Neurocomputing 320 (2018): 47 – 59

DOI: https://doi.org/10.1016/j.neucom.2018.08.049

Modified Frank–Wolfe Algorithm for Enhanced Sparsity in Support Vector Machine Classifiers

Carlos M. Alaíz[1,*], Johan A.K. Suykens[2]

[1] *Dpto. Ing. Informática, Universidad Autónoma de Madrid, 28049 Madrid, Spain*
[2] *Dept. Electrical Engineering (ESAT–STADIUS), KU Leuven, B-3001 Leuven, Belgium*

Abstract

This work proposes a new algorithm for training a re-weighted ℓ_2 Support Vector Machine (SVM), inspired on the re-weighted Lasso algorithm of Candès *et al.* and on the equivalence between Lasso and SVM shown recently by Jaggi. In particular, the margin required for each training vector is set independently, defining a new weighted SVM model. These weights are selected to be binary, and they are automatically adapted during the training of the model, resulting in a variation of the Frank–Wolfe optimization algorithm with essentially the same computational complexity as the original algorithm.

As shown experimentally, this algorithm is computationally cheaper to apply since it requires less iterations to converge, and it produces models with a sparser representation in terms of support vectors and which are more stable with respect to the selection of the regularization hyper-parameter.

Keywords: Support Vector Machines, Sparsity, Frank–Wolfe, Lasso

1. Introduction

Regularization is an essential mechanism in Machine Learning that usually refers to the set of techniques that attempt to improve the estimates by biasing them away from their sample-based values towards values that are deemed to be

[*] Corresponding author.
Email addresses: carlos.alaiz@inv.uam.es (Carlos M. Alaíz), johan.suykens@esat.kuleuven.be (Johan A.K. Suykens)

more “physically plausible” [1]. In practice, it is often used to avoid over-fitting, apply some prior knowledge about the problem at hand or induce some desirable properties over the resulting learning machine. One of these properties is the so called sparsity, which can be roughly defined as expressing the learning machines using only a part of the training information. This has advantages in terms of the interpretability of the model and its manageability, and also preventing the over-fitting. Two representatives of this type of models are the Support Vector Machines (SVM; [2]) and the Lasso model [3], based on inducing sparsity at two different levels. On the one hand, the SVMs are sparse in their representation in terms of the training patterns, which means that the model is characterized only by a subsample of the original training dataset. On the other hand, the Lasso models induce sparsity at the level of the features, in the sense that the model is defined only as a function of a subset of the inputs, hence performing an implicit feature selection.

Recently, Jaggi [4] showed an equivalence between the optimization problems corresponding to a classification ℓ_2-SVM and a constrained regression Lasso. As explored in this work, this connection can be useful to transfer ideas from one field to the other. In particular, and looking for sparser SVMs, in this paper the reweighted Lasso approach of Candès *et al.* [5] is taken as the basis to define first a weighted ℓ_2-SVM, and then to propose a simple way of adjusting iteratively the weights that leads to a Modified Frank–Wolfe algorithm. This adaptation of the weights does not add an additional cost to the algorithm. Moreover, as shown experimentally the proposed approach needs less iterations to converge than the standard Frank–Wolfe, and the resulting SVMs are sparser and much more robust with respect to changes in the regularization hyper-parameter, while retaining a comparable accuracy.

In summary, the contributions of this paper can be stated as follows:

(i) The definition of a new weighted SVM model, inspired by the weighted Lasso and the connection between Lasso and SVM. This definition can be further extended to a re-weighted SVM, based on an iterative scheme to define the weights.

(ii) The proposal of a modification of the Frank–Wolfe algorithm based on the re-weighting scheme to train the SVM. This algorithm results in a sparser SVM model, which coincides with the model obtained using a standard SVM training algorithm over only an automatically-selected subsample of the original training data.

(iii) The numerical comparison of the proposed model with the standard SVM over a number of different datasets. These experiments show that the proposed algorithm requires less iterations while providing a sparser model which is also more stable against modifications of the regularization parameter.

The remaining of the paper is organized in the following way. Section 2 summarizes some results regarding the connection of SVM with Lasso. The weighted and re-weighted SVMs are introduced in Section 3, whereas the proposed modified Frank–Wolfe algorithm is presented in Section 4. The performance of this algorithm is tested through some numerical experiments in Section 5, and Section 6 ends the paper with some conclusions and pointers to further work.

Notation

N denotes the number of training patterns, and D the number of dimensions. The data matrix is denoted by $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^\intercal \in \mathbb{R}^{N \times D}$, where each row correspond to the transpose of a different pattern $\mathbf{x}_i \in \mathbb{R}^D$. The corresponding vector of targets is $\mathbf{y} \in \mathbb{R}^N$, where $y_i \in \{-1, +1\}$ denotes the label of the i-th pattern. The identity matrix of dimension N is denoted by $\mathbf{I}_N \in \mathbb{R}^{N \times N}$.

2. Preliminaries

This section covers some preliminary results concerning the Support Vector Machine (SVM) formulation, its connection with the Lasso model, and the re-weighted Lasso algorithm, which are included since they are the basis of the proposed algorithm.

2.1. SVM Formulation

The following ℓ_2-SVM classification model (this model is described for example in [6]), crucial in [4], will be used as the starting point of this work:

$$\min_{\mathbf{w},\rho,\xi} \left\{ \frac{1}{2} \left\| \mathbf{w} \right\|_2^2 - \rho + \frac{C}{2} \sum_{i=1}^{N} \xi_i^2 \right\} \text{ s.t. } \mathbf{w}^\intercal \mathbf{z}_i \geq \rho - \xi_i \,, \tag{1}$$

where $\mathbf{z}_i = y_i \mathbf{x}_i$. Straightforwardly, the corresponding Lagrangian dual problem can be expressed as:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \left\{ \boldsymbol{\alpha}^\intercal \hat{\mathbf{K}} \boldsymbol{\alpha} \right\} \text{ s.t. } 0 \leq \alpha_i \leq 1, \; \sum_{i=1}^{N} \alpha_i = 1 \,, \tag{2}$$

where $\hat{\mathbf{K}} = \mathbf{Z}\mathbf{Z}^\intercal + \frac{1}{C}\mathbf{I}_N$. A non-linear SVM can be considered simply by substituting $\mathbf{Z}\mathbf{Z}^\intercal$ by the (labelled) kernel matrix $\mathbf{K} \circ \mathbf{y}\mathbf{y}^\intercal$ (where $\circ$ denotes the Hadamard or component-wise product).

It should be noticed that the feasible region of Problem (2) is just the probability simplex, and the objective function is simply a quadratic term.

2.2. Connection between Lasso and SVM

There exists an equivalence between the SVM dual Problem (2) and the following problem, which corresponds to a constrained Lasso regression model:

$$\min_{\mathbf{w} \in \mathbb{R}^D} \left\{ \left\| \mathbf{X}\mathbf{w} - \mathbf{y} \right\|_2^2 \right\} \text{ s.t. } \left\| \mathbf{w} \right\|_1 \leq 1 \,, \tag{3}$$

where in this case the vector $\mathbf{y} \in \mathbb{R}^N$ does not need to be binary. In particular, a problem of the form of Problem (2) can be rewritten in the form of Problem (3) and vice-versa [4].

This relation is only at the level of the optimization problem, which means that an ℓ_2-SVM model can be trained using the same approach as for training the Lasso model and the other way around (as done in [7]), but it cannot be extended to a prediction phase, since the Lasso model is solving a regression problem, whereas the SVM solves a classification one. Moreover, the number of dimensions and the number of patterns flip when transforming one problem into the other. Nevertheless, and as illustrated in this paper, this connection can be valuable by itself to inspire new ideas.

2.3. Re-Weighted Lasso

The re-weighted Lasso (RW-Lasso) was proposed as an approach to approximate the ℓ_0 norm by using the ℓ_1 norm and a re-weighting of the coefficients [5]. In particular, this approach was initially designed to approximate the problem

$$\min_{\mathbf{w}\in\mathbb{R}^D} \; \{\|\mathbf{w}\|_0\} \; \text{ s.t. } \; \mathbf{y} = \mathbf{X}\mathbf{w} \, ,$$

by minimizing weighted problems of the form:

$$\min_{\mathbf{w}\in\mathbb{R}^D} \; \left\{ \sum_{i=1}^{D} t_i |w_i| \right\} \; \text{ s.t. } \; \mathbf{y} = \mathbf{X}\mathbf{w} \, , \tag{4}$$

for certain weights $t_i > 0$, $i = 1, \ldots, D$. An iterative approach was proposed, where the previous coefficients are used to define the weights at the current iterate:

$$t_i^{(k)} = \frac{1}{|w_i^{(k-1)}| + \epsilon} \, , \tag{5}$$

what results in the following problem at iteration k:

$$\min_{\mathbf{w}^{(k)}\in\mathbb{R}^D} \; \left\{ \sum_{i=1}^{D} \frac{1}{|w_i^{(k-1)}| + \epsilon} |w_i^{(k)}| \right\} \; \text{ s.t. } \; \mathbf{y} = \mathbf{X}\mathbf{w}^{(k)} \, .$$

The idea is that if a coefficient is small, then it could correspond to zero in the ground-truth model, and hence it should be pushed to zero. On the other side, if the coefficient is large, it most likely will be different from zero in the ground-truth model, and hence its penalization should be decreased in order not to bias its value.

