Multi-prototype Support Vector Machine

Fabio Aiolli
Dept. of Computer Science
University of Pisa, Italy
aiolli@di.unipi.it

Alessandro Sperduti
Dept. of Pure and Applied Mathematics
University of Padova, Italy
sperduti@math.unipd.it

Abstract

We extend multiclass SVM to multiple prototypes per class. For this framework, we give a compact constrained quadratic problem and we suggest an efficient algorithm for its optimization that guarantees a local minimum of the objective function. An annealed process is also proposed that helps to escape from local minima. Finally, we report experiments where the performance obtained using linear models is almost comparable to that obtained by state-of-art kernel-based methods but with a significant reduction (of one or two orders) in response time.

1 Introduction

Automatic multiclass classification, i.e. the process to automatically assign exactly one from a prefixed set of labels to a stream of input instances, is a central task for many real world problems like speech recognition, OCR, text categorization etc. Many supervised learning methods have been studied that help on these tasks. Recently, kernel-based methods, like SVM, have been well studied especially for binary settings, and they yielded state-of-art performance in many different tasks. SVM searches for a high margin linear discriminant model in a very high dimensional space (feature space) where examples are implicitly mapped via a function $\Phi(x)$. Since kernel-based algorithms need only of dot products in feature space, it is possible to resort to the so called kernel trick if these dot products can be computed efficiently with a kernel function $K(x, y) = \Phi(x) \cdot \Phi(y)$. When the number of examples (or support vectors) is large, these approaches tend to be less efficient w.r.t. the time spent for classifying new vectors since they require to work with the implicit characterization of the discriminant model.

One of the most effective SVM extension able to deal with general multiclass problems has been recently provided [Crammer and Singer, 2000] and it has shown very good results. In this paper, we extend the multiclass SVM to the multi-prototype setting where for each class there can be more than one prototype vector. This allowed us to get very expressive decision functions by using simple models without necessarily requiring the use of kernels.

1.1 Preliminaries and notation

Let $S = \{(x_1, c_1), .., (x_n, c_n)\}$ be a set of n training examples, $x_i \in X \subseteq \mathbb{R}^d$ and $c_i \in C = \{1, .., m\}$. . single-label multi-class classifier is a function $h : X \mapsto C$ that maps instances $x \in X$ to labels $c \in C$. We focus on the class of *winneMake-all* linear classifiers with form:

$$h_M(x) = C\left(\arg\max_{r \in R}\{M_r \cdot x\}\right) \qquad \text{(i)}$$

where M_r is the r-th prototype vector, R is the set of prototype indexes and $C(r)$ the function returning the class associated to the prototype indexed r. Given a classifier $h_M(x)$ and a training example (x_i, c_i) we will say that $h_M(x)$ misclassifies x_i if $h_M(x_i) \neq c_i$.

In the following, we denote by y_i^r the constant that is equal to 1 if $C(r) = c_i$ and -1 otherwise. For a given example x_i, we denote by $P_i = \{r \in R : y_i^r = 1\}$ the set of "positive" prototypes indexes and by $N_i = \{r \in R : y_i^r = -1\}$ the set of "negative" prototypes indexes. Finally, the real value $s_r(x) = M_r \cdot x$ is referred to as *similarity score* (or simply *score)* of the r-th prototype on the instance x.

2 Single-prototype multi-class SVM

An effective multi-class extension to SVM has been already proposed in [Crammer and Singer, 2000]. The resulting classifier is of the same form of (1) where each class has associated exactly one prototype, i.e. $R \equiv C$ and $C(r) = r$. The solution is obtained through the minimization of a convex quadratic constrained problem. In this section, we present a simpler derivation of an equivalent formulation that will be extended in the next section to the multi-prototype framework.

