A PARTAN-Accelerated Frank-Wolfe Algorithm for Large-Scale SVM Classification

Emanuele Frandi
ESAT-STADIUS
KU Leuven, Belgium
efrandi@esat.kuleuven.be

Ricardo Ñanculef
Department of Informatics
Federico Santa María University, Chile
jnancu@inf.utfsm.cl

Johan A. K. Suykens, *Fellow IEEE*
ESAT-STADIUS
KU Leuven, Belgium
johan.suykens@esat.kuleuven.be

***Abstract*—Frank-Wolfe algorithms have recently regained the attention of the Machine Learning community. Their solid theoretical properties and sparsity guarantees make them a suitable choice for a wide range of problems in this field. In addition, several variants of the basic procedure exist that improve its theoretical properties and practical performance. In this paper, we investigate the application of some of these techniques to Machine Learning, focusing in particular on a Parallel Tangent (PARTAN) variant of the FW algorithm that has not been previously suggested or studied for this type of problems. We provide experiments both in a standard setting and using a stochastic speed-up technique, showing that the considered algorithms obtain promising results on several medium and large-scale benchmark datasets for SVM classification.**

I. Introduction

The Frank-Wolfe algorithm (hereafter FW) is a classical method for convex optimization that has seen a substantial revival in interest from researchers [1], [2], [3]. Recent results have shown that the family of FW algorithms enjoys powerful theoretical properties such as iteration complexity bounds that are independent of the problem size, provable primal-dual convergence rates, and sparsity guarantees that hold during the whole execution of the algorithm [4], [2]. Furthermore, several variants of the basic procedure exist which can improve the convergence rate and practical performance of the basic FW iteration [5], [6], [7], [8]. Finally, the fact that FW methods work with projection-free iterations is an essential advantage in applications such as matrix recovery, where a projection step (as needed, e.g., by proximal methods) has a super-linear complexity [2], [9]. As a result, FW is now considered a suitable choice for large-scale optimization problems arising in several contexts such as Machine Learning, statistics, bioinformatics and other fields [10], [11], [12]. In the context of SVM classification, for example, FW methods have been shown to perform well on large-scale datasets with hundreds of thousands of examples, thus providing a promising alternative to solvers such as Active Set methods and SMO [13], [14], whose applicability is often limited to small and medium scale problems [11], [7].

In this paper, we consider the application of some well-known variants of the FW algorithm to Machine Learning problems, focusing in particular on a type of FW iteration known in the literature as PARTAN, which to the best of our knowledge has not previously been employed for this kind of application. Using several benchmark SVM datasets, we show that this variant is able to accelerate the standard FW method, obtaining on average a $2.52\times$ speedup in CPU time. Furthermore, we show how some FW variants indeed display a faster convergence rate in practice using a primal-dual stopping criterion, though their advantage is limited when the value of the tolerance parameter is not too strict. Finally, to further improve running times on large problems, we consider a random sampling speedup technique, and elaborate on its advantages and drawbacks, particularly on the existence of a tradeoff between iteration complexity and risk of a premature convergence.

Structure of the Paper

The paper is organized as follows. Section II provides a general overview of the FW method and its modifications, their theoretical properties and some applications to Machine Learning, while in Section III we examine in more detail the PARTAN variant of FW. Then, in Section IV, we perform numerical experiments on SVM problems to assess the performance of the considered methods, and close the paper by summarizing our conclusions in Section V.

II. The Frank-Wolfe Method and its Variants

The FW algorithm [1] is a general method to solve optimization problems of the form

$$\min_{\alpha \in \Sigma} \; f(\alpha), \tag{1}$$

where $f : \mathbb{R}^m \to \mathbb{R}$ is a convex differentiable function with Lipschitz continuous gradient, and $\Sigma \subset \mathbb{R}^m$ a compact convex set. The main idea behind the FW iteration is to exploit a linear model of the objective function at the current iterate to define a new search direction. In its basic form, the standard FW algorithm can be schematized as in Algorithm 1.

A. Theoretical Properties

We summarize here, for the sake of completeness, some well-known primal-dual convergence results for the FW algorithm. Proofs for these results can be found in [2], [15].

Proposition 1 (Sublinear convergence)**.** *Let α^* be an optimal solution for problem (1). Then, for $k \geq 1$, the iterates of*

Algorithm 1 The general FW algorithm.

1: **Input:** an initial guess α^0.
2: **for** $k = 0, 1, \dots$ **do**
3: Define a search direction $d_{\text{FW}}^{(k)} = u^{(k)} - \alpha^{(k)}$, where

$$u^{(k)} \in \underset{u \in \Sigma}{\operatorname{argmin}} (u - \alpha^{(k)})^T \nabla f(\alpha^{(k)}). \tag{2}$$

4: Choose a stepsize $\lambda^{(k)}$, either via the line-search

$$\lambda^{(k)} \in \underset{\lambda \in [0,1]}{\operatorname{argmin}} f(\alpha^{(k)} + \lambda d_{\text{FW}}^{(k)})$$

or with the rule $\lambda^{(k)} = 2/(k+2)$ [2].
5: Update:

$$\alpha^{(k+1)} = \alpha^{(k)} + \lambda^{(k)} d_{\text{FW}}^{(k)} = (1 - \lambda^{(k)})\alpha^{(k)} + \lambda^{(k)} u^{(k)}.$$

6: **end for**

Algorithm 1 satisfy

$$f(\alpha^{(k)}) - f(\alpha^*) \leq \frac{4C_f}{k+2},$$

where C_f is the curvature constant of the objective [2].