This approach is based on a constrained formulation that does not allow for training errors, since the resulting model will always satisfy $\mathbf{y} = \mathbf{X}\mathbf{w}$. A possible implementation of the idea of Problem (4) without such a strong assumption is the following:

$$\min_{\mathbf{w}\in\mathbb{R}^D} \; \left\{ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \right\} \; \text{ s.t. } \; \sum_{i=1}^{D} t_i |w_i| \leq \rho \, , \tag{6}$$

where the errors are minimized and the weighted ℓ_1 regularizer is included as a constraint (equivalently, the regularizer could be also added to the objective function [8]). The iterative procedure to set the weights can be similar to the one explained above, for example defining the weights at iteration k as:

$$t_i^{(k)} = \frac{c^{(k)}}{|w_i^{(k-1)}| + \epsilon} \quad ; \quad c^{(k)} = \frac{\rho}{\sum_{j=1}^{D} \frac{|w_j^{(k-1)}|}{|w_j^{(k-1)}|+\epsilon}} \approx \frac{\rho}{D} \, ,$$

where the constant $c^{(k)}$ allows the existence of a fixed point, in the sense that if $\mathbf{w}^{(k)} = \mathbf{w}^{(k-1)}$ then the constraint of Problem (6) is trivially satisfied.

Although there have been different advances and alternative approaches in the definition of weighting schemes for sparse recovery (for instance [9]), the advantage of Problem (4) is that it is focused on the ℓ_1 norm, which is the one that appears in the connection between Lasso and SVM. Moreover, the particular form of the weighting scheme, be it (5) or a more sophisticated one, will not be used in the remaining of the paper.

2.4. Towards a Sparser SVM

One important remark regarding the RW-Lasso is that the re-weighting scheme breaks the equivalence with the SVM explained in Section 2.2, i.e., one cannot simply apply the RW-Lasso approach to solve the SVM problem in order to get more sparsity in the dual representation (i.e. fewer support vectors). Instead, an analogous scheme will be directly included in the SVM formulation in the section below.

More specifically, and as shown in Fig. 1, the connection between Lasso and SVM suggests to apply a weighting scheme also for SVM. In order to set the weights, an iterative procedure (analogous to the RW-Lasso) seems to be the natural step, although this would require to solve a complete SVM problem at each iteration. Finally, an online procedure to determine the weights, that are adapted directly at the optimization algorithm, will lead to a modification of the Frank–Wolfe algorithm.

It should be stated that a weighted SVM has been already proposed in [10], but that model differs from the approach described here. In particular, the weighing of [10] refers to the primal problem (through different regularization

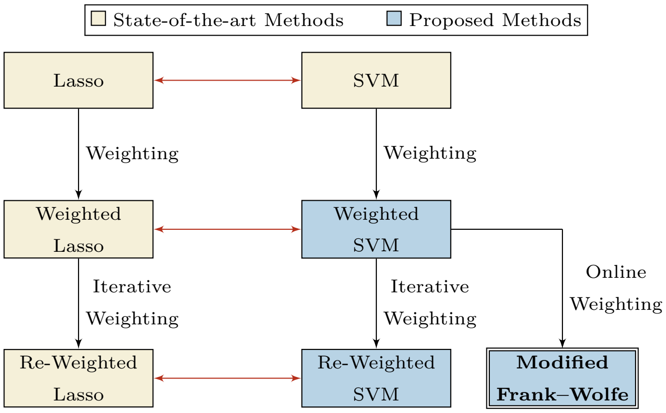


Figure 1: Scheme of the relation between the proposed methods and the inspiring Lasso variants.

parameters associated to each pattern) whereas in this work the weighting refers directly to the dual problem. As explained in Section 3.1, both models are not equivalent.

It is also import to notice that there are other SVM formulations that tend to produce a sparser representation in terms of support vectors, such as the Linear Programming Support Vector Machines (LPSVMs; [11]), or the 1-norm Support Vector Machines [12]. Nevertheless, the focus of this work is not to get the sparsest SVM possible, but to propose a small modification of the training algorithm that can lead to a sparser (and faster in convergence) model. On the other side, both the LPSVM and 1-norm SVM are designed as the solution of other optimization problems, whereas as shown in Section 4.2.3 the approach proposed here leads to a solution to Problem (1) but posed over a subsample of the original data. Finally, it would be interesting and a possible line of future work to check if the proposed modification of the Frank–Wolfe algorithm can also be translated to the training algorithms of sparser SVMs, such as LPSVMs and 1-norm SVMs, to get even sparser models.

3. Weighted and Re-Weighted SVMs

In this section the weighted SVM model is proposed. Furthermore, a re-weighting scheme to define iteratively the weights is sketched.

3.1. Weighted SVM

In order to transfer the weighting scheme of RW-Lasso to an SVM framework, the most natural idea is to directly change the constraint of Problem (2) to introduce the scaling factors t_i. This results in the following Weighted-SVM (W-SVM) dual optimization problem:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \left\{ \boldsymbol{\alpha}^\intercal \hat{\mathbf{K}} \boldsymbol{\alpha} \right\} \text{ s.t. } 0 \leq \alpha_i , \ \sum_{i=1}^N t_i \alpha_i = 1 \, , \tag{7}$$

for a fixed vector of weights $\mathbf{t}$. This modification relates with the primal problem as stated in the proposition below.

Proposition 1. *The W-SVM primal problem corresponding to Problem* (7) *is:*

$$\min_{\mathbf{w}, \rho, \xi} \left\{ \frac{1}{2} \left\| \mathbf{w} \right\|_2^2 - \rho + \frac{C}{2} \sum_{i=1}^N \xi_i^2 \right\} \ \textit{s.t.} \ \mathbf{w}^\intercal \mathbf{z}_i \geq t_i \rho - \xi_i \, . \tag{8}$$

Proof. The Lagrangian of Problem (8) is:

$$\mathcal{L}(\mathbf{w}, \rho, \xi; \boldsymbol{\alpha}) = \frac{1}{2} \left\| \mathbf{w} \right\|_2^2 - \rho + \frac{C}{2} \sum_{i=1}^N \xi_i^2 + \sum_{i=1}^N \alpha_i (-\mathbf{w}^\intercal \mathbf{z}_i + t_i \rho - \xi_i) \, ,$$

with derivatives with respect to the primal variables:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= \mathbf{w} - \mathbf{Z} \boldsymbol{\alpha} = 0 & \Longrightarrow \quad & \mathbf{w} = \mathbf{Z} \boldsymbol{\alpha} \, ; \\ \frac{\partial \mathcal{L}}{\partial \rho} &= -1 + \sum_{i=1}^N t_i \alpha_i = 0 & \Longrightarrow \quad & \sum_{i=1}^N t_i \alpha_i = 1 \, ; \\ \frac{\partial \mathcal{L}}{\partial \xi} &= C \xi - \boldsymbol{\alpha} = 0 & \Longrightarrow \quad & \xi = \frac{1}{C} \boldsymbol{\alpha} \, . \end{aligned}$$

Substituting into the Lagrangian, the following objective function for the dual problem arises:

$$\frac{1}{2} \left\| \mathbf{Z} \boldsymbol{\alpha} \right\|_2^2 - \rho + \frac{C}{2C^2} \left\| \boldsymbol{\alpha} \right\|_2^2 - \left\| \mathbf{Z} \boldsymbol{\alpha} \right\|_2^2 + \rho \sum_{i=1}^N t_i \alpha_i - \frac{1}{C} \left\| \boldsymbol{\alpha} \right\|_2^2 = -\frac{1}{2} \left\| \mathbf{Z} \boldsymbol{\alpha} \right\|_2^2 - \frac{1}{2C} \left\| \boldsymbol{\alpha} \right\|_2^2 \, .$$

Hence, the resulting dual problem is:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \left\{ \|\mathbf{Z}\boldsymbol{\alpha}\|_2^2 + \frac{1}{C}\|\boldsymbol{\alpha}\|_2^2 \right\} \text{ s.t. } 0 \le \alpha_i, \; \sum_{i=1}^{N} t_i\alpha_i = 1 \,,$$

which coincides with Problem (7). □

Therefore, the effect of increasing the scaling factor t_i in the W-SVM dual formulation is equivalent to increasing the margin required for the i-th pattern in the primal formulation. Thus, intuitively an increase of t_i should facilitate the i-th pattern to become a support vector. This influence is numerically illustrated in Fig. 2, where the value of one weight t_i is varied to analyse its influence over the corresponding multiplier α_i in a binary classification problem with $N = 100$ and $D = 2$. The other weights are just fixed equal to one, but before solving the problem all the vector $\mathbf{t}$ is normalized so that its maximum is still equal to one in order to preserve the scale. This experiment is done for three different values of C (10^{-3}, 1 and 10^3) and for the weights corresponding to the maximum, minimum and an intermediate value of the multiplier of the standard (unweighted) SVM. Clearly t_i and α_i present a proportional relationship, so the larger t_i is, the larger the obtained multiplier α_i becomes (until some point of saturation), confirming the initial intuition.

