve in general. However, as we will see in the following, it is prone to an efficient optimization procedure that approximates a global optimum. At this point, it is worth noticing that, since this formulation is itself an (heuristic) approximation to the structural risk minimization principle where the parameter C rules the trade-off between keeping the VC-dimension low and minimizing the training error, a good solution of the problem in Eq. (8), even if not optimal, can nevertheless give good results in practice. As we will see, this claim seems confirmed by the results obtained in the experimental work.

In the following section we demonstrate that when the assignment is fixed for each pattern, the problem results tractable and we are able to give an efficient procedure to solve the associated problem.

5. Optimization with Static Assignments

Let suppose that the assignments are kept fixed. In this case, the reduced problem becomes convex and it can be solved as described above by resorting to the optimization of the Wolfe dual problem. In this case, the Lagrangian is:

$$\begin{array}{rcl} L_\pi(M,\xi,\theta,\alpha,\lambda) & = & \frac{1}{2}||M||^2 + C\sum_i \langle \pi_i, \xi_i \rangle + \\ & & \sum_{i,r\in\mathcal{P}_i} \alpha_i^r (\theta_i + 1 - \xi_i^r - \langle M_r, \mathbf{x}_i \rangle) - \\ & & \sum_{i,r\in\mathcal{P}_i} \lambda_i^r \xi_i^r + \\ & & \sum_{i,r\in\mathcal{N}_i} \alpha_i^r (\langle M_r, \mathbf{x}_i \rangle - \theta_i), \end{array} \tag{9}$$

subject to the constraints $\alpha_i^r, \lambda_i^r \geq 0$.

As above, by differentiating the Lagrangian of the reduced problem and imposing the optimality conditions, we obtain:

$$\begin{array}{rcl} \frac{\partial L_\pi(M,\xi,\theta,\alpha,\lambda)}{\partial M_r} = 0 & \Leftrightarrow & M_r = \sum_i y_i^r \alpha_i^r \mathbf{x}_i \\ \frac{\partial L_\pi(M,\xi,\theta,\alpha,\lambda)}{\partial \xi_i^r} = 0 & \Leftrightarrow & C\pi_i^r - \alpha_i^r - \lambda_i^r = 0 \Leftrightarrow \alpha_i^r \leq C\pi_i^r \\ \frac{\partial L_\pi(M,\xi,\theta,\alpha,\lambda)}{\partial \theta_i} = 0 & \Leftrightarrow & \sum_{r\in\mathcal{P}_i} \alpha_i^r = \sum_{r\in\mathcal{N}_i} \alpha_i^r \end{array} \tag{10}$$

Notice that the second condition requires the dual variables associated to (positive) prototypes not assigned to a pattern to be 0. By denoting now as y_i the unique index $r \in \mathcal{P}_i$ such that $\pi_i^r = 1$, once using the conditions of Eq. (10) in Eq. (9) and omitting constants that do not change the obtained solution, the reduced problem can be restated as:

$$\begin{array}{l} \max_\alpha \sum_{i,r} \alpha_i^r - ||M(\alpha)||^2 \\ \text{subject to: } \left\{ \begin{array}{l} \forall i, r,\ \alpha_i^r \geq 0 \\ \forall i,\ \alpha_i^{y_i} = \sum_{r\in\mathcal{N}_i} \alpha_i^r \leq C \\ \forall i, r \in \mathcal{P}_i \setminus \{y_i\},\ \alpha_i^r = 0. \end{array} \right. \end{array} \tag{11}$$

It can be trivially shown that this formulation is consistent with the formulation of the SProtSVM dual given above. Moreover, when kernels are used, the score function for the r-th prototype and a pattern $\mathbf{x}$ can be formulated as in the single-prototype case as

$$f_r(\mathbf{x}) = \langle M_r, \phi(\mathbf{x}) \rangle = \sum_{i=1}^{n} y_i^r \alpha_i^r k(\mathbf{x}, \mathbf{x}_i).$$

Thus, when patterns are statically assigned to the prototypes via constant vectors π_i, the convexity of the associated MProtSVM problem implies that the optimal solution for the primal problem in Eq. (8) can be found through the maximization of the Lagrangian as in problem in Eq. (11). Assuming an equal number q of prototypes per class, the dual involves $n \times m \times q$ variables which leads to a very large scale problem. Anyway, the independence of constraints among the different patterns allows for the separation of the variables in n disjoint sets of $m \times q$ variables.

The algorithms we propose for the optimization of the problem in Eq. (11) are inspired by the ones already presented in (Crammer and Singer, 2000, 2001) consisting in iteratively selecting patterns from the training set and greedily optimizing with respect to the variables associated to that pattern. In particular, the authors propose a fixed-point procedure for the optimization of the reduced problem.

In the following, we first show that the pattern related problem can be further decomposed until the solution for a minimal subset of two variables is required. This is quite similar to the SMO procedure for binary SVM. Then, a training algorithm for this problem can be defined by iterating this basic step.

5.1 The Basic Optimization Step

In this section the basic step corresponding to the simultaneous optimization of a subset of variables associated to the same pattern is presented. Let pattern $\mathbf{x}_p$ be fixed. Since we want to enforce the linear constraint $\sum_{r\in\mathcal{N}_p}\alpha_p^r+\lambda_p=C$, $\lambda_p\geq 0$, from the second condition in Eq. (10), two elements from the set of variables $\{\alpha_p^r, r\in\mathcal{N}_p\}\cup\{\lambda_p\}$ will be optimized in pair while keeping the solution inside the feasible region. In particular, let ζ_1 and ζ_2 be the two selected variables, we restrict the updates to the form $\zeta_1\leftarrow\zeta_1+\nu$ and $\zeta_2\leftarrow\zeta_2-\nu$ with optimal choices for ν.

In order to compute the optimal value for ν we first observe that an additive update ΔM_r to the prototype r will affect the squared norm of the prototype vector M_r of an amount

$$\Delta||M_r||^2=||\Delta M_r||^2+2\langle M_r,\Delta M_r\rangle.$$

Then, we examine separately the two ways a pair of variables can be selected for optimization.

(Case 1) We first show how to analytically solve the problem associated to an update involving a single variable α_p^r, $r\in\mathcal{N}_p$ and the variable $\alpha_p^{y_p}$. Note that, since λ_p does not influence the value of the objective function, it is possible to solve the associated problem with respect to the variable α_p^r and $\alpha_p^{y_p}$ in such a way to keep the constraint $\alpha_p^{y_p}=\sum_{r\in\mathcal{N}_p}\alpha_p^r$ satisfied and afterwards to enforce the constraints $\lambda_p=C-\sum_{s\in\mathcal{N}_p}\alpha_p^s\geq 0$. Thus, in this case we have:

$$\alpha_p^r\leftarrow\alpha_p^r+\nu \text{ and } \alpha_p^{y_p}\leftarrow\alpha_p^{y_p}+\nu.$$

Since $\Delta M_r=-\nu\mathbf{x}_p$, $\Delta M_{y_p}=\nu\mathbf{x}_p$ and $\Delta M_s=0$ for $s\notin\{r,y_p\}$, we obtain

$$\Delta||M||^2=\Delta||M_r||^2+\Delta||M_{y_p}||^2=2\nu^2||\mathbf{x}_p||^2+2\nu(f_{y_p}(\mathbf{x}_p)-f_r(\mathbf{x}_p))$$

and the difference obtained in the Lagrangian value will be

$$\Delta L(\nu)=2\nu(1-f_{y_p}(\mathbf{x}_p)+f_r(\mathbf{x}_p)-\nu||\mathbf{x}_p||^2).$$

Since this last formula is concave in ν, it is possible to find the optimal value when the first derivative is null, i.e.

$$\hat{\nu}=\arg\max_{\nu}\Delta L(\nu)=\frac{1-f_{y_p}(\mathbf{x}_p)+f_r(\mathbf{x}_p)}{2||\mathbf{x}_p||^2} \tag{12}$$

If the values of α_p^r and $\alpha_p^{y_p}$, after being updated, turn out to be not feasible for the constraints $\alpha_p^r\geq 0$ and $\alpha_p^{y_p}\leq C$, we select the unique value for ν such to fulfill the violated constraint bounds at the limit ($\alpha_p^r+\nu=0$ or $\alpha_p^{y_p}+\nu=C$ respectively).

(Case 2) Now, we show the analytic solution of the associated problem with respect to an update involving a pair of variables $\alpha_p^{r_1},\alpha_p^{r_2}$ such that $r_1,r_2\in\mathcal{N}_p$ and $r_1\neq r_2$. Since, in this case, the update must have zero sum, we have:

$$\alpha_p^{r_1}\leftarrow\alpha_p^{r_1}+\nu \text{ and } \alpha_p^{r_2}\leftarrow\alpha_p^{r_2}-\nu$$

In this case, $\Delta M_{r_1} = -\nu \mathbf{x}_p$, $\Delta M_{r_2} = \nu \mathbf{x}_p$ and $\Delta M_s = 0$ for $s \notin \{r_1, r_2\}$, thus

$$\Delta||M||^2 = \Delta||M_{r_1}||^2 + \Delta||M_{r_2}||^2 = 2\nu^2||\mathbf{x}_p||^2 + 2\nu(f_{r_2}(\mathbf{x}_p) - f_{r_1}(\mathbf{x}_p))$$

leading to an Lagrangian improvement equals to

$$\Delta L(\nu) = 2\nu(f_{r_1}(\mathbf{x}_p) - f_{r_2}(\mathbf{x}_p) - \nu||\mathbf{x}_p||^2).$$

Since also this last formula is concave in ν, it is possible to find the optimal value

$$\hat{\nu} = \arg\max_{\nu} \Delta L(\nu) = \frac{f_{r_1}(\mathbf{x}_p) - f_{r_2}(\mathbf{x}_p)}{2||\mathbf{x}_p||^2} \tag{13}$$

Similarly to the previous case, if the values of the $\alpha_p^{r_1}$ and $\alpha_p^{r_2}$, after being updated, turn out to be not feasible for the constraints $\alpha_p^{r_1} \geq 0$ and $\alpha_p^{r_2} \geq 0$, we select the unique value for ν such to fulfill the violated constraint bounds at the limit (in this case, considering $f_{r_1}(\mathbf{x}_p) \leq f_{r_2}(\mathbf{x}_p)$ and thus $\hat{\nu} \leq 0$ with no loss in generality, we obtain $\alpha_p^{r_1} + \nu = 0$ or $\alpha_p^{r_2} - \nu = C$ respectively).