Choice of the stopping criterion. As a consequence of Proposition 1, we immediately have that Algorithm 1 requires $\mathcal{O}(1/\varepsilon)$ iterations to obtain an *ε-approximate solution*, i.e. a solution $\alpha^{(k)}$ s.t. $f(\alpha^{(k)}) - f(\alpha^*) \leq \varepsilon$. However, given that the *primal gap* $f(\alpha^{(k)}) - f(\alpha^*)$ is not a computable quantity, this fact cannot be exploited directly. Instead, the stopping condition for FW algorithms is usually based on the following *duality gap* criterion [2]:

$$\Delta_d^{(k)} := \max_{u \in \Sigma} (\alpha^{(k)} - u)^T \nabla f(\alpha^{(k)}) \leq \varepsilon \ . \tag{3}$$

This is motivated by the fact that the duality gap provides an upper bound for the primal gap, i.e. $f(\alpha^{(k)}) - f(\alpha^*) \leq \Delta_d^{(k)}$, while at the same time enjoying the same asymptotic guarantees.

Proposition 2 (Primal-dual convergence)**.** *After $K \geq 2$ iterations, Algorithm 1 produces at least one iterate $\alpha^{(\bar{k})}$, $1 \leq \bar{k} \leq K$, s.t.*

$$\Delta_d^{(\bar{k})} \leq \frac{27C_f}{2(K+2)}. \tag{4}$$

From Proposition 2, it immediately follows that the $\mathcal{O}(1/\varepsilon)$ complexity bound holds for Δ_d as well. Furthermore, the above results give the tolerance parameter ε a clean interpretation as a tradeoff between optimization accuracy and overall computational complexity.

B. Variants on the Classical Iteration

Though endowed with solid theoretical properties, the standard FW algorithm exhibits a rather slow convergence rate, and is known to be prone to stagnation, as $d_{\text{FW}}^{(k)}$ tends to become nearly orthogonal to the gradient when nearing a solution [11]. Solutions to this drawback date back to the 1970s, and mostly consist in algorithmic variations where an alternative search direction is added to avoid stalling.

Among the most well-known variants of this kind is the Modified Frank-Wolfe method (MFW) [5], [6], [7], [8]. In the modified FW iteration, we define an alternative search direction by *maximizing* the linear model:

$$v^{(k)} \in \underset{v \in \Sigma}{\operatorname{argmax}} (v - \alpha^{(k)})^T \nabla f(\alpha^{(k)}),$$

and then setting $d_{\text{A}}^{(k)} = \alpha^{(k)} - v^{(k)}$. The best descent direction is then selected, i.e. we choose $d_{\text{A}}^{(k)}$ if $\nabla f(\alpha^{(k)})^T d_{\text{A}}^{(k)} \leq \nabla f(\alpha^{(k)})^T d_{\text{FW}}^{(k)}$, and stick to the standard $d_{\text{FW}}^{(k)}$ otherwise.

Another option is to use a pairwise (or "swap") FW iteration as proposed in [7], where the alternative search direction is defined as $d_{\text{SW}}^{(k)} = u^{(k)} - v^{(k)}$. In this case, the choice between $d_{\text{FW}}^{(k)}$ and $d_{\text{SW}}^{(k)}$ is based on a greedy criterion, i.e. we select the step that yields the best function value. It can be proved that the resulting procedure enjoys properties analogous to those of the MFW algorithm.

As the specialization of these algorithms has already been presented extensively in [7], we do not discuss them further, and refer to the literature for implementation details. In a similar vein, other options for improving the FW iterations, such as conjugate direction based FW or FW with optimization on a 2-dimensional convex hull [16], are not included in this paper due to space constraints.

From a theoretical point of view, these variants often enjoy improved convergence guarantees. In particular, under suitable hypotheses [1], a linear convergence rate in primal gap can be obtained, i.e. for sufficiently large k we have

$$\frac{f(\alpha^{(k+1)}) - f(\alpha^*)}{f(\alpha^{(k)}) - f(\alpha^*)} \leq M,$$

with $M \in (0, 1)$ a constant.

However, to the best of our knowledge, no analogous results improving Proposition 2 were obtained for the duality gap, meaning that there is no *a priori* guarantee that a stopping criterion based on Δ_d is able to capture the improved behaviour of the algorithm. Furthermore, as the linear convergence results are asymptotic in nature, it is not possible in general to predict whether for a given tolerance the linear rate will kick in before the algorithm stops. Some of the experiments in Section IV aim precisely at investigating these issues and their practical impact.

C. Applications to Machine Learning

One of the most prominent examples of applications of FW-based algorithms to the field of Machine Learning is given by the binary nonlinear L_2-SVM training problem [17]:

$$\min_{\alpha \in \mathbb{R}^m} \quad f(\alpha) = \tfrac{1}{2}\alpha^T K \alpha \qquad \text{s.t. } \sum_{i=1}^{m} \alpha_i = 1, \ \alpha \geq 0 \ , \tag{5}$$

[1] We refer to the specialized literature for the detailed analyses [6], [7], [8], noting that the necessary hypotheses are satisfied for the test problems used in this paper.

Here, K is a positive definite kernel matrix, and the feasible set Σ is the unit simplex, whose vertices are the coordinate vectors $e_1, \ldots, e_m$. It is easy to see that in this case

$$u^{(k)} = e_{i_*^{(k)}}, \qquad \text{where } i_*^{(k)} \in \operatorname*{argmin}_{i=1,\ldots,m} \nabla f(\alpha^{(k)})_i. \tag{6}$$

Though FW methods can in principle be applied to any SVM formulation giving rise to a compact and convex feasible set, the L_2-SVM is chosen here because of its convenience. The geometry of the unit simplex yields indeed very simple formulas for the key steps in the FW iteration, which from a computational perspective leads to an extremely efficient implementation [7].

It we denote $\mathcal{I}^{(k)} = \{i \,|\, \alpha_{[i]}^{(k)} > 0\}$, it follows directly from (6) that at iteration k the solution can be expressed in terms of at most $k+|\mathcal{I}^{(0)}|$ data points [2], [7], or in other words that the number of Support Vectors is bounded during the entire run of the algorithm, which constitutes a substantial advantage of FW methods in comparison to methods with dense iterates. This holds true in particular for nonlinear SVM problems with datasets where the solution is sparse (in terms of the number of SVs defining the classification model), on which the latter suffer from the so-called "curse of kernelization" and are unable to recover the sparsity of the solution [18]. In addition, Proposition 2 implies that the total number of iterations is independent of the dataset size m. Together with the sparsity certificate, this also implies that the memory requirement for the whole algorithm is bounded independently of m.

