Minimum Enclosing Ball for Anomaly Detection

Francesco Sartori Matteo Pernini Irene Caria Marco Furlan

Abstract

In this paper we analyze the problem of Anomaly Detection. We address the task using two algorithms that compute $(1+\epsilon)$-approximation to Minimum Enclosing Ball (MEB) and since they are closely related to the Frank-Wolfe (FW) algorithm, we first focus on the Away-step FW and the Pairwise FW algorithms. Then we choose 2 different datasets: one with more instances than attributes, and one viceversa. On these, we test the two MEB algorithms both on performance and on their capability in finding anomalies.

1 Introduction

The Frank-Wolfe Algorithm (also known as *conditional gradient* or *reduce gradient*) is one of the earliest first-order method for constrained convex optimization. It is nowadays widely used in plenty of different contexts, including adversarial attacks, SVM training, minimum enclosing ball problems and others. It was initially proposed in the 50's by Marguerite Frank and Philip Wolfe to solve quadratic programming problems. In this work we provide a general description of the main results and clarify some relevant variants of the Frank-Wolfe algorithm that have been successfully applied in practice. In the second part of the paper we show how to apply this tools to the specific problem of computing a $(1+\epsilon)$ approximation to the radius of the minimum enclosing ball (MEB) of a set of points $\mathcal{A}$.

The general problem we address here has the following form:

$$\min_{x \in \mathcal{M}} f(x), \quad \mathcal{M} = conv(\mathcal{A})$$

where $\mathcal{M}$ can be thought as the convex hull of a finite set of vectors $\mathcal{A} \subseteq \mathbb{R}^d$, that we called atoms. The objective function $f(x)$ is assumed to be μ-strongly convex and with L-Lipschitz continuous gradient over $\mathcal{M}$.

Algorithm 1 Frank-Wolfe algorithm (1956)

```
1: Let $x^{(0)} \in \mathcal{A}$
2: for $t = 0...T$ do
3:     $\hat{x}_t := \arg\min_{x\in\mathcal{A}} \langle (x - x^{(t)}), \nabla f(x^{(t)}) \rangle$
4:     Update $x^{(t+1)} := x^{(t)} + \alpha^{(t)}(\hat{x}^{(t)} - x(t))$ with $\alpha^{(t)} \in (0, 1]$
5: end for
```

At each step of the for-loop, Algorithm 1 defines a feasible direction (that is called a *Frank-Wolfe direction*) towards the search atom s_t and moves along that direction by minimizing the linear approximation of the objective function over $\mathcal{M}$. The main idea is then to solve this linearized problem instead of the original non-linear one, which is harder to handle. The minimization happens in line 3 of the algorithm, where a linear minimization oracle of the following form is required:

$$LMO_{\mathcal{A}}(r) \in \arg\min_{x\in\mathcal{A}} \langle r, x \rangle$$

Line 4 of Algorithm 1 shows how to update the iterate. To find the best value for the step-size α^t a fix or pre-defined value can be used, but in the following implementations a line-search is used. One of the reasons why Frank-Wolfe algorithm gained so much popularity in recent years is the sparsity of its iterates, that can be thought as a sparse convex combination of the atoms of the domain; in particular iteration t will be a combination of at most $(t + 1)$ atoms, so that we can write $x^{(t)} = \sum_{v\in\mathcal{S}^{(t)}} \alpha_v^{(t)} v$, where $\mathcal{S}^{(t)}$ is the *active set* at iteration t containing atoms discovered in the previous iterations and such that their weight $\alpha_r^{(t)}$ is greater then zero for $r \leq t$.

2 Improved Variants of the Frank-Wolfe Algorithm

In terms of convergence, and being x^* an optimal solution which lies on the boundary of $\mathcal{M}$, the iterates of Algorithm 1 satisfy $f(x^{(t)}) - f(x^*) \leq O(\frac{1}{t})$, which means that the convergence rate is sublinear. The main reason for that is the use of the $LMO_{\mathcal{A}}$ previously defined, that leads to the so called *zig-zagging phenomenon*: as the iterates come closer to the face of the polytope containing x^*, the feasible directions found by Algorithm 1 start to zig-zag between the two vertices where the optimal solution lies, dramatically slowing down the convergence rate. Therefore some variants of the original algorithm were studied to face this problem and improve the convergence rate. In the following the analysis will be focused on two of them: the Away-Step Frank-Wolfe and the Pairwise Frank-Wolfe algorithm. Both of these algorithms, as well as other Frank-Wolfe variants, show, in the strongly convex case, a linear convergence rate, which is proved in [1]. The main result in this regard is the following theorem:

Theorem 1. *Suppose that f has L-Lipschitz gradient and is μ-strongly convex over $\mathcal{M} = conv(\mathcal{A})$. Let $M = diam(\mathcal{M})$ and $\delta = PWidth(\mathcal{A})$. Then the suboptimality h_t of the iterates of all the four variants of the FW algorithm decreases geometrically at each step that is not a drop step nor a swap step (i.e. when $\gamma_t < \gamma_{max}$, called a good step), that is:*

$$h_{t+1} \leq (1-\rho)h_t, \qquad where \quad \rho := \frac{\mu}{4L}\frac{\delta}{M}^2 \tag{1}$$

Let $k(t)$ be the number of good steps up to iteration t. We have $k(t) = t$ for Fully Corrective Frank-Wolfe (FCFW); $k(t) \geq \frac{t}{2}$ for Min Norm Point (MNP) and Away Step Frank-Wolfe (AFW); and $k(t) \geq \frac{t}{3|\mathcal{A}|!+1}$ for Pairwise Frank-Wolfe (PFW). This yields a global linear convergence rate of $h_t \leq h_0 \exp(-\rho k(t))$ for all variants. If $\mu = 0$ (general convex), then $h_t = O(\frac{1}{k(t)})$ instead.

Note that $FCFW$ and MNP are two other variants of the original Frank-Wolfe, that are not analyzed in this paper. For the complete proof we link to paper [1], but, even if it cannot be discussed in detail here, we want to give at least a general intuition. Since by assumption the objective function is strongly convex with L-Lipschitz gradient, the LCG inequality can be used:

$$f(x^{(t+1)}) \leq f(x^t + \gamma d_t) \leq f(x^{(t)}) + \gamma\langle \nabla f(x^{(t)}), d_t\rangle + \frac{\gamma^2}{2}L\|d_t\|^2 \quad \forall \gamma \in [0, \gamma_{max}] \tag{2}$$

We now define the suboptimality error $h_t := f(x^{(t)}) - f(x^*)$ and let $r_t := -\nabla f(x^{(t)})$. We also suppose for now that:

$$\gamma_{max} \geq \gamma_t^* := \frac{\langle r_t, d_t\rangle}{L\|d_t\|^2} \tag{3}$$

If we set $\gamma = \gamma_t^*$ and subtract $f(x^*)$ on both sides we get:

$$h_t - h_{t+1} \geq \frac{\langle r_t, d_t\rangle^2}{2L\|d_t\|^2} = \frac{1}{2L}\langle r_t, \hat{d_t}\rangle^2 \tag{4}$$

where $\hat{d_t} := \frac{d_t}{\|d_t\|}$. We define now the error vector $e_t := x^* - x^{(t)}$ and by μ-strong convexity we have:

$$f(x^{(t)} + \gamma e_t) \geq f(x^{(t)}) + \gamma\langle \nabla f(x^{(t)}), e_t\rangle + \frac{\gamma^2}{2}\mu\|e_t\|^2 \quad \forall \gamma \in [0,1] \tag{5}$$

The right hand side of the inequality (5) is lower bounded by $\gamma := \frac{\langle r_t, e_t\rangle^2}{\mu\|e_t\|^2}$. This means that we can apply on the left side any value of $\gamma \in [0,1]$ and still end up with an inequality that holds true. In particular we can use $\gamma = 1$, so to get:

$$h_t \leq \frac{\langle r_t, e_t\rangle^2}{2\mu\|e_t\|^2} = \frac{\langle r_t, \hat{e_t}\rangle^2}{2\mu} \tag{6}$$

Finally from a simple comparison between (4) and (6) we get:

$$h_t - h_{t+1} \geq \frac{\mu}{L} \frac{\langle r_t, \hat{d}_t \rangle^2}{\langle r_t, \hat{e}_t \rangle^2} h_t \tag{7}$$

To get a linear convergence rate we would like to lower bound by a positive constant the term in front of h_t. This can be done by comparing the scalar product between the negative gradient and the update direction with the pairwise Frank-Wolfe direction. If we take a look at the condition on lines 6-9 in Algorithm 2, d_t is always chosen so to maximize the scalar product with the negative gradient. Then we have:

$$2\langle r_t, d_t \rangle \geq \langle r_t, d_t^{FW} \rangle + \langle r_t, d_t^{A} \rangle = \langle r_t, d_t^{FW} + d_t^{A} \rangle = \langle r_t, d_t^{PFW} \rangle \tag{8}$$

where d_t^{PFW} is the pairwise direction at iteration t. We then get:

$$\langle r_t, d_t \rangle \geq \frac{\langle r_t, d_t^{PFW} \rangle}{2} \tag{9}$$