Note that, when a kernel is used, the norm in the feature space can be substituted with the diagonal component of the kernel matrix, i.e. $||\mathbf{x}_p||^2 = k(\mathbf{x}_p, \mathbf{x}_p) = K_{pp}$ while the scores can be maintained in implicit form and computed explicitly when necessary.

To render the following exposition clearer, we try to compact the two cases in one. This can be done by defining the update in a slightly different way, that is, for each pair $(r_a, r_b) \in (\mathcal{P}_p \cup \mathcal{N}_p) \times \mathcal{N}_p$ we define:

$$\alpha_p^{r_a} \leftarrow \alpha_p^{r_a} + y_p^{r_a}\nu \text{ and } \alpha_p^{r_b} \leftarrow \alpha_p^{r_b} - y_p^{r_b}\nu$$

and hence the improvement obtained for the value of the Lagrangian is

$$V_{r_a,r_b}^p(\nu) = 2\nu\left(\frac{1}{2}(y_p^{r_a} - y_p^{r_b}) - f_{r_a}(\mathbf{x}_p) + f_{r_b}(\mathbf{x}_p) - \nu k(\mathbf{x}_p, \mathbf{x}_p)\right) \tag{14}$$

where the optimal value for the ν is

$$\hat{\nu} = \frac{\frac{1}{2}(y_p^{r_a} - y_p^{r_b}) - f_{r_a}(\mathbf{x}_p) + f_{r_b}(\mathbf{x}_p)}{2k(\mathbf{x}_p, \mathbf{x}_p)}$$

subject to the constraints

$$\alpha_p^{r_a} + y_p^{r_a}\nu > 0,\ \alpha_p^{r_b} - y_p^{r_b}\nu > 0,\ \alpha_{y_p} + \frac{1}{2}(y_p^{r_a} - y_p^{r_b})\nu \leq C.$$

The basic step algorithm and the updates induced in the scoring functions are described in Figure 1 and Figure 2, respectively.

5.2 New Algorithms for the Optimization of the Dual

In the previous section we have shown how it is possible to give an explicit optimal solution of the reduced problem obtained by fixing all the variables apart for the two variables under consideration.

In this section, we analyze different algorithms that are based on the step given above. The basic idea is the same as SMO for SVM (Platt, 1998), that is to repeat a process in which

`BasicStep`(p, r_a, r_b)

$\nu = \frac{\frac{1}{2}(y_p^{r_a} - y_p^{r_b}) - f_{r_a}(\mathbf{x}_p) + f_{r_b}(\mathbf{x}_p)}{2K_{pp}}$

if $(\alpha_p^{r_a} + y_p^{r_a}\nu < 0)$ **then** $\nu = -y_p^{r_a}\alpha_p^{r_a}$

if $(\alpha_p^{r_b} - y_p^{r_b}\nu < 0)$ **then** $\nu = y_p^{r_b}\alpha_p^{r_b}$

if $(\alpha_p^{y_p} + \frac{1}{2}(y_p^{r_a} - y_p^{r_b})\nu > C)$ **then** $\nu = 2\frac{C - \alpha_p^{y_p}}{y_p^{r_a} - y_p^{r_b}}$

return ν

Figure 1: The basic optimization step: explicit optimization of the reduced problem with two variables, namely $\alpha_p^{r_a}$ and $\alpha_p^{r_b}$.

`BasicUpdate`(p, r_a, r_b, ν)

$\alpha_{r_a} = \alpha_{r_a} + y_p^{r_a}\nu; \quad \alpha_{r_b} = \alpha_{r_b} + y_p^{r_b}\nu;$

$f_{r_a}(\mathbf{x}_p) = f_{r_a}(\mathbf{x}_p) + y_p^{r_a}\nu K_{pp}; \quad f_{r_b}(\mathbf{x}_p) = f_{r_b}(\mathbf{x}_p) - y_p^{r_b}\nu K_{pp};$

Figure 2: Updates done after the basic optimization step has been performed and the optimal solution found. K_{pp} denotes the p-th element of the kernel matrix diagonal.

- a minimal subset of independent multipliers are selected
- the analytic solution of the reduced problem obtained by fixing all the variables but the ones we selected in the previous step is found.

In our case, a minimal set of two variables associated to the same example are selected at each iteration. As we showed in the last section, each iteration leads to an increase of the Lagrangian. This, together with the compactness of the feasible set guarantees the convergence of the procedure. Moreover, this optimization procedure can be considered incremental in the sense that the solution we have found at one step forms the initial condition when a new subset of variables are selected for optimization. Finally, it should be noted that for each iteration the scores of the patterns in the training set must be updated before to be used in the selection phase. The general optimization algorithm just described is depicted in Figure 3.

In the following, we present three alternative algorithms for the optimization of the problem in Eq. (11) which differ in the way they choose the pairs to optimize through the iterations, i.e. the `OptimizeOnPattern` procedure.

The first practical and very simple algorithm for solving the problem in Eq. (11) can be derived from the steps given above where at each iteration a pair of multipliers is selected and then optimized according to the analytic solution given in the previous section until some convergence criterion

```
OptimizeStaticProblem(φ_V)
    repeat
      PatternSelection(p) // Heuristically choose an example p based on Eq. (14)
      OptimizeOnPattern(p, φ_V)
    until converge.
```

Figure 3: High-level procedure for the optimization of a statically assigned multi-prototype SVM. The parameter φ_V is the tolerance when checking optimality in the `OptimizeOnPattern` procedure.

```
BasicOptimizeOnPattern(p, φ_V)
    Heuristically choose two indexes r_a ≠ r_b based on Eq. (14)
    ν = BasicStep(p, r_a, r_b)
    BasicUpdate(p, r_a, r_b, ν)
```

Figure 4: SMO-like algorithm for the optimization of statically assigned multi-prototype SVM.

is fulfilled. Eq. (14) gives a natural method for the selection of the two variables involved, i.e. take the two indexes that maximize the value of that formula. Finally, once chosen two variables to optimize, the basic step in the algorithm in Figure 1 provides the optimal solution. This very general optimization algorithm will be referred to as `BasicOptimizeOnPattern` and it is illustrated in Figure 4.

A second method to solve the optimization problem in Eq. (11) is given in the following and can be also considered as an alternative method to the Crammer and Singer fixed-point algorithm for the optimization over a single example (Crammer and Singer, 2001). This method consists in fixing an example and iterating multiple times the basic step described above on pairs of variables chosen among that associated to the pattern into consideration until some convergence conditions local to the pattern under consideration are matched. Notice that this algorithm requires just a single step in the binary single-prototype case. In Figure 5 the pseudo-code of the proposed pattern optimization algorithm referred to as `AllPairsOptimizeOnPattern` is presented. At each step, the algorithm applies the basic step to the $m(m-1)/2$ pairs of variables associated with the pattern chosen for optimization until a certain condition on the value of the increment of the Lagrangian is verified. Iterating multiple times the basic step described above on pairs of variables chosen among that associated to a given pattern it is guaranteed to find the optimality condition for the pattern. The optimization step of this reduced problem can require the optimization over all the $q^2m(m-1)/2$ pairs of variables not constrained to 0 associated with the selected pattern. Thus the complexity of

`AllPairsOptimizeOnPattern`(p, φ_V)

$t = 0, V(0) = 0.$

do

$t \leftarrow t+1, V(t) = 0$

For each $r_1 \neq r_2$

$\nu = \texttt{BasicStep}(p, r_1, r_2)$

$V(t) = V(t) + 2\nu\left(\frac{1}{2}(y_p^{r_1} - y_p^{r_2}) - f_{r_1}(\mathbf{x}_p) + f_{r_2}(\mathbf{x}_p) - \nu K_{pp}\right)$

`BasicUpdate`(p, r_1, r_2, ν)

until $(V(t) \leq \varphi_V)$

Figure 5: Algorithm for the incremental optimization of the variables associated with a given pattern of a staticallly assigned multi-prototype SVM

the optimization of the reduced problem is $O((mq)^2 I)$ where I is the number of iterations.

Now, we perform a further step by giving a third algorithm that is clearly faster than the previous versions having at each iteration a complexity $O(mq)$. For this we give an intuitive derivation of three optimality conditions. Thus, we will show that if a solution is such that all these conditions are not fulfilled, then this solution is just the optimal one since it verifies the KKT conditions.

First of all, we observe that for the variables $\{\alpha_p^r, \lambda_p\}$ associated to the pattern $\mathbf{x}_p$ to be optimal, the value ν returned by the basic step must be 0 for each pair. Thus, we can consider the two cases above separately. For the first case, in order to be able to apply the step, it is necessary for one of the following two conditions to be verified:

$$(\Psi_1) \qquad (\alpha_p^{y_p} < C) \wedge (f_{y_p}(\mathbf{x}_p) < \max_{r \in \mathcal{N}_p} f_r(\mathbf{x}_p) + 1)$$
$$(\Psi_2) \quad (\alpha_p^{y_p} > 0) \wedge (f_{y_p}(\mathbf{x}_p) > \max_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r(\mathbf{x}_p) + 1)$$

In fact, in Eq. (12), when there exists $r \in \mathcal{N}_p$ such that $f_{y_p}(\mathbf{x}_p) < f_r(\mathbf{x}_p) + 1$, the condition $\hat{\nu} > 0$ holds. In this case the pair $(\alpha_p^{y_p}, \alpha_p^r)$ can be chosen for optimization. Thus, it must be $\alpha_p^{y_p} < C$ in order to be possible to increase the values of the pair of multipliers. Alternatively, if $\alpha_p^{y_p} > 0$ and there exists an index r such that $\alpha_p^r > 0$ and $f_{y_p}(\mathbf{x}_p) > f_r(\mathbf{x}_p) + 1$ then $\hat{\nu} < 0$ and (at least) the pair $(\alpha_p^{y_p}, \alpha_p^k)$ where $k = \arg\max_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r(\mathbf{x}_p)$ can be chosen for optimization. Finally, from Eq. (13), we can observe that in order to have $\hat{\nu} \neq 0$, we need the last condition to be verified:

$$(\Psi_3) \quad (\alpha_p^{y_p} > 0) \wedge (\max_{r \in \mathcal{N}_p} f_r(\mathbf{x}_p) > \min_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r(\mathbf{x}_p))$$

In fact, in Eq. (13), if there exists a pair $(\alpha_p^{r_a}, \alpha_p^{r_b})$ such that $f_{r_a}(\mathbf{x}_p) > f_{r_b}(\mathbf{x}_p)$ and $\alpha_p^{r_b} > 0$, the condition $\hat{\nu} > 0$ holds for this pair and it can be chosen for optimization.