Another related problem that can be tackled with a FW method is the Lasso problem with a 1-norm constraint

$$\min_{\alpha \in \mathbb{R}^m} \quad f(\alpha) := \|A\alpha - b\|_2^2 \quad \text{s.t. } \|\alpha\|_1 \leq t\,,$$

where $A \in \mathbb{R}^{n \times m}$ is a measurement matrix and $b \in \mathbb{R}^n$. In this case, the advantage of a FW-based method would be the possibility to well approximate the solution of high-dimensional problems using a reduced set of explanatory variables. Indeed, from Proposition 1 we have that at most $\mathcal{O}(1/\varepsilon)$ "active" features are required to reach an ε-approximate optimality, independently of the dimensionality of the feature space in which the observations have been embedded.

Finally, matrix recovery problems with nuclear norm regularization of the form

$$\min_{\alpha \in \mathbb{R}^{n \times m}} \quad f(\alpha) := \|\mathcal{A}(\alpha) - b\|_2^2 \quad \text{s.t. } \|\alpha\|_* \leq t\,,$$

where $\mathcal{A} : \mathbb{R}^{n \times m} \to \mathbb{R}^p$ is a linear operator and $b \in \mathbb{R}^p$, have also been successfully tackled with FW-based solvers [9]. The motivation here is mainly that FW methods do not require projection steps. The solution of the linear approximation step can be obtained in a fast way by solving a largest eigenvalue problem, as opposed to proximal methods that require a full SVD of the gradient matrix at each iteration, which is prohibitive for large-scale problems.

As a motivating example, we consider the SVM problem (5) for the experiments in this paper, not only because of its significance, but also to allow for a comparison with the results obtained in previous research efforts [19], [20], [11], [7], [21].

III. PARTAN Frank-Wolfe Iterations

Another variant of the FW algorithm that has been proposed and successfully employed (for example, in traffic assignment applications [22], [23], [24], [16]) consists in an adaptation of the method of Parallel Tangents (PARTAN) to FW iterations [25]. To the best of our knowledge, though, this scheme has not yet been investigated in Machine Learning applications.

The basic idea, as seen from Figure 1, is to incorporate previous information by performing an averaging between the classical FW step and the previous iterate. First, an intermediate FW step is defined:

$$\tilde{\alpha} = (1 - \lambda^{(k)})\alpha^{(k)} + \lambda^{(k)} u^{(k)}. \tag{7}$$

Then, the previous iterate is used to define an extra search direction:

$$\alpha^{(k+1)} = \tilde{\alpha} + \mu^{(k)}(\tilde{\alpha} - \alpha^{(k-1)}). \tag{8}$$

Stepsizes $\lambda^{(k)}$ and $\mu^{(k)}$ can be determined via line-search.

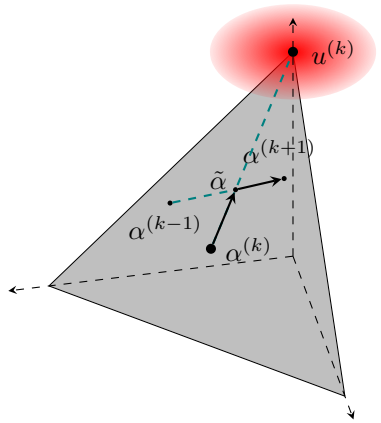
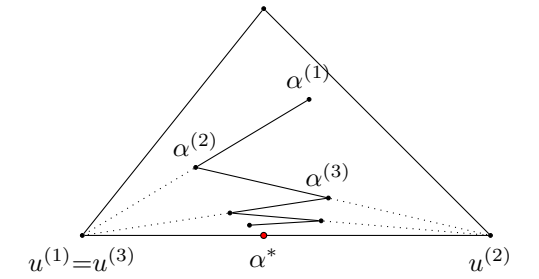


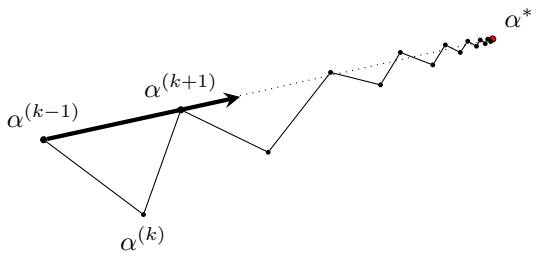
Fig. 1. Sketch of the search directions used by PARTAN iterations.

A geometrical interpretation of the PARTAN method can be obtained by looking at the typical behaviour of a standard FW iteration near a solution: the fact that the search direction of the FW method tends to become orthogonal to $\nabla f(\alpha^{(k)})$ close to the optimum can easily lead to a zigzagging trajectory, as seen from Figure 2(a). A simple way to circumvent this behaviour consists in performing an extra line-search along the line connecting $\alpha^{(k-1)}$ to $\tilde{\alpha}$ (which corresponds to a basic FW step from $\alpha^{(k)}$). The case depicted in Figure 2(b) shows how PARTAN is able to avoid traversing the "sawtooth" in the trajectory, directly moving towards a point closer to the solution. It is apparent how this approach is especially advantageous if the stepsizes can be computed by a closed formula, as is the case, e.g., for quadratic objective functions.

When specialized to the SVM problem (5), the algorithm assumes a simpler form, as the key steps in each iteration can be performed analytically. The necessary formulas, which are obtained via elementary algebraic manipulations, are reported in the Appendix. For the purposes of the discussion here, it suffices to mention that the cost per iteration of the PARTAN



(a)



(b)

Fig. 2. A geometrical interpretation of the PARTAN-FW iteration.

method is nearly equivalent to that of the standard FW, as also demonstrated by the numerical results in the next section. Regarding the stopping criterion, the duality gap can be conveniently computed as

$$\Delta_d^{(k)} = \nabla f(\alpha^{(k)})_{i_*^{(k)}} - 2f(\alpha^{(k)})\,. \tag{9}$$

We summarize the overall procedure in Algorithm 2.

