A FAST TRAINING ALGORITHM FOR UNBIASED PROXIMAL SVM

Felipe A. C. de Bastos and Marcello L. R. de Campos

Electrical Engineering Program
COPPE/Federal University of Rio de Janeiro
P.O.Box 68504, 21945-970, Rio de Janeiro, RJ, Brazil
fcaetano@ipd.eb.mil.br and campos@lps.ufrj.br

ABSTRACT

This paper presents a new algorithm for fast training of unbiased Proximal Support Vector Machines. PSVM was first introduced as an alternative to SVM classifiers that usually require a large amount of computation time for training. Unfortunately PSVM may present poor performance, especially for low values of a regularization parameter C, due to biased optimal hyperplanes. The proposed algorithm, named UPSVM (Unbiased Proximal Support Vector Machines), uses a slightly different approach to circumvent this problem, such that an unbiased optimal hyperplane is always obtained. Simulations show that the proposed algorithm performs better than PSVM and Sequential Minimal Optimization (SMO) with respect to training time with similar probability of correct pattern classification.

1. INTRODUCTION

Support Vector Machines (SVM) can be used to classify an M-dimensional pattern in one of two different classes. The classification surface is a hyperplane on the input space (linear SVMs) or on the feature space (non-linear SVMs) obtained after applying a non-linear transformation to the input space. For linear classifiers, the separating hyperplane in the input space is given by

$$\mathbf{x}^T\mathbf{w}_o + b_o = 0 \tag{1}$$

where $\mathbf{w} \in \mathcal{R}^{\mathcal{M}}$ and $b \in \mathcal{R}$ denote weight vector and bias, respectively, and $\mathbf{x} \in \mathcal{R}^{\mathcal{M}}$ is a vector in the input space. An input pattern, $\mathbf{x}_i$, is said to belong to class C_1 if $\mathbf{x}_i^T\mathbf{w}_o + b_o \geq 0$, and to belong to class C_2 otherwise. For non-linear classifiers, the separating hyperplane in the feature space is given by

$$g^T(\mathbf{x})\mathbf{w}_o + b_o = 0 \tag{2}$$

were $g(\mathbf{x})$ is a mapping function that maps the M-dimensional input space into the L-dimensional feature space.

In the case of linearly separable patterns the support vectors are defined as the closest data to the optimal hyperplane, i.e., without loss of generality, for a linear classifier, a support vector $\mathbf{x}_S$ must satisfy

$$\mathbf{x}_S^T\mathbf{w}_o + b_o = \begin{cases} 1 & \text{if } \mathbf{x}_S \in C_1 \\ -1 & \text{if } \mathbf{x}_S \in C_2 \end{cases} \tag{3}$$

The distance between the hyperplanes defined by Eq. 3 is called margin of separation. It can be easily proved that the margin of separation is inversely proportional to the magnitude of $\mathbf{w}$. When input patterns are not linearly separable (soft margin), the support vectors are still given by Eq. 3, but some input patterns may lay inside the margin of separation, or on the wrong side of the optimal hyperplane. In this case, classification error may occur, and the optimal hyperplane parameters are obtained by the minimization of the training error along with the maximization of the margin of separation. The latter can be understood as a way to obtain classification error minimization for unknown data. This can be achieved by solving the following linearly-constrained convex minimization problem:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \left[J(\mathbf{w},\boldsymbol{\xi}) = \left(\frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{p}\sum_{i=1}^{N}\xi_i^p \right) \right] \quad \text{subject to} \quad \mathbf{D}(\mathbf{X}^T\mathbf{w} + b\mathbf{e}) \geq \mathbf{e} - \boldsymbol{\xi} \tag{4}$$