Note that in Ψ_2 and Ψ_3 the condition $\alpha_p^{y_p} > 0$ is redundant and serves to assure that the second condition makes sense. In fact, when the first condition is not verified, we would have $\alpha = \mathbf{0}$ and the second condition is undetermined.

Summarizing, we can give three conditions of non-optimality. This means that whenever at least one among these conditions is verified the solution is not optimal. They are

$$\begin{array}{ll} (a) & (\alpha_p^{y_p} < C) \wedge (f_{y_p}(\mathbf{x}_p) < \max_{r\in\mathcal{N}_p} f_r(\mathbf{x}_p) + 1) \\ (b) & (\alpha_p^{y_p} > 0) \wedge (f_{y_p}(\mathbf{x}_p) > \max_{r\in\mathcal{N}_p, \alpha_p^r>0} f_r(\mathbf{x}_p) + 1) \\ (c) & (\alpha_p^{y_p} > 0) \wedge (\max_{r\in\mathcal{N}_p} f_r(\mathbf{x}_p) > \min_{r\in\mathcal{N}_p, \alpha_p^r>0} f_r(\mathbf{x}_p)) \end{array} \tag{15}$$

Now we are able to demonstrate the following theorem showing that when no one of these conditions are satisfied the conditions of optimality (KKT conditions) are verified:

Theorem 1 *Let α be an admissible solution for the dual problem in Eq. (11) not satisfying any of the conditions in Eq. (15), then α is an optimal solution.*

Proof. We consider the *Kuhn-Tucker* theorem characterizing the optimal solutions of convex problems. We know from theoretical results about convex optimization that for a solution α to be optimal a set of conditions are both necessary and sufficient. These conditions are the one reported in Eq. (10) plus the so-called *Karush-Kuhn-Tucker* (KKT) complementarity conditions that in our case correspond to:

$$\begin{array}{lll} (a) & \forall p, r \in \mathcal{P}_p, & \alpha_p^r(\theta_p + 1 - \xi_p^r - f_r(\mathbf{x}_p)) = 0 \\ (b) & \forall p, r \in \mathcal{P}_p, & \lambda_p^r \xi_p^r = 0 \\ (c) & \forall p, v \in \mathcal{N}_p, & \alpha_p^v (f_v(\mathbf{x}_p) - \theta_p) = 0. \end{array} \tag{16}$$

Then, we want to show that these KKT complementary conditions are satisfied by the solution α_p for every $p \in \{1, \ldots, n\}$. To this end let us fix an index p and consider a solution where all the conditions in Eq. (15) are not satisfied. We want to show that the KKT conditions in Eq. (16) are verified in this case.

First of all, we observe that for all the variables associated to a positive prototype $r \in \mathcal{P}_p$ not assigned to the pattern $\mathbf{x}_p$, that is such that $\pi_p^r = 0$, from Eq. (10) we trivially have $\alpha_p^r = 0$ and $\lambda_p^r = 0$ thus verifying all conditions in Eq. (16).

Let now consider the case $0 < \alpha_p^{y_p} < C$. In this case the non applicability of condition in Eq. (15)c says that $\theta_p = \max_{v\in\mathcal{N}_p} f_v(\mathbf{x}_p)$ and $\forall v \in \mathcal{N}_p,\ \alpha_p^v > 0 \Rightarrow f_v(\mathbf{x}_p) = \theta_p$ that is the condition in Eq. (16)c holds. Moreover, the condition in Eq. (15)a, if not satisfied, implies $f_{y_p}(\mathbf{x}_p) \geq \theta_p + 1$, thus $\alpha_p^{y_p} = 0$ and $\xi_p^{y_p} = 0$ thus satisfying the conditions in Eq. (16)a and Eq. (16)b.

Let now consider the case $\alpha_p^{y_p} = 0$. The conditions in Eq. (16)a and Eq. (16)c follow immediately. In this case, Eq. (15)b and Eq. (15)c are not satisfied. For what concerns the Eq. (15)a it must be the case $f_{y_p}(\mathbf{x}_p) \geq \max_{v\in\mathcal{N}_p} f_v(\mathbf{x}_p) + 1$ and so $\xi_p^{y_p} = 0$ thus verifying the condition in Eq. (16)b.

Finally, in the case $\alpha_p^{y_p} = C$, from Eq. (10) we have $\lambda_p = 0$ and hence the condition in Eq. (16)b is verified. Moreover, from the fact that Eq. (15)c is not satisfied $\forall v \in \mathcal{N}_p : \alpha_p^v > 0 \Rightarrow \theta_p = \max_{r\in\mathcal{N}_p} f_r(\mathbf{x}_p) \leq f_v(\mathbf{x}_p) \Rightarrow \theta_p = f_v(\mathbf{x}_p)$ and the condition in Eq. (16)c holds. Moreover, from condition in Eq. (15)b we obtain $f_{y_p}(\mathbf{x}_p) \leq \theta_p + 1$ and $\xi_p^{y_p} = \theta_p + 1 - f_{y_p}(\mathbf{x}_p)$ thus implying the truth of the condition in Eq. (16)a. □

`OptKKTOptimizeOnPattern`($\mathbf{x}_p$,φ_V)

$\forall r,\ f_r := f_r(\mathbf{x}_p) = \sum_{i=1}^{n} y_i^r \alpha_i^r k(\mathbf{x}_i, \mathbf{x}_p)$, $K_{pp} = k(\mathbf{x}_p, \mathbf{x}_p)$;

do

if $(\alpha_p^{y_p} = 0)$ **then** $\{$

$r_1 := \arg\max_{r \in \mathcal{N}_p} f_r$;

$\nu_1 :=$ `BasicStep`(p, y_p, r_1); $V_1 := 2\nu_1(1 - f_{y_p} + f_{r_1} - \nu_1 K_{pp})$;

$k := 1\}$

else $\{$

$r_1 := \arg\max_{r \in \mathcal{N}_p} f_r$; $r_2 := \arg\max_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r$; $r_3 := \arg\min_{r \in \mathcal{N}_p, \alpha_p^r > 0} f_r$;

$\nu_1 :=$ `BasicStep`(p, y_p, r_1); $V_1 := 2\nu_1(1 - f_{y_p} + f_{r_1} - \nu_1 K_{pp})$;

$\nu_2 :=$ `BasicStep`(p, y_p, r_2); $V_2 := 2\nu_2(1 - f_{y_p} + f_{r_2} - \nu_2 K_{pp})$;

$\nu_3 :=$ `BasicStep`(p, r_1, r_3); $V_3 := 2\nu_3(f_{r_1} - f_{r_3} - \nu_3 K_{pp})$;

$k := \arg\max_j V_j$; $\}$

case k **of** $\{$

1: `BasicUpdate`(p,y_p,r_1,ν_1);

2: `BasicUpdate`(p,y_p,r_2,ν_2);

3: `BasicUpdate`(p,r_1,r_3,ν_3); $\}$

until $(V_k \leq \varphi_V)$;

Figure 6: Algorithm for the optimization of the variables associated with a given pattern $\mathbf{x}_p$ and a tolerance φ_V.

All the conditions in Eq. (15) can be checked in time linear with the number of classes. If none of these conditions are satisfied, this means that the condition of optimality has been found. This consideration suggests an efficient procedure that is presented in Figure 6 that searches to greedily fulfill these conditions of optimality and it is referred to as `OptKKTOptimizeOnPattern`. Briefly, the procedure first checks if the condition $\alpha_p = \mathbf{0}$ holds. In this case, two out of the three conditions do not make any sense and the choice of the pair to optimize is mandatory. Otherwise, three indexes (r_1, r_2, r_3) are found defining the conditions in Eq. (15). Then for every pair associated to the condition a basic step is performed and the pair obtaining the larger improvement in the Lagrangian is chosen for the effective update.

5.3 Selection Criteria and Cooling Schemes

The efficiency of the general scheme in Figure 3 is tightly linked to the strategy based on which the examples are selected for optimization.

The algorithm proposed in (Crammer and Singer, 2001) is just an instance of the same scheme. In that work, by using the KKT conditions of the optimization problem, the authors derive a quantity $\psi_i \geq 0$ for each example and show that this value needs to be equal to zero at the optimum. Thus, they use this value to drive the optimization process. In the baseline implementation, the example that maximizes ψ_i is selected. Summarizing, their algorithm consists of a main loop which is composed of: (*i*) an example selection, via the ψ_i quantity, (*ii*) an invocation of a fixed-point algorithm that is able to approximate the solution of the reduced pattern-related problem and (*iii*) the computation of the new value of ψ_i for each example. At each iteration, most of the computation time is spent on the last step since it requires the computation of one row of the kernel matrix, that one relative to the pattern with respect to which they have just optimized. This is why it is so important a strategy that tries to minimize the total number of patterns selected for optimization. Their approach is to maintain an active set containing the subset of patterns having $\psi_i \geq \varepsilon$ where ε is a suitable accuracy threshold. Cooling schemes, i.e. heuristics based on the gradual decrement of this accuracy parameter, are used for improving the efficiency with large datasets.

In our opinion, this approach has however some drawbacks:

i) while $\psi_i \approx 0$ gives us the indication that the variables associated to the pattern $\mathbf{x}_i$ are almost optimal and it would be better not to change them, the actual value ψ_i does not give us information about the improvement we can obtain choosing those variables in the optimization;

ii) cooling schemes reduce the incidence of the above problem but, as we will see, they do not always perform well;

iii) at each iteration, the fixed point optimization algorithm is executed from scratch, and previously computed solutions obtained for an example can't help when the same example is chosen again in future iterations; in addition, it is able to find just an *approximated* solution for the associated pattern-related problem.

According to the above-mentioned considerations, it is not difficult to define a number of criteria to drive a 'good' pattern selection strategy, which seem to be promising. We consider the following three procedures which return a value V_p that we use for deciding if a pattern has to be selected for optimization. Namely:

i) Original KKT as defined in Crammer and Singer's work (here denoted KKT): in this case, the value of V_p corresponds to the ψ_p;

ii) Approximate Maximum Gain (here denoted AMG): in this case the value of V_p is computed as: $\max_{r_1 \neq r_2} V^p_{r_1,r_2}(\hat{\mathbf{v}})$ as defined in Eq. (14). Notice that this is a lower bound of the total increment in the Lagrangian obtained when the pattern p is selected for optimization and the optimization on the variables associated to it is completed;

iii) True Maximum Gain (here denoted BMG): in this case the value is computed using iteratively Eq. (14) and it represents the actual increment in the Lagrangian obtained when the pattern p is selected for optimization.