As another illustration, Fig. 3 shows a small toy example of three patterns, which allows to represent the feasible set in two dimensions as the convex hull of the three vertices. The value of one weight t_i is changed in the set $\{10^{-2}, 10^{-1}, 1, 10^1, 10^2\}$, whereas the other two weights are kept fixed to 1. As before, increasing the weight pushes the solution towards the corresponding pattern. Moreover, the last row in Fig. 3 shows the same example but with a three dimensional representation, so that it is more clear the effect of decreasing t_1 in the feasible set, basically lengthening the triangle and increasing its angle with respect to the horizontal plane, until the point where the triangle becomes an unbounded rectangle ($t_1 = 0$) completely vertical. Taking into consideration that the solution of the unconstrained problem (for $C \neq \infty$) is the origin, decreasing t_1 is moving away the first vertex from the unconstrained solution, thus

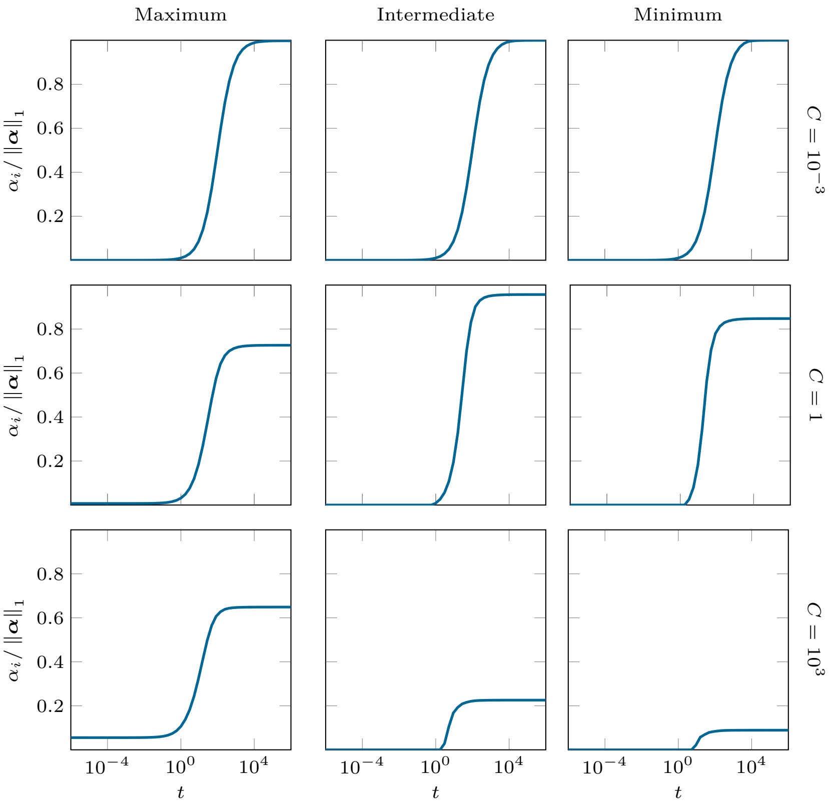


Figure 2: Evolution of the SVM coefficient α with respect to the weight t, for C equal to 10^{-3} (first row), 1 (second row) and 10^3 (third row), and for the patterns corresponding to the maximum (first column), an intermediate (second column) and the minimum (third column) initial value of α.

making less likely to assign a non-zero coefficient to that point unless it really decreases the objective function.

It is mandatory to state the differences between the W-SVM proposed here and the previous model proposed in [10]. First of all, the formulations over which both approach are based are different. But, even if the same starting SVM model were used, both weighting schemes are essentially different:

- Lapin *et al.* propose a modification of the primal SVM formulation so that the cost associated is different for each pattern. This means that the

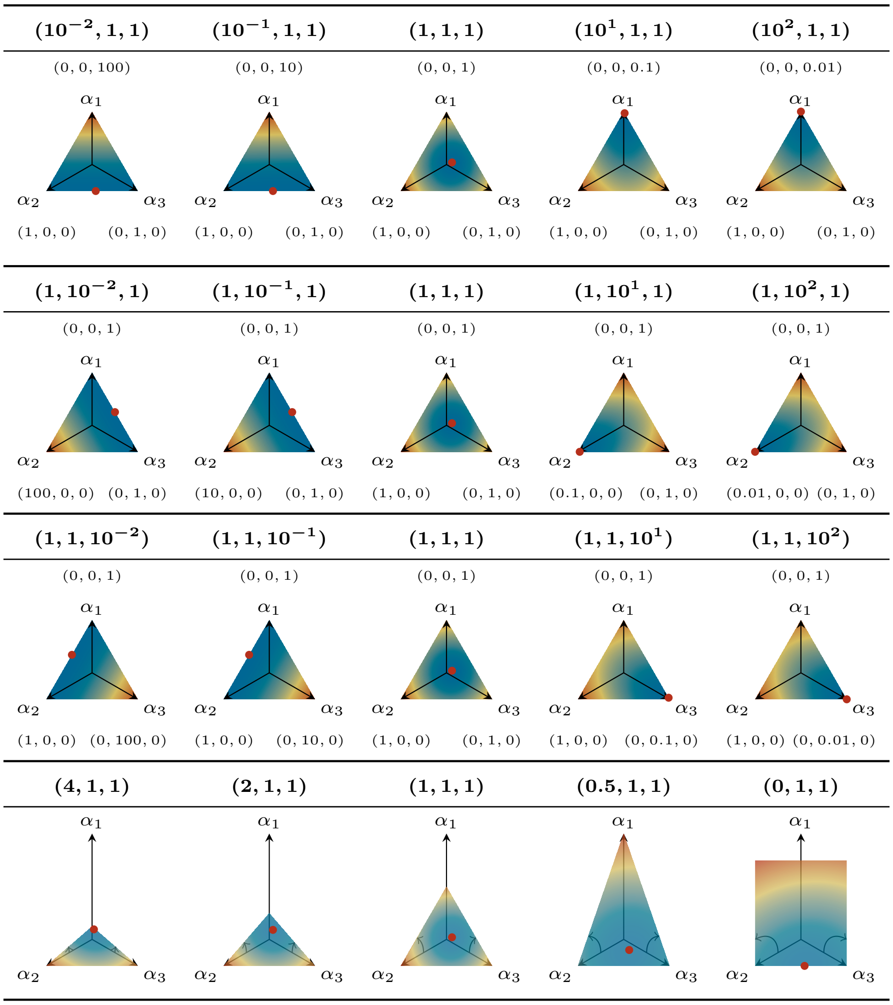


Figure 3: Example of the feasible region and the solution for a problem with three patterns, for different values of the weighting vector $\mathbf{t}$. For each plot, the value of $\mathbf{t}$ is shown above in boldface. The three rows correspond to changes in t_1, t_2 and t_3 respectively, and the weighted probability simplex is represented as the convex hull of the three vertices. The fourth row corresponds again to changes in t_1 but with a 3-dimensional representation keeping the same aspect ratio for all the axis, and also including the limit case $t_1 = 0$ where α_1 is not upper bounded. The solution of the constrained optimization problem is shown with a red dot [●].

loss associated to that pattern is multiplied by a constant.

- The W-SVM proposed here introduces the weights directly into the dual problem, what results into a modification of the margin required for each pattern in the primal problem. Considering again the loss associated to each pattern, the "insensitivity" zone (the region of predictions that are associated to a zero loss) is widened or narrowed according to a constant.

Hence, both approaches are fundamentally different, and the effects that they produce are not equivalent.

3.2. Re-Weighted SVM

Once Problem (7) has been defined, and provided that the scaling factors seem to influence the sparsity of the solution (as illustrated in Figs. 2 and 3), a procedure to set the weighting vector $\mathbf{t}$ is needed.

In parallelism with the original RW-Lasso, but considering that in this case the relation between the weight t_i and the corresponding optimal multiplier α_i is directly proportional, the following iterative approach, namely Re-Weighted SVM (RW-SVM), arises naturally:

1. At iteration k, the following W-SVM problem is solved:

$$\begin{aligned} \boldsymbol{\alpha}^{\star(k)} = \ & \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \left\{\boldsymbol{\alpha}^\intercal \hat{\mathbf{K}} \boldsymbol{\alpha}\right\} \\ & \text{s.t. } 0 \le \alpha_i ,\ \textstyle\sum_{i=1}^N t_i^{(k)} \alpha_i = 1\,. \end{aligned} \tag{9}$$

2. The weighting vector for the next iteration, $\mathbf{t}^{(k)}$, is updated as:

$$t_i^{(k+1)} = f_{\mathrm{mon}}\left(\alpha_i^{\star(k)}\right) ,$$

   where $f_{\mathrm{mon}} : \mathbb{R} \to \mathbb{R}$ is some monotone function.

This approach has two main drawbacks. The first one is how to select the function f_{mon}. This also implies selecting some minimum and maximum values to which the weights $t_i^{(k)}$ should saturate, so it is not a trivial task, and it can greatly influence the behaviour of the model. The second drawback is that this approach requires to solve Problem (9) at each iteration, which means training completely a W-SVM model (with a complexity that should not differ from that of training a standard SVM) on each iteration, and hence the overall computational cost can be much larger. Although this is in fact an affordable drawback if the objective is solely to approach the ℓ_0 norm as it was the case in the original paper of RW-Lasso [5], in the case of the SVM the aim is to get sparser models in order to reduce their complexity and to improve the performance specially in large datasets, and hence it does not make sense to need for this several iterations.

As a workaround, the next section proposes an online modification of the weights that leads to a simple modification of the training algorithm for SVMs.

4. Modified Frank–Wolfe Algorithm

This section proposes a training algorithm to get sparser SVMs, which is based on an online modification of the weighting vector $\mathbf{t}$ of a W-SVM model. In particular, the basis of this proposal is the Frank–Wolfe optimization algorithm.

4.1. Frank–Wolfe Algorithm

The Frank–Wolfe algorhtm (FW; [13]) is a first order optimization method for constrained convex optimization. There are several versions of this algorithm, in particular the basis of this work is the Pairwise Frank–Wolfe [14, 15]. Roughly speaking, it is based on using at each iteration a linear approximation of the objective function to select one of the vertices as the target towards which the current estimate of the solution will move (the *forward* node), and another vertex as that from which the solution will move away (the *away* node), and then updating the solution in the direction that goes from the *away* node to the *forward* one using the optimal step length. At the end, the linear approximation boils down to selecting the node corresponding to the smallest partial derivative as the *forward* node, and that with the largest derivative as the *away* node.

This general algorithm can be used in many different contexts, and in particular it has been successfully applied to the training of SVMs [16, 17, 18]. Specifically, for the case of Problem (2), the following definitions and results are employed.