where $\mathbf{X} \in \mathcal{R}^{\mathcal{M}\times\mathcal{N}}$ is a matrix of observations, $\mathbf{x}_i \in \mathcal{R}^{\mathcal{M}}$ is the ith column of $\mathbf{X}$, $\mathbf{D}$ is a diagonal matrix with class label D_{ii} equal to 1 if $\mathbf{x}_i$ belongs to class C_1 or -1 otherwise. $\boldsymbol{\xi} \in \mathcal{R}^{\mathcal{N}}$ is a vector whose ith element ξ_i is zero if and only if $\mathbf{x}_i$ is bounded by the respective support hyperplane, otherwise ξ_i is equal to the distance from $\mathbf{x}_i$ to the respective support hyperplane. Vector $\mathbf{e}$ has all its elements equal to one, and p defines the classifier type. When $p = 1$ the obtained classifier is called L_1 large margin SVM (L1SVM), and when $p = 2$ it is called L_2 large margin SVM (L2SVM). Parameter C is the regularization factor that measures how much emphasis is given to the minimization of the training error.

The L1SVM classifier can be obtained by the solution of the dual-problem described below [1]:

$$\min_{\boldsymbol{\alpha}} \left[J(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{D}\mathbf{K}\mathbf{D}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{e} \right] \quad \text{subject to: } \begin{cases} \boldsymbol{\alpha}^T\mathbf{D}\mathbf{e} = 0 \\ 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, N \end{cases} \tag{5}$$

where $\boldsymbol{\alpha}$ is the Lagrange multiplier vector associated with the constraint given by Eg. 4, $\mathbf{K} \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$ is a kernel matrix whose element $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = g^T(\mathbf{x}_i)g(\mathbf{x}_j)$ for non-linear classifiers, or $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$ in the case of linear classifiers, and N is the number of training observations. A radial-basis function (RBF) kernel, defined below, is often used in non-linear classifiers:

$$K_{ij} = \exp\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right) \tag{6}$$

where σ is a parameter defined by the user. Figure 1 shows the use of a hyperplane for the classification of input vectors from two non-linearly separable classes.

After training, an optimal $\boldsymbol{\alpha}_o$ is obtained. The distance from the input vector $\mathbf{y}$ to the optimal hyperplane is given by

$$\text{dist}(\mathbf{y}) = \mathbf{k}_\mathbf{y}^T\mathbf{D}\boldsymbol{\alpha}_o + b_o \tag{7}$$

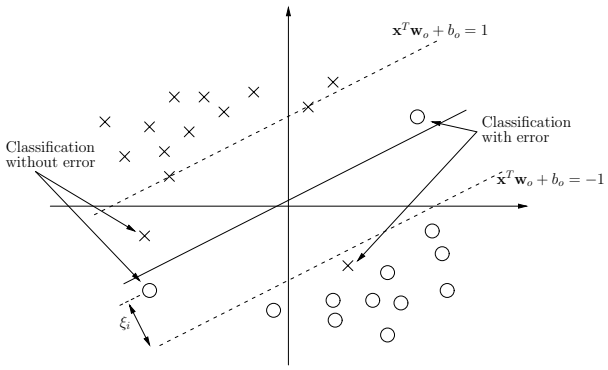

Fig. 1. Example for non-linearly separable classes.

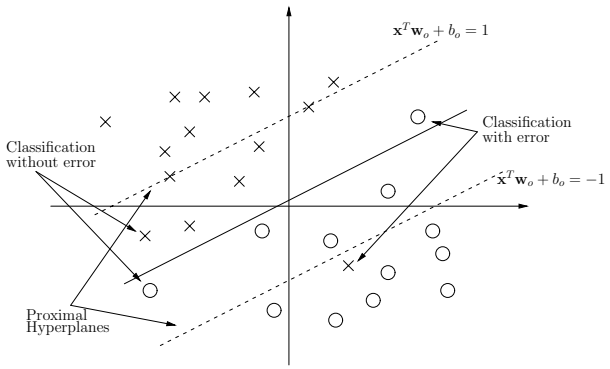

Fig. 2. Example for non-linearly separable classes with proximal hyperplanes.