At the begin of each iteration, a threshold for pattern selection θ_V is computed. For each example of the training set one of the above strategies is applied to it and the example is selected for optimization if the value returned is greater than the threshold. The definition of the threshold θ_V can be performed either by a cooling scheme that decreases its value as the iteration proceeds or in a data dependent way. In our case, we have used a logarithmic cooling scheme since this is the one that has shown the best results for the original Crammer and Singer approach. In addition, we propose two new schemes for the computation of the value θ_V: MAX where the threshold is computed as $\theta_V = \mu \cdot \max_p V_p$, $0 \leq \mu \leq 1$, and MEAN where the threshold is computed as $\theta_V = \frac{1}{n}\sum_{p=1}^{n} V_p$.

5.4 Experiments with Pattern Selection

Experiments comparing the proposed pattern selection approaches versus the Crammer and Singer one has been conducted using a dataset consisting of 10705 digits randomly taken from the NIST-3 dataset. The training set consisted of 5000 randomly chosen digits.

The optimization algorithm has been chosen among: *i)* The base-line Crammer and Singer original fixed-point procedure (here denoted CS); *ii)* `AllPairsOptimizeOnPatterns` (here denoted ALL); *iii)* `BasicOptimizeOnPatterns` (here denoted BAS). In the first experiments we used a cache for the kernel matrix of size 3000 that was able to contain all the matrix rows associated to the support vectors. For all the following experiments a AMD K6-II, 300MHz, with 64MB of memory has been used.

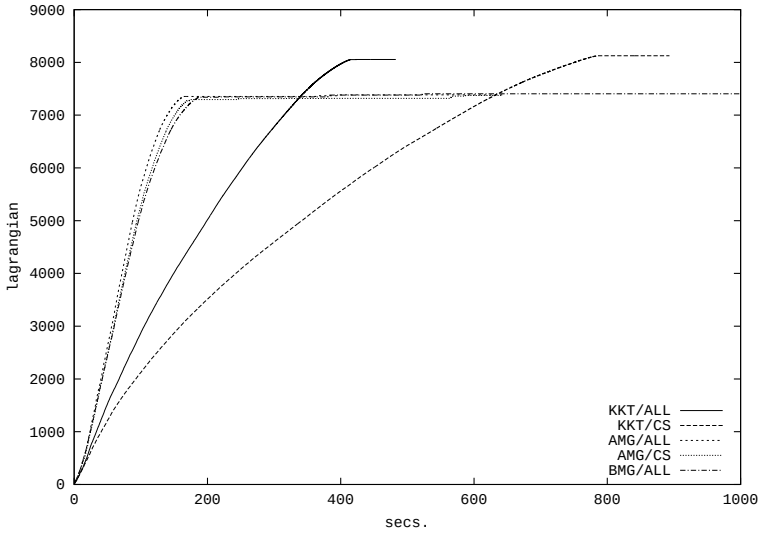

Figure 7: The effect of the logarithmic cooling scheme on different selection/optimization strategies.

In Figure 7 the effect of the application of the logarithmic scheme of cooling to the different selection/optimization strategies is shown. It is possible to note that even if the proposed selection strategies largely improve the convergence rate, the optimal solution can not be reached. This clearly shows how cooling schemes of the same family of that proposed in (Crammer and Singer, 2001) are not suitable for these new proposed selection strategies. This is mostly due to the fact that the

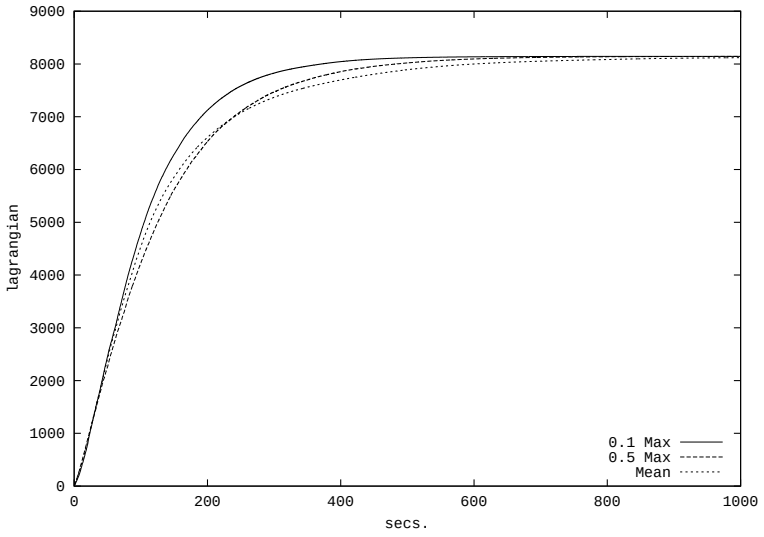

Figure 8: Comparison of different heuristics for the computation of the value θ_V for the SMO-like algorithm.

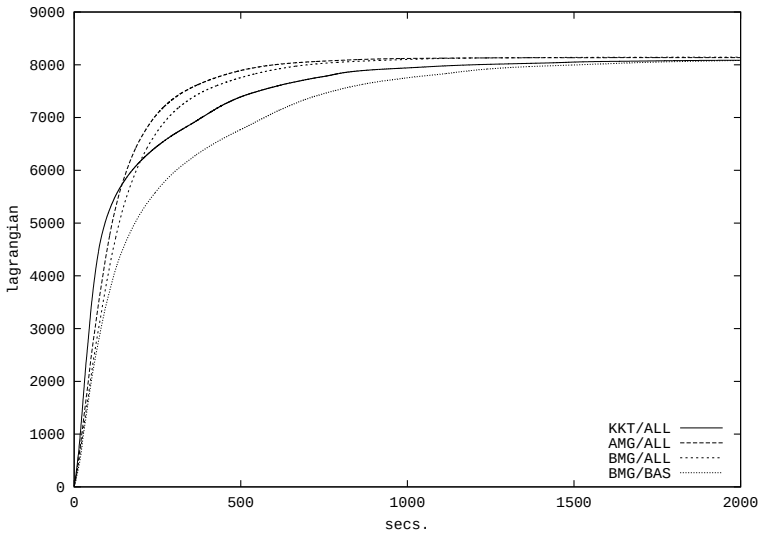

Figure 9: Comparison of different selection strategies using the heuristic MEAN.

logarithmic function is very slow to converge to zero, and because of that, the value returned by the strategies will be soon below the threshold. In particular the logarithmic function remains on a value of about 0.1 for many iterations. While this value is pretty good for the accuracy of the KKT solution, it is not sufficient for our selection schemes. In Figure 8 different heuristics for the computation of the value θ_V of the selection strategy of the SMO-like algorithm are compared. In this case the very simple heuristics MAX and MEAN reach similar performance, which is much better than the baseline scheme. In Figure 9, given the heuristic MEAN, different selection strategies

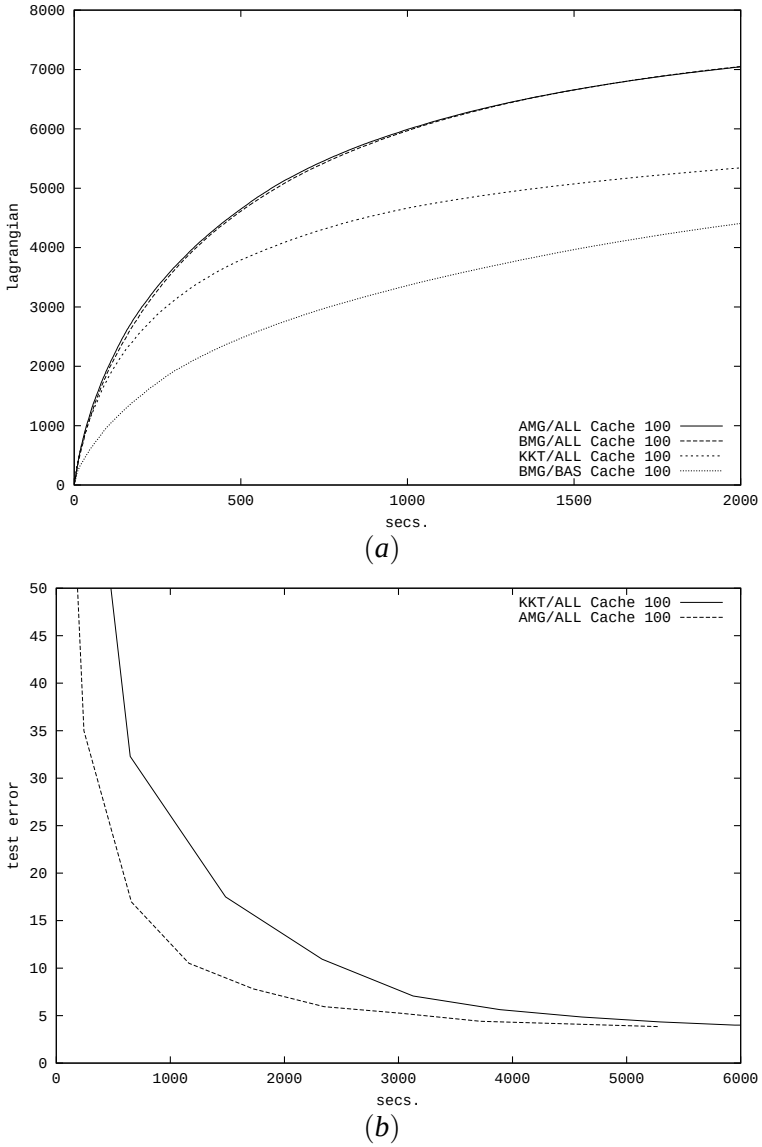

Figure 10: The effect of the cache limitation: (a) Lagrangian value versus time; (b) test performance versus time.

are compared. In this case, the new strategies slightly outperform the one based on Crammer and Singer's KKT conditions. Actually, as we will see in the following, this slight improvement is due to the big size of the cache of kernel matrix rows that prevents the algorithm suffering of the large amount of time spent in the computation of kernels that are not present in the cache.

In order to reproduce conditions similar to the ones occurring when dealing with large datasets, the size of the cache of kernel matrix rows has been reduced to 100 rows. As it is possible to see in figure 10-a a decrease in the performance is evident for each method, however, this decrease becomes more evident when KKT conditions are used as the pattern selection strategy. From the

same figure we can see also a quite poor performance when the basic version of the SMO-like is used as a global optimization method. This demonstrates how important is to solve the overall problem one pattern at time. In fact, this leads to a decrease of the total number of patterns selected for optimization and consequently to a decrease of the number of kernel computations. This puts also in evidence the amount of time spent in kernel computation versus the amount of time spent in the optimization. Figure 10-b clearly shows that the same argument can be applied to the recognition accuracy.