Algorithm 2 The PARTAN-FW algorithm for problem (5).

Input: an initial estimate $\alpha^{(0)}$ and a tolerance ε.
Compute $\alpha^{(1)}$ via a standard FW step.
Search for $i_*^{(1)} \in \operatorname{argmax}_i \nabla f(\alpha^{(1)})_i$.
Initialize the duality gap as in (9).
Set $k = 1$.
while $\Delta_d^{(k)} > \epsilon$ **do**
  Compute the optimal FW steplength as in (11).
  Compute the function value after the intermediate FW step as in (12).
  Compute W_k as in (14).
  Compute the optimal PARTAN steplength as in (13).
  Perform the PARTAN step (8) as:

$$\alpha^{(k+1)} = (1 + \mu^{(k)} - \lambda^{(k)} - \mu^{(k)}\lambda^{(k)})\alpha^{(k)} - \mu^{(k)}\alpha^{(k-1)} + (\lambda^{(k)} + \lambda^{(k)}\mu^{(k)})e_{i_*^{(k)}}.$$

  Update the function value as in (15).
  Set $k := k + 1$.
  Search for $i_*^{(k)} \in \operatorname{argmax}_i \nabla f(\alpha^{(k)})_i$.
  Update the duality gap as in (9).
end while

IV. Numerical Results

In this section, we assess the performance of all the considered variants of FW on the binary classification problem (5), using the benchmark datasets listed in Table I. The number of examples in the training set and test set are denoted by m and t, respectively, while n denotes the number of features.

Dataset	m	t	n
Adult a9a	32,561	16,281	123
Web w8a	49,749	14,951	300
IJCNN1	49,990	91,701	22
USPS-ext	266,079	75,383	675
KDD99-binary	395,216	98,805	38
RCV1-binary	677,399	20,242	47,236

TABLE I
List of the benchmark datasets for problem (5).

All the experiments are performed with an RBF kernel. Due to the size of the datasets, the SVM regularization parameter is selected by a simple approach, where a single validation set is built by randomly extracting 70% of the training examples, and the remaining 30% is reserved for testing [2]. For the RCV1 dataset, we used the value suggested in [26]. The kernel width is selected according to the heuristic in [17]. The algorithms are coded in C++ and run in Linux on a 3.40 GHz Intel i7 machine with 16 GB of main memory.

In the first experiment, we set $\varepsilon = 10^{-4}$ in the stopping criterion (3) and evaluate the performance of the proposed methods in terms of test accuracy, CPU time (in seconds), number of iterations and model size (number of SVs). Results are reported in Table II.

It can be seen that all the algorithms generally exhibit a good performance. In particular, the PARTAN variant in Algorithm 2 yields the most consistent results, improving on the running times of the plain FW by a factor of 2.52 on average. Results relative to test accuracy and model sizes are fairly stable, with no particular variant outperforming the others in most cases, though it should be noted that PARTAN is often able to find a smaller SV set. This means that the number of spurious points (i.e. active examples which are not part of the true SV set) selected by the FW iterations is potentially reduced, as especially evident on the **Web w8a** dataset.

We can also see how the reduction in computational time for PARTAN-FW is roughly proportional to the decrease in the number of iterations, which confirms our intuition that using the PARTAN algorithm on SVM does not imply a higher iteration complexity than that of the standard FW [3]. Seeing how this technique provides a systematic speedup with no evident drawbacks when compared to the standard FW, we recommend it over the latter for large-scale SVM problems.

A potentially relevant observation is that the benefit of using the PARTAN-accelerated iteration is related to a good extent to the sparsity of the solution. The advantage is indeed more

[2]The same values of the hyper-parameters are used for all the methods.
[3]Note that this also holds true for the other variants [7].

	FW	MFW	SWAP	PARTAN
Adult a9a				
Acc (%)	84.21	83.29	83.53	84.00
Time	1.58e + 02	1.57e + 02	2.26e + 02	1.07e + 02
Iter	2.02e + 04	2.00e + 04	1.76e + 04	1.34e + 04
SVs	1.39e + 04	1.28e + 04	1.41e + 04	1.18e + 04
Web w8a				
Acc (%)	99.30	99.32	99.28	99.30
Time	3.78e + 02	3.16e + 02	3.56e + 02	1.07e + 02
Iter	1.65e + 04	1.38e + 04	9.24e + 03	4.62e + 03
SVs	6.92e + 03	4.48e + 03	4.97e + 03	2.83e + 03
IJCNN1				
Acc (%)	98.50	98.22	98.40	98.36
Time	5.13e + 01	4.57e + 01	5.09e + 01	1.98e + 01
Iter	1.59e + 04	1.41e + 04	1.35e + 04	5.48e + 03
SVs	3.23e + 03	2.73e + 03	3.16e + 03	2.73e + 03
USPS-ext				
Acc (%)	99.52	99.52	99.53	99.52
Time	1.98e + 03	8.44e + 02	9.46e + 02	7.38e + 02
Iter	2.15e + 04	9.17e + 03	8.10e + 03	7.79e + 03
SVs	3.93e + 03	3.64e + 03	3.67e + 03	3.51e + 03
KDD99-binary				
Acc (%)	99.94	99.93	99.94	99.93
Time	7.02e + 02	5.26e + 02	5.51e + 02	1.89e + 02
Iter	1.71e + 04	1.27e + 04	7.82e + 03	4.37e + 03
SVs	5.25e + 03	3.63e + 03	3.86e + 03	2.82e + 03
RCV1-binary				
Acc (%)	97.55	96.64	97.17	97.50
Time	1.37e + 04	1.36e + 04	1.71e + 04	1.23e + 04
Iter	3.77e + 04	3.81e + 04	3.65e + 04	3.38e + 04
SVs	3.75e + 04	3.58e + 04	3.65e + 04	3.37e + 04

TABLE II
COMPARISON OF DIFFERENT VARIANTS OF FW ON BENCHMARK SVM DATASETS.

apparent on problems where the size of the SV set is a small fraction of the total number of examples, with the **KDD99-binary** dataset being a prominent example.