In the multiclass setting, in order to have a correct classification of the instance x_i, the prototype of the correct class should have a score that is greater than the maximum among the scores of the prototypes associated to incorrect classes. We can formally write the constraints for a correct classification of the example x_i with a margin greater or equal to 1 by requiring:

$$M_{y_i} \cdot x_i \geq \theta_i + 1 \text{ and } \theta_i = \max_{r \neq y_i} M_r \cdot x_i,$$

where $y_i = d$ is the index of the prototype vector associated to the correct class for the example x_i. To allow for margin

violations, for each example, we introduce soft margin slack variables $\xi_i \geq 0$, s.t.

$$\xi_i = [\theta_i + 1 - M_{y_i} \cdot x_i]_+,$$

where $[x]_+$ denotes the hinge loss equal to x if $x > 0$ and 0 otherwise. Notice that the value ξ_i is an upper bound to the 0 - 1 loss for the example X_i, and consequently its average value over the training set will be an upper bound for the empirical error.

Motivated by the structural risk minimization (SRM) principle in [Vapnik, 1998], we want to search for a set of prototype vectors $M = \{Mi, .., M\backslash c\backslash\}$ with small norm such to minimize the empirical error on the training set. For this, we can formulate the problem in a SVM-style by requiring a set of prototypes of minimal norm that fulfill the soft constraints given by the classification requirements. Thus, the single-prototype multiclass SVM (SProtSVM) will result in:

$$\min_{M,\xi,\theta} \tfrac{1}{2}||M||^2 + \gamma \textstyle\sum_i \xi_i \quad \text{subject to: } \begin{cases} \forall i, r \neq y_i,\ M_r \cdot x_i \leq \theta_i, \\ \forall i,\ M_{y_i} \cdot x_i \geq \theta_i + 1 - \xi_i \\ \forall i, \xi_i \geq 0 \end{cases} \tag{2}$$

Notice that, as desired, the optimal solution for θ_i will be the maximum value among the negative scores for the instance X_i. This problem 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:

$$\begin{aligned} L(M,\xi,\theta,\alpha,\lambda) &= \tfrac{1}{2}||M||^2 + \gamma \textstyle\sum_i \xi_i + \\ &\quad \textstyle\sum_{i,r\neq y_i} \alpha_i^r (M_r \cdot x_i - \theta_i) + \\ &\quad \textstyle\sum_i \alpha_i^{y_i} (\theta_i + 1 - \xi_i - M_{y_i} \cdot x_i) - \\ &\quad \textstyle\sum_i \lambda_i \xi_i \\ &= \tfrac{1}{2}||M||^2 - \textstyle\sum_{i,r} y_i^r \alpha_i^r (M_r \cdot x_i - \theta_i) + \\ &\quad \textstyle\sum_i \alpha_i^{y_i} + \sum_i (\gamma - \alpha_i^{y_i} - \lambda_i)\xi_i, \end{aligned} \tag{3}$$

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 the set of constraints (KKT conditions) that the variables must fulfill in order to be an optimal solution:

$$\begin{aligned} \tfrac{\partial L(M,\xi,\theta,\alpha,\lambda)}{\partial M_r} = 0 &\Leftrightarrow M_r = \textstyle\sum_i y_i^r \alpha_i^r x_i \\ \tfrac{\partial L(M,\xi,\theta,\alpha,\lambda)}{\partial \xi_i} = 0 &\Leftrightarrow \gamma - \alpha_i^{y_i} - \lambda_i = 0 \Leftrightarrow \alpha_i^{y_i} \leq \gamma \\ \tfrac{\partial L(M,\xi,\theta,\alpha,\lambda)}{\partial \theta_r} = 0 &\Leftrightarrow \alpha_i^{y_i} = \textstyle\sum_{r\neq y_i} \alpha_i^r \end{aligned} \tag{4}$$