This inequality is crucial, because now we can use the result that the scalar product with the pairwise direction is lower bounded by the *Pyramidal Width*, a quantity that depends only on the geometry of $\mathcal{M}$. In particular we use the following theorem:

Theorem 2. *Let $x \in \mathcal{M} = conv(\mathcal{A})$ be a suboptimal point and $\mathcal{S}$ be an active set for x. Let x^* be an optimal point and corresponding error direction $\hat{e} = \frac{(x^* - x)}{\|x^* - x\|}$, and negative gradient $r := -\nabla f(x)$ (and so $\langle r, \hat{e} \rangle > 0$). Let $d = s - v$ be the Pairwise FW direction obtained over $\mathcal{A}$ and $\mathcal{S}$ with negative gradient r. Then*

$$\frac{\langle r, d \rangle}{\langle r, \hat{e} \rangle} \geq PWidth(\mathcal{A}) \tag{10}$$

This result can be used to prove the linear convergence rate of the Away-Step and the Pairwise algorithm. In the following paragraphs we describe in more details the two of them and illustrate their performance in a practical experiment.

2.1 Away-Step Frank-Wolf algorithm

The first variant makes use of *away-steps*, which adds the possibility in each iteration not only to move towards a specific atom, but also to move *away* from an active atom in $\mathcal{S}^{(t)}$, which is chosen to be the atom of maximum ascent in its direction. This method improves also the sparsity of the iterates and under some conditions (strong-convexity) it drives to a faster linear convergence rate. Algorithm 2 must keep track of the active set $\mathcal{S}^{(t)}$ as well, because the search for the worst atom happens in the active set, which contains at the beginning only the starting atom $x^{(0)}$ and then is updated at each iteration, by adding or removing elements. Line 3 of Algorithm 2 performs the same search for the Frank-Wolfe direction as in the original algorithm, by using $LMO_{\mathcal{A}}(\nabla f(x^{(t)}))$.

In line 4 an away-direction d_t^A is defined in such a way that the atom v_t maximizes the scalar product with the gradient, that is to say the potential descent given by $\langle -\nabla f(x^{(t)}), d_t^A \rangle = g_t^A$. An important remark here is that the active set is typically small, smaller anyway than $\mathcal{A}$, which means that the search is easier than the previous one based on $LMO_{\mathcal{A}}$. Then the duality gap g_t^{FW}, that is the scalar product between the negative gradient and the FW direction, is used as stopping condition, since its role of upper bound on the suboptimality error is well known from the theory (and this stopping condition is the same as in Algorithm 1). A simple comparison between g_t^{FW} and g_t^A in line 6 allows the algorithm to choose if a Frank-Wolfe step or an Away step is required. The value of γ_{max} in lines 7 and 9 is chosen in such a way that the updated iteration in line 11 stays in $\mathcal{M}$. This choice of γ_{max} is, on closer inspection, conservative and in some specific cases it is possible to have a bigger step-size that is still feasible. In case the algorithm performs a Frank-Wolfe step, if $\gamma_t = 1$, then $S^{(t+1)} = \{s_t\}$, otherwise $S^{(t+1)} = S^{(t)} \cup \{s_t\}$. Furthermore $\alpha_{s_t}^{(t+1)} := (1-\gamma_t)\alpha_{s_t}^{(t)} + \gamma_t$ and $\alpha_v^{(t+1)} := (1-\gamma_t)\alpha_v^{(t)}$ for $v \in S^{(t)} \setminus \{s_t\}$. In case of an away-step, if $\gamma_t = \gamma_{max}$ we perform a *drop* step, which means that the atom v_t is removed from the active set and $S^{(t+1)} = S^{(t)} \setminus \{v_t\}$; otherwise $S^{(t+1)} = S^{(t)}$. Finally, we have $\alpha_{v_t}^{(t+1)} := (1+\gamma_t)\alpha_{v_t}^{(t)} - \gamma_t$ and $\alpha_v^{(t+1)} := (1+\gamma_t)\alpha_v^{(t)}$ for $v \in S^{(t)} \setminus \{v_t\}$.