Let f denote the (scaled) objective function of Problem (2) (or Problem (7)), with gradient and partial derivatives:

$$
\begin{aligned}
f(\boldsymbol{\alpha}) &= \frac{1}{2}\boldsymbol{\alpha}^{\intercal}\hat{\mathbf{K}}\boldsymbol{\alpha}\,; && (10)\\
\nabla f(\boldsymbol{\alpha}) &= \hat{\mathbf{K}}\boldsymbol{\alpha}\,; \\
\frac{\partial f}{\partial \alpha_i}(\boldsymbol{\alpha}) &= \hat{\mathbf{k}}_i^{\intercal}\boldsymbol{\alpha}\,, && (11)
\end{aligned}
$$

where $\hat{\mathbf{k}}_i^{\intercal}$ is the i-th row of $\hat{\mathbf{K}}$. Let $\mathbf{d}$ denote the direction in which the current solution will be updated. The optimal step-size can be computed by solving the problem:

$$\min_{\gamma}\ \{f(\boldsymbol{\alpha}+\gamma\mathbf{d})\}\,, \qquad (12)$$

and truncating the optimal step, if needed, in order to remain in the convex hull of the nodes, i.e., to satisfy the constraints of Problem (2) (or, equivalently, of Problem (7)). Straightforwardly, Problem (12) can be solved simply taking the derivative with respect to γ and making it equal to zero:

$$
\begin{aligned}
\frac{\partial f}{\partial \gamma}(\boldsymbol{\alpha}+\gamma\mathbf{d}) = \mathbf{d}^{\intercal}\hat{\mathbf{K}}(\boldsymbol{\alpha}+\gamma\mathbf{d}) = 0 \\
\implies \gamma = -\frac{\mathbf{d}^{\intercal}\hat{\mathbf{K}}\boldsymbol{\alpha}}{\mathbf{d}^{\intercal}\hat{\mathbf{K}}\mathbf{d}}\,.
\end{aligned}
$$

It should be noticed that $\hat{\mathbf{K}}\boldsymbol{\alpha}$ is the gradient of f at the point $\boldsymbol{\alpha}$, and thus there is no need to compute it again (indeed, the gradient times the direction is minus the FW gap, that can be used as a convergence indicator). Moreover, if the direction $\mathbf{d} = \sum_{i\in\mathcal{U}} d_i\mathbf{e}_i$ is sparse, then $\hat{\mathbf{K}}\mathbf{d} = \sum_{i\in\mathcal{U}} d_i\hat{\mathbf{k}}_i$ only requires to compute the columns of the kernel matrix corresponding to the set of updated variables $\mathcal{U}$. In particular, in the Pairwise FW only the columns of the *forward* and *away* nodes are used to determine γ and to keep the gradient updated.

The whole procedure for applying FW to the SVM training is summarized in Alg. 1.

4.2. Modified Frank–Wolfe Algorithm

The idea of the proposed Modified Frank–Wolfe (M-FW) is to modify the weights t_i, i.e., the margin required for each training pattern, directly on each

Algorithm 1 Pairwise Frank–Wolfe algorithm for SVM.

```
procedure TrainSVM($\hat{\mathbf{K}}, \epsilon$)          ▶ • Kernel $\hat{\mathbf{K}} \in \mathbb{R}^{N \times N}$.
                                                  • Precision $\epsilon \in \mathbb{R}$.
  Initialization.
    set $i_0 \in \{1, \dots, N\}$                        ▶ Initial vertex.
    $\boldsymbol{\alpha} \leftarrow \mathbf{e}_{i_0}$                                 ▶ Initial point.
    $\mathbf{g} \leftarrow \hat{k}_{i_0}$                                 ▶ Initial gradient.
    repeat                                        ▶ Main Loop.
  Update of Coefficients.
        $s \leftarrow \arg\min_i g_i$                      ▶ Select forward node.
        $v \leftarrow \arg\max_i g_i$                      ▶ Select away node.
        $\mathbf{d} \leftarrow \mathbf{e}_s - \mathbf{e}_v$                           ▶ Build update direction.
        $\delta \leftarrow -\mathbf{g} \cdot \mathbf{d}$                           ▶ FW gap.
        $\gamma \leftarrow \min\{\max\{\delta/(\mathbf{d}^\intercal \hat{\mathbf{K}} \mathbf{d}), 0\}, \alpha_v\}$    ▶ Compute step length.
        $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \gamma \mathbf{d}$                         ▶ Point update.
        $\mathbf{g} \leftarrow \mathbf{g} + \gamma \hat{\mathbf{k}}_s - \gamma \hat{\mathbf{k}}_v$                 ▶ Gradient update.
    until $\delta \leq \epsilon$                              ▶ Stopping criterion.
end procedure
```

inner iteration of the algorithm, hence with an overall cost similar to that of the original FW. In particular, and since according to Figs. 2 and 3 the relation between each weight and the resulting coefficient seems to be directly proportional, an incremental procedure with binary weights is defined, leading to a new training algorithm for SVM. Specifically, the training vectors will be divided into two groups, the working vectors, with a weight $t_i = 1$, and the idle vectors, with a weight $t_i = 0$. The proposed M-FW will start with only one initial working vector, and at each iteration, the idle vector with the smaller negative gradient (if there is any) will be added to the working set. After that, the coefficients of the working vectors will be updated by using a standard FW pair-wise step.

The intuition behind this algorithm is the following. The standard FW algorithm applied to the SVM training will activate (make non-zero) the coefficient of a certain vector if its partial derivative is better (smaller) than that of the already active coefficients, i.e., if that vector is “less bad” than the others. On the other side, the M-FW will only add a coefficient to the working set if its partial derivative is negative (hence, that coefficient would also be activated without the simplex constraint), i.e., the vector has to be somehow "good" by itself.

In what follows, the M-FW algorithm is described in more detail.

4.2.1. Preliminaries

The set of working vectors is denoted by $\mathcal{W} = \{i \mid 1 \leq i \leq N, t_i = 1\}$, and that of idle vectors as $\bar{\mathcal{W}} = \{i \mid 1 \leq i \leq N, t_i = 0\}$. The dual problem becomes:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \left\{ \boldsymbol{\alpha}^\intercal \hat{\mathbf{K}} \boldsymbol{\alpha} \right\} \text{ s.t. } 0 \leq \alpha_i, \ \sum_{i \in \mathcal{W}} \alpha_i = 1 \, .$$

Thus, the coefficients for the points in $\mathcal{W}$ have to belong to the probability simplex of dimension $|\mathcal{W}|$, whereas the coefficients for $\bar{\mathcal{W}}$ only have a non-negative constraint.

4.2.2. Algorithm

The proposed M-FW algorithm to train an SVM is summarized in Alg. 2. This algorithm is very similar to Alg. 1, except for the initialization and control of the working set in Lines 3 and 7 to 14, the search for the *forward* and *away* nodes of Lines 15 and 16 (which is done only over the working set) and the stopping criterion of Line 22 (which requires both that the dual gap is small enough and that no new vertices have been activated).

4.2.3. Convergence

Regarding the convergence of the M-FW algorithm, the following theorem states that this algorithm will provide a model that is equivalent to a standard SVM model trained only over a subsample[1] of the training patterns.

Theorem 1. *Algorithm 2 converges to a certain vector $\boldsymbol{\alpha}^\star \in \mathbb{R}^N$. In particular:*

[1] Due to its sparse nature, an SVM is expressed only in terms of the support vectors. Nevertheless, the proposed M-FW provides an SVM trained over a subsample of the training set, although not all the vectors of this subsample have to become support vectors.

Algorithm 2 Modified Frank–Wolfe algorithm for SVM.

```
procedure $\textsc{Train}\text{SVM}^{\text{M-FW}}(\hat{\mathbf{K}}, \epsilon)$    ▶ • Kernel $\hat{\mathbf{K}} \in \mathbb{R}^{N \times N}$.
                                                  • Precision $\epsilon \in \mathbb{R}$.
Initialization.
    set $i_0 \in \{1, \dots, N\}$    ▶ Initial vertex.
    $\mathcal{W} \leftarrow \{i_0\}$    ▶ Initial working set.
    $\boldsymbol{\alpha} \leftarrow \mathbf{e}_{i_0}$    ▶ Initial point.
    $\mathbf{g} \leftarrow \hat{k}_{i_0}$    ▶ Initial gradient.
    repeat    ▶ Main Loop.
Activation of Coefficients.
        if $|\mathcal{W}| < N$ then
            $b_{\text{chng}} \leftarrow$ false    ▶ Flag for changes.
            $u \leftarrow \arg\min_{i \in \bar{\mathcal{W}}} g_i$    ▶ Select node.
            if $g_u < 0$ then
                $\mathcal{W} \leftarrow \mathcal{W} \cup \{u\}$    ▶ Activate node.
                $b_{\text{chng}} \leftarrow$ true    ▶ Mark change.
            end if
        end if
Update of Working Coefficients.
        $s \leftarrow \arg\min_{i \in \mathcal{W}} g_i$    ▶ Select forward node.
        $v \leftarrow \arg\max_{i \in \mathcal{W}} g_i$    ▶ Select away node.
        $\mathbf{d} \leftarrow \mathbf{e}_s - \mathbf{e}_v$    ▶ Build update direction.
        $\delta \leftarrow -\mathbf{g} \cdot \mathbf{d}$    ▶ FW gap.
        $\gamma \leftarrow \min\{\max\{\delta / (\mathbf{d}^\intercal \hat{\mathbf{K}} \mathbf{d}), 0\}, \alpha_v\}$    ▶ Compute step length.
        $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \gamma \mathbf{d}$    ▶ Point update.
        $\mathbf{g} \leftarrow \mathbf{g} + \gamma \hat{\mathbf{k}}_s - \gamma \hat{\mathbf{k}}_v$    ▶ Gradient update.
    until $\delta \leq \epsilon$ and not $b_{\text{chng}}$    ▶ Stopping criterion.
end procedure
```

(i) The working set converges to a set $\mathcal{W}^\star$.