On the other hand, the FW variants show no advantage over the standard algorithm on the **RCV1-binary** problem. This is arguably because the number of SVs is basically the same as the total number of iterations. Since the FW algorithm spends all of its iterations adding new vertices (i.e. examples corresponding to nonzero components of $\alpha^{(k)}$) to the model, the usual slowdown behaviour of FW, where the algorithm cycles between the same vertices readjusting their weights, is not observed. As such, there is little benefit in adding modified FW directions. The same phenomenon is observed, on a smaller scale, on the **Adult a9a** dataset.

This is consistent with the fact that FW methods are best used to solve sparse problems, as also suggested by their theoretical properties. Conversely, their usefulness is more limited when the solution is dense, as the incremental nature of the algorithm provides no particular advantage in this case.

As far as the difference in performance between the variants is concerned, we remark that the theoretical results in Section I are of asymptotic nature, and as such it is difficult to assess their practical impact with a fixed value of ε, which might be too large to observe a faster convergence compared to the standard FW. We investigate this issue in the next paragraph.

A. Considerations on the Iteration Complexity

We now attempt to better assess the practical difference between the standard FW and its variants, in order to understand how and when the latter can give a subtantial advantage. In particular, we want to estabilish whether the improved convergence predicted by the theory can be observed experimentally when using a duality gap-based stopping criterion. To this end, we apply all the considered variants of FW to the datasets **Adult a9a**, **Web w8a** and **IJCNN1**, using increasingly strict tolerances $\varepsilon \in \{10^{-3}, \dots, 10^{-6}\}$, and monitoring the number of iterations needed to trigger the stopping condition. We do not attempt to solve the larger scale problems here, as the smallest value of ε would lead to prohibitive running times, and we remark that this experiment aims exclusively at providing an insight on the convergence speed of the algorithms. Results are shown in Table III.

	ε	1e − 03	1e − 04	1e − 05	1e − 06
Adult a9a					
FW	Time	2.24e + 01	1.58e + 02	1.46e + 03	1.42e + 04
	Iter	2.82e + 03	2.02e + 04	1.84e + 05	1.79e + 06
MFW	Time	2.20e + 01	1.57e + 02	5.48e + 02	1.21e + 03
	Iter	2.77e + 03	2.00e + 04	6.80e + 04	1.50e + 05
SWAP	Time	3.07e + 01	2.26e + 02	6.46e + 02	1.30e + 03
	Iter	2.61e + 03	1.76e + 04	5.18e + 04	1.09e + 05
PARTAN	Time	1.73e + 01	1.07e + 02	5.09e + 02	5.43e + 03
	Iter	2.15e + 03	1.64e + 04	6.21e + 04	6.63e + 05
Web w8a					
FW	Time	4.47e + 01	3.78e + 02	3.73e + 03	4.02e + 04
	Iter	1.93e + 03	1.65e + 04	1.63e + 05	1.75e + 06
MFW	Time	4.27e + 01	3.16e + 02	1.52e + 03	3.78e + 03
	Iter	1.86e + 03	1.38e + 04	6.38e + 04	1.65e + 05
SWAP	Time	6.42e + 01	3.56e + 02	1.31e + 03	8.63e + 03
	Iter	1.69e + 03	9.24e + 03	4.29e + 04	3.46e + 05
PARTAN	Time	1.83e + 01	1.07e + 02	5.97e + 02	3.32e + 03
	Iter	7.77e + 02	4.62e + 03	2.57e + 04	1.44e + 05
IJCNN1					
FW	Time	5.77e + 00	5.13e + 01	5.49e + 02	6.48e + 03
	Iter	1.72e + 03	1.59e + 04	1.68e + 05	1.97e + 06
MFW	Time	5.16e + 00	4.57e + 01	2.38e + 02	6.81e + 02
	Iter	1.51e + 03	1.41e + 04	7.12e + 04	2.05e + 05
SWAP	Time	6.94e + 00	5.09e + 01	2.56e + 02	6.56e + 02
	Iter	1.21e + 03	1.35e + 04	6.85e + 04	1.77e + 05
PARTAN	Time	3.09e + 00	1.98e + 01	1.60e + 02	1.83e + 03
	Iter	7.83e + 02	5.48e + 03	4.34e + 04	4.99e + 05

TABLE III
ITERATION COMPLEXITY OF DIFFERENT VARIANTS OF FW.

From the results, it is clear how the standard FW behaves according to the $\mathcal{O}(1/\varepsilon)$ iteration bound, with the number of iterations increasing 10-fold every time the tolerance parameter decreases by one order of magnitude. This corresponds to the duality gap decreasing as $\mathcal{O}(1/k)$, as predicted by Proposition 2. This result suggests that, though (4) is an upper bound of the duality gap (and thus in turn of the primal gap), it gives in practice a good indication of the number of iterations that we can expect from the standard FW algorithm, therefore implying that the computational effort can be predicted and

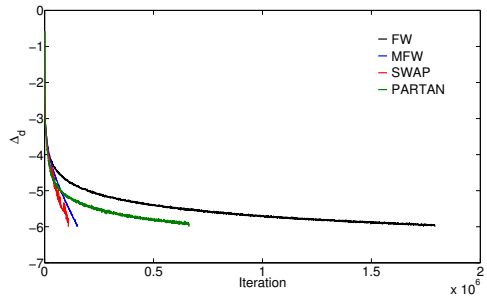
controlled by appropriately tuning the tolerance parameter. The modified variants, in contrast, enjoy a faster convergence rate, ending up gaining a computational advantage of one order or magnitude or more with respect to the plain FW when the strictest tolerance value is used. It is interesting to note how the three variants analyzed here do not always provide the same improvement. As the results presented in this work are only preliminary, it is difficult to establish whether this is simply due to the methods having different convergence factors (e.g. because they enjoy convergence rates with the same asymptotic behaviour but differing by a constant) or is intrinsically related to the nature of the algorithms (for example, PARTAN starts out with a substantial advantage over the other variants at $\varepsilon = 10^{-4}$, but is outperformed by MFW when seeking for a more accurate solution).

