Frank-Wolfe method (and variants) applied to SVM

OPTIMIZATION FOR DATA SCIENCE - FINAL PROJECT

CAMILO BETANCOURT NIETO

1 INTRODUCTION

This project aims to implement and analyze some optimization algorithms in the context of the training of Support Vector Machines (SVM). More specifically, taking into consideration the theoretical properties of the Frank-Wolfe algorithm and some of its variants (the Away-steps Frank Wolfe and the Pairwise Frank-Wolfe), the goal of the project is to train a soft-margin SVM for each of three datasets that will be referred from now on as *Breast cancer*, *Phishing* and *Skin*.

I based my work on the guidelines provided by the *Optimization for data science* lectures taught by Francesco Rinaldi and three papers: *An Equivalence between the Lasso and Support Vector Machines* (Jaggi, 2014), *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization* (Jaggi, 2013), and *On the Global Linear Convergence of Frank-Wolfe Optimization Variants* (Lacoste-Julien & Jaggi, 2015).

2 FORMALIZATION OF THE PROBLEM

2.1 SOFT-MARGIN SVM

Following the structure proposed by the first paper (Jaggi, An Equivalence between the Lasso and Support Vector Machines, 2014), a SVM can be understood as a tool used for two-class classification tasks which tries to separate the data in two groups by means of a hyperplane that maximizes the distance from the decision boundary to the closest of the data points (the margin). However, the soft-margin SVM variant allows some misclassification errors, although it attempts to minimize them. Mathematically, this problem could be modeled as follows:

$$\min_{\bar{w}\in\mathbb{R}^d,\ \rho\in\mathbb{R},\ \xi\in\mathbb{R}^n} \quad \frac{1}{2}||\bar{w}||^2 - \rho + \frac{C}{2}\sum_i \xi_i^2$$

$$s.t. \quad y_i \cdot \bar{w}^T X_i \geq \rho - \xi_i \quad \forall\, i \in [1 \dots n].$$

Where:

- $\bar{w}$ is the vector that characterizes the hyperplane.
- ρ is the margin parameter. Usually, this value is fixed to one, but in this case, it is convenient to treat it as another parameter instead.
- ξ_i is a slack variable added for each data point and corresponds to the possibility to make classification mistakes with a penalization to the objective function in the case that the point violates the margin.

- C is the regularization parameter that controls the importance that the misclassification errors have in the model.
- y_i is the label of each data point. It could be-1 or 1 (the two classes).
- X_i is the vector of features of each data point.
- n is the number of data points.
- d is the number of features.

it is important to highlight that this formulation is made for a soft-margin SVM with squared loss (ℓ_2-loss) and without an offset term. However, following the guidelines established in the paper, in my implementation the offset term was included by adding an extra feature at the end of each feature vector with a fixed value of 1. This allows the resulting hyperplane to have flexibility and not be constrained to pass through the origin. Also, with respect to the regularization parameter, I used $C = 1$ in my implementation.

2.2 THE DUAL FORMULATION FOR A SOFT-MARGIN SVM

Still following the reasoning of the paper (Jaggi, 2014), this soft-margin SVM problem can be reformulated to obtain a structure that is convenient to handle from an optimization point of view. This process implies switching from a *primal* problem (the original one) to its corresponding *dual* problem (the reformulated one).

The Lagrangian of the previous formulation (with its n constraints) is useful to obtain the *dual* problem and can be written as follows:

$$L(\overline{w}, \rho, \xi, \alpha) := \frac{1}{2}||w||^2 - \rho + \frac{C}{2}\sum\nolimits_i \xi_i^2 + \sum\nolimits_i \alpha_i(-w^T Z_i + \rho - \xi_i).$$

Where α_i is a non-negative Lagrange multiplier for each of the n constraints and $Z \in \mathbb{R}^{d \times n}$ is a data matrix constructed by stacking as columns each of the $Z_i := y_i X_i$.