5.5 Brief Discussion

The type of strategies we have analyzed in earlier sections are very similar to the ones used by SMO (Platt, 1998), modified SMO (Keerthi et al., 1999) and svmlight (Joachims, 1999) algorithms for binary SVM. In these cases, linear constraints involving dual variables which are related to different patterns (derived by KKT conditions over the bias term) are present. However, in our case, as in the Crammer and Singer's algorithm (Crammer and Singer, 2001), constraints involve dual variables which are related to the same pattern (but over different prototypes). This makes a difference in the analysis since it turns out that it is convenient to optimize as much as possible the reduced problem obtained for a single pattern as this optimization does not require the computation of new kernels. This claim is supported by our experimental results comparing BMG-ALL vs. BMG-BAS in Figure 10.

Also, we have shown experimentally that the use of heuristics based on the increase of the Lagrangian tend to be faster than KKT based ones, when used for pattern selection (compare BMG-ALL vs. KKT-ALL in Figure 10). This can be due to the fact that the number of different patterns selected along the overall optimization process tends to be smaller and this largely compensates the inefficiency derived by the computation of the increase of the Lagrangian and the thresholds. On the other hand, according to the same experimental analysis, KKT conditions help when used for the choice of pairs to optimize in the reduced problems obtained for a given pattern. According to these considerations, this mixed approach has been adopted in the experiments that follow.

6. Optimization of General MProtSVM

By now, we have analyzed the (static) problem obtained when the assignment is given. In this section, we describe methods for the optimization with respect to the assignments π as well. Naturally, the full problem is no longer convex. So, we first present an efficient procedure that guarantees to reach a stationary point of the objective function of the problem in Eq. (8) associated to MProtSVM. Then, we insert it in a stochastic search framework with the aim to improve the quality of the solutions we find.

6.1 Greedy Optimization of MProtSVM

In the following, an algorithm for the optimization of the problem in Eq. (8) is described. The algorithm consists of two steps: a step in which, fixed the values for the set of variables α, we select the assignments π's in such a way to minimize the primal value, followed by a step in which the optimization of the variables α is performed once fixed the assignments. Each of these steps will lead to an improvement of the objective function thus guaranteeing the convergence to a stationary point.

Let suppose to start by fixing an initial assignment $\pi(1)$ for the patterns. As we have already seen, the associated problem is then convex and can be efficiently solved for example by using the general scheme in Figure 3. Once that the optimal value for the primal, let say $P^*_{\pi(1)}$, has been reached, we can easily observe that the solution can be further improved by updating the assignments in such a way to associate each pattern $\mathbf{x}_i$ to a positive prototype having associated the minimal slack value, i.e. by setting the vector $\pi_i(2)$ so to have the unique 1 corresponding to the best performing positive prototype. However, with this new assignment $\pi(2)$, the variables α may no longer fulfill the second admissibility condition in Eq. (10). If this is the case, it simply means that the current solution $M(\alpha)$ is not optimal for the primal (although still admissible). Furthermore, α cannot be optimal for the dual given the new assignment since it not even admissible. Thus, a Lagrangian optimization, done by keeping the constraints dictated by the admissibility conditions in Eq. (10) satisfied for the new assignment, is guaranteed to obtain a new α with a better optimal primal value $P^*_{\pi(2)}$, i.e. $P^*_{\pi(2)} \leq P^*_{\pi(1)}$. For the optimization algorithm to succeed, however, KKT conditions on α have to be restored in order to return back to a feasible solution and then finally resuming the Lagrangian optimization with the new assignment $\pi(2)$. Admissibility conditions can be simply restored by setting $\alpha_i = \mathbf{0}$ whenever there exists any $r \in \mathcal{P}_i$ such that the condition $\alpha^r_i > 0 \wedge \pi^r_i = 0$ holds. Note that, when the values assigned to the slack variables allow to define a new assignment for π corresponding to a new problem with a better optimal primal value, then, because of convexity, the Lagrangian of the corresponding dual problem will have an optimal value that is strictly smaller than the optimal dual value of the previous problem.

Performing the same procedure over different assignments, each one obtained from the previous one by the procedure described above, implies the convergence of the algorithm to a fixed-point consisting of a stationary point for the primal problem when no improvements are possible and the KKT complementarity conditions are all fulfilled by the current solution.

One problem with this procedure is that it can result onerous when dealing with large datasets or when using many prototypes since, in this case, many complete Lagrangian optimizations have to be performed. For this, we can observe that for the procedure to work, at each step, it is sufficient to stop the optimization of the Lagrangian when we find a value for the primal which is better than the last found value and this is going to happen for sure since the last solution was found not optimal. This requires only a periodic check of the primal value when optimizing the Lagrangian.

6.2 Stochastic Modifications for MProtSVM Optimization

Another problem with the procedure given in the previous section is that it leads to a stationary point (either a local minima or a saddle point) that can be very far from the best possible solution. Moreover, it is quite easy to observe that the problem we are solving is combinatorial. In fact, since the induced problem is convex for each possible assignment, then there will exist a unique optimal primal value $P^*(\pi, \alpha^*(\pi))$ associated with optimal solutions $\alpha^*(\pi)$ for the assignments π. Thus, the overall problem can be reduced to find the best among all possible assignments. However, when assuming an equal number q of prototype vectors for each class, there are q^n possible solutions with many trivial symmetries.

Given the complexity of the problem we are trying to optimize, we propose to resort to stochastic search techniques. Specifically, the approach we suggest can be considered an instance of Iterated Local Search (ILS). ILS is a family of general purpose metaheuristics for finding good solutions of combinatorial optimization problems (Lourenco et al., 2002). These algorithms are based on

building a sequence of solutions by first perturbing the current solution and then applying local search to that modified solution.

In the previous section, a way to perform approximated local search has been given. Let us now consider how to perturb a given solution. We propose to perform a perturbation that is variable with time and is gradually cooled by a simulated annealing like procedure.

For this, let us view the value of the primal as an energy function

$$E(\pi) = \frac{1}{2}||M||^2 + C\sum_i \langle \pi_i, \xi_i \rangle.$$

Let suppose to have a pattern $\mathbf{x}_i$ having slack variables ξ_i^r, $r \in \mathcal{P}_i$, and suppose that the probability for the assignment to be in the state of nature s (i.e. with the s-th component set to 1) follows the law

$$p_i(s) \propto e^{-\Delta E_s/T}$$

where T is the temperature of the system and $\Delta E_s = C(\xi_i^s - \xi_i^{y_i})$ the variation of the system energy when the pattern $\mathbf{x}_i$ is assigned to the s-th prototype. By multiplying every term $p_i(s)$ by the normalization term $e^{C(\xi_i^{y_i} - \xi_i^0)/T}$ where $\xi_i^0 = \min_{r \in \mathcal{P}_i} \xi_i^r$ and considering that probabilities over alternative states must sum to one, i.e. $\sum_{r \in \mathcal{P}_i} p_i(r) = 1$, we obtain

$$p_i(s) = \frac{1}{Z_i} e^{-\frac{C(\xi_i^s - \xi_i^0)}{T}} \tag{17}$$

with $Z_i = \sum_{r \in \mathcal{P}_i} e^{-C(\xi_i^r - \xi_i^0)/T}$ the partition function.

Thus, when perturbing the assignment for a pattern $\mathbf{x}_i$, each positive prototype s will be selected with probability $p_i(s)$. From Eq. (17) it clearly appears that, when the temperature of the system is low, the probability for a pattern to be assigned to a prototype different from the one having minimal slack value tends to 0 and we obtain a behavior similar to the deterministic version of the algorithm. The simulated annealing is typically implemented by decreasing the temperature, as the number of iterations increases, by a monotonic decreasing function $T = T(t, T_0)$.

Summarizing, an efficient realization of the ILS-based algorithm is obtained by substituting the true local optimization with one step of the algorithm in Section 6.1 and is given in Figure 11.

7. Generalization Ability of MProtSVM

In this section, we give a theoretical analysis of the generalization ability of the MProtSVM model. For simplicity, we consider MProtSVM with a fixed number q of prototypes per class. We first assume training data being separated by a MProtSVM model and we give a margin based upper bound on the error that holds with high probability on a independently generated set of examples. Then, we give a growth-function based bound on the error which do not assume linear separability of training data.

Margin based generalization bound Let us suppose that an *i.i.d.* sample $\mathcal{S}$ of n examples and a model M are given such that the condition in Eq. (7) holds for every example in $\mathcal{S}$, i.e.

$$\forall (\mathbf{x}_i, c_i) \in \mathcal{S}, \exists r \in \mathcal{P}_i : \ \langle M_r, \mathbf{x}_i \rangle \geq \theta_i + 1 \text{ and } \theta_i = \max_{r \in \mathcal{N}_i} \langle M_r, \mathbf{x}_i \rangle.$$

Annealed_MProtSVM()

$T := T_0$; randomly initialize $\pi(1)$;

compute the primal $E(1) := P_{\pi(1)}(0)$;

for $t = 1,..,t_{max}$

do for all the examples $\mathbf{x}_p \in S$

$\alpha_p =$ OptimizeOnPattern($\mathbf{x}_p$, ε);

until $P_{\pi(t)}(\alpha) < E(t)$;

compute a new assignment $\pi(t+1)$ using $T(t, T_0)$ in Eq. (17);

compute the new primal $E(t+1) := P_{\pi(t+1)}(\alpha)$;

restore KKT conditions on α /*see Section 6*/

end;

Figure 11: Fast annealed algorithm for the optimization of MProtSVM.

With this assumption, fixing a pattern $\mathbf{x}_p$, there will be at least one slack variable ξ_p^r, associated with it, equal to zero. In fact, the condition $\xi_p^r = 0$ is true at least in the case $r = y_p$, where y_p is the positive prototype associated to the pattern $\mathbf{x}_p$, i.e. such that $\pi_p^{y_p} > 0$.

To give the margin-based bound on the generalization error, we use the same technique as in an(Platt et al., 2000) for general Perceptron DDAG[1] (and thus SVM-DAG also), i.e. we show how the original multiclass problem can be reduced into one made of multiple binary decisions. The structure of our proof resembles the one given in (Crammer and Singer, 2000) for single-prototype multiclass SVM.

A Perceptron DDAG is a rooted binary DAG with N leaves labelled by the classes where each of the $K = m(m-1)$ internal nodes is associated with a perceptron able to discriminate between two classes. The nodes are arranged in a triangle with the single root node at the top, two nodes in the second layer and so on until the final layer of m leaves. The i-th node in layer $j < m$ is connected to the i-th and $(i+1)$-st node in the $(j+1)$-st layer. A Perceptron DDAG based classification can also be though of as operating on a list of classes with associated a set of perceptrons, one for each different pair of classes in the list. The evaluation of new patterns is made by evaluating the pattern with the perceptron discriminating the classes in the first and in the last position of the list. The losing class between the two is eliminated from the list. This process is repeated until only one class remains in the list and this class is returned.