Algorithm 2 Away-steps Frank-Wolfe algorithm: **AFW**$(x^{(0)}, \mathcal{A}, \epsilon)$

```
1: Let $x^{(0)} \in \mathcal{A}$, and $\mathcal{S}^{(0)} := \{x^{(0)}\}$;
2: for $t = 0...T$ do
3:     Let $s_t := LMO_{\mathcal{A}}(\nabla f(x^{(t)}))$ and $d_t^{FW} := s_t - x^{(t)}$;
4:     Let $v_t \in \arg\max_{v \in \mathcal{S}^{(t)}} \langle \nabla f(x^{(t)}), v \rangle$ and $d_t^A := x^{(t)} - v_t$;
5:     if $g_t^{FW} := \langle -\nabla f(x^{(t)}), d_t^{FW} \rangle \leq \epsilon$ then return $x^{(t)}$;
6:     end if
7:     if $g_t^{FW} := \langle -\nabla f(x^{(t)}), d_t^{FW} \rangle \geq \langle -\nabla f(x^{(t)}), d_t^A \rangle$ then
8:         $d_t := d_t^{FW}$ and $\gamma_{max} := 1$
9:     else
10:        $d_t := d_t^A$ and $\gamma_{max} := \frac{\alpha_{v_t}}{(1-\alpha_{v_t})}$
11:    end if
12:    Line-search: $\gamma_t \in \arg\min_{\gamma \in [0, \gamma_{max}]} f(x^{(t)} + \gamma d_t)$
13:    Update $x^{t+1} := x^{(t)} + \gamma_t d_t$
14:    Update $\mathcal{S}^{t+1} := \{v \in \mathcal{A} \quad s.t. \quad \alpha_v^{t+1} > 0\}$
15: end for
```

2.2 Pairwise Frank-Wolfe algorithm

The next variant is the so called *Pairwise Frank-Wolfe* algorithm, whose pseudocode is very similar to the *Away-step* one (it computes a Frank-Wolfe direction and an Away direction), with the main difference that it does not perform any drop step, but at each iteration it moves weight from the away atom to the Frank-Wolfe atom, while all the other α remain unchanged. According to this swap of mass we have: $\alpha_{v_t}^{(t+1)} = \alpha_{v_t}^{(t)} - \gamma$ and $\alpha_{s_t}^{(t+1)} = \alpha_{s_t}^{(t)} + \gamma$ for some $\gamma \leq \gamma_{max} := \alpha_{v_t}^{(t)}$.

Algorithm 3 Pairwise Frank-Wolfe algorithm: $\mathbf{PFW}(x^{(0)}, \mathcal{A}, \epsilon)$

```
1:  Let $x^{(0)} \in \mathcal{A}$, and $\mathcal{S}^{(0)} := \{x^{(0)}\}$;
2:  for $t = 0 ... T$ do
3:      Let $s_t := LMO_{\mathcal{A}}(\nabla f(x^{(t)}))$ and $d_t^{FW} := s_t - x^{(t)}$;
4:      Let $v_t \in \arg\max_{v \in \mathcal{S}^{(t)}} \langle \nabla f(x^{(t)}), v \rangle$ and $d_t^A := x^{(t)} - v_t$;
5:      if $g_t^{FW} := \langle -\nabla f(x^{(t)}), d_t^{FW} \rangle \leq \epsilon$ then return $x^{(t)}$;
6:      end if
7:      $d_t = d_t^{PFW} := s_t - v_t$ and $\gamma_{max} := \alpha_{v_t}$
8:      Line-search: $\gamma_t \in \arg\min_{\gamma \in [0, \gamma_{max}]} f(x^{(t)} + \gamma d_t)$
9:      Update $x^{t+1} := x^{(t)} + \gamma_t d_t$
10:     Update $\mathcal{S}^{t+1} := \{v \in \mathcal{A} \quad s.t. \quad \alpha_v^{t+1} > 0\}$
11: end for
```

2.3 Experiment for the AFW and PFW algorithms

To confront the performance of the AFW (Away-Steps Frank-Wolfe) and of the PFW (Pairwise Frank-Wolfe) we coded the following experiment shown in the paper [1].