Furthermore, if the *Karush–Kuhn–Tucker* optimality conditions are sought by differentiating L with respect to the original variables and after following the mathematical procedures described by the paper, the following *dual* problem is obtained:

$$\min_{\alpha} \quad \alpha^T \left(Z^T Z + \frac{1}{C} \boldsymbol{I}_n \right) \alpha$$

$$s.t. \quad \begin{matrix} \alpha \geq \mathbf{0} \\ \alpha^T \mathbf{1} = 1 \end{matrix}$$

Where $\mathbf{1}$ is an all-ones vector, $\mathbf{0}$ is an all-zeros vector and $\boldsymbol{I}_n$ is the $n \times n$ identity matrix.

An important observation is that the problem's constraints now conform a unit simplex $\triangle$, which corresponds to the following (Jaggi, 2014):

$$\triangle := \left\{ x \in \mathbb{R}^d \mid x \geq \mathbf{0}\,, \sum\nolimits_{\boldsymbol{i}} x_i = 1 \right\} = conv(\{\mathbf{1}_i, i = 1, \dots, d\}).$$

it should be noted that the unit simplex is the *convex hull* of the canonical basis of $\mathbb{R}^d$. This property is quite convenient from an optimization perspective, as will be described later on. With this in mind and in accordance with the paper, finally, the *dual* problem can be seen as an optimization problem of the form

$$\min_{\alpha \in \triangle} \quad ||A\alpha||^2.$$

where

$$A := \begin{pmatrix} Z \\ \frac{1}{\sqrt{C}} \boldsymbol{I}_n \end{pmatrix} \in \mathbb{R}^{(d+n)\times n}.$$

As will be further described during the rest of the document, this last formulation of the *dual* problem allows us to solve it in a convenient way because of its structure. For this reason, this is the optimization task that will be solved.

Nonetheless, after finding the solution for the *dual* problem, it is still necessary to obtain the primal solution. By considering the optimality conditions described in the paper, it is possible to get the *primal* solution w from a *dual* optimal α as $w = Z\alpha$.

2.3 OBJECTIVE FUNCTION AND GRADIENT

Now that the optimization problem is defined, we see that our objective function is quadratic and can be written in the following form:

$$f(\alpha) = ||A\alpha||^2 = (A\alpha)^T(A\alpha) = \alpha^T A^T A\, \alpha.$$

Therefore, the function's gradient, which is used repeatedly for the implementation of the algorithms, can be obtained as

$$\nabla f(\alpha) = 2A^T A\alpha.$$

Lastly, the Hessian matrix corresponds to $2A^T A$.

3 ALGORITHMS AND IMPLEMENTATION DETAILS

3.1 GENERAL CONSIDERATIONS

The detailed description and some theoretical properties of the original Frank-Wolfe algorithm are found in *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization* (Jaggi, 2013), while the explanation of the Away-steps Frank-Wolfe and the Pairwise Frank Wolfe variants, along with their characteristics, are depicted in *On the Global Linear Convergence of Frank-Wolfe Optimization Variants* (Lacoste-Julien & Jaggi, 2015).

However, it is relevant to mention some details that are important for the implementation of this project. In the first place, all algorithms used the same starting point: a vector of all-zeros, except for the first coordinate, which had a value of 1. In addition, there are some more aspects that are common to all algorithms that will now be further described.

3.1.1 The Frank-Wolfe problem

Firstly, as mentioned before, the *dual* problem's structure can be exploited, given that we are dealing with a constrained optimization task over a unit simplex.

This applies for all the algorithms considered. The reason is that, in all cases, to do the updates and to check the stopping conditions at each iteration k it is necessary to solve the so-called *Frank-Wolfe problem*. This problem, adapted to the notation I have been using to go along with the *dual* formulation, can be written as follows (Rinaldi, 2023):

$$\min_{\alpha \in \triangle} \nabla f(\alpha_k)^T(\alpha - \alpha_k)$$

while the optimal $\alpha \in \triangle$ that solves the problem will be noted as $\hat{\alpha}$.