We can attempt to shed some more light on this issue by plotting in Figure 3 the duality gap (in logarithmic scale), obtained with $\varepsilon = 10^{-6}$, against the iteration number for all the variants of the algorithm. From the graphs, it can be seen how the MFW algorithm seems to exhibit the best primal-dual convergence rate, with oscillations in the duality gap being very small [4]. The SWAP algorithm performs even better on two of the datasets (though exhibiting larger oscillations), but substantially worse on **Web w8a**. The PARTAN variant seems instead to show a behaviour similar to that of the standard FW, but with a better convergence factor, an observation which is consistent with the results obtained in Tables II and III. It might also be worth noticing that $\Delta_d^{(k)}$ is only an upper bound of the optimality measure $f(\alpha^{(k)}) - f(\alpha^*)$, thus occasional oscillating values of the duality gap do not imply that the solution is getting less accurate.

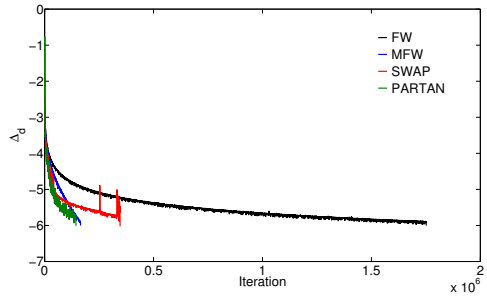
Overall, though we are well aware that a more representative batch of problems would be needed to draw more solid conclusions, the results in Table III show that the considered FW variants indeed display a faster convergence rate in practice using a primal-dual stopping criterion. It is not obvious, however, whether the results on the duality gap given by Proposition 2 can be improved under suitable hypotheses. They also show how the traditional FW is not a suitable method if one wants to use stricter values of ε, for example because the application at hand requires a higher optimization accuracy. This confirms on a Machine Learning problem the well-known intuition that the standard FW step stagnates when close to a solution, unless extra search directions which do not get orthogonal to the gradient are added [11].

It should be noted, indeed, that a good choice of ε is application-dependent. In SVMs for classification, for instance, it is well known that the test accuracy is often relatively insensitive to ε after a certain threshold. On the other hand, different applications, such as function estimation, could be more sensitive to the accuracy (in an optimization sense) of the obtained model. Therefore, while they may not appear very relevant in the context of classification SVMs, the improved properties of some FW modifications may be of importance for other related tasks. In this case, we would recommend the use of a FW variant rather than the standard algorithm.

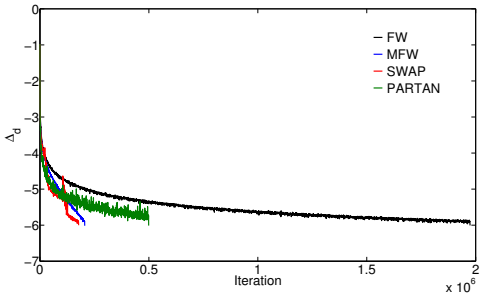
[4]It is important to observe that, as opposed to $f(\alpha^{(k)})$, $\Delta_d^{(k)}$ is not a monotonically decreasing quantity.



(a)



(b)



(c)

Fig. 3. Duality gap behaviour of FW algorithms for $\varepsilon = 10^{-6}$ on the datasets **Adult a9a** (a), **Web w8a** (b), and **IJCNN1** (c).

B. Results with Randomized Iterations

As the total number of iterations required by a FW algorithm can be large, devising a convenient way to solve the subproblem (2) is recommended in order to make the algorithm more viable on large-scale datasets. A typical situation arises when (2) has an analytical solution or it is easy to solve due to the problem structure [7], [27]. This is the case, for example, for all the problems introduced in Section II-C. Still, the resulting complexity usually depends on the problem size (for example, in (6) it is proportional to m), and can thus be impractical when handling large-scale data.

A simple and yet effective way to avoid the dependence on m is to look for the solution of (2) by exploring only a fixed

number of extreme points on the boundary of Σ [17], [28], [21]. In the case of (5), for example, this means extracting a sample $\mathcal{S} \subseteq \{1, \dots, m\}$ and solving

$$i_{\mathcal{S}}^{(k)} \in \underset{i \in \mathcal{S}}{\operatorname{argmin}} \nabla f(\alpha^{(k)})_i \,. \tag{10}$$

The cost of an iteration becomes in this case $\mathcal{O}(|\mathcal{S}||\mathcal{I}^{(k)}|)$, rather than $\mathcal{O}(m|\mathcal{I}^{(k)}|)$ as in (6)[5].

The stopping criterion, however, is not applicable without computing the entire gradient $\nabla f(\alpha^{(k)})$, which is not done in the randomized case. As a possible alternative, we can use the approximate quantity

$$\Delta_{\mathcal{S}}(\alpha^{(k)}) := 2f(\alpha^{(k)}) - \nabla f(\alpha^{(k)})_{i_{\mathcal{S}}^{(k)}}.$$

Since $\Delta_{\mathcal{S}}(\alpha^{(k)}) \leq \Delta_d^{(k)}$, this simplification entails a tradeoff between the reduction in computational cost and risk of a premature stopping. Although this can be acceptable in contexts such as SVM classification, where solving the optimization problem with a high accuracy is usually not needed, it is important to make sure that the impact of this approximation is kept to an acceptable level. This issue has been discussed in detail in [21]. Here, to mitigate the effect of a possible early stopping, we implement a simple safeguard strategy where the sampling (10) is repeated twice in case $\Delta_{\mathcal{S}}(\alpha^{(k)}) \leq \varepsilon$.