(ii) The components of $\boldsymbol{\alpha}^\star$ corresponding to $\mathcal{W}^\star$ conform the solution of the standard SVM Problem (2) *posed over the subset $\mathcal{W}^\star$ of the set of training patterns. The remaining components $\alpha_i^\star$, for $i \notin \mathcal{W}^\star$, are equal to zero.*

Proof.

(i) Let $\mathcal{W}^{(k)}$ denote the working set at iteration k. At iteration $k+1$, the set $\mathcal{W}^{(k+1)}$ will be either equal to $\mathcal{W}^{(k)}$ or equal to $\mathcal{W}^{(k)} \cup \{u\}$ for some

$u \notin \mathcal{W}^{(k)}$. Hence, $\mathcal{W}^{(k)} \subseteq \mathcal{W}^{(k+1)}$ for all k. Moreover, $\mathcal{W}^{(k)}$ is always a subset of the whole set of training vectors $\mathcal{T} = \{1, \cdots, N\}$, i.e. $\mathcal{W}^{(k)} \subseteq \mathcal{T}$ for all k. Since $\{\mathcal{W}^{(k)}\}$ is a monotone nondecreasing sequence of subsets of a finite set $\mathcal{T}$, then $\mathcal{W}^{(k)} \uparrow \mathcal{W}^\star \subseteq \mathcal{T}$, as proved next. Let $\{k_1, \ldots, k_{N'}\}$ be those iterations in which the working set grows, $\mathcal{W}^{(k_i)} \subset \mathcal{W}^{(k_i+1)}$. Obviously, the number of such N' iterations is finite with $N' \leq N$ since no more than N elements can be added to the working set. Therefore, $\forall k \geq k_{N'}$, $\mathcal{W}^{(k)} = \mathcal{W}^{(k'_N)} = \mathcal{W}^\star \subseteq \mathcal{T}$.

(ii) Provided that $\mathcal{W}^{(k)} \subseteq \mathcal{W}^\star$ for all k, then for $i \notin \mathcal{W}^\star$ the corresponding coefficients will never be updated (they cannot be selected in Lines 15 and 16 of Alg. 2), so they would conserve their initial value, i.e. $\alpha_i^{(k)} = 0$ for all $i \notin \mathcal{W}^\star$ and for all k.

With respect to the convergence of $\alpha_i^{(k)}$ for $i \in \mathcal{W}^\star$, it suffices to consider the iterations after the convergence of the working set, $k \geq k_{N'}$. Let $\boldsymbol{\alpha}_{\mathcal{W}^\star}^{(k)} \in \mathbb{R}^{N'}$ be the vector composed by the coefficients of the working patterns. Using (10) and since the coefficients of idle vectors are equal to zero (proved above):

$$f\left(\boldsymbol{\alpha}^{(k)}\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} \hat{k}_{ij}\alpha_i^{(k)}\alpha_j^{(k)} = \sum_{i\in\mathcal{W}^\star}\sum_{j\in\mathcal{W}^\star} \hat{k}_{ij}\alpha_i^{(k)}\alpha_j^{(k)} = f_{\mathcal{W}^\star}\left(\boldsymbol{\alpha}_{\mathcal{W}^\star}^{(k)}\right),$$

where $f_{\mathcal{W}^\star}$ denotes the objective function of Problem (2) posed only over the subset $\mathcal{W}^\star$ of the original training set. A similar result can be obtained for the components of the gradient using (11):

$$\frac{\partial f}{\partial \alpha_i}\left(\boldsymbol{\alpha}^{(k)}\right) = \sum_{j=1}^{N} \hat{k}_{ij}\alpha_j^{(k)} = \sum_{j\in\mathcal{W}^\star}^{N} \hat{k}_{ij}\alpha_j^{(k)} = \frac{\partial f_{\mathcal{W}^\star}}{\partial \alpha_i}\left(\boldsymbol{\alpha}_{\mathcal{W}^\star}^{(k)}\right).$$

Therefore, once the working set has converged both the objective function and the partial derivatives of the working set computed in Alg. 2 are equal to those computed in Alg. 1 when this algorithm is applied only over the vectors of the working set. Hence, in the remaining iterations M-FW reduces to the standard FW algorithm but considering only the vertices

Dataset	Tr. Size	Te. Size	Dim.	Maj. Class (%)
`ijcnn`	49 990	91 701	22	90.4
`mgamma`	13 020	6 000	10	64.8
`australian`	621	69	14	55.5
`breast`	615	68	10	65.0
`diabetes`	692	76	8	65.1
`german`	900	100	24	70.0
`heart`	243	27	13	55.6
`ionosphere`	316	35	34	64.1
`iris`	135	15	4	66.7
`mushrooms`	7 312	812	112	51.8
`sonar`	188	20	60	53.4
`miniboone`	100 000	29 596	50	71.8

Table 1: Description of the datasets.

in $\mathcal{W}^\star$, which converges to the solution of Problem (2) over the subset $\mathcal{W}^\star$ [15].

□

It is worth mentioning that, although the proposed M-FW algorithm converges to an SVM model trained over a subsample $\mathcal{W}^\star$ of the training data, this subsample will (as shown in Section 5) depend on the initial point of the algorithm.

5. Experiments

In this section the proposed M-FW algorithm will be compared with the standard FW algorithm over several classification tasks. In particular, the binary datasets that will be used for the experiments are described in Table 1, which includes the size of the training and test sets, the number of dimensions and the percentage of the majority class (as a baseline accuracy). All of them belong to the LibSVM repository [19] except for `mgamma` and `miniboone`, which belong to the UCI repository [20].

5.1. Preliminary Experiments

The first experiments will be focused on the first two datasets of Table 1, namely `ijcnn` and `mgamma`, which are the largest ones except for `miniboone`.

5.1.1. Set-Up

The standard SVM model trained using FW (SVM) and the model resulting from the proposed M-FW algorithm (denoted by $\mathsf{SVM}^{\mathsf{M\text{-}FW}}$, which as shown in Theorem 1 is just an SVM trained over a subsample $\mathcal{W}^\star$ of the original training set) will be compared in terms of their accuracies, the number of support vectors and the number of iterations needed to achieve the convergence during the training algorithm. Two different kernels will be used, the linear and the RBF (or Gaussian) ones. With respect to the hyper-parameters of the models, the value of both C and the bandwidth σ (in the case of the RBF kernel) will be obtained through 10-fold Cross Validation (CV) for `mgamma`, whereas for the largest dataset `ijcnn` only C will be tuned, and σ will be fixed as $\sigma = 1$ in the RBF kernel (this value is similar to the one used for the winner of the IJCNN competition [21]). Once the hyper-parameters are tuned, both models will be used to predict over the test sets. The stopping criterion used is $\epsilon = 10^{-5}$.

5.1.2. Results

The test results are summarized in Table 2. Looking first at the accuracies, both models SVM and $\mathsf{SVM}^{\mathsf{M\text{-}FW}}$ are practically equivalent in three of the four experiments, where the differences are insignificant, whereas for `ijcnn` with the linear kernel the accuracy is higher in the case of $\mathsf{SVM}^{\mathsf{M\text{-}FW}}$. Regarding the number of support vectors, $\mathsf{SVM}^{\mathsf{M\text{-}FW}}$ gets sparser models for `ijcnn` with linear kernel and `mgamma` with RBF kernel, whereas for the other two experiments both models get a comparable sparsity. Finally, and with respect to the convergence of the training algorithms, $\mathsf{SVM}^{\mathsf{M\text{-}FW}}$ shows an advantage when dealing with linear kernels, whereas for the RBF ones both approaches are practically equivalent.

It should be noticed that, for these larger datasets, only one execution is done per dataset and kernel, and hence it is difficult to get solid conclusions.

Data	**K.**	**Accuracy (%)**		**Number SVs**		**Number Iters.**	
		SVM	**SVM$^{\text{M-FW}}$**	**SVM**	**SVM$^{\text{M-FW}}$**	**SVM**	**SVM$^{\text{M-FW}}$**
ijcnn	lin	92.17	93.20	2.01E+4	8.00E+3	5.39E+4	1.75E+4
	rbf	98.83	98.81	4.99E+3	4.98E+3	3.31E+4	3.38E+4
mgamma	lin	78.22	78.26	1.19E+4	1.04E+4	1.68E+5	7.03E+4
	rbf	87.94	87.98	8.25E+3	7.46E+3	3.10E+4	3.06E+4