The convenience that the structure of the problem provides is that optimizing over a simplex has a straightforward methodology that is not so costly to implement. To explain this in more detail, we first observe that, since at each iteration α_k and $\nabla f(\alpha_k)$ are fixed, they become vectors of constants. Using this fact and the definition of the unit simplex, the *Frank-Wolfe problem* can be seen as a *linear programming problem* with the following form (Rinaldi, 2023):

$$\min_{\alpha} \nabla f(\alpha_k)^T \alpha$$

$$s.t. \quad \begin{matrix} \alpha \geq \mathbf{0} \\ \alpha^T \mathbf{1} = 1 \end{matrix}$$

Using the *Fundamental Theorem of Linear Programming* we can conclude that, if the problem has a solution, the optimal point is a vertex of the polytope that defines the feasible set, which is the unit simplex in this case. Therefore, to obtain the optimal solution $\hat{\alpha} \in \triangle$ it suffices to compute $\nabla f(\alpha_k)^T \alpha$ for all α that are vertices and then selecting the minimum. This is equivalent to just calculating the whole gradient and finding the coordinate corresponding to the minimum component. That way it is possible to find the vertex corresponding to $\hat{\alpha}_k$ at each iteration with a computational cost of $O(d)$.

3.1.2 The duality gap and the stopping condition

Another aspect to consider is that, since we are dealing with constrained optimization problems, only checking the optimality conditions with respect to the norm of the gradient does not work in this case. Alternatively, what is done for the Frank-Wolfe algorithms (Jaggi, 2013) is to work with the *duality gap*.

To elaborate on this, in general for any constrained convex optimization problem of the form $\min_{x \in D} f(x)$ and a given point $x \in D$, it is possible to define a simple dual function

$$w(x) = \min_{s \in D} f(x) + \nabla f(x)^T (s - x).$$

Furthermore, the *duality gap* can be defined as follows (Rinaldi, 2023):

$$g(x) = f(x) - w(x) = -\min_{s \in D} \nabla f(x)^T (s - x) = \max_{s \in D} \nabla f(x)^T (x - s)$$

Because of the convexity of the function, for all pairs $x, z \in D$ it holds that $f(x) \geq w(z)$. Therefore, it also holds that $f(x^*) \geq w(x)$, where $f(x^*)$ is the minimum value of f. Finally, if we subtract $f(x)$ on both sides of the inequality and flip the signs we get that

$$f(x) - f(x^*) \leq f(x) - w(x) = g(x),$$

so the *duality gap* can be interpreted as a certificate for the approximation to the optimal value (Jaggi, 2013), which is cheap to obtain once the *Frank-Wolfe* problem has been solved. Adapting these results to this project and the notation I have been using, the algorithms can be stopped when the gap is small enough:

$$g(\alpha_k) = -\nabla f(\alpha_k)^T (\hat{\alpha} - \alpha_k) \leq \varepsilon$$

In my implementation, ε was fixed at 10^{-3}. However, all algorithms were programmed to stop after reaching a fixed number of iterations or a specific time limit. For this project, the maximum number of iterations was set to 50 000, while the maximum time allowed was 300 seconds.

3.2 THE FRANK-WOLFE IMPLEMENTATION

The algorithm was implemented following the description provided by Jaggi (Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization, 2013). The only additional aspect that is left to be described in detail is the updating procedure of the α vector when moving towards the optimum. However, I already mentioned this process is closely related to the *Frank-Wolfe problem*. This is because the update direction d_k involves the solution $\hat{\alpha}_k$ of the problem at each iteration:

$$\alpha_{k+1} = \alpha_k + \beta_k d_k$$

Where $d_k = \hat{\alpha}_k - \alpha_k$.

On the other hand, β_k is the step-size of the update which, for the original Frank-Wolfe implementation of this project, was carried out in two ways (following the guidelines of Jaggi (2013)):

- With a diminishing step-size:

$$\beta_k = \frac{2}{k+2}$$

- With a line-search using the Armijo Rule. According to the lecture notes of the course (Rinaldi, 2023), this is a procedure that, for a starting step-size Δ_k and two fixed parameters $\delta \in (0,1)$ and $\gamma \in (0, 1/2)$ the steps $\beta = \delta^m \Delta_k$, for $m = 0, 1, 2, \dots$ are attempted until the following condition is satisfied:

$$f(\alpha_k + \beta d_k) \leq f(\alpha_k) + \gamma\beta \nabla f(\alpha_k)^T d_k,$$

  Then, we can set $\beta_k = \beta$.