In Table IV, we report the results obtained with a randomization technique, taking $|\mathcal{S}| = 194$ [6], averaged over 10 runs. Note that we do not attempt to run the randomization technique on problems for which this strategy is not beneficial. Taking **RCV1-binary** as an example, it is already clear, from the structure of the dataset and the results in Table II, that using a random sampling would not provide any advantage: the SV set size being of order 10^4, an iteration would have a complexity in the order of millions of floating point operations, which is actually much larger than the size of the whole dataset. In general, we do not recommend using a random sampling for problems with dense SV sets, as in order to obtain a computational gain the number of samples would have to be too small, possibly leading to an inaccurate solution.

First of all, note that the effect of sampling is substantially problem-dependent. The best computational gains are obtained on the problems **Web w8a**, **USPS-ext** and **KDD99-binary**, with the latter two being the largest and most sparse datasets. It should be noted that the reduction in CPU time is attributable both to the reduced iteration complexity and to the smaller iteration count, the latter being due to the approximate stopping criterion employed. This is not observed on all the datasets, however. For example, the total number of iterations on **Adult a9a** and **IJCNN1** is comparable to that of the deterministic case. The MFW algorithm also appears to be overall less sensitive to this particular issue.

It is interesting to note that this phenomenon does not necessarily lead to a loss in test accuracy. This is possibly due to the nature of the SVM classification models, which do not require a very accurate solution of the optimization problem to build a decision function with a good predictive capability.

[5]It should also be noted that a clever implementation allows to eliminate the $|\mathcal{I}^{(k)}|$ factor from (6) in the SVM case [21].

[6]This value corresponds to a probability of at least 0.98 that $i_*^{(k)}$ lies in the 2% smallest components of $\nabla f(\alpha^{(k)})$. See Theorem 6.33 in [28].

	FW	MFW	SWAP	PARTAN
Adult a9a				
Acc (%)	83.94	83.86	83.52	84.08
Time	1.69e + 02	1.63e + 02	1.64e + 02	1.08e + 02
Iter	1.89e + 04	1.98e + 04	1.65e + 04	1.24e + 04
SVs	1.35e + 04	1.24e + 04	1.35e + 04	1.11e + 04
Web w8a				
Acc (%)	99.17	99.21	99.16	99.10
Time	8.39e + 01	6.69e + 01	7.12e + 01	4.89e + 01
Iter	6.78e + 03	9.10e + 03	4.97e + 03	3.25e + 03
SVs	3.66e + 03	3.01e + 03	3.19e + 03	2.31e + 03
IJCNN1				
Acc (%)	98.57	97.98	98.34	98.37
Time	3.45e + 01	2.10e + 01	2.21e + 01	1.95e + 01
Iter	1.12e + 04	1.14e + 04	7.10e + 03	5.76e + 03
SVs	4.12e + 03	2.45e + 03	3.43e + 03	3.34e + 03
USPS-ext				
Acc (%)	99.53	99.57	99.55	99.53
Time	2.19e + 02	2.60e + 02	2.25e + 02	2.07e + 02
Iter	3.61e + 03	6.49e + 03	3.59e + 03	3.28e + 03
SVs	2.96e + 03	2.48e + 03	2.94e + 03	2.86e + 03
KDD99-binary				
Acc (%)	99.73	99.93	99.78	99.82
Time (s)	2.84e + 01	9.46e + 01	2.03e + 01	1.20e + 01
Iter	1.88e + 03	1.29e + 04	1.57e + 03	1.15e + 03
SVs	1.71e + 03	2.35e + 03	1.45e + 03	1.11e + 03

TABLE IV
COMPARISON OF DIFFERENT VARIANTS OF FW ON BENCHMARK SVM DATASETS (RANDOMIZED ITERATION).

V. CONCLUSIONS

The results presented in this paper show that the family of FW algorithms obtains promising results on several benchmark SVM classification tasks, offering a solid and fast alternative to the classical solvers used in this field.

While the experimental results presented here are preliminary, they provide the first example of a successful application of the PARTAN-FW iteration to Machine Learning problems, showing that this variant is able to accelerate the basic FW iteration in a systematic way. On the other hand, the advantage of other modified FW algorithms is especially apparent when employing stricter tolerance parameters, arguably due to the stronger theoretical properties of the enhanced iterations.

Finally, we have shown how, on larger scale problems, a randomization technique can be employed to reduce the computational effort with satisfactory results, with some caveats related to the tradeoff between complexity and optimization accuracy which is inherent to this kind of strategy.

Experiments on different machine learning applications (such as Lasso and matrix recovery problems) are currently being investigated, and will be the subject of another paper.

ACKNOWLEDGMENTS

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 and 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; Flemish Government: FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants; iMinds Medical Information Technologies SBO 2014; IWT: POM II SBO 100031; Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017). The second author received funding from CONICYT Chile through FONDECYT Project 11130122.

APPENDIX
IMPLEMENTATION DETAILS

We report here, for the sake of completeness, the analytical formulas used in the implementation of Algorithm 2 for the SVM problem (5). To simplify the equations, we use the shorthand notations $f^{(k)} = f(\alpha^{(k)})$ and $\nabla f^{(k)} = \nabla f(\alpha^{(k)})$.

After some elementary algebraic manipulations, we obtain that the optimal steplength value for step (7) is given by

$$\lambda^{(k)} = \frac{2f^{(k)} - \nabla f^{(k)}_{i^{(k)}_*}}{2f^{(k)} - 2\nabla f^{(k)}_{i^{(k)}_*} - K_{i^{(k)}_*, i^{(k)}_*}}. \tag{11}$$

After this step, the objective value becomes

$$\begin{aligned}\tilde{f} = (1-\lambda^{(k)})^2 f^{(k)} + \lambda^{(k)}(1-\lambda^{(k)})\nabla f^{(k)}_{i^{(k)}_*} \\ - \tfrac{1}{2}(\lambda^{(k)})^2 K_{i^{(k)}_*, i^{(k)}_*}.\end{aligned} \tag{12}$$