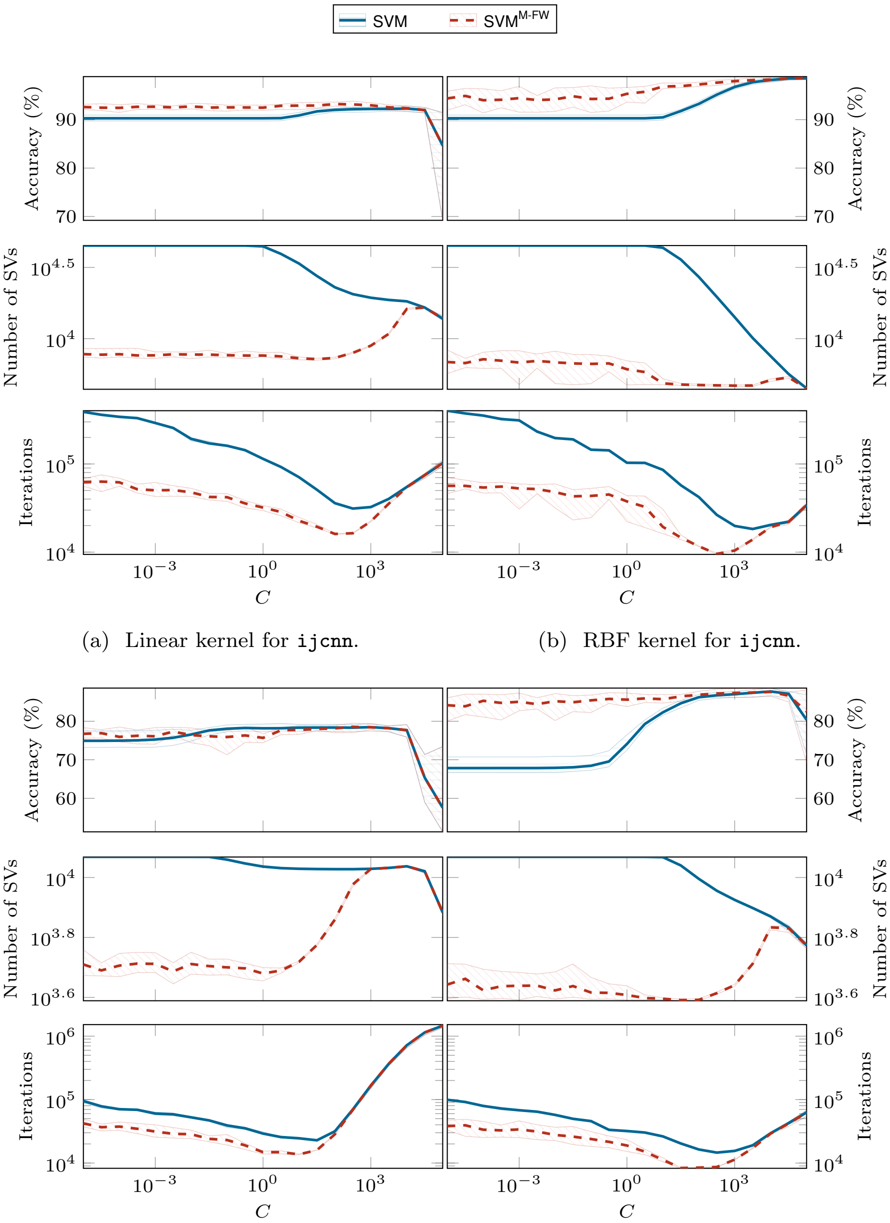
Table 2: Test results for the larger datasets.

Hence, it can be interesting to analyse the performance of the models during the CV phase, as done below.

5.1.3. Robustness w.r.t. Hyper-Parameter C

The evolution with respect to the parameter C of the accuracy, the number of support vectors and the number of training iterations is shown in Fig. 4 for both SVM and the proposed SVM$^{\text{M-FW}}$. For the RBF kernel, the curves correspond to the optimum value of σ for SVM. Observing the plots of the accuracy, SVM$^{\text{M-FW}}$ turns out to be much more stable than SVM, getting an accuracy almost optimal and larger than that of SVM in a wide range of values of C. Moreover, this accuracy is achieved with a smaller number of support vectors and with less training iterations. At some point, when the value of C is large enough, both SVM and SVM$^{\text{M-FW}}$ perform the same since all the support vectors of SVM also become working vectors during the training of SVM$^{\text{M-FW}}$, and both algorithms FW and M-FW provide the same model.

A possible explanation for the stability of the accuracy of SVM$^{\text{M-FW}}$ is its inherent regularization, in the sense that requiring a negative gradient entry in order to add a working vector prevents the model from using all the vectors as support vectors, even if the regularization parameter C is small. Furthermore, the stability of SVM$^{\text{M-FW}}$ suggests to fix C beforehand in order to get rid of a tuning parameter. This option will be explored in the next bunch of experiments.



(a) Linear kernel for `ijcnn`.

(b) RBF kernel for `ijcnn`.

(c) Linear kernel for `mgamma`.

(d) RBF kernel for `mgamma`.

Figure 4: Evolution of the validation results for `ijcnn` and `mgamma`, using both the linear and the RBF kernel for the optimum σ of SVM, both for the standard SVM and the proposed SVM$^{\text{M-FW}}$. The striped regions represent the range between minimum and maximum for the 10 partitions, whereas the lines in the middle represent the average values.

5.2. Exhaustive Experiments

In the following experiments, the smaller 9 datasets of the second block of Table 1 will be used to compare exhaustively three models: SVM, the proposed $\mathsf{SVM}^{\mathsf{M\text{-}FW}}$, and an alternative $\mathsf{SVM}^{\mathsf{M\text{-}FW}}$ model with a fixed regularization parameter (denoted as $\mathsf{SVM}^{\mathsf{M\text{-}FW}}_{\mathsf{FP}}$), in particular $C = 1$ (normalized).

5.2.1. Set-Up

As in the previous experiments, the hyper-parameters will be obtained using 10-fold CV (except for $\mathsf{SVM}^{\mathsf{M\text{-}FW}}_{\mathsf{FP}}$, where C is fixed and only σ will be tuned for the RBF kernel). The stopping criterion is again $\epsilon = 10^{-5}$. Once trained, the models will be compared over the test set.

Furthermore, in order to study the significance of the differences between the models, the whole procedure, including the CV and the test phase, will be repeated 10 times for different training/test partitions of the data (with a proportion 90 %/10 %).

5.2.2. Results

The results are detailed in Table 3, which includes for each of the three models the mean and standard deviation of the accuracy, the number of support vectors and the number of training iterations over the 10 partitions. The colours represent the rank of the models for each dataset and kernel, where the same rank is used if there is no significant difference between the models[2].

The results are averaged as a summary in Table 4, where they are included as a percentage with respect to the reference SVM. This table shows that $\mathsf{SVM}^{\mathsf{M\text{-}FW}}$ allows to reduce the number of support vectors, and of training iterations, to a 30.1 % and a 26.5 %, whereas the accuracy only drops to a 99.8 %. Moreover, using the $\mathsf{SVM}^{\mathsf{M\text{-}FW}}_{\mathsf{FP}}$ approach allows to avoid tuning C, while reducing the support vectors and iterations to a 26.0 % and a 8.0 %, with a drop of the accuracy to only the 99.7 % of the SVM accuracy.

[2] Using a Wilcoxon signed rank test for zero median, with a significance level of 5%.

Data	K.	SVM	$\text{SVM}^{\text{M-FW}}$	$\text{SVM}^{\text{M-FW}}_{\text{FP}}$
			Accuracy (%)	
australia	lin	85.65 ± 4.4	86.09 ± 4.2	85.65 ± 4.1
	rbf	85.94 ± 4.5	85.36 ± 4.1	85.22 ± 4.1
breast	lin	96.92 ± 1.8	96.49 ± 1.7	96.49 ± 2.1
	rbf	96.63 ± 2.1	96.49 ± 1.9	96.78 ± 1.7
diabetes	lin	77.35 ± 3.9	78.52 ± 3.0	76.57 ± 4.4
	rbf	77.48 ± 3.2	77.09 ± 3.3	75.92 ± 4.6
german	lin	76.70 ± 3.3	76.60 ± 4.1	76.70 ± 4.8
	rbf	76.70 ± 5.2	76.30 ± 4.2	76.00 ± 4.9
heart	lin	82.59 ± 6.5	83.33 ± 7.3	84.44 ± 7.2
	rbf	83.33 ± 8.2	83.33 ± 6.6	84.44 ± 7.6
ionospher	lin	82.65 ± 6.9	82.65 ± 6.9	81.79 ± 7.7
	rbf	92.61 ± 6.4	91.19 ± 6.3	92.03 ± 5.7
iris	lin	100.00 ± 0.0	100.00 ± 0.0	100.00 ± 0.0
	rbf	100.00 ± 0.0	99.33 ± 2.1	99.33 ± 2.1
mushrooms	lin	100.00 ± 0.0	100.00 ± 0.0	100.00 ± 0.0
	rbf	100.00 ± 0.0	100.00 ± 0.0	100.00 ± 0.0
sonar	lin	72.60 ± 7.3	70.21 ± 8.4	71.67 ± 10.5
	rbf	87.00 ± 4.6	88.50 ± 6.4	87.00 ± 5.1
			Number SVs	
australia	lin	5.94E+2 ± 5.8E+1	4.40E+2 ± 8.6E+1	2.01E+2 ± 6.6
	rbf	5.14E+2 ± 7.8E+1	4.45E+2 ± 9.0E+1	2.14E+2 ± 5.1E+1
breast	lin	1.35E+2 ± 1.5E+1	7.23E+1 ± 1.9E+1	5.52E+1 ± 2.4
	rbf	2.93E+2 ± 1.1E+2	5.56E+1 ± 1.7E+1	5.62E+1 ± 9.4
diabetes	lin	6.37E+2 ± 2.8E+1	5.01E+2 ± 1.1E+2	3.61E+2 ± 8.1
	rbf	6.06E+2 ± 2.1E+1	4.94E+2 ± 1.2E+2	3.73E+2 ± 2.1E+1
german	lin	8.04E+2 ± 1.5E+1	6.18E+2 ± 1.2E+2	5.21E+2 ± 9.5
	rbf	7.88E+2 ± 3.8E+1	7.01E+2 ± 1.1E+2	4.82E+2 ± 2.4E+1
heart	lin	2.18E+2 ± 2.8E+1	1.01E+2 ± 4.1	1.02E+2 ± 4.8
	rbf	1.92E+2 ± 3.1E+1	1.27E+2 ± 1.6E+1	1.28E+2 ± 2.4E+1
ionospher	lin	2.10E+2 ± 1.9E+1	2.02E+2 ± 2.7E+1	1.40E+2 ± 7.4
	rbf	1.72E+2 ± 3.4E+1	8.19E+1 ± 2.3E+1	7.20E+1 ± 7.4
iris	lin	9.91E+1 ± 1.8E+1	2.60 ± 1.9	2.60 ± 1.9
	rbf	1.35E+2 ± 0.0	2.70 ± 4.8E−1	2.70 ± 4.8E−1
mushrooms	lin	8.45E+2 ± 1.5E+2	1.12E+2 ± 2.1E+1	1.40E+2 ± 8.7
	rbf	7.31E+3 ± 5.2E−1	2.74E+1 ± 2.0	2.72E+1 ± 1.8
sonar	lin	1.04E+2 ± 3.6E+1	1.12E+2 ± 2.5E+1	1.36E+2 ± 3.7
	rbf	1.44E+2 ± 2.9E+1	6.50E+1 ± 1.0E+1	7.10E+1 ± 8.6
			Number Iters.	
australia	lin	2.65E+4 ± 5.6E+4	9.98E+4 ± 9.9E+4	6.41E+2 ± 1.6E+1
	rbf	1.26E+4 ± 1.2E+4	1.74E+4 ± 1.4E+4	7.40E+2 ± 1.2E+2
breast	lin	2.06E+4 ± 6.1E+4	8.93E+2 ± 7.6E+2	1.90E+2 ± 1.0E+1
	rbf	3.09E+3 ± 7.5E+3	2.56E+3 ± 7.3E+3	2.29E+2 ± 3.8E+1
diabetes	lin	1.42E+4 ± 3.1E+4	9.35E+3 ± 1.6E+4	1.16E+3 ± 5.9E+1
	rbf	1.11E+4 ± 9.6E+3	9.33E+3 ± 1.0E+4	1.39E+3 ± 1.8E+2
german	lin	7.51E+4 ± 1.6E+5	6.51E+4 ± 1.6E+5	2.02E+3 ± 3.2E+1
	rbf	7.90E+3 ± 8.4E+3	7.82E+3 ± 5.2E+3	1.38E+3 ± 7.5E+1
heart	lin	3.14E+4 ± 9.6E+4	5.81E+2 ± 1.5E+2	3.80E+2 ± 1.9E+1
	rbf	1.66E+4 ± 1.6E+4	5.26E+2 ± 1.7E+2	3.90E+2 ± 7.7E+1
ionospher	lin	9.98E+4 ± 1.0E+5	9.36E+4 ± 1.1E+5	8.91E+2 ± 4.8E+1
	rbf	1.24E+3 ± 6.2E+2	8.07E+2 ± 8.0E+2	2.16E+2 ± 2.8E+1
iris	lin	2.63E+2 ± 6.3E+1	5.40 ± 1.1E+1	5.40 ± 1.1E+1
	rbf	1.22E+3 ± 0.0	2.86E+1 ± 1.8E+1	1.67E+1 ± 1.0E+1
mushrooms	lin	7.49E+3 ± 1.6E+3	6.98E+2 ± 1.3E+2	5.21E+2 ± 2.6E+1
	rbf	5.61E+4 ± 3.1E+3	2.85E+2 ± 2.7E+1	1.61E+2 ± 1.4E+1
sonar	lin	4.36E+4 ± 4.2E+4	2.22E+4 ± 3.2E+4	2.66E+3 ± 1.1E+2
	rbf	9.67E+2 ± 6.0E+2	2.65E+2 ± 1.7E+2	1.96E+2 ± 2.3E+1