where

$$b_o = d_S - \mathbf{k}_{x_S}^T \mathbf{D}\boldsymbol{\alpha}_o \tag{8}$$

with $d_S = \pm 1$, $\mathbf{k_y} = [K(\mathbf{y}, \mathbf{x}_1), K(\mathbf{y}, \mathbf{x}_2), \ldots, K(\mathbf{y}, \mathbf{x}_r)]^T$, $\mathbf{x}_i$ is an input training vector associated with a non-zero α_i, and $\mathbf{y}$ is a vector not used during training. If dist$(\mathbf{y}) \geq 0$ then $\mathbf{y} \in C_1$, otherwise $\mathbf{y} \in C_2$. Eq. 7 can be used also in the case of linear classifiers, although the classification can be further simplified:

$$\mathbf{w}_o = \mathbf{X}\mathbf{D}\boldsymbol{\alpha}_o \tag{9}$$

2. THE PSVM FORMULATION

PSVM was first introduced by Mangasarian and Fung [2] as a fast training algorithm for a modified SVM problem. The PSVM associated with DAGSVM [3] was used by Li and others in [4] for the generalization of the PSVM classifier for more than 2 classes. The formulation for the linear case is described below:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \left[J(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2} C \boldsymbol{\xi}^T \boldsymbol{\xi} + \frac{1}{2} \left(\mathbf{w}^T \mathbf{w} + b^2 \right) \right]$$
$$\text{subject to} \quad \boldsymbol{\xi} = \mathbf{e} - \mathbf{D}\left(\mathbf{X}^T \mathbf{w} + \mathbf{e}b \right) \tag{10}$$

In Eq. 10 above, the term b^2 added to the objective function is an artificial tool included to obtain an analytic solution for $\mathbf{w}$, b, and $\boldsymbol{\xi}$ based on a small system of linear equations. A similar approach had been used before in the formulation of the Active SVM proposed by Mangasarian and Musicant [5]. In section 4, we will show that the inclusion of b^2 in Eq. 10 does not improve classification performance, but rather, it will decrease the probability of correct classification on training and test sets for small values of C. This happens because much effort is made to minimize the magnitudes of $\mathbf{w}$ and b simultaneously, which implies that b be unnecessarily small and the optimal hyperplane be biased, i.e., closer to the origin than necessary.

Eq. 10 also shows that the traditional SVM inequality constraint is replaced by an equality constraint. This drastically changes the nature of the support hyperplanes ($\mathbf{w}_o^T \mathbf{x} + b_o = \pm 1$). These hyperplanes are no longer bounding hyperplanes. Instead, they correspond to "proximal hyperplanes," around which the points of each class are clustered. Figure 2 illustrates this approach.

Using Lagrange for solving the PSVM minimization problem and considering $\boldsymbol{\alpha}$ as a vector of Lagrange multipliers, the solution for the PSVM problem can be obtained as:

$$\boldsymbol{\alpha}_o = \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{e} \tag{11}$$

$$\mathbf{w}_o = \mathbf{X}\mathbf{D}\boldsymbol{\alpha}_o \tag{12}$$

$$b_o = \mathbf{e}^T \mathbf{D}\boldsymbol{\alpha}_o \tag{13}$$

$$\boldsymbol{\xi} = \frac{\boldsymbol{\alpha}_o}{C} \tag{14}$$

where $\mathbf{H} = [\mathbf{X}^T \quad -\mathbf{e}]$.

Eq. 11 was obtained after substituting $\mathbf{w}_o$, b_o, and $\boldsymbol{\xi}$, given by Eqs. 12 to 14, respectively, in the equality constraint given in Eq. 10. We can use the matrix inversion Lemma [6] to obtain:

$$\boldsymbol{\alpha}_o = C \left[\mathbf{I} - \mathbf{H} \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \right] \mathbf{e} \tag{15}$$

The solution given by Eq. 15 is better than the one given by Eq. 11 because it involves the inversion of an $M \times M$ matrix, where M is the number of observation parameters which is much smaller than N, the number of observations used in training.