The first approach simply shrinks the step-size with each iteration. Recalling that, for the constrained optimization problem we are dealing with, the optimality is not pursued by only reducing the gradient's norm, shrinking the step-size allows that the update is eventually small. In contrast, the Armijo Rule tries to guarantee a decrease of the objective function with respect to a reference model established by the parameters used. Thus, the difference between both strategies is that the diminishing step-size only shrinks the updates as the iterations increase, while the Armijo Rule attempts to obtain a *better* reduction at each step.

In my implementation, the parameters used were $\delta = 0.5$ and $\gamma = 10^{-2}$, while the starting step-size at each iteration was equal to the one that would be obtained with the diminishing step-size:

$$\Delta_k = \frac{2}{k+2}$$

Once the implementation details have been described, it is worth mentioning that, according to Jaggi (2013), the theoretical convergence rate is

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

where C_f is the *curvature constant* of the function, which is related to the *Lipschitz constant* C as follows:

> If ∇f is L-Lipschitz continuous on D w.r.t. some arbitrary chosen norm $\|\cdot\|$, then $C_f \leq diam_{\|\cdot\|}(D)^2 L$ where $diam_{\|\cdot\|}(\cdot)$ denotes the $\|\cdot\|$-diameter (Jaggi, 2013).

3.3 THE AWAY-STEPS FRANK-WOLFE IMPLEMENTATION

This variant and the Pairwise Frank-Wolfe are closely related to the original method. The main differences lie in the way that the direction of the update is computed. For this variant, to improve the algorithm's performance by avoiding the zig-zag effect that the original Frank-Wolfe carries (Lacoste-Julien & Jaggi, 2015) it is necessary to choose between two directions: d_k^{FW} and d_k^A.

The first one, d_k^{FW}, is equal to the direction was described in the previous section (the one obtained by solving the *Frank-Wolfe problem*).

In contrast, according to Lacoste-Julien and Jaggi (2015), the other direction for the notation used in this project is $\mathrm{d}_k^A = \alpha_k - \hat{\alpha}_k^A$, where

$$\hat{\alpha}_k^A = \arg\max_{\alpha \in S_k} \nabla f(\alpha_k)^T (\alpha - \alpha_k)$$

and, in this case, S_k is a subset of the vertices of the unit simplex $\triangle$ that characterizes iteration *k.* What this means is that it is possible to write $\alpha_k = \sum_{\alpha \in S_k} \pi_k^\alpha \alpha$ where S_k is "the *active set*" (Lacoste-Julien & Jaggi, 2015), which, for this project, refers to the values in α_r for $r < k$ that have a non-zero weight $\pi_k^\alpha > 0$. In other words, S_k is the set of currently used vertices at each iteration.

Again, because of the problem's structure, a similar logic to the one used to solve the *Frank-Wolfe problem* over the unit simplex $\triangle$ can be used to solve this new problem and find the *Away-step* direction d_k^A. What is done is to calculate the maximum component of the gradient among the non-zero coordinates of α. That procedure allows us to find the vertex corresponding to $\hat{\alpha}_k^A$ and, with it, the corresponding d_k^A.

Then, after the two directions are found, they are compared in terms of the dot product with the gradient following procedure described by Lacoste-Julien and Jaggi (2015). If $\nabla f(\alpha_k)^T \mathrm{d}_k^{FW} \leq \nabla f(\alpha_k)^T \mathrm{d}_k^A$, the Frank-Wolfe direction is chosen ($\mathrm{d_k} = \mathrm{d}_k^{FW}$) and the maximum possible step-size for the iterate $\bar{\beta}$ is set at 1. On the contrary, $\mathrm{d_k} = \mathrm{d}_k^A$ and $\bar{\beta}$ is set to the maximum value that allows the next update to stay within the unit simplex $\triangle$. For the formulation that we are dealing with, $\bar{\beta} = \pi_{\hat{\alpha}_k^A}/(1 - \pi_{\hat{\alpha}_k^A})$.