The steplength for the PARTAN step (8) is then given by

$$\mu^{(k)} = \frac{\lambda^{(k)}\nabla f^{(k-1)}_{i^{(k)}_*} - (1-\lambda^{(k)})W^{(k)} - 2\tilde{f}}{2(\tilde{f} + (1-\lambda^{(k)})W^{(k)} - \lambda^{(k)}\nabla f^{(k-1)}_{i^{(k)}_*} + f^{(k-1)})}, \tag{13}$$

where

$$\begin{aligned}W^{(k)} = &-2(1+\mu^{(k-1)})(1-\lambda^{(k-1)})f^{(k-1)} \\ &- (1+\mu^{(k-1)})\lambda^{(k-1)}\nabla f^{(k-1)}_{i^{(k-1)}_*} - \mu^{(k-1)}W^{(k-1)}\end{aligned} \tag{14}$$

is a quantity that can be computed recursively starting from $W^{(1)} = (\alpha^{(0)})^T K \alpha^{(1)}$. Finally, the updated objective value after the PARTAN iteration is

$$\begin{aligned}f^{(k+1)} = (1+\mu^{(k)})^2\tilde{f} + \mu^{(k)}(1+\mu^{(k)}_k)(1-\lambda^{(k)})W_k \\ - \mu^{(k)}(1+\mu^{(k)})\lambda^{(k)}\nabla f^{(k-1)}_{i^{(k)}_*} + (\mu^{(k)})^2 f^{(k-1)}.\end{aligned} \tag{15}$$

REFERENCES

[1] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 1, pp. 95–110, 1956.

[2] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in *Proceedings of the 30th ICML*, 2013.

[3] Z. Harchaoui, A. Juditski, and A. Nemirovski, "Conditional gradient algorithms for norm-regularized smooth convex optimization," *Mathematical Programming*, vol. 13, no. 1, pp. 1–38, 2014.

[4] K. Clarkson, "Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm," *ACM Transactions on Algorithms*, vol. 6, no. 4, pp. 63:1–63:30, 2010.

[5] P. Wolfe, "Convergence theory in nonlinear programming," in *Integer and Nonlinear Programming*, J. Abadie, Ed. North-Holland, 1970, pp. 1–36.

[6] J. Guélat and P. Marcotte, "Some comments on Wolfe's "away step"," *Mathematical Programming*, vol. 35, pp. 110–119, 1986.

[7] R. Ñanculef, E. Frandi, C. Sartori, and H. Allende, "A novel Frank-Wolfe algorithm. Analysis and applications to large-scale SVM training," *Information Sciences (in press)*, 2014.

[8] S. Lacoste-Julien and M. Jaggi, "An affine invariant linear convergence analysis for Frank-Wolfe algorithms," *arXiv.org*, 2014.

[9] M. Signoretto, E. Frandi, Z. Karevan, and J. Suykens, "High level high performance computing for multitask learning of time-varying models," in *IEEE Symposium on Computational Intelligence in Big Data*, 2014.

[10] A. Argyriou, M. Signoretto, and J. Suykens, "Hybrid algorithms with applications to sparse and low rank regularization," in *Regularization, Optimization, Kernels, and Support Vector Machines*, J. Suykens, M. Signoretto, and A. Argyriou, Eds. Chapman & Hall/CRC, 2014.

[11] E. Frandi, M. G. Gasparo, S. Lodi, R. Ñanculef, and C. Sartori, "Training support vector machines using Frank-Wolfe methods," *IJPRAI*, vol. 27, no. 3, 2011.

[12] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher, "Block-coordinate Frank-Wolfe optimization for structural SVMs," in *Proceedings of the 30th ICML*, 2013.

[13] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in kernel methods: support vector learning*. MIT Press, 1999, pp. 169–184.

[14] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in kernel methods: support vector learning*. MIT Press, 1999, pp. 185–208.

[15] G. Lan, "The complexity of large-scale convex programming under a linear optimization oracle," *arXiv.org*, 2014.

[16] M. Mitradjieva and P. O. Lindberg, "The stiff is moving - conjugate direction Frank-Wolfe methods with applications to traffic assignment," *Transportation Science*, vol. 47, no. 2, pp. 280–293, 2012.

[17] I. Tsang, J. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *JMLR*, vol. 6, pp. 363–392, 2005.

[18] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training," *JMLR*, vol. 13, pp. 3103–3131, 2012.

[19] E. Frandi, M. G. Gasparo, S. Lodi, R. Ñanculef, and C. Sartori, "A new algorithm for training SVMs using approximate minimal enclosing balls," in *Proceedings of the 15th Iberoamerican Congress on Pattern Recognition*. Springer, 2010, pp. 87–95.

[20] E. Frandi, M. G. Gasparo, R. Ñanculef, and A. Papini, "Solution of classification problems via computational geometry methods," *Recent Advances in Nonlinear Optimization and Equilibrium Problems: a Tribute to Marco D'Apuzzo, Quaderni di Matematica*, vol. 27, pp. 201–226, 2012.

[21] E. Frandi, R. Ñanculef, and J. Suykens, "Complexity issues and randomization strategies in Frank-Wolfe algorithms for machine learning," in *7th NIPS Workshop on Optimization for Machine Learning*, 2014.

[22] M. Florian, J. Guélat, and H. Spiess, "An efficient implementation of the "PARTAN" variant of the linear approximation method for the network equilibrium problem," *Networks*, vol. 17, no. 3, pp. 319–339, 1987.

[23] L. J. LeBlanc, R. V. Helgason, and D. E. Boyce, "Improved efficiency of the Frank-Wolfe algorithm for convex network programs," *Transportation Science*, vol. 19, no. 4, pp. 445–462, 1985.

[24] Y. Arezki and D. van Vliet, "A full analytical implementation of the PARTAN/Frank-Wolfe algorithm for equilibrium assignment," *Transportation Science*, vol. 24, no. 1, pp. 58–62, 1990.

[25] D. G. Luenberger and Y. Ye, *Linear and Non-Linear Programming*, 3rd ed. Springer, 2008.

[26] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *JMLR*, vol. 5, pp. 361–397, 2004.

[27] G. Liuzzi and F. Rinaldi, "Solving l_0-penalized problems with simple constraints via the Frank-Wolfe reduced dimension method," *Optimization Letters*, vol. 9, no. 1, pp. 57–74, 2015.

[28] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.