By using the fact $\alpha_i^{y_i} = \frac{1}{2}\sum_r \alpha_i^r$ and substituting conditions (4) in (3) and omitting constants that do not change the solution, the problem can be restated as:

$$\max_\alpha \textstyle\sum_{i,r} \alpha_i^r - ||M(\alpha)||^2 \quad \text{subject to: } \begin{cases} \forall i, r,\ \alpha_i^r \geq 0 \\ \forall i, \alpha_i^{y_i} = \sum_{r\neq y_i} \alpha_i^r \leq \gamma \end{cases} \tag{5}$$

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

3 Multi-prototype multi-class SVM

In this section, the SProtSVM model previously defined will be extended to the case of multiple prototypes per class. The basic idea is that, given multiple prototype vectors, we have a correct classification iff *at least* one of the prototypes associated to the correct class has a score higher than the maximum of the scores of the prototypes associated to incorrect classes.

In this case, we write the constraints for a correct classification of the example X_i with a margin greater or equal to 1 requiring:

$$\exists r \in P_i :\ M_r \cdot x_i \geq \theta_i + 1 \text{ and } \theta_i = \max_{r\in N_i} M_r \cdot x_i.$$

To allow for margin violations, for each example x_i, we introduce soft margin slack variables $\xi_i^r \geq 0$, one for each positive prototype, s.t.

$$\forall r \in P_i,\ \xi_i^r = [\theta_i + 1 - M_r \cdot x_i]_+$$

Given a pattern X_i, we arrange the soft margin slack variables f J* in a vector $\boldsymbol{\xi}_i \in \mathbb{R}^{|P_i|}$. Let us now introduce, for each example X_i a new vector $\boldsymbol{\pi}_i \in \{0,1\}^{|P_i|}$, whose components are all zero except one component that is 1. In the following, we refer to $\boldsymbol{\pi}_i$ as the assignment of the pattern x_i. Notice that the dot product $\boldsymbol{\pi}_i \cdot \boldsymbol{\xi}_i$ is always an upper bound to the 0—1 loss for the example X; independently from its assignment.

Now, we are ready to formulate the 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 will result in:

$$\min_{M,\xi,\theta,\pi} \tfrac{1}{2}||M||^2 + \gamma \textstyle\sum_i \boldsymbol{\pi}_i \cdot \boldsymbol{\xi}_i \quad \text{subject to: } \begin{cases} \forall i, r \in N_i,\ M_r \cdot x_i \leq \theta_i, \\ \forall i, r \in P_i,\ M_r \cdot x_i \geq \theta_i + 1 - \xi_i^r \\ \forall i, r \in P_i, \xi_i^r \geq 0 \end{cases} \tag{6}$$

Unfortunately, this is a mixed integer problem that is not convex and it is a difficult problem in general but, as we will see in the following, it is prone to an efficient optimization procedure that approximates the global optimum. At this point, it is worth noticing that, since the effective formulation is itself an (heuristic) approximation to the structural risk minimization principle, a good solution, although not optimal, can anyway give good results. As we see after, this claim is confirmed by the results obtained in our experimental work.

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

$$\begin{aligned} L_\pi(M,\xi,\theta,\alpha,\lambda) &= \tfrac{1}{2}||M||^2 + \gamma \textstyle\sum_i \boldsymbol{\pi}_i \cdot \boldsymbol{\xi}_i + \\ &\quad \textstyle\sum_{i,r\in P_i} \alpha_i^r (\theta_i + 1 - \xi_i^r - M_r \cdot x_i) - \\ &\quad \textstyle\sum_{i,r\in P_i} \lambda_i^r \xi_i^r + \\ &\quad \textstyle\sum_{i,r\in N_i} \alpha_i^r (M_r \cdot x_i - \theta_i), \end{aligned} \tag{7}$$