Table 3: Test results for the exhaustive experiments (10 repetitions). The colour indicates the rank (the darker, the better).

	SVM	$\text{SVM}^{\text{M-FW}}$	$\text{SVM}^{\text{M-FW}}_{\text{FP}}$
Accuracy	100.00	99.80	99.67
Number SVs	100.00	30.12	25.95
Number Iters.	100.00	26.45	7.98

Table 4: Geometric mean of the test results as a percentage with respect to SVM for the exhaustive experiments.

5.3. Evolution over a Large Dataset

This section shows the evolution of the training algorithms over a larger dataset, namely the `miniboone` shown in Table 1, for the three approaches SVM, $\text{SVM}^{\text{M-FW}}$, and $\text{SVM}^{\text{M-FW}}_{\text{FP}}$.

5.3.1. Set-Up

In this experiment the only kernel used is the RBF one. In order to set the hyper-parameters C and σ, 10-fold CV is applied over a small subsample of 5 000 patterns. Although this approach can seem quite simplistic, it provides good enough parameters for the convergence comparison, which is the goal of this experiment. In the case of $\text{SVM}^{\text{M-FW}}_{\text{FP}}$, C is fixed as $C = 1$, and the optimal σ of SVM is directly used instead of tuning it, so that no validation is done for this model.

Once C and σ are selected, the models are trained over the whole training set during 40 000 iterations. During this process, intermediate models are extracted every 5 000 iterations, simulating different selections of the stopping criterion ϵ. These intermediate models (trained using 5 000, 10 000, 15 000... iterations) are used to predict over the test set, and thus they allow to analyse the evolution of the test accuracy as a function of the number of training iterations.

5.3.2. Results

The results are shown in Fig. 5, which includes the evolution of the number of support vectors and the test accuracy.

It can be observed that the standard SVM starts with the higher accuracy, but it is rapidly matched by $\text{SVM}^{\text{M-FW}}_{\text{FP}}$, and later by $\text{SVM}^{\text{M-FW}}$. Nevertheless, all

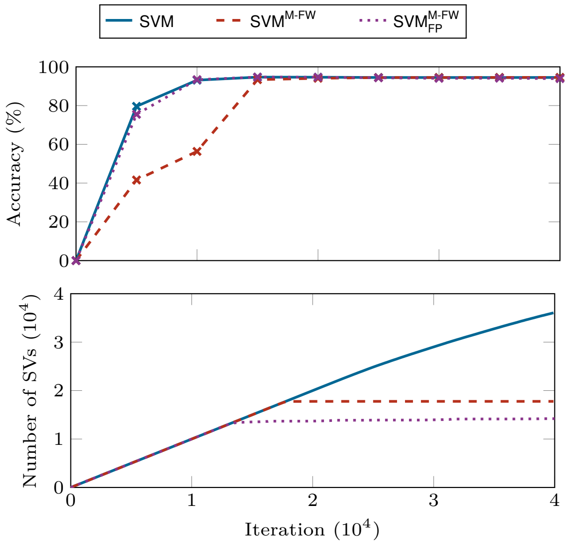


Figure 5: Evolution of the training for `miniboone` with RBF kernel, for the standard SVM, the proposed SVM$^{\text{M-FW}}$ and the parameter free SVM$^{\text{M-FW}}_{\text{FP}}$. The accuracy corresponds to the test set.

of the models get finally a comparable and stable accuracy, and they reach it at approximately the same number of iterations (around 15 000).

The main difference can be seen in the evolution of the number of support vectors. In the first iterations, all the models introduce a new support vector at each iteration, but first SVM$^{\text{M-FW}}_{\text{FP}}$ and second SVM$^{\text{M-FW}}$ saturate this number presenting a final almost flat phase. On the contrary, although SVM reduces slightly the rate of growth of the number of support vectors, it continues adding more patterns to the solution during the whole training. This means that, if the stopping criterion is not carefully chosen for SVM, this model will use much more support vectors than needed, with the corresponding increase in its complexity. On the other side, SVM$^{\text{M-FW}}$ and SVM$^{\text{M-FW}}_{\text{FP}}$ (both models trained with M-FW) limit successfully the number of support vectors, providing sparser models with the same accuracy as SVM.

As a remark, it should be noticed that for SVM$^{\text{M-FW}}_{\text{FP}}$ no validation phase was needed, since C is fixed beforehand, and for σ the optimal of SVM was used.

This suggests again that $\mathsf{SVM}_{\mathsf{FP}}^{\mathsf{M\text{-}FW}}$ can be applied successfully with $C = 1$ and only tuning σ if the RBF kernel is to be used.

5.4. Dependence on the Initialization

Another aspect of the proposed algorithm is its dependence on the initialization. Whereas the standard SVM is trained by solving a convex optimization problem with unique solution in the non-degenerate case, the proposed method summarized in Alg. 2 starts with an initial working vector that influences the resulting model, since it will determine the final subset of working vectors $\mathcal{W}^\star$.

5.4.1. Set-Up

A comparison of the models obtained using different initial working vectors will be done to study the variability due to the initialization. In particular, for all 9 smaller datasets of Table 1 and in this case only for the linear kernel with the parameters obtained in Section 5.2 (no CV process is repeated), one model per possible initial point will be trained, so that at the end there will be as many models as training patterns for each partition.

5.4.2. Results

A first measure for the dependence on the initialization are the differences between the sets of support vectors of the models. Table 5 shows in the second column the average overlap between these sets of support vectors for every pair of models with different initializations, quantified as the percentage of support vectors that are shared on both models over the total number of support vectors[3]. The two easiest datasets, `iris` and `mushrooms`, show the smallest overlaps (around 30 %) and hence the highest dependence on the initialization. This is not surprising, since for example in the `iris` dataset there are many hyperplanes that separate both classes perfectly. The remaining datasets show

[3] In particular, there are $N(N-1)/2$ measures per each one of the 10 repetitions, since there are N different possible initializations (as many as training patterns).