We consider the constrained Lasso problem $\min_{x \in \mathcal{M}} f(x) = ||Ax - b||^2$, where $\mathcal{M} = 20 \cdot L_1$ is a scaled L_1-ball. We use $A \in \mathbb{R}^{200 \times 500}$ random Gaussian matrix, and $b = Ax^*$ noisy measurement, where x^* is a sparse vector with 50 entries ± 1 and 10% of additive noise. We use the linear minimization oracle as proposed in the algorithms above, that is $LMO_{\mathcal{M}}(\nabla f(x)) \in \arg\min_{y \in \mathcal{M}} \langle \nabla f(x), y \rangle$, where $\nabla f(x) = 2 \cdot A^T(Ax - b)$.

The results are shown in Figure 1. Similarly to what we can see from the same graph in the paper [1], the Pairwise Frank-Wolfe shows a faster shrinking of the duality gap, which indicates a faster convergence to the optimal point. More generally, the Pairwise Frank Wolfe algorithm often outperforms the Away-Steps Frank-Wolfe algorithm, especially in case of sparse solutions, that is solutions whose active set $S^{(t)}$ is small, or equivalently the optimal solution lies on a low-dimensional face of $\mathcal{M}$ (ref. [1]). Furthermore, the plot shows the linear convergence of the two FW variants, confirming empirically the theoretical results.

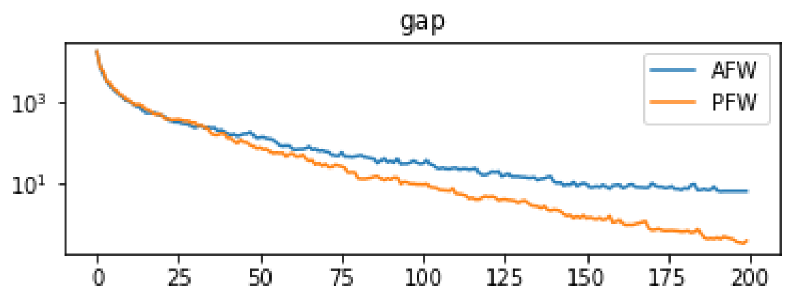

Figure 1: Duality gap g_t^{FW} (y axis) vs iterations (x axis) for the Away-Steps Frank-Wolfe and the Pairwise Frank-Wolfe

3 Minimum Enclosing Ball problem

In this section we want to focus on the Minimum Enclosing Ball problem analyzed in the paper (Yildirim [3]): given a finite set of points $\mathcal{A} = \{a^1, ..., a^m\} \subset \mathbb{R}^n$ and $\epsilon > 0$, the minimum enclosing ball MEB($\mathcal{A}$) can be computed by solving the following optimization problem:

$$\begin{aligned} &\min_{c,\rho} \quad \rho \\ &\text{subject to} \\ &\qquad \|a^i - c\| \leq \rho, \quad i = 1, ..., m. \end{aligned} \tag{11}$$

where $c \in \mathbb{R}^n$ is the center and $\rho \in \mathbb{R}$ is the radius of the ball. Before proceeding with the discussion of the problem, let us give some important definitions. The ball centered at c with radius ρ is defined as:

$$\mathcal{B}_{c,\rho} := \{x \in \mathbb{R}^n : \|x - c\| \leq \rho\}$$

Given $\epsilon > 0$, a ball $\mathcal{B}_{c,\rho}$ is said to be a $(1+\epsilon)-$approximation to MEB($\mathcal{A}$) if

$$\mathcal{A} \subset \mathcal{B}_{c,\rho}, \quad \rho \leq (1+\epsilon)\rho_{\mathcal{A}}$$

where $\mathcal{B}_{c_{\mathcal{A}},\rho_{\mathcal{A}}} = \text{MEB}(\mathcal{A})$. A subset $\chi \subseteq \mathcal{A}$ is said to be an $\epsilon-$core set of $\mathcal{A}$ if

$$\rho_\chi \leq \rho_{\mathcal{A}} \leq (1+\epsilon)\rho_\chi$$

where $\mathcal{B}_{c_\chi,\rho_\chi} = \text{MEB}(\chi)$. In our project we will consider an equivalent formulation of the problem (11) :

$$\begin{aligned} &\min_{c,\gamma} \quad \gamma \\ &\text{subject to} \\ &\qquad (a^i)^T a^i - 2(a^i)^T c + c^T c \leq \gamma, \quad i = 1, ..., m. \end{aligned} \tag{12}$$

where $\gamma := \rho^2$ and we squared the constraints. Now we can approach this problem through the Lagrange multipliers method:

$$\max_u \quad \Phi(u) := \sum_{i=1}^m u_i (a^i)^T a^i - \left(\sum_{i=1}^m u_i a^i\right)^T \left(\sum_{i=1}^m u_i a^i\right) \tag{13}$$

subject to

$$\sum_{i=1}^m u_i = 1$$
$$u_i \geq 0, \quad i = 1, ..., m.2$$

Due to a simple manipulation of the Karush-Kuhn-Tucker optimality conditions $\exists u^* \in \mathbb{R}^m$ such that

$$\gamma_{\mathcal{A}} = \Phi(u^*)$$

which implies that $u^* \in \mathbb{R}^m$ is an optimal solution of (13) and that strong duality holds between (12) and (13).