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:

$$\begin{array}{lll} \frac{\partial L_P(M,\xi,\theta,\alpha,\lambda)}{\partial M_r} = 0 & \Leftrightarrow & M_r = \sum_i y_i^r \alpha_i^r x_i \\ \frac{\partial L_P(M,\xi,\theta,\alpha,\lambda)}{\partial \xi_i^r} = 0 & \Leftrightarrow & \gamma\pi_i^r - \alpha_i^r - \lambda_i^r = 0 \Leftrightarrow \alpha_i^r \leq \gamma\pi_i^r \\ \frac{\partial L_P(M,\xi,\theta,\alpha,\lambda)}{\partial \theta_i} = 0 & \Leftrightarrow & \sum_{r\in P_i} \alpha_i^r = \sum_{r\in N_i} \alpha_i^r \end{array}$$

Notice that the second condition requires the dual variables associated to positive prototypes and not assigned to the associated pattern to be 0. By denoting now as y_i the unique index $r \in P_i$ such that $\pi_i^r = 1$, once using the conditions (8) in (7) and omitting constants that do not change the obtained solution, the reduced problem can be restated as:

$$\begin{array}{l} \max_\alpha \sum_{i,r} \alpha_i^r - ||M(\alpha)||^2 \\ \text{subject to: } \left\{ \begin{array}{l} \forall i, r,\ \alpha_i^r \geq 0 \\ \forall i,\ \alpha_i^{y_i} = \sum_{r\in N_i} \alpha_i^r \leq \gamma \\ \forall i, r \in P_i \setminus \{y_i\},\ \alpha_i^r = 0 \end{array} \right. \end{array} \quad (9)$$

It can be trivially shown that this formulation is consistent with the formulation for SProtSVM's given above.

3.1 Optimization with static assignments

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 (6) can be found through the maximization of the lagrangian in (9).

Assuming an equal number *q* of prototypes per class, the dual involves *n* x *m* x *q* variables which lead 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 x *q* variables.

The algorithm we propose for the optimization of the problem in (9) is inspired by the ones already presented in [Crammer and Singer, 2000; Aiolli and Sperduti, 2002a] and consists in iteratively selecting patterns from the training set and optimizing with respect to the variables associated to that pattern. From the convexity, each iteration leads to an increase of the lagrangian until no more improvement is possible since the optimal solution for the lagrangian has been found.

After selecting a pattern X_i, to enforce the constraint $\sum_{r\in N_i} \alpha_i^r + \lambda_i = \gamma,\ \lambda_i \geq 0$, two elements from the set of variables $\{\alpha_i^r, r \in N_i\} \cup \{\lambda_i\}$ will be optimized in pair while keeping the solution inside the feasible region. In particular, let w_i and w_2 be the two selected variables, we restrict the updates to the form $\omega_1 \leftarrow \omega_1 + \rho$ and $\omega_2 \leftarrow \omega_2 - \rho$ with optimal choices for *p*. Iterating the application of this basic step over different pairs and then doing the same for different patterns will guarantee to reach the optimum for the overall problem.

Let now compute the optimal value for *p*. First of all, we observe that the update will affect the squared norm of the prototype vector M_r of an amount

$$\Delta||M_r||^2 = ||\Delta M_r||^2 + 2M_r \cdot \Delta M_r$$

Now, we show how to analytically solve the associated problem with respect to an update involving a single variable $\alpha_p^r, r \in N_i$ and the variable λ_p. 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 $\alpha_p^{y_p}$ to keep the constraint $\alpha_p^{y_p} = \sum_{r\in N_p} \alpha_p^r$ satisfied and then to enforce the constraints $\lambda_p = \gamma - \sum_{s\in N_i} \alpha_p^s \geq 0$. Thus, in this case we have:

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

Since $\Delta M_r = -\rho x_p$, $\Delta M_{y_i} = \rho x_p$ and $\Delta M_s = 0$ for $s \notin \{r, y_i\}$, we obtain $\Delta||M||^2 = \Delta||M_r||^2 + \Delta||M_{y_i}||^2 = 2\rho^2||x_p||^2 + 2\rho(s_{y_p}(x_p) - s_r(x_p))$ and $\Delta L(\alpha) = 2\rho(1 - s_{y_p}(x_p) + s_r(x_p) - \rho||x_p||^2)$. Since this last formula is concave on ρ, it is possible to find the optimal value when the first derivative is null, i.e.