For the non-linear case, the PSVM classifier is given by the following equations (see [2] for the details):

$$\boldsymbol{\alpha}_o = \left(\frac{1}{C}\mathbf{I} + \mathbf{G}\mathbf{G}^T \right)^{-1} \mathbf{e} \tag{16}$$

$$b_o = \mathbf{e}^T \mathbf{D}\boldsymbol{\alpha}_o \tag{17}$$

$$\boldsymbol{\xi} = \frac{\boldsymbol{\alpha}_o}{C} \tag{18}$$

where $\mathbf{G} = [\mathbf{K} \quad -\mathbf{e}]$.

The PSVM leads to an analytic solution for a modified SVM problem . Its main advantage compared to traditional SVM problems is its low computational complexity for the linear case. For the non-linear case, reduced-kernel techniques [2] can be used to reduce the $N \times N$ dimensionality of the kernel matrix $\mathbf{K}$. In the next section, a new algorithm is proposed that will eliminate the dependence of the objective function to the bias parameter b and therefore eliminates a disadvantage of PSVM: the biased optimal hyperplane obtained for small values of the regularization parameter C.

3. UNBIASED PROXIMAL SVM

3.1. Linear UPSVM

As discussed in the last section, the minimization of b^2 does not lead to improved classifier generalization or to less training errors. Instead, it can decrease the probability of correct classification of test data and increase the number of training errors. A new algorithm is proposed that does not take into account parameter b

in the objective function to be minimized. Using proximal hyperplanes, the new problem for the linear case can be mathematically formulated as

$$\min_{\mathbf{w},\boldsymbol{\xi}} \left[J(\mathbf{w},\boldsymbol{\xi}) = \frac{1}{2} C \boldsymbol{\xi}^T \boldsymbol{\xi} + \frac{1}{2} \mathbf{w}^T \mathbf{w} \right] \quad \text{subject to} \quad \boldsymbol{\xi} = \mathbf{e} - \mathbf{D}\left(\mathbf{X}^T \mathbf{w} + \mathbf{e}b\right) \tag{19}$$

Substituting the equality constraint into the objective function in Eq. 19, we obtain

$$\min_{\mathbf{w},b} \left[J(\mathbf{w},b) = \frac{1}{2} C \left(\mathbf{w}^T \mathbf{X}\mathbf{X}^T \mathbf{w} + 2b\mathbf{w}^T \mathbf{X}\mathbf{e} + b^2 N + N - 2\mathbf{w}^T \mathbf{X}\mathbf{D}\mathbf{e} - 2b\mathbf{e}^T \mathbf{D}\mathbf{e} \right) + \frac{1}{2} \mathbf{w}^T \mathbf{w} \right] \tag{20}$$

Eq. 20 shows the dependence of the objective function to the hyperplane parameters $\mathbf{w}$ and b. In order to find a solution for $\mathbf{w}$ and b, the gradient of $J(\mathbf{w}, b)$ is calculated, which gives

$$\frac{\partial J}{\partial \mathbf{w}} = C\left(\mathbf{X}\mathbf{X}^T \mathbf{w} + b\mathbf{X}\mathbf{e} - \mathbf{X}\mathbf{D}\mathbf{e}\right) + \mathbf{w} = \mathbf{0} \tag{21}$$

$$\frac{\partial J}{\partial b} = C\left(\mathbf{w}^T \mathbf{X}\mathbf{e} + bN - \mathbf{e}^T \mathbf{D}\mathbf{e}\right) = 0 \tag{22}$$