Once the direction and the maximum feasible step-size are chosen, the updates can be made with the same formula that is used for the original Frank-Wolfe:

$$\alpha_{k+1} = \alpha_k + \beta_\mathrm{k} d_k$$

Where $\beta_\mathrm{k} \in (0, \bar{\beta}]$. For my implementation of the Away-steps Frank-Wolfe, β_k was computed by means of a line search using the Armijo Rule with the same parameters as the previous algorithm and the starting step-size $\Delta_k = \bar{\beta}$.

3.4 THE PAIRWISE FRANK-WOLFE IMPLEMENTATION

This variant is quite similar to the Away-steps Frank-Wolfe because it also requires obtaining d_k^{FW} and d_k^A to avoid the zig-zag effect of the original Frank-Wolfe. However, the difference lies in the way that the final direction is determined (Lacoste-Julien & Jaggi, 2015).

In this case, the final direction of each iterate can be computed as $d_k = \mathrm{d}_k^{FW} + \mathrm{d}_k^{A}$. Also, for this variant, the maximum feasible step-size slightly differs and is set to $\bar{\beta} = \pi_{\hat{\alpha}_k^A}$. However, the formula for the updates is the same as the one used for the Away-steps Frank-Wolfe.

Regarding the implementation of the algorithm for this project, again, the line-search was performed using the Armijo Rule with the same parameters that were used for the other two algorithms and the starting step-size was again set as $\Delta_k = \bar{\beta}$.

4 DATASETS

To test the behavior of the algorithms, three datasets were used for this project: *Breast cancer*, *Phishing* and *Skin*. Since the goal is to train soft-margin SVM by using the optimization methods described above, all the chosen datasets had a binary response (only two output labels) that were converted into -1 and 1. The complete description of each dataset is found in their corresponding repository[1], however, for the sake of clarity, it is worth noting that the *Breast cancer* datasets contains samples that could be either benign or malignant, the *Phishing* examples could be either phishy or legitimate and the *Skin* data points consist of images (characterized by their RGB) that could be skin or non-skin samples.

The table below shows the number of samples and the number of features (before adding the feature corresponding to the offset term) for each dataset:

Table 1. Number of simples and number of features of the datasets

Dataset	*Number of samples*	*Number of features*
Breast cancer	683	10
Phishing	11 055	68
Skin	245 057	3

However, not all samples were used for training. Although the focus of this project is on testing the optimization performance rather than the classification results, it is still relevant to verify the accuracy of the classification task implementation. To achieve this, the data was divided into training sets and test sets (with a random selection of 80% of the data for training). The sizes of the resulting datasets are shown below:

Table 2. Training set and test set sizes

Dataset	*Training set samples*	*Test set samples*
Breast cancer	546	137
Phishing	8 844	2 211
Skin	196 045	49 012

After adding the offset term, when organizing the data to solve the dual problem, as explained in section 2.2, the resulting A matrix dimensions are 556×546 for *Breast cancer,* $8\,913 \times 8\,844$ for *Phishing* and $196\,049 \times 196\,045$ for *Skin.*

[1] All datasets are available in https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

5 RESULTS AND DISCUSSION

5.1 CLASSIFICATION RESULTS

While the main objective is to evaluate the optimization side of the soft-margin SVM training procedure by using the Frank-Wolfe method and its variants, the classification results are also shown for the purpose of verifying that the implementation was correct. To do this, after solving the optimization *dual* problem and returning to the *primal*, as described in section 2.2, each data point of the test set was classified by checking the sign of $w^T X_j \; \forall \; j \in [1 \dots n_{test}]$. After this, it was possible to obtain the overall classification accuracy for each dataset and each algorithm (Frank-Wolfe with diminishing step-size, Frank-Wolfe with Armijo Rule, Away-steps Frank-Wolfe and Pairwise Frank-Wolfe). Also, as an attempt to avoid interpretation mistakes because of imbalances between output classes, the *F1 score* (which incorporates both precision and recall) was also computed.