Similarly, MProtSVM classification can be thought of as operating on a list. Suppose the index of prototypes $r \in R = \{1,\ldots,mq\}$ are ordered according to their class $\mathcal{C}(r) \in \{1,\ldots,m\}$. Then,

1. Note that the term "perceptron" here simply denotes a linear decision function which is not necessarily produced by the "perceptron algorithm".

given a new pattern, we compare the scores obtained by the two prototypes in the head and in the tail of the list and the loser is removed from the list. This is done until only prototypes of the same class remain on the list and this is the class returned for the pattern under consideration. It is easy to show that this procedure is equivalent to the rule in Eq. (2).

In the following, we will refer to the following theorem giving a bound on the generalization error of a Perceptron DDAG:

Theorem 2 *(Platt et al., 2000) Suppose we are able to classify a random sample of labelled examples using a Perceptron DDAG on m classes containing K decision nodes with margin γ_i at node i, then we can bound the generalization error with probability greater than $1-\delta$ to be less than*

$$\frac{1}{n}(130R^2D'\log(4en)\log(4n)+\log(\frac{2(2n)^K}{\delta}))$$

where $D'=\sum_{i=1}^{K}\gamma_i^{-2}$, and R is the radius of a ball containing the support of the distribution.

Note that, in this theorem, the margin γ_i for a perceptron $(\mathbf{w}_i,b_i)$ associated to the pair of classes (r,s) is computed as $\gamma_i=\min_{c_p\in\{r,s\}}|\langle\mathbf{w}_i,\mathbf{x}_p\rangle-b_i|$. Moreover, we can observe that the theorem depends only on the number of nodes (number of binary decisions) and does not depend on the particular architecture of the DAG.

Going back to MProtSVM, for the following analysis we define the hyperplane $\mathbf{w}_{rs}=M_r-M_s$ for each pair of prototypes indexes r,s such that $\mathcal{C}(r)<\mathcal{C}(s)$. and the *support* of the hyperplane $\mathbf{w}_{rs}$ as the subset of patterns

$$\Gamma_{rs}=\{i\in\{1,\ldots,n\}:(r\in\mathcal{P}_i\wedge\pi_i^r>0)\vee(s\in\mathcal{P}_i\wedge\pi_i^s>0)\}.$$

Now, we can define the margin of the classifier $h_{rs}(\mathbf{x})=\langle\mathbf{w}_{rs},\mathbf{x}\rangle$ as the minimum of the (geometrical) margins of the patterns associated to it, i.e.

$$\gamma_{rs}=\min_{i\in\Gamma_{rs}}\frac{|h_{rs}(\mathbf{x}_i)|}{||\mathbf{w}_{rs}||} \tag{18}$$

Note that, from the hypothesis of separation of the examples and from the way we defined the margin, we have $|h_{rs}(\mathbf{x}_i)|\geq 1$ and hence the lower bound on the margin $\gamma_{rs}\geq||\mathbf{w}_{rs}||^{-1}$.

Now, we can show that the maximization of these margins leads to a small generalization error by demonstrating the following result.

Lemma 3 *Suppose we are able to classify a random sample of labelled examples using a MProtSVM with q prototypes for each of the m classes with margin γ_{rs} when $\mathcal{C}(r)<\mathcal{C}(s)$, then we can bound the generalization error with probability greater than $1-\delta$ to be less than*

$$\frac{1}{n}(130R^2D\log(4en)\log(4n)+\log(\frac{2(2n)^K}{\delta}))$$

where $D=\sum_{r,s:\ \mathcal{C}(r)<\mathcal{C}(s)}\gamma_{rs}^{-2}$, $K=\frac{1}{2}q^2m(m-1)$, and R is the radius of a ball containing the support of the distribution.

Proof. First of all, we show that an MProtSVM can be reduced to a Perceptron DDAG. Let be given prototype indices $r \in R = \{1,\ldots,mq\}$ ordered according to their class $\mathcal{C}(r) \in \{1,\ldots,m\}$. Consider two cursors r and s initially set to the first and the last prototype in R, respectively. Now, we build a DAG node for (r,s) based on the classifier h_{rs}. Then, recursively, left and right edges are built associated to nodes $(r,s-1)$ and $(r+1,s)$ respectively. This is made until the condition $\mathcal{C}(r) = \mathcal{C}(s) = t$ holds. When this is the case, a leaf node is built instead with label t. This construction is based on the fact that there is not need to compare the scores obtained by prototypes associated to the same class.

We show now that the number of nodes in the skeleton of a DAG D which is built in this way is exactly $K = \frac{1}{2}q^2m(m-1)$. In fact, consider the DAG D' obtained by keeping on constructing DAG nodes (r,s) until the condition $r = s$ holds, instead of just $\mathcal{C}(r) = \mathcal{C}(s)$. This graph would be the same that would have been obtained by considering mq classes with one prototype each. Note that D' is balanced and it consists of $\frac{1}{2}mq(mq-1)$ nodes. It follows that, to obtain the DAG D, for each class y, we subtract the subDAG constructed by considering all possible $k_y = q(q-1)/2$ pairs of prototypes associated to that class.

Summarizing, the number of nodes of the DAG D is the number of nodes of the balanced DAG D' minus the total number of mk_y subDAG nodes. That is we get:

$$K = \frac{1}{2}mq(mq-1) - m(\frac{1}{2}q(q-1)) = \frac{1}{2}q^2m(m-1).$$

Now, we can apply Theorem 2, by considering a Perceptron DDAG with K nodes associated to pairs $r,s:\ \mathcal{C}(r) < \mathcal{C}(s)$ and the margin for the node (r,s) defined as in Eq. (18).□

By now, we have demonstrated that the minimization of the term $D = \sum_{r,s:\ \mathcal{C}(r)<\mathcal{C}(s)} \gamma_{rs}^{-2}$ proportional to the margin of the nodes of the Perceptron DDAG we have constructed, leads to a small generalization error. This result can then be improved by showing how these margins are linked to the norm of the MProtSVM matrix M and finally proving the following theorem.

Theorem 4 *Suppose we are able to classify a random sample of n labelled examples using a MProtSVM with q prototypes for each of the m classes and matrix M, then we can bound the generalization error with probability greater than* $1-\delta$ *to be less than*

$$\frac{1}{n}\left(130R^2q(m-1+q)||M||^2\log(4en)\log(4n) + \log(\frac{2(2n)^K}{\delta})\right)$$

where $K = \frac{1}{2}q^2m(m-1)$ *and R is the radius of a ball containing the support of the distribution.*

Proof. First of all, note that we have $\gamma_{rs}^{-2} \leq ||\mathbf{w}_{rs}||^2 = ||M_r - M_s||^2$ and $\sum_r M_r = 0$. The second condition can be easily verified. In fact, from conditions in Eq. (10), it follows

$$\sum_r M_r = \sum_r \sum_i y_i^r \alpha_i^r \mathbf{x}_i = \sum_i (\underbrace{\sum_r y_i^r \alpha_i^r}_{0})\mathbf{x}_i = 0.$$

Now, we have

$$\sum_{r,s:\ \mathcal{C}(r)<\mathcal{C}(s)} ||M_r - M_s||^2 \quad = \quad q(m-1)\sum_r ||M_r||^2 - 2\sum_{r,s:\mathcal{C}(r)<\mathcal{C}(s)} \langle M_r, M_s\rangle. \tag{19}$$

The second term of the equation above is

$$\begin{aligned} \sum_{r,s:\ \mathcal{C}(r)<\mathcal{C}(s)} \langle M_r, M_s \rangle &= \tfrac{1}{2}\left(\sum_{r,s} \langle M_r, M_s \rangle - \sum_{r,s,\ \mathcal{C}(r)=\mathcal{C}(s)} \langle M_r, M_s \rangle\right) \\ &= -\tfrac{1}{2} \sum_{r,s:\ \mathcal{C}(r)=\mathcal{C}(s)} \langle M_r, M_s \rangle \\ &= -\tfrac{1}{2}\left(\sum_r ||M_r||^2 + \sum_{r\neq s:\ \mathcal{C}(r)=\mathcal{C}(s)} \langle M_r, M_s \rangle\right). \end{aligned} \tag{20}$$

where the following inequality holds

$$\begin{aligned} \sum_{r\neq s,\ \mathcal{C}(r)=\mathcal{C}(s)} \langle M_r, M_s \rangle &\leq \sum_{j=1}^{m} \sum_{r\neq s:\ \mathcal{C}(r)=\mathcal{C}(s)=j} \langle M_r, M_s \rangle \\ &\leq q^2 \sum_{y=1}^{m} ||\tilde{M}_y||^2 \\ &\leq q^2 \sum_r ||M_r||^2 \end{aligned} \tag{21}$$

once we set $\tilde{M}_y = \arg\max_{r:\ \mathcal{C}(r)=y} ||M_r||$. Finally, substituting back Eq. (21) in Eq. (20) and Eq. (20) in Eq. (19) we obtain:

$$D \leq q(m-1+q) \sum_r ||M_r||^2$$

and the theorem easily follows. Note that this bound nicely generalizes the case of single prototype per class already shown in (Crammer and Singer, 2000).□

Growth function based generalization bound In the following, we give another kind of analysis of the generalization capability of our model based on the growth function. In order to do that, it is convenient to show that our multi-prototype model is equivalent to a three-layer network of perceptrons where the weights of the second and third layer are decided before learning. Thus, the free parameters of the network are only the weights of perceptrons in the first layer. As before, with no loss of generality, we assume to have q prototypes for each class $c \in \mathcal{Y}$.

Given a MProtSVM $H_M(\cdot)$, the corresponding network $\mathcal{N}_{H_M}$ is constructed as follows (we assume threshold perceptrons $(\mathbf{w}, \theta)$ with output $o(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle - \theta)$):

- First layer: $\forall r, s \in \Omega,\ \mathcal{C}(r) < \mathcal{C}(s)$, define the perceptron $h^{(1)}_{rs}$ with weight vector $\mathbf{w}^{(1)}_{rs} = M_r - M_s$ and $\theta^{(1)}_{rs} = 0$;
- Second layer (AND): $\forall u \in \mathcal{Y} = \{1, \ldots, m\}$, $\forall v \in \Omega : \mathcal{C}(v) = u$ define the perceptron $h^{(2)}_{uv}$, taking input from all $h^{(1)}_{rs}$ such that $r = v$ or $s = v$ and connection equal to 1 if $r = v$, -1 otherwise; set threshold to the value $\theta^{(2)}_{uv} = q(m-1) - 0.5$ ("on" if all the inputs are 1).
- Third layer (OR): $\forall w$ *suchthat* $w \in \mathcal{Y} = \{1, \ldots, m\}$ define the perceptron $h^{(3)}_w$, taking input from all $h^{(2)}_{uv}$ *suchthat* $w = u$ and connections all equal to 1; set the threshold to the value $\theta^{(3)}_w = 1/2$ ("on" if any input is 1).