Supposing the same number of training observations for classes C_1 and C_2, then $\mathbf{e}^T\mathbf{D}\mathbf{e} = 0$. Solving Eqs. 21 and 22, yields

$$\mathbf{w}_o = \left[\frac{1}{C}\mathbf{I} + \mathbf{X}\left(\mathbf{I} - \frac{1}{N}\mathbf{e}\mathbf{e}^T\right)\mathbf{X}^T\right]^{-1} \mathbf{X}\mathbf{D}\mathbf{e} \tag{23}$$

$$b_o = -\frac{\mathbf{e}^T \mathbf{X}^T \mathbf{w}_o}{N} \tag{24}$$

An analytic solution for a Proximal SVM has been found, but without taking into account the minimization of b. It means that the optimal hyperplane will be unbiased even for small values of C. The solution in Eqs. 23 and 24 also involves the inversion of an $M \times M$ matrix, which guarantees low computational complexity ($O(N)$).This complexity is much lower than that of the Sequential Minimal Optimization (SMO) algorithm [7] used to train L1SVM. Simulations have shown that the proposed UPSVM algorithm is faster than SMO during training and has similar performance for classifying test data.

3.2. Computational Complexity for the Linear UPSVM

The number of floating-point multiplications and additions can be calculated for the linear UPSVM described by Eqs. 23 and 24 as follows:

Table 1. Computational complexity for the linear UPSVM

OPERATION		MULT.	ADD.
$\mathbf{X}\mathbf{D}\mathbf{e}$		0	$M(N-1)$
$\mathbf{X}\mathbf{X}^T$	(T.1)	$M(M+1)N/2$	$M(M+1)(N-1)/2$
$\mathbf{X}\mathbf{e}/N$		M	$(N-1)$
$\mathbf{X}\mathbf{e}(\mathbf{X}\mathbf{e})^T/N$	(T.2)	$M(M+1)/2$	0
$\mathbf{I}/C + (\text{T.1}) - (\text{T.2})$		1	$M(M+1)/2$
Choleski [6]		$M^3/6$	$M^3/6$
Back./Fwd. subst. [6]		$M^2 + M + 2$	$M^2 + M$
Calculation of b		$NM + 1$	$(N-1)(M-1)$

As can be seen the number of floating-point operations for the linear UPSVM is approximately equal to NM^2 for N much greater than M, and its computational complexity is $O(N)$.

3.3. Non-linear UPSVM

To obtain a non-linear UPSVM we will proceed in the same way that traditional SVM problems were generalized [1], i.e., substituting $\mathbf{X}^T\mathbf{X}$ by the kernel matrix $\mathbf{K}$. From Eq. 23,

$$\mathbf{w}_o = C\left[\mathbf{I} - \mathbf{X}\left(\frac{\mathbf{I}}{C} + \mathbf{A}\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{A}\mathbf{X}^T\right]\mathbf{X}\mathbf{D}\mathbf{e} \tag{25}$$

where

$$\mathbf{A} = \left(\mathbf{I} - \frac{1}{N}\mathbf{e}\mathbf{e}^T\right) \tag{26}$$

If

$$\boldsymbol{\alpha}_o = C\mathbf{D}\left[\mathbf{I} - \left(\frac{\mathbf{I}}{C} + \mathbf{A}\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{A}\mathbf{X}^T\mathbf{X}\right]\mathbf{X}\mathbf{D}\mathbf{e} \tag{27}$$

we can rewrite Eqs. 25 and 24 as

$$\mathbf{w}_o = \mathbf{X}\mathbf{D}\boldsymbol{\alpha}_o \tag{28}$$

and

$$b_o = -\frac{\mathbf{e}^T\mathbf{X}^T\mathbf{w}_o}{N} = -\frac{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{D}\boldsymbol{\alpha}_o}{N} \tag{29}$$

Note that Eq. 28 has the same form of Eq. 9. Therefore $\boldsymbol{\alpha}_o$ can be understood as a Lagrange multiplier vector, although it was not calculated explicitly. Substituting $\mathbf{X}^T\mathbf{X}$ by $\mathbf{K}$ we obtain the non-linear UPSVM as

$$\boldsymbol{\alpha}_o = C\mathbf{D}\left[\mathbf{I} - \left(\frac{\mathbf{I}}{C} + \mathbf{A}\mathbf{K}\right)^{-1}\mathbf{A}\mathbf{K}\right]\mathbf{D}\mathbf{e} \tag{30}$$

$$b_o = -\frac{1}{N}\mathbf{e}^T\mathbf{K}\mathbf{D}\boldsymbol{\alpha}_o \tag{31}$$

Also note that Eq. 30 involves inversion of an $N \times N$ matrix. As was also done in [2], reduced kernel techniques can be used to reduce computational complexity.