The classification results for the *Breast cancer* dataset (the smallest one) are almost identical[2] for all methods considered:

Table 3. Classification results – Breast cancer

Method	*Accuracy*	*F1 score*
Frank-Wolfe with DS	0.978	0.982
Frank-Wolfe with AR	0.978	0.982
Away-steps Frank-Wolfe	0.978	0.982
Pairwise Frank-Wolfe	0.978	0.982

Likewise, even though the *Phishing* dataset shows different results among methods, the differences are small and, in all cases, both the overall accuracy and the *F1 score* surpass 0.9:

Table 4. Classification results – Phishing

Method	*Accuracy*	*F1 score*
Frank-Wolfe with DS	0.914	0.918
Frank-Wolfe with AR	0.938	0.944
Away-steps Frank-Wolfe	0.943	0.948
Pairwise Frank-Wolfe	0.934	0.941

Lastly, the largest dataset (*skin*) shows the most contrasting results from the classification point of view:

Table 5. Classification results – Skin

Method	*Accuracy*	*F1 score*
Frank-Wolfe with DS	0.792	0.884
Frank-Wolfe with AR	0.792	0.884
Away-steps Frank-Wolfe	0.810	0.864
Pairwise Frank-Wolfe	0.876	0.921

We can notice a more dramatic difference between the best and worst performing methods. The method that consistently outperforms the others with respect to both accuracy and the *F1 score* is the Pairwise Frank-Wolfe, which tries to correct the zig-zag effect of the original Frank-Wolfe by adding the two computed directions together. Perhaps, since the *Skin* file is much larger than the other datasets, the efficiency of the algorithms becomes a key aspect when training the soft-margin SVM. Since all algorithms are using the same time and

[2] All classification results are rounded to three decimals.

iterations limitations, if the methods do not reach the optimal stopping condition (i.e., when the gap remains greater than ε), a more efficient optimization algorithm could yield more precise classification results as the solution gets closer to the optimal point. The efficiency of the algorithms will become clearer in the next section.

5.2 Optimization results

From the optimization point of view, following the lines in the Lacoste-Julien and Jaggi paper (On the Global Linear Convergence of Frank-Wolfe Optimization Variants, 2015), the performance was measured by plotting the iterations versus the *duality gap* for each dataset. Additionally, I included a time versus *duality gap* plot for each case.

The plots of the *Breast cancer* dataset are presented below. In addition, I included a table that summarizes the final value of the *duality gap* and the objective function, along with the corresponding time and iterations required to reach those values:

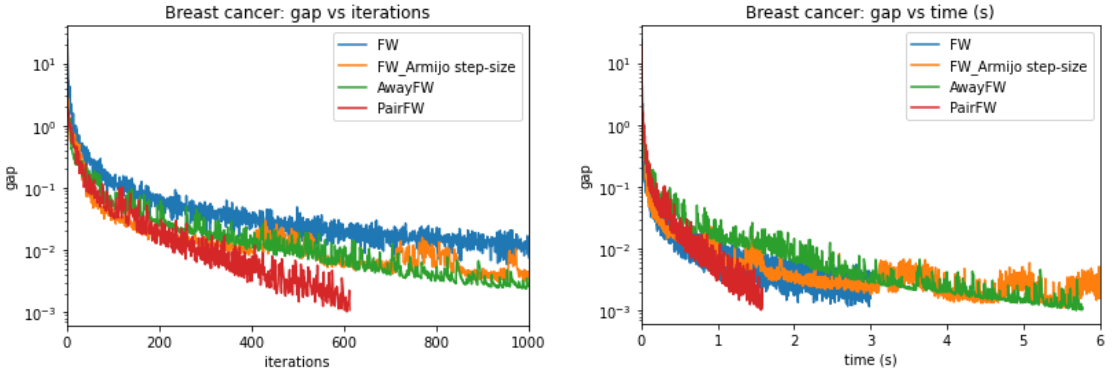


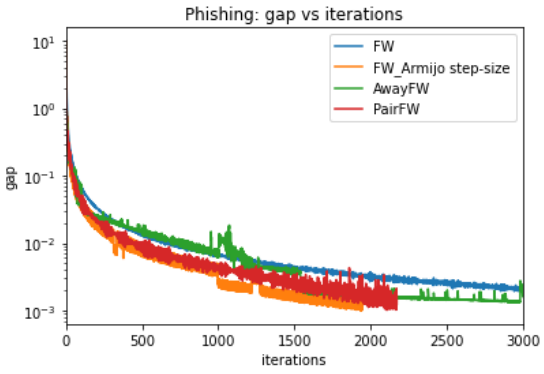
Table 6. Breast cancer: final duality gap and objective function

Method	*Duality gap*	*Objective function*	*Iterations*	*Time (s)*
Frank-Wolfe with DS	0.00099	0.01955	4 559	2.99
Frank-Wolfe with AR	0.00098	0.02003	3 727	6.80
Away-steps Frank-Wolfe	0.00096	0.01956	1 704	5.77
Pairwise Frank-Wolfe	0.00088	0.01955	613	1.58

We can observe that all methods reached convergence (they obtained a *duality gap* $\leq \varepsilon$). Moreover, the Pairwise Frank-Wolfe was the fastest one and the one that required the least amount of iterations.

With respect to the iterations, the Frank-Wolfe with diminishing step-size and the Pairwise Frank-Wolfe are the methods that show the smallest and the greatest reduction per iteration, respectively. However, there is not a clear pattern among the other methods. Regarding the time, there are not so graphically obvious patterns, although we can notice that the two previously mentioned methods stopped before the other two. In both cases (iterations and time) there is a steep reduction at the beginning and then the lines flatten and show a less dramatic reduction.

Furthermore, the *Phishing* dataset yielded the following results:



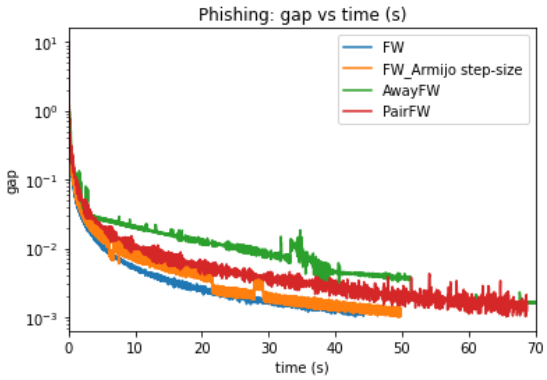


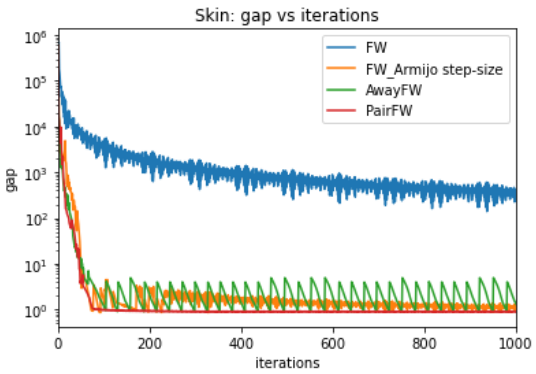
Table 7. Phishing: final duality gap and objective function

Method	*Gap*	*Objective function*	*Iterations*	*Time (s)*
Frank-Wolfe with DS	0.00099	0.00063	5 049	44.24
Frank-Wolfe with AR	0.00099	0.00079	1 948	49.81
Away-steps Frank-Wolfe	0.00100	0.00080	4 889	226.73
Pairwise Frank-Wolfe	0.00097	0.00077	2 174	68.77

In this case, the iterations vs *duality gap* plot shows a similar behavior as before, although the Pairwise Frank-Wolfe is not as efficient as in the previous case. On the other hand, the time vs *duality gap* plot indicates that the original Frank-Wolfe (with both step-size approaches) stopped before the implemented variants. Additionally, even though all methods reached convergence again, what stands out for the *Phishing* data is that the Frank-Wolfe with Armijo Rule was surprisingly efficient.