See Figure 12 for an example of network construction when $q = 2$ and $m = 3$. Notice that, by construction, for any input there will be no two activated perceptrons $h^{(2)}_{uv}$ and $h^{(2)}_{\hat{u}\hat{v}}$ such that $u \neq \hat{u}$. So, only one out of the perceptrons at the third layer will be activated, and its index will correspond to the predicted class.

The constructed network has $\sigma = \frac{q^2 m(m-1)}{2}$ perceptrons at the first layer. Since only these perceptrons have trainable weights, the VC-dimension of the network only depends on these free parameters (apart for the hard-threshold functions).

We are now ready to state the following result.

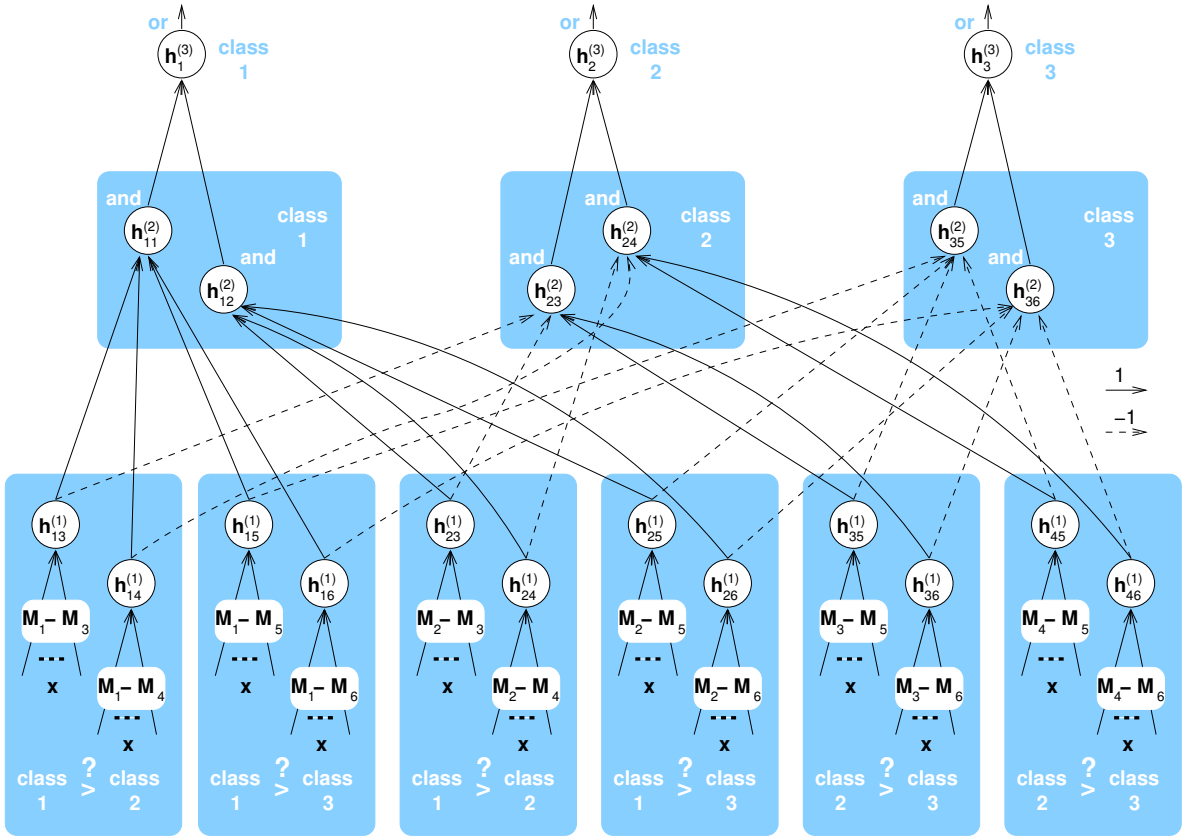

Figure 12: Example of network construction when $q = 2$ and $m = 3$. With this setting, prototypes M_1 and M_2 are associated with class 1, prototypes M_3 and M_4 are associated with class 2, and prototypes M_5 and M_6 are associated with class 3.

Theorem 5 *For any $0 < \delta < 1$, any MProtSVM $H_M(\cdot)$ with q prototypes for each of the m classes, given $\mathcal{S}$ a sample of size n drawn i.i.d. from $\mathcal{D}_{\mathbb{R}^d \times \{1,\ldots,m\}}$, with probability at least $1-\delta$*

$$err_{\mathcal{D}}(H_M) \leq err_{\mathcal{S}}(H_M) + \sqrt{4\frac{1+(qm+\frac{1}{2})ln(qm)+\frac{dq^2m(m-1)}{2}ln(2en/d)-ln(\delta/4)}{n}} + \frac{1}{n}.$$

Proof. By the above construction, the class of functions computable by a MProtSVM with q prototypes for each of the m classes is contained in the class of functions computable by three-layer perceptrons defined as above. This class of functions is completely characterized by the set of collective states that the σ perceptrons at the first layer can assume. It is well known (Kearns and Vazirani, 1994) that, by Sauer Lemma, the growth function of a single perceptron (with 0 threshold) is bounded from above by the quantity $(en/d)^d$, and so the growth function of our class of networks is bounded from above by the quantity $(en/d)^{d\sigma}$.

This bound, however, does not consider that not all the possible configurations of σ bits can be generated by the first layer. In fact, by construction of the network, we have that if $h_{rs}^{(1)}(\mathbf{x}) = 1$

and $h^{(1)}_{s\hat{s}}(\mathbf{x}) = 1$, then for sure $h^{(1)}_{r\hat{s}}(\mathbf{x}) = 1$, as well as, if $h^{(1)}_{rs}(\mathbf{x}) = 0$ and $h^{(1)}_{s\hat{s}}(\mathbf{x}) = 0$, then for sure $h^{(1)}_{r\hat{s}}(\mathbf{x}) = 0$. This is the result of the fact that, given an input vector $\mathbf{x}$, the outputs of the first layer perceptrons are fully determined by the total order over the MProtSVM prototypes induced by the score functions $f_r(\cdot)$. Thus we can compute an upper bound on the proportion of 'legal' configurations by considering all possible permutations of the qm prototypes divided by all possible configurations, i.e., 2^σ. Notice that this is an upper bound since when considering prototypes of the same class, we do not care about their relative order.

So we can bound the growth function of our class of networks by

$$\frac{(qm)!}{2^\sigma}(en/d)^{d\sigma} < \sqrt{2\pi qm}(qm/e)^{qm} e^{\frac{1}{12qm}} 2^{-\sigma}(en/d)^{d\sigma},$$

where the last inequality has been obtained by using Stirling's formula.

Making explicit the value of σ, the right term of the above inequality can be written as

$$\sqrt{2\pi}(qm)^{qm+\frac{1}{2}}(n/d)^{\frac{dq^2m(m-1)}{2}} e^{\frac{6q^3m^2(m-1)(d-ln(2))-12q^2m^2+1}{12qm}}.$$

Now, we can apply Theorem 4.1 in (Vapnik, 1998) (involving the logarithm of the growth function for a sample of dimension $2n$) obtaining

$$\begin{aligned} err_{\mathcal{D}}(H_M(\cdot)) &\leq err_{\mathcal{S}}(H_M(\cdot)) + \\ &\quad + \sqrt{4\frac{ln(\sqrt{2\pi}(qm)^{qm+\frac{1}{2}}(\frac{2n}{d})^{\frac{dq^2m(m-1)}{2}} e^{\frac{6q^3m^2(m-1)(d-ln(2))-12q^2m^2+1}{12qm}}) - ln(\frac{\delta}{4})}{n}} + \frac{1}{n} \\ &\leq err_{\mathcal{S}}(H_M(\cdot)) + \sqrt{4\frac{1+(qm+\frac{1}{2})ln(qm)+\frac{dq^2m(m-1)}{2}ln\frac{2en}{d} - ln\frac{\delta}{4}}{n}} + \frac{1}{n} \end{aligned}$$

□

8. Experimental Results

In the following, we report experiments we have done for testing the complexity and the generalization performance of the MProtSVM model with respect to other state-of-the-art algorithms. We choose to compare our model against results already published in literature on different datasets instead of doing experiments with those methods directly. This is because we have not available the code for all those methods and hence a re-implementation would be necessary. This can potentially introduce errors or uncorrect use of the methods and it is far more onerous for us. For this, we experimented on our model trying to replicate the initial conditions of published results as much as possible in such a way to obtain fair comparisons.

For all the following experiments, the linear kernel $K(\mathbf{x},\mathbf{y}) = (\langle \mathbf{x},\mathbf{y}\rangle + 1)$ has been used. Moreover, the annealing process required by MProtSVM has been implemented by decreasing the temperature of the system with the exponential law:

$$T(t,T_0) = T_0(1-\tau)^t$$

where t is the current iteration, $0 < \tau < 1$ and $T_0 > 0$ are external parameters. We used $T_0 = 10$ for all the following experiments. In addition, the only free parameter C of MProtSVM has been selected

by performing validation of the model on a subset of the training set with values $C = \{10^k, k = -2, .., 2\}$.

Initially, we tested our model against three multiclass datasets that we briefly describe in the following:

NIST: it consists of a 10-class task of 10705 digits randomly taken from the NIST-3 dataset. The training set consists of 5000 randomly chosen digits, while the remaining 5705 digits are used in the test set.

USPS: it consists of a 10-class OCR task (digits from 0 to 9) whose input are the pixels of a scaled digit image. There are 7291 training examples and 2007 test examples.