After obtaining $\boldsymbol{\alpha}_o$ and b_o from the training procedure described by Eqs. 30 and 31, the classification task is accomplished by calculating the distance from the test vector $\mathbf{y}$ to the optimal hyperplane. This can be done, without knowing vector $\mathbf{w}_o$, by Eq. 7 using b_o calculated by Eq. 31. If dist($\mathbf{y}$) is positive then $\mathbf{y} \in C_1$, otherwise $\mathbf{y} \in C_2$.

4. EXPERIMENTAL RESULTS

UPSVM was compared with PSVM and SMO based on probability of correct classification on a test set, variance of the optimal hyperplane estimated parameters, and training time.

4.1. Example 1: Linearly Separable Classes

For class C_1, each observation lies inside a circle with center at origin and unitary radius. For class C_2, each observation lies inside a circle with center at $(2, 0)$ and radius equal to 0.5. In this experiment, 100 classifiers were generated for each one of the three methods being tested (SMO, PSVM, and UPSVM). For each comparison, we trained the three methods with the same set of 400 randomly generated observations (200 for C_1 and 200 for C_2). The test set consisted of 100 subsets of 1000 randomly generated observations (500 observations per class).

Table 2 shows the average probability of correct classification on the test data set and the training times normalized by the smallest value. Table 3 shows the variance of the estimated parameters $(\mathbf{w}_o, b_o)$. Comparison is carried out for different values of C.

Table 2. Average probability and normalized training time

C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10	100
SMO	N/A	N/A	100 24.38	100 17.95	100 24.82	100 16.64	100 18.16
PSVM	77.9 1.85	88.14 1.77	98.53 1.79	99.93 1.79	100 1.779	100 1.79	100 1.82
UPSVM	100 1.12	100 1.00	100 1.03	100 1.31	100 1.02	100 1.02	100 1.01

Table 3. Estimated parameter variance ($\times 10^4$)

C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10	100
SMO	N/A	N/A	2 9 8	39 60 175	268 381 672	675 1549 1326	675 1550 1370
PSVM	< 1 < 1 < 1	< 1 < 1 < 1	< 1 6 1	2 21 5	2 30 6	2 31 6	2 31 6
UPSVM	< 1 < 1 < 1	< 1 < 1 < 1	1 3 3	2 20 6	2 29 6	2 31 6	2 31 6

In table 2, probability of correct classification and training time were not computed for SMO for low values of C (10^{-4} and 10^{-3}) because it saturated, i.e., all Lagrange multipliers were equal to C. As can be seen in table 2, UPSVM was 13 to 24 times faster than SMO for the same value of C and still maintained the same performance. When comparing PSVM and UPSVM it is important to note that UPSVM was a little faster and that PSVM was not able to classify correctly two linearly separable classes for C less than 1. This was due to an unnecessary low value of $|b|$ that implied in a biased optimal hyperplane.

As can be seen in table 3 the variance of the estimated parameters grows with C. Large variances means that an optimal hyperplane obtained after only one training has a smaller probability of being close to the theoretical optimal hyperplane. This result suggests that it is interesting to work with low values of C whenever possible. UPSVM was the only one that achieved good performance in this case. Furthermore, the proximal hyperplanes were closer to their theoretical expected value than support hyperplanes obtained by SMO. This was true especially for large values of C, as suggested by the larger variances obtained by SMO when compared to the variances obtained by the other two methods.