Lemma 1. *Let $\mathcal{A} = \{a^1, ..., a^m\}$. The minimum enclosing ball of $\mathcal{A}$ exists and is unique. Let $u^* \in \mathbb{R}^m$ denote the optimal solution of (13), then MEB$(\mathcal{A}) = \mathcal{B}_{c_{\mathcal{A}}, \rho_{\mathcal{A}}}$, where*

$$c_{\mathcal{A}} = \sum_{i=1}^m u_i^* a^i, \qquad \rho_{\mathcal{A}} = \sqrt{\Phi(u^*)}.$$

3.1 The first algorithm that computes a $(1+\epsilon)-$approximation to MEB($\mathcal{A}$)

Algorithm 4 The first algorithm that computes a $(1+\epsilon)-$approximation to MEB($\mathcal{A}$)

```
Require: Input set of points $\mathcal{A} = \{a^1, ..., a^m\} \subset \mathbb{R}^n$, $\epsilon > 0$
 1: $\alpha \longleftarrow \arg\max_{i=1,...,m} \|a^i - a^1\|^2$,    $\beta \longleftarrow \arg\max_{i=1,...,m} \|a^i - a^\alpha\|^2$;
 2: $u_i^0 \longleftarrow 0, \quad i = 1, ..., m$
 3: $u_\alpha^0 \longleftarrow 1/2, \quad u_\beta^0 \longleftarrow 1/2$;
 4: $\chi_0 \longleftarrow \{a^\alpha, a^\beta\}$;
 5: $c^0 \longleftarrow \sum_{i=1}^m u_i^0 a^i$;
 6: $\gamma^0 \longleftarrow \Phi(u^0)$;
 7: $\kappa \longleftarrow \arg\max_{i=1,...,m} \|a^i - c^0\|^2$;
 8: $\delta_0 \longleftarrow \left(\|a^\kappa - c^0\|^2 / \gamma^0\right) - 1$;
 9: k$\longleftarrow$ 0;
10: while $\delta_k > (1+\epsilon)^2 - 1$, do
11:     $\lambda^k \longleftarrow \delta_k / [2(1+\delta_k)]$;
12:     $k \longleftarrow k + 1$;
13:     $u^k \longleftarrow (1 - \lambda^{k-1}) u^{k-1} + \lambda^{k-1} e^\kappa$;
14:     $c^k \longleftarrow (1 - \lambda^{k-1}) c^{k-1} + \lambda^{k-1} a^\kappa$;
15:     $\chi_k \longleftarrow \chi_{k-1} \cup \{a^\kappa\}$;
16:     $\gamma^k \longleftarrow \Phi(u^k)$;
17:     $\kappa \longleftarrow \arg\max_{i=1,...,m} \|a^i - c^k\|^2$;
18:     $\delta_k \longleftarrow \left(\|a^\kappa - c^k\|^2 / \gamma^k\right) - 1$;
19: end while
20: Output $c^k, \chi_k, u^k, \sqrt{(1+\delta_k)\gamma^k}$.
```

The Algorithm 4 at first computes an initial guess for the center and the radius for the MEB($\mathcal{A}$) by selecting the furthest point $a^\alpha \in \mathcal{A}$ from $a^1 \in \mathcal{A}$ and then by selecting the furthest point $a^\beta \in \mathcal{A}$ from a^α (see step 1 to 6). Then we compute the furthest point of $\mathcal{A}$ from the center we have just obtained and we want to move our ball towards this point (see step $7-8$). Finally, we iterate this procedure until we get a $(1+\epsilon)-$approximation of our ball. Algorithm 4 is the adaptation of the classical Frank Wolfe algorithm to the problem (13) since each iteration linearize the quadratic objective function $\Phi(u)$ at the current state u^k. It is important to note that Algorithm 4 uses only first order approximation to Φ, so it will take expectantly more iterations than other algorithms that use second order approximations. Therefore, each iteration will require a lower number of computations. It has been proved that this algorithm converges in $O(1/\epsilon)$ iterations with a overall complexity bound of $O(mn/\epsilon)$. Furthermore, the core set returned by the algorithm has size $O(1/\epsilon)$.

3.2 The second algorithm that computes a $(1+\epsilon)-$approximation to MEB($\mathcal{A}$)

Algorithm 5 The second algorithm that computes a $(1+\epsilon)-$approximation to MEB($\mathcal{A}$)

```
Require: Input set of points $\mathcal{A} = \left\{a^1, ..., a^m\right\} \subset \mathbb{R}^n$, $\epsilon > 0$
 1: $\alpha \longleftarrow \arg\max_{i=1,...,m} \|a^i - a^1\|^2$,    $\beta \longleftarrow \arg\max_{i=1,...,m} \|a^i - a^\alpha\|^2$;
 2: $u_i^0 \longleftarrow 0, \quad i = 1, ..., m$
 3: $u_\alpha^0 \longleftarrow 1/2, \quad u_\beta^0 \longleftarrow 1/2$;
 4: $\chi_0 \longleftarrow \left\{a^\alpha, a^\beta\right\}$;
 5: $c^0 \longleftarrow \sum_{i=1}^m u_i^0 a^i$;
 6: $\gamma^0 \longleftarrow \Phi(u^0)$;
 7: $\kappa \longleftarrow \arg\max_{i=1,...,m} \|a^i - c^0\|^2, \quad \xi \longleftarrow \arg\min_{i:a^i \in \chi_0} \|a^i - c^0\|^2$;
 8: $\delta_0^+ \longleftarrow \left(\|a^\kappa - c^0\|^2/\gamma^0\right) - 1, \quad \delta_0^- \longleftarrow 1 - \left(\|a^\xi - c^0\|^2/\gamma^0\right)$;
 9: $\delta_0 \longleftarrow \max\left\{\delta_0^+, \delta_0^-\right\}$
10: k$\longleftarrow$ 0;
11: while $\delta_k > (1+\epsilon)^2 - 1$, do
12:     if $\delta_k > \delta_k^-$ then
13:         $\lambda^k \longleftarrow \delta_k / \left[2(1+\delta_k)\right]$;
14:         $k \longleftarrow k+1$;
15:         $u^k \longleftarrow (1-\lambda^{k-1})u^{k-1} + \lambda^{k-1}e^\kappa$;
16:         $c^k \longleftarrow (1-\lambda^{k-1})c^{k-1} + \lambda^{k-1}a^\kappa$;
17:         $\chi_k \longleftarrow \chi_{k-1} \cup \{a^\kappa\}$;
18:     else
19:         $\lambda^k \longleftarrow \min\left\{\frac{\delta_k^-}{\left[2(1+\delta_k^-)\right]}, \frac{u_\xi^k}{1-u_\xi^k}\right\}$;
20:         if $\lambda^k = u_\xi^k/(1-u_\xi^k)$ then
21:             $\chi_{k+1} \longleftarrow \chi_k \setminus \left\{a^\xi\right\}$;
22:         else
23:             $\chi_{k+1} \longleftarrow \chi_k$;
24:         end if
25:         $k \longleftarrow k+1$;
26:         $u^k \longleftarrow (1+\lambda^{k-1})u^{k-1} + \lambda^{k-1}e^\xi$;
27:         $c^k \longleftarrow (1+\lambda^{k-1})c^{k-1} + \lambda^{k-1}a^\xi$;
28:     end if
29:     $\gamma^k \longleftarrow \Phi(u^k)$;
30:     $\kappa \longleftarrow \arg\max_{i=1,...,m} \|a^i - c^k\|^2, \quad \xi \longleftarrow \arg\min_{i:a^i \in \chi_k} \|a^i - c^k\|^2$;
31:     $\delta_k^+ \longleftarrow \left(\|a^\kappa - c^k\|^2/\gamma^k\right) - 1, \quad \delta_k^- \longleftarrow 1 - \left(\|a^\xi - c^k\|^2/\gamma^k\right)$;
32:     $\delta_k \longleftarrow \max\left\{\delta_k^+, \delta_k^-\right\}$
33: end while
34: Output $c^k, \chi_k, u^k, \sqrt{(1+\delta_k)\gamma^k}$.
```

The Algorithm 5 starts in the same way of the Algorithm 4 (from step 1 to step 6), instead at step 7 it computes not only the furthest point from the center, but also the closest one. As the Away-step Frank Wolfe Algorithm 2, Algorithm 5 decides at each iteration to perform either a step like the one computed by Algorithm 4 or an away step that moves the center away from the closest point $a^{\xi} \in \chi_k$. Algorithm 5 requires at most twice as many iterations as that required by Algorithm 4, but asymptotically the bound is the same. Furthermore, also the asymptotic overall complexity is still $O(mn/\epsilon)$. Finally, it is important to notice that even if the asymptotic bound on the size of the core set is the same as Algorithm 4, in practice the Algorithm 5 is able to compute smaller core sets since, when $\lambda^k = u^k_{\xi}/(1 - u^k_{\xi})$ (drop-iteration), a^{ξ} is removed from the working core set. Another important difference with Algorithm 4 is that the second algorithm enjoys linear convergence. Since Algorithm 4 and Algorithm 5, as we already said, are respectively applications of classical Frank Wolfe and Away-step Frank Wolfe, it is not surprising that the former $(1 + \epsilon)-$approximation algorithm is sub-linear and the latter linear, however none of the Away-step Frank Wolfe convergence proofs are directly applicable to our case. This is the reason why it is necessary to introduce a perturbation in the primal formulation (problem 12) as shown in (Yildirim [3] and Robinson [5]).

3.3 Heart Disease dataset

The Heart Disease dataset collects data from Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D. There are 14 attributes:

1. age;
2. sex;
3. chest pain type;
4. resting blood pressure (in mm Hg on admission to the hospital);
5. serum cholestoral in mg/dl;
6. fasting blood sugar > 120 mg/dl ($1 =$ true; $0 =$ false);
7. resting electrocardiographic results;
8. maximum heart rate achieved;
9. exercise induced angina ($1 =$ yes; $0 =$ no);
10. ST depression induced by exercise relative to rest;
11. the slope of the peak exercise ST segment;
12. number of major vessels (0-3) colored by flourosopy;
13. thal;

14. diagnosis of heart disease;

We decided to not consider the attributes 11, 12 and 13, since they have too many missing values (denoted with -9 in the dataset). In addition, we removed the few rows with other missing values. In order to have all features in the same range, we use the MinMaxScaler to rescale our space. We finally considered the last attribute 14 as our label; this last consists of the number of heart attack (from 0 to 4). Therefore, our final dataset has 261 instances and 10 attributes.

3.4 Arcene dataset

The Arcene dataset was obtained from two sources: The National Cancer Institute (NCI) and the Eastern Virginia Medical School (EVMS). All the data consist of mass-spectra obtained with the SELDI technique. The samples include patients with cancer (ovarian or prostate cancer), and healthy or control patients. The original features indicate the abundance of proteins in human sera having a given mass value. Based on those features one must separate cancer patients from healthy patients. There are no missing values. Also for this dataset, we decided to use the MinMaxScaler. The labels were collected in another file, so our final dataset has 100 instances and 10000 attributes.

3.5 Experiments for the MEB problem

We decided to use the datasets presented in Section 3.3 and Section 3.4, since the former has a number of instances greater than the number of attributes ($n >> m$) while the latter has a number of instances smaller than the number of attributes ($m >> n$).

ϵ	**Time**		**Coreset size**		**Iterations**	
	MEB1	MEB2	MEB1	MEB2	MEB1	MEB2
1	0.006119	0.008051	2	2	0	0
0.1	0.016605	0.025138	6	6	4	4
0.01	0.191513	0.122724	10	8	66	25
0.001	1.808183	0.441618	10	8	642	93
0.0001	20.370467	0.911522	10	8	5965	205

Table 1: Heart Disease results

ϵ	**Time**		**Coreset size**		**Iterations**	
	MEB1	MEB2	MEB1	MEB2	MEB1	MEB2
1	0.017535	0.016511	2	2	0	0
0.1	0.031350	0.041328	5	5	3	3
0.01	0.067873	0.120693	9	9	9	13
0.001	0.476887	0.279441	10	11	106	39
0.0001	3.292613	0.501910	11	11	808	72

Table 2: Arcene results

For both the datasets we decided to run Algorithm 4 and Algorithm 5 by changing the value of ϵ from 1 (trivial) to 0.0001. As we expected from the theory results (Section 3.1 and Section 3.2), when ϵ goes to zero, time, core-set size and iterations increase. Looking at Table 1 and Table 2, we can see that the $first\_MEB$ algorithm is slower and makes many more iterations than $second\_