LETTER: it consists of a task with 26 classes consisting of alphabetic letters A-Z. Inputs are measures of the printed font glyph. The first 15000 examples are used for training and the last 5000 for testing.

q	LVQ2.1 Error %	MProtSVM Error %
1	7.43	6.45
5	4.68	3.63
10	4.35	3.28
15	3.52	2.80

Table 1: Comparison of generalization performances between MProtSVM and LVQ with increasing number of prototypes/codewords (NIST dataset, $\tau = .05$, $\beta = 0.002 \times q$).

q	USPS Error (%)
1	8.12
3	6.13
5	5.83
10	5.48
15	5.23
20	5.00

q	LETTER Error (%)
1	21.36
3	9.64
5	6.42
10	4.84
15	3.16
20	2.94

Table 2: (a) Test error of MProtSVM on the USPS dataset ($\tau = .05$, $\beta = 0.00137 \times q$), with an increasing number of prototypes; (b) Test error of MProtSVM on the LETTER dataset ($\tau = .05$, $\beta = 0.00043 \times q$), with an increasing number of prototypes.

A first set of experiments have been performed to compare the generalization performance of our (linear) model versus LVQ (Kohonen et al., 1996), which seemed to us the most comparable model, into an OCR task. For this, we have reported the results obtained by the LVQ2.1 version of the algorithm in (Sona et al., 2000) on the NIST problem. Configurations with a higher number of codewords started to overfit the data. As it can be seen in Table 1, MProtSVM performs significantly

better with the same number of parameters. This can be due to the more effective control of the margin for our model w.r.t. LVQ models. On the same dataset, the tangent-distance based TVQ algorithm (Aiolli and Sperduti, 2002b) has obtained the best result, a remarkable 2.1% test error, and polynomial SVM's have obtained a 2.82% test error. These last results should not surprise since their models are well suited for OCR tasks. Here and in the following experiments we report the value of the factor $\beta = (m \times q)/n$ defined as the number of prototypes produced as a fraction of the cardinality of the training set. This represent a sort of factor of compression in the model.

A second set of experiments have been performed to test the MProtSVM model against state-of-the-art methods on two well known datasets: UCI Irvine USPS and LETTER. The obtained results are reported in Table 2. As it is possible to see, by combining a reasonably high number of linear prototypes, we have been able to obtain performances almost comparable with the ones obtained using non-linear models. In fact, on the USPS dataset, we obtained a 4.63% error using an our own SProtSVM implementation with polynomial kernel of degree 3 and without further preprocessing of the data. Finally, a 5.63% test error performance has been obtained using 1-NN. Concerning the LETTER dataset, the results should be compared to versus the 1.95% obtained in (Crammer and Singer, 2001) by SProtSVM with exponential kernel and to the 4.34% obtained by 1-NN. Although obtained with a slightly different split of the LETTER dataset (16000 examples for training and 4000 for test), we would like to mention the results reported in (Michie et al., 1994) where LVQ yielded a 7.9%.

From these experiments it is clear that MProtSVM returns far more compact models with respect to state of the art non-linear kernel methods allowing a (one or two order) reduced response time in classification while preserving a good generalization performance. In fact, the above experiments have shown very low values for the compression factor β (e.g. 26×20 prototypes in the LETTER dataset gives $\beta = 0.013$ and 10×20 prototypes for USPS gives $\beta = 0.0274$). Notice that β can be directly compared with the fraction of support vectors in kernel machines. Thus, MProtSVMs also give us a way to decide (before training) the compression factor we want to obtain.

	— Vectors —		**——— Errors ———**					
Dataset	**SVM**	**RVM**	**SVM**	**RVM**	**MProtSVM**			
					q=1	q=3	q=5	q=10
Banana	135.2	11.4	10.9	**10.8**	46.0	12.8	[11.0]	11.0
Breast Cancer	116.7	6.3	**26.9**	29.9	28.2	**26.9**	27.5	[27.0]
German	411.2	12.5	22.6	**22.2**	[23.6]	23.8	23.5	23.7
Image	166.6	34.6	3.0	3.9	15.0	3.2	2.7	[**2.5**]
Titanic	93.7	65.3	**22.1**	23.0	[22.5]	22.2	22.2	22.2
Waveform	146.4	14.6	10.3	10.9	13.3	10.8	**10.0**	[10.2]

Table 3: Comparison of solution complexity and generalization error of MProtSVM with respect to SVM and Tipping's RVM on a set of UCI binary datasets. The results quoted for SVM and RVM are taken from (Tipping, 2001). Values in brackets are the ones obtained using the model suggested by model selection performed over the number of prototypes.

To further validate our claim, we made a comparison of our technique against others that explicitly try to obtain compact models. In Table 3 we reported the results obtained with six binary

Dataset	# Features	Train Size	Test Size
Banana	2	400	4900
Breast Cancer	9	200	77
German	20	700	300
Image	18	1300	1010
Titanic	3	150	2051
Waveform	21	400	4600

Table 4: General information about the UCI binary datasets used in the experiments.

problems (see Table 4 for general information about these datasets) from the benchmark of Rätsch available over the web[2], exactly the ones used by (Tipping, 2001) of which we are reporting the obtained results for RVM and SVM. They correspond to averages over the first 10 splits of the collection. For each dataset, it is reported the average number of support vectors generated by SVM and RVM, the generalization error obtained with these two methods and the results obtained using four very simple MProtSVM configurations, namely made of 1, 3, 5 and 10 prototypes per class (2, 6, 10 and 20 vectors in total, respectively[3]). It is possible to see how very compact and simple models performs as good as (sometimes better than) state-of-the-art methods. Values in brackets represent the error value obtained using the model suggested by the validation procedure we have performed over the number of models per class.

Finally, in Table 5 we report an example of the values obtained for the objective function of the primal problem in Eq. (8) along with their corresponding test errors obtained using different configurations and lowering the simulated annealing parameter τ on the USPS dataset. As expected, once fixed a raw in the table, better values for the primal can generally be obtained with lower values of τ. Moreover, as the number of prototypes per class increases, the choice of small τ tends to be more crucial. Anyway, higher values for τ, and thus not optimal values for the primal, can nevertheless lead to good generalization performances. Notice that from the fact that the primal value is just a way to approximate the theoretical SRM principle and from the non-optimality of the parameter C in these experiments, better values for the primal does not necessarily correspond to better values for the test error.

9. Conclusions

We have proposed an extension of multiclass SVM able to deal with several prototypes per class. This extension defines a non-convex problem. We suggested to solve this problem by using a novel efficient optimization procedure within an annealing framework where the energy function corresponds to the primal of the problem. Experimental results on some popular benchmarks demonstrated that it is possible to reach very competitive performances by using few linear models per class instead of a single model per class with kernel. This allows the user to get very compact models which are very fast in classifying new patterns. Thus, according to the computational constraints, the user may decide how to balance the trade-off between better accuracy and speed of

2. http://ida.first.gmd.de/∼raetsch
3. When one prototype per class is used in a binary problem, as in this case, MProtSVM actually generates two vectors that are the same with sign inverted. Thus, they can be compacted into one vector only with no loss of information.

q	$\tau = 0.2$	$\tau = 0.1$	$\tau = 0.05$	$\tau = 0.03$
3	7.44626 (6.33%)	7.28049 (6.03%)	7.08138 (6.13%)	7.04274 (6.48%)
5	7.49136 (6.08%)	7.27318 (5.63%)	7.10498 (5.83%)	7.00946 (5.58%)
10	7.82233 (5.58%)	7.51780 (5.88%)	7.27596 (5.48%)	7.12517 (5.23%)
15	7.82222 (5.33%)	7.57009 (5.73%)	7.38722 (5.33%)	7.22250 (5.53%)
20	7.78410 (5.48%)	7.79388 (5.72%)	7.49125 (5.38%)	7.21303 (5.53%)

Table 5: Primal values and generalization error obtained with different configurations varying the parameter τ for the USPS dataset.

classification. Finally, it should be noted that the proposed approach compares favorably versus LVQ, a learning procedure that, similarly to the proposed approach, returns a set of linear models.

Preliminary experiments with kernels have shown negligible improvements that makes us to consider this extension not worthwhile of further investigations. An alternative more interesting extension would be to try to combine different types of kernels together in the same model.

Acknowledgments

This work has been developed when Fabio Aiolli was a PhD student and postdoc at the Department of Computer Science, Univ. of Pisa. The research was partially supported by Italian MIUR project 2003091149_005. We would like to thank Y. Singer, K. Crammer, and the anonymous reviewers for their valuable suggestions on how to improve the paper.

References

F. Aiolli and A. Sperduti. An efficient SMO-like algortihm for multiclass SVM. In *Proceedings of IEEE workshop on Neural Networks for Signal Processing*, pages 297–306, 2002a.

F. Aiolli and A. Sperduti. A re-weighting strategy for improving margins. *Artificial Intelligence Journal*, 137/1-2:197–216, 2002b.

F. Aiolli and A. Sperduti. Multi-prototype support vector machine. In *Proceedings of International Joint Conference of Artificial Intelligence (IJCAI)*, 2003.

E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 2000.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 35–46, 2000.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based machines. *Journal of Machine Learning Research*, 2(Dec):265–292, 2001.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

G. M. Fung, O. L. Mangasarian, and A. J. Smola. Minimal kernel classifiers. *Journal of Machine Learning Research*, 3:303–321, 2002.

Y. Guermeur, A. Elisseeff, and H. Paugam-Moisy. A new multi-class SVM based on a uniform convergence result. In *Proceedings of the IJCNN*, 2000.

T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. B. Schlkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for SVM classifier design. Technical Report CD-99-14, Control Division, Dept. of Mechanical and Production Engineering, National University of Singapore, 1999.

T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. Lvq_pak: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, January 1996. http://www.cis.hut.fi/nnrc/nnrc-programs.html.

H. R. Lourenco, O. C. Martin, and T. Stutzle. Iterated local search. *Handbook of Metaheuristics*, Ed. F. Glover and G. Kochenberger, International Series in Operations Research & Management Science(57):321–353, 2002.

D. Michie, D. Speigelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

J. Platt, N. Cristianini, and J. Shawe Taylor. Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K. R. Muller, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2000.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, 1998.

J. R. Quinlan. *C4.5: Programs for Empirical Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

D. E. Rumelhart, G. E. Hinton, and R.J Williams. Learning internal representation by error propagation. In *Parallel Distributed Processing - Explorations in the Microstructure of cognition*, chapter 8, pages 318–362. MIT Press, 1986.

B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Muller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 5 (10):1000–1017, 1999.

B. Schölkopf and C. Burges and V. Vapnik. Extracting support data for a given task. In *First International Conference on Knowledge Discovery & Data Mining*, pages 252–257, 1995.

D. Sona, A. Sperduti, and A. Starita. Discriminant pattern recognition using transformation invariant neurons. *Neural Computation*, 12(6), 2000.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

J. Weston and C. Watkins. Multiclass support vector machines. In M. Verleysen, editor, *Proceedings of ESANN99*. D. Facto Press, 1999.