Data	SVs Overlap (%)	Accuracy (%)		
	SVM$^{\text{M-FW}}$ Ini.	SVM	SVM$^{\text{M-FW}}$	SVM$^{\text{M-FW}}$ Ini.
austra	95.69 ± 4.9	85.65 ± 4.4	86.09 ± 4.2	86.06 ± 3.9
breast	83.96 ± 4.2	96.92 ± 1.8	96.49 ± 1.7	96.45 ± 1.7
diabet	97.26 ± 2.2	77.35 ± 3.9	78.52 ± 3.0	77.69 ± 3.6
german	92.24 ± 4.8	76.70 ± 3.3	76.60 ± 4.1	76.86 ± 3.7
heart	81.44 ± 3.1	82.59 ± 6.5	83.33 ± 7.3	82.87 ± 7.6
ionosp	97.66 ± 3.5	82.65 ± 6.9	82.65 ± 6.9	83.16 ± 6.4
iris	30.99 ± 25.0	100.00 ± 0.0	100.00 ± 0.0	99.66 ± 1.5
mushro	32.61 ± 5.2	100.00 ± 0.0	100.00 ± 0.0	100.00 ± 0.0
sonar	96.69 ± 2.3	72.60 ± 7.3	70.21 ± 8.4	70.47 ± 8.7

Table 5: Results for the initialization dependence, including the overlap of the different sets of support vectors for SVM$^{\text{M-FW}}$, and the accuracies of SVM, SVM$^{\text{M-FW}}$ and SVM$^{\text{M-FW}}$ considering all possible initializations.

an overlap above 80 %, and there are 4 datasets above 95 %. Therefore, the influence on the initialization will depend strongly on the particular dataset.

Nevertheless, looking at the accuracies included in Table 5, and specifically comparing the results of SVM$^{\text{M-FW}}$ when considering only one or all the possible initializations (columns 4 and 5), it seems that there is no noticeable difference between them. In particular, and reducing the table to a single measure, the average error is 86.05 % for SVM, 85.99 % for SVM$^{\text{M-FW}}$ and 85.91 % for SVM$^{\text{M-FW}}$ considering all the initializations.

Moreover, as an additional experiment Fig. 6 shows the results of an extra 10-fold CV for the `heart` dataset with linear kernel, including the results of SVM$^{\text{M-FW}}$ with all the possible initializations. It can observed that SVM$^{\text{M-FW}}$ performs basically the same in average when changing the initial vector, in terms of all three the accuracy, the number of support vectors and the number of iterations, although obviously the distance between minimum and maximum value for each C (striped region in the plots) increases since more experiments are included.

Therefore, it can be concluded that, although the proposed method can depend strongly on the initialization for some datasets, it seems that the resulting models are comparable in terms of accuracy, number of support vectors and

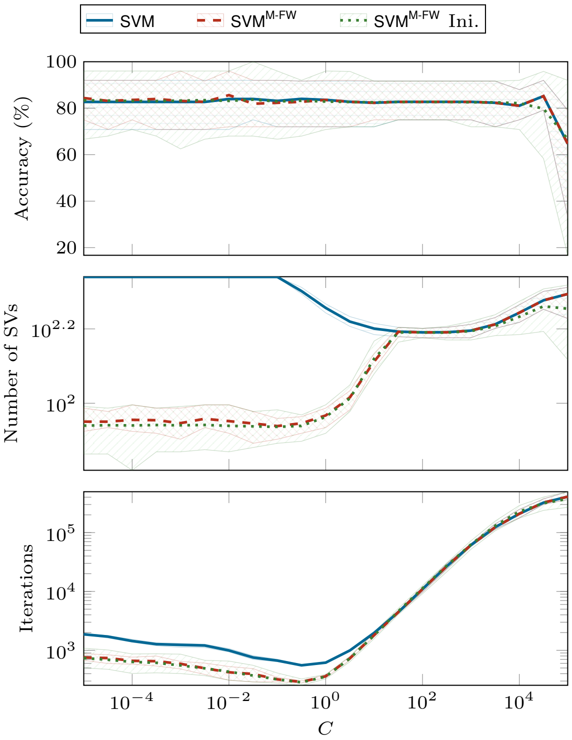


Figure 6: Evolution of the validation results for `heart` with linear kernel, for the standard SVM, the proposed SVM$^{\text{M-FW}}$ and SVM$^{\text{M-FW}}$ considering all possible initializations. The striped regions represent the range between minimum and maximum for the 10 partitions (10 times the number of training patterns when considering all the possible initializations), whereas the lines in the middle represent the average values.

required training iterations. On the other side, it should be noticed that trying to establish a methodology to initialize in a clever way the algorithm would probably need of a considerable overhead, since the computational advantage of Frank–Wolfe and related methods is that they compute the gradient incrementally because the changes only affect a few coordinates. A comparison between all the possible initial vertices, leaving aside heuristics, would require the use of the whole kernel matrix, what could be prohibitive for large datasets.

6. Conclusions

The connection between Lasso and Support Vector Machines (SVMs) has been used to propose an algorithmic improvement in the Frank–Wolfe (FW) algorithm used to train the SVM. This modification is based on the re-weighted Lasso to enforce more sparsity, and computationally it just requires an additional conditional check at each iteration, so that the overall complexity of the algorithm remains the same. The convergence analysis of this Modified Frank–Wolfe (M-FW) algorithm shows that it provides exactly the same SVM model that one would obtain applying the original FW algorithm only over a subsample of the training set. Several numerical experiments have shown that M-FW leads to models comparable in terms of accuracy, but with a sparser dual representation, requiring less iterations to be trained, and much more robust with respect to the regularization parameter, up to the extent of allowing to fix this parameter beforehand, thus avoiding its validation.

Possible lines of extension of this work are to explore other SVM formulations, for example based on the ℓ_1 loss, which should allow for even more sparsity. The M-FW algorithm could also be applied to the training of other machine learning models such as non-negative Lasso, or even to general optimization problems that permit a certain relaxation of the original formulation.

Acknowledgments

The authors would like to thank the following organizations. • EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923). This paper reflects only the authors' views, the Union is not liable for any use that may be made of the contained information. • Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants. • Flemish Government: – FWO: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants. – IWT: SBO POM (100031); PhD/Postdoc grants. • iMinds Medical Information Technologies SBO 2014. • Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017). • Fundación BBVA: project FACIL–Ayudas Fundación BBVA a Equipos de Investigación Científica 2016. • UAM–ADIC Chair for Data Science and Machine Learning.

References

[1] J. H. Friedman, Regularized discriminant analysis, Journal of the American statistical association 84 (405) (1989) 165–175. `doi:10.2307/2289860`.

[2] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297. `doi:10.1007/bf00994018`.

[3] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society. Series B (Methodological) (1996) 267–288.

[4] M. Jaggi, An equivalence between the lasso and support vector machines, in: J. A. K. Suykens, M. Signoretto, A. Argyriou (Eds.), Regularization, optimization, kernels, and support vector machines, Chapman and Hall/CRC, 2014, pp. 1–26.

[5] E. J. Candès, M. B. Wakin, S. P. Boyd, Enhancing sparsity by reweighted ℓ_1 minimization, Journal of Fourier analysis and applications 14 (5-6) (2008) 877–905. `doi:10.1007/s00041-008-9045-x`.

[6] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, K. R. Murthy, A fast iterative nearest point algorithm for support vector machine classifier design, IEEE transactions on neural networks 11 (1) (2000) 124–136. `doi:10.1109/72.822516`.

[7] C. M. Alaíz, A. Torres, J. R. Dorronsoro, Solving constrained lasso and elastic net using ν-SVMs, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2015, i6doc.com, 2015, pp. 267–272.

[8] H. Zou, The adaptive lasso and its oracle properties, Journal of the American statistical association 101 (476) (2006) 1418–1429. doi:10.1198/016214506000000735.

[9] D. Wipf, S. Nagarajan, Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions, IEEE Journal of Selected Topics in Signal Processing 4 (2) (2010) 317–329.

[10] M. Lapin, M. Hein, B. Schiele, Learning using privileged information: Svm+ and weighted svm, Neural Networks 53 (2014) 95–108. doi:10.1016/j.neunet.2014.02.002.

[11] W. Zhou, L. Zhang, L. Jiao, Linear programming support vector machines, Pattern recognition 35 (12) (2002) 2927–2936.

[12] J. Zhu, S. Rosset, R. Tibshirani, T. J. Hastie, 1-norm support vector machines, in: Advances in neural information processing systems, 2004, pp. 49–56.

[13] M. Frank, P. Wolfe, An algorithm for quadratic programming, Naval research logistics quarterly 3 (1-2) (1956) 95–110. doi:10.1002/nav.3800030109.

[14] M. Jaggi, Revisiting Frank–Wolfe: Projection-free sparse convex optimization., in: ICML (1), 2013, pp. 427–435.

[15] S. Lacoste-Julien, M. Jaggi, On the global linear convergence of frank-wolfe optimization variants, in: Advances in Neural Information Processing Systems, 2015, pp. 496–504.

[16] B. Gärtner, M. Jaggi, Coresets for polytope distance, in: Proceedings of the twenty-fifth annual symposium on Computational geometry, ACM, 2009, pp. 33–42. doi:10.1145/1542362.1542370.

[17] H. Ouyang, A. Gray, Fast stochastic frank-wolfe algorithms for nonlinear svms, in: Proceedings of the 2010 SIAM International Conference on Data Mining, SIAM, 2010, pp. 245–256. doi:10.1137/1.9781611972801.22.

[18] E. Frandi, R. Nanculef, M. G. Gasparo, S. Lodi, C. Sartori, Training support vector machines using Frank–Wolfe optimization methods, International Journal of Pattern Recognition and Artificial Intelligence 27 (03) (2013) 1360003. doi:10.1142/s0218001413600033.

[19] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 1–27, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. doi:10.1145/1961189.1961199.

[20] M. Lichman, UCI machine learning repository (2013).
URL http://archive.ics.uci.edu/ml

[21] C.-C. Chang, C.-J. Lin, IJCNN 2001 challenge: Generalization ability and text decoding, in: Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on, Vol. 2, IEEE, 2001, pp. 1031–1036. doi:10.1109/ijcnn.2001.939502.