$$\hat{\rho} = \arg\max_\rho \Delta L(\alpha) = \frac{1 - s_{y_p}(x_p) + s_r(x_p)}{2||x_p||^2}$$

If the values of $a^r{}_p$ and *aft*, after being updated, would turn out to be not feasible for the constraints $a^r{}_p > 0$ and *aft* < 7, we select the value for *p* such to fulfill the violated constraint bounds at the limit.

Now, we show how to analytically solve the associated problem with respect to an update involving a pair of variables $a^{\wedge 1}$, $a_\rho{}^2$ such that *r*\, r_2 *E* N_i and = ^ *r-i.* Since, in this case, the update must have zero sum, we have:

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

In this case, $\Delta M_{r_1} = -\rho x_p$, $\Delta M_{r_2} = \rho x_p$ and $\Delta M_s = 0$ for $s \notin \{r_1, r_2\}$, we obtain $\Delta||M||^2 = \Delta||M_{r_1}||^2 + \Delta||M_{r_2}||^2 = 2\rho^2||x_p||^2 + 2\rho(s_{r_2}(x_p) - s_{r_1}(x_p))$ and $\Delta L(\alpha) = 2\rho(s_{r_1}(x_p) - s_{r_2}(x_p) - \rho||x_p||^2)$. Since also this last formula is concave on ρ, it is possible to find the optimal value

$$\hat{\rho} = \arg\max_\rho \Delta L(\alpha) = \frac{s_{r_1}(x_p) - s_{r_2}(x_p)}{2||x_p||^2}$$

Similarly to the previous case, if the values of the $\alpha_p^{r_1}$ and $a^r{}_\rho{}^2$, after being updated, would 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 value for *p* such to fulfill the violated constraint bounds at the limit.

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. This has been exploited in [Aiolli and Sperduti, 2002a] to devise an incremental algorithm that uses the method on the reduced problem of a single example and then iterates over different examples. This optimization procedure can be considered incremental in the sense that the solution previously found for a given pattern forms the initial condition when the pattern is selected again for optimization. In this version, the basic step of optimization of the reduced problem can require the optimization over all the $((m-1) \times q + 1)((m-1) \times q)/2$ pairs of variables not constrained to 0 associated with the selected pattern. Thus the complexity of the optimization of the reduced problem is $o(I \times (m \times q)^2)$ where *I* is the number of iterations.

Now, we perform a further step by giving an algorithm that at each iteration has a complexity *o(m x q).* For this, we observe that for the variables $\{\alpha_p^r, \lambda_p\}$ associated to the pattern

x_p to be optimal, the feasible analytic solution p must be 0 for each pair.

In particular, for the first case above, in order to be able to apply the step, it is required that, one of the following two conditions are verified, i.e.:

$$(s_{y_i}(x_p) < \max_{r \in N_p} s_r(x_p) + 1) \wedge (\alpha_p^{y_i} < \gamma)$$
$$\vee$$
$$(s_{y_i}(x_p) > \max_{r \in N_p, \alpha_p^r > 0} s_r(x_p) + 1) \wedge (\alpha_p^{y_i} > 0)$$

while for the second case we must have:

$$\max_{r \in N_p} s_r(x_p) > \min_{r \in N_p, \alpha_p^r > 0} s_r(x_p).$$

Notice that these facts can be checked in linear time. It is easy to show that the solution obtained at a certain step can be improved iff one of these facts are verified. This suggests an efficient procedure, shown in Figure I(Top), that tries to greedily fulfill the previous condition of optimality.

3.2 Optimization of general MProtSVM

In this section, we present an efficient procedure that guarantees to reach a local minimum of the objective function of the problem in (6) associated to MProtSVM. This procedure will try to optimize also with respect to the assignments π_i.

Let suppose to start with a given assignment $\pi(1)$ for the patterns. In this case, as we have already seen, the associated problem is convex and can be efficiently solved by using the algorithm given in Section 3.1. Once that the optimal value for the primal $P^*_{\pi(1)}$ has been reached, we observe that the solution can be further improved by updating the assignments in such a way to assign each pattern x_i to a positive prototype with minimal slack value, i.e. by setting the vector $\pi_i(2)$ such to have the unique 1 corresponding to the best performing positive prototype. However, with this new assignment 7r(2), the variables α might not fulfill the second KKT condition in eq. (8) anymore. If it is the case, it simply means that the current solution α is not optimal for the new assignment. Thus, a lagrangian optimization done by satisfying constraints dictated by KKT conditions for the new assignment is guaranteed to obtain a new α with a better optimal primal value $P^*_{\pi(2)}$. For the optimization algorithm to succeed, however, the KKT conditions on the α must be restored in order to return back to a feasible solution and then finally resuming the lagrangian optimization with the new assignment 7r(2). We restore the KKT conditions by setting $\alpha_i = 0$ whenever there exists $r \in P_i$ such that $\alpha_i^r > 0 \wedge \pi_i^r = 0$.

Performing the same procedure over different assignments, each one obtained from the previous one by the procedure just mentioned, implies the convergence of the algorithm to a local solution for the primal problem when no improvements are possible and the KKT conditions are all fulfilled by the current solution. This is due to the fact that every step induces an improvement on the primal value.

The first problem with this procedure is that it results onerous when dealing with many prototypes since we must perform many lagrangian optimizations. For this, we 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

OptimizeOnPattern(x_p,ϵ)

$\forall r, s_r := s_r(x_p), K_{pp} = ||x_p||^2$;
do
 if $(\alpha_p^{y_p} = 0)$ **then** {
  $r_1 := \arg\max_{r \in N_p} s_r$; $\rho_1 := \min\{\frac{1 - s_{y_p} + s_{r_1}}{2K_{pp}}, \gamma\}$;
  $V_1 := 2\rho_1(1 - s_{y_p} + s_{r_1} - \rho_1 K_{pp})$; $k := 1$}
 else {
  $r_1 := \arg\max_{r \in N_p} s_r$;
  $r_2 := \arg\max_{r \in N_p, \alpha_p^r > 0} s_r$;
  $r_3 := \arg\min_{r \in N_p, \alpha_p^r > 0} s_r$;
  $\rho_1 := \min\{\frac{1 - s_{y_p} + s_{r_1}}{2K(x_p, x_p)}, \gamma - \alpha_p^{y_p}\}$;
  $\rho_2 := \max\{\frac{1 - s_{y_p} + s_{r_2}}{2K(x_p, x_p)}, -\alpha_p^{r_2}\}$;
  $\rho_3 := \min\{\frac{s_{r_1} - s_{r_3}}{2K(x_p, x_p)}, \alpha_p^{r_3}\}$;
  $V_1 := 2\rho_1(1 - s_{y_p} + s_{r_1} - \rho_1 K_{pp})$;
  $V_2 := 2\rho_2(1 - s_{y_p} + s_{r_2} - \rho_2 K_{pp})$;
  $V_3 := 2\rho_3(s_{r_1} - s_{r_3} - \rho_3 K_{pp})$;
  $k := \arg\max_j V_j$ };
 case k **of** {
  1: $\alpha_p^{y_p} := \alpha_p^{y_p} + \rho_1$; $\alpha_p^{r_1} := \alpha_p^{r_1} + \rho_1$;
   $s_{y_p} := s_{y_p} + \rho_1 K_{pp}$; $s_{r_1} := s_{r_1} - \rho_1 K_{pp}$;
  2: $\alpha_p^{y_p} := \alpha_p^{y_p} + \rho_2$; $\alpha_p^{r_2} := \alpha_p^{r_2} + \rho_2$;
   $s_{y_p} := s_{y_p} + \rho_2 K_{pp}$; $s_{r_2} := s_{r_2} - \rho_2 K_{pp}$;
  3: $\alpha_p^{r_1} := \alpha_p^{r_1} + \rho_3$; $\alpha_p^{r_3} := \alpha_p^{r_3} - \rho_3$;
   $s_{r_1} := s_{r_1} - \rho_3 K_{pp}$; $s_{r_3} := s_{r_3} + \rho_3 K_{pp}$}
until $(V_k \leq \epsilon)$;
return $\{\alpha\}$

AnnealedMProtSVM()

$T := T_0$; randomly initialize $\pi(1)$;
compute the primal $E(1) := P_{\pi(1)}(0)$;
for $t = 1, .., t_{max}$
 do for all the examples $x_p \in S$
  α_p = OptimizeOnPattern(x_p , ϵ);
 until $P_{\pi(t)}(\alpha) < E(t)$;
 compute a new assignment $\pi(t+1)$ using $T(t, T_0)$;
 compute the new primal $E(t+1) := P_{\pi(t+1)}(\alpha)$;
 restore KKT conditions on α; /*see Section 3.2*/
end;

Figure 1: *Top:* algorithm for the optimization of the variables associated with a given pattern x_p and a tolerance *e. Bottom:* algorithm for the optimization of MProtSVM.

for the primal better than the last found and this is going to happen for sure since the last solution has been found not optimal. This requires only a periodic check of the primal value when optimizing the lagrangian.

The second and more stringent problem is that the procedure will lead to a local minimum that can be very far from the best possible. For this, we suggest to update the assignments on the basis of a stochastic annealed function instead of the hard decision function described above.

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

$$E(\pi) = \frac{1}{2}||M||^2 + \gamma \sum_{i \in S} \pi_i \cdot \xi_i$$

Let suppose to have a pattern X_i having slack variables ξ_i^r, $r \in P_i$ and suppose that the probability for the assignment to be in the state *s* (i.e. with the 5-th component set to 1) follows the law

$$p_i(s) \propto \exp(-\Delta E_s / T)$$

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

$$p_i(s) = \exp(-\frac{\gamma(\xi_i^s - \xi_i^0)}{T})/Z$$

with $Z = \sum_{r \in P_i} \exp(-\gamma(\xi_i^r - \xi_i^0)/T)$ the partition function.

Thus, when assigning the pattern *Xi,* each positive prototype *s* will be selected with probability *Pi(s).* Notice 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 not annealed 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 $T = T(t,T_0)$. The full algorithm is depicted in Figure 1 (Bottom).

4 Experimental Results

We tested our model against three datasets that we briefly describe in the following:

N1ST: 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 26-class task (alphabetic letters A-Z). Inputs are measures of the printed font glyph. The first 16000 examples are used for training and the last 4000 for testing.

Even if any kernel function can be in principle used, for the following preliminary experiments, we used the linear kernel $K(x, y) = (x \cdot y + 1)$. This allowed us to write an optimized code for training and classification that works directly with the explicit (compact) version of the model *M.*

Since we are primarily interested into the evaluation of MProtSVM w.r.t. SProtSVM, for each dataset, we performed validation on the parameter γ for the SProtSVM model and we re-used the obtained value for training every MProtSVM generated for the same dataset. This approach seems us anyway a pessimistic estimate of the performance of MProtSVM.

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 0 < r < 1 and T_0 > 0 are external parameters. We used To = 10 for all the following experiments.

q	LVQ2.1 Error %	MProtSVM Error %
1	7.43	6.45
5	4.68	3.70
10	4.35	3.28
15	3.52	3.24

Table 1: Comparison of generalization performances between LVQ and MProtSVM increasing the number of codebooks/prototypes on the NIST dataset ($\beta = 0.2 \times q$).

q	USPS Error (%)	q	LETTER Error (%)
1	8.12	1	21.36
3	6.13	2	13.42
5	5.83	3	9.64
10	5.48	5	6.42
15	5.33	10	4.84
20	5.38	15	3.16

Table 2: (a) Test error of MProtSVM on the USPS dataset ($\gamma = 0.01, \tau = .05, \beta = 0.137 \times q$), with an increasing number of prototypes; (b) Test error of MProtSVM on the LETTER dataset ($\gamma = 10, \tau = 0.1, \beta = 0.043 \times q$), with an increasing number of prototypes.

A 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. For this, we have reported the best results that have been obtained by LVQ on the NIST dataset. Specifically, they have been obtained with the LVQ2.1 version of the algorithm (see [Sona and Sperduti, 2000]). As it is possible to see in Table 1, MProtSVM performs significantly better when few prototypes per class are used, while the difference gets lower when the number of prototypes per class increases. This can be due to the more effective control of the margin for SVM 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. In

	$\tau = 0.2$	$\tau = 0.1$	$\tau = 0.05$	$\tau = 0.03$
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 3: Primal values and generalization error obtained with different configurations varying the parameter r (USPS dataset).

addition, we tested the MProtSVM on the UCI Irvine USPS and LETTER datasets. As it is possible to see in Table 2, by combining a reasonably high number of linear prototypes, we have obtained performances almost comparable with the ones obtained in literature by using non-linear models. In fact, on the USPS dataset, we have been able to get a 4.63% error, using a SProtSVM with polynomial kernel of degree 3 and without preprocessing the data, while for LETTER, we refer to the 1.95% obtained in [Crammer and Singer, 2001] by SProtSVM with exponential kernel. Although obtained with a slightly different split of the LETTER dataset (15000 examples for training and 5000 for test), we would like to mention the results reported in [Michie *et al.,* 1994] where LVQ and k-NN yielded a 7.9% and 6.8% error, respectively.

Notice that MProtSVM returns far more compact models with respect to state of the art non-linear kernel methods, thus allowing a (one or two order) reduced response time in classification. Defining a sort of model complexity factor β = 100 x (m x *q)/n,* i.e. the number of prototypes produced as a fraction of the cardinality of the training set, the above experiments have shown very low value for β (e.g. 15 x 26 prototypes in the LETTER dataset gives β = 0.65% and 20 x 10 prototypes for USPS gives β = 2.74%). 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 complexity of the model.

Finally, in Table 3 we have reported the values of the objective function of the primal problem in (6) along with their corresponding test errors obtained with the USPS dataset using different configurations and lowering the parameter r. As expected, fixed a raw in the table, better values for the primal can be obtained with lower values of r. Moreover, as the number of prototypes per class increases, the choice of small *r* tends to be more crucial. Anyway, higher values for r, and thus not optimal values for the primal, can neverthless lead to good generalization performances. This can be due to the fact that the primal value is just a way to approximate the theorethical SRM principle.

5 Conclusion

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 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 per class.

References

[Aiolli and Sperduti, 2002a] Fabio Aiolli and Alessandro Sperduti. An efficient smo-like algorithm for multiclass svm. In *Proceedings of IEEE workshop on Neural Networks/or Signal Processing,* 2002.

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

[Crammer and Singer, 2000] Koby Crammer and Yoram 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.

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

[Kohonen *et al.,* 1996] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. Lvq_pak: The learning vector quantization program package. Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, Rakentajanaukio 2 C, SF-02150 Espoo, Finland, January 1996. http://www.cis.hut.fi/nnrc/nnrc-programs.html.

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

[Sona and Sperduti, 2000] Diego Sona and Alessandro Sperduti. Discriminant pattern recognition using transformation-invariant neurons. *Neural Computation,* 12:1355-1370,2000.

[Vapnik, 1998] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.