A different aspect to remark is that, like in the *Breast cancer* case, the direction obtained with the Pairwise Frank-Wolfe algorithm seems to result in better performance compared to the Away-steps Frank-Wolfe variant.

Finally, the results for *Skin,* the largest dataset, are the following:



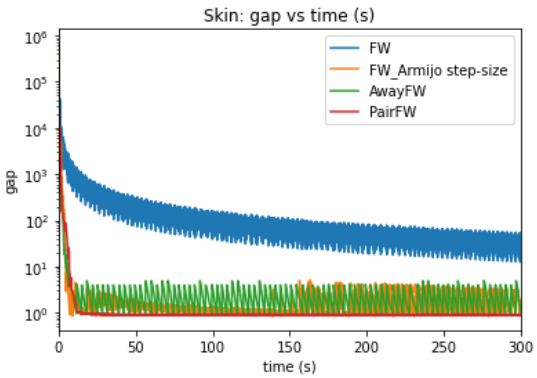


Table 8. Skin: final duality gap and objective function

Method	*Gap*	*Objective function*	*Iterations*	*Time (s)*
Frank-Wolfe with DS	15.24507	0.05783	10 433	300
Frank-Wolfe with AR	0.89739	0.38164	3 039	300
Away-steps Frank-Wolfe	1.10145	0.40814	2 439	300
Pairwise Frank-Wolfe	0.91222	0.40437	2 405	300

In both plots it is evident that the Frank-Wolfe with diminishing step-size is the least efficient method in terms of iterations and time. In addition, unlike the other methods, the Pairwise Frank-Wolfe shows a less *bumpy* optimization process, in the sense that the decrease in the *duality gap* appears to be more consistent throughout the iterations (in

contrast to the other methods, were from one step to the other the *duality gap* sometimes increased significantly). Also, even though none of the methods reached full convergence for the established time budget, the algorithms that resulted in the smallest duality gaps were the Frank-Wolfe with Armijo Rule and the Pairwise Frank-Wolfe. These characteristics could explain why the Pairwise Frank-Wolfe was the approach that yielded the best classification results, as explained in the previous section.

In conclusion, considering all the results for all the datasets, implementing the Armijo Rule approach, instead of using the diminishing step-size for the Frank-Wolfe results in a better performing algorithm. This makes sense as the Armijo Rule tries to guarantee a certain reduction at each update, while the diminishing step-size approach simply shrinks the size of the step.

Also, it appears that in this experiment the Frank-Wolfe variants do improve the original method, as the theory suggests (given that the variants are designed to avoid the zig-zag effect of the original algorithm). Moreover, the Pairwise Frank-Wolfe showed a more consistent performance than the Away-steps Frank-Wolfe in terms of iterations and time efficiency. Therefore, it can be concluded that, in this scenario, the approach to compute the Pairwise Frank-Wolfe updates yielded better empirical results than the other variant.

6 REFERENCES

Chang, C.-C., & Lin, C.-J. (2011). *LIBSVM : a library for support vector machines.* Obtenido de ACM Transactions on Intelligent Systems and Technology: http://www.csie.ntu.edu.tw/~cjlin/libsvm

CSIE @ NTU. (s.f.). *LIBSVM Data: Classification, Regression, and Multi-label.* Obtenido de https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. *JMLR: Workshop and Conference Proceedings*, 28.

Jaggi, M. (2014). An Equivalence between the Lasso and Support Vector Machines. *arXiv:1303.1152v2 [cs.LG] 25 Apr 2014*.

Lacoste-Julien, S., & Jaggi, M. (2015). On the Global Linear Convergence of Frank-Wolfe Optimization Variants. *arXiv:1511.05932v1 [math.OC] 18 Nov 2015*.

Rinaldi, F. (2023). Optimization for data science [Lecture notes]. University of Padova.