4.2. Example 2: Non-Separable Classes

Each class consists of bidimensional Gaussian stochastic processes. Observations from class C_1 have mean $[0\ 0]^T$ and covariance matrix $0.5\mathbf{I}$. For the second class, observations have mean $[2\ 0]^T$ and covariance matrix $4\mathbf{I}$. We proceeded in the same way as was done for linearly separable classes. The probability of correct classification for test data sets and the training times were obtained and their mean values are shown on table 4. For each training an RBF kernel was used with σ^2 equal to 4. This experiment is equal to the one discussed in [1] for which the probability of correct classification is equal to 81.51.

Probability of correct classification and training time were not computed for SMO for low values of C (10^{-4} to 10^{-2}) because it saturated. In this experiment UPSVM was not always faster than SMO, but obtained similar performance and was approximately 4 and 38 times faster than SMO for C equal to 10 and 100 respectively. For other values of C both methods obtained almost the same training time. Another interesting result is that the training time increases with C when training a non-linear SMO classifier, whereas it was almost constant for the other two methods. It can also be seen that UPSVM was more than twice faster than PSVM in training. All three methods obtained classifiers with probability of correct classification very close to the optimal value except the PSVM for C equal to 10^{-4} and 10^{-3} due to biased optimal hyperplane in the feature space.

Table 4. Average probability and normalized training time

C	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10	100
SMO	N/A	N/A	N/A	81.29 1.00	81.10 1.40	80.65 5.21	80.58 48.67
PSVM	75.62 3.36	79.61 3.36	81.23 3.36	81.09 3.36	80.97 3.36	81.07 3.37	81.06 3.38
UPSVM	81.05 1.33	81.06 1.33	81.25 1.33	81.09 1.34	81.00 1.35	81.15 1.36	80.82 1.35

5. CONCLUSIONS

We introduced a new unbiased proximal SVM algorithm that showed better (in some cases much better) training times when compared to known PSVM and SMO classifiers, while maintained similar probability of correct pattern classification. In the simulations shown, the proposed algorithm was 13 to 24 times faster than SMO for a linear kernel and 4 to 38 times faster for a non-linear kernel and large values of the regularization parameter C. The UPSVM training algorithm was also faster than PSVM. When a non-linear kernel was used, both the variance of the estimated parameters and SMO training time, increased with C, which suggests that low values of C should be favored. However, for small values of C, the SMO algorithm had difficulties to find the support vectors, whereas the PSVM was biased. In these cases, the proposed UPSVM algorithm is presented as a good alternative.

6. REFERENCES

[1] S. Haykin, *Neural Networks - A Comprehensive Foundation*, Prentice Hall, 2nd. edition, 1999.

[2] O.L. Mangasarian and G. Fung, "Proximal support vector machine classifi ers," in *Proceedings International Conference on Knowledge, Discovery, and Data Mining*, 2001, pp. 64–70.

[3] J.C. Platt, N. Cristianini, and J.S. Taylor, "Large margin DAGS for multiclass classifi cation," in *Advances in Neural Information Processing Systems (NIPS)*, 2000, vol. 12, pp. 547–553.

[4] K.-L. Li, Z.-F. Tian, and H.-H. Huang, "A novel multiclass svm classifi er based on DDAG," in *Proceedings Of First International Conference On Machine Learning And Cybernetics*, 2002, vol. 3, pp. 1203–1207.

[5] O.L. Mangasarian and D.R. Musicant, "Active support vector machine classifi cation," in *Advances in Neural Information Processing Systems (NIPS)*, 2000, vol. 13, pp. 577–583.

[6] G. H. Gollub and C. F. Van Loan, *Matrix computations*, Johns Hopkins, 3rd edition, 1996.

[7] J.C. Platt, *Advances in Kernel Methods — Support Vector Learning*, vol. 12, chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pp. 41–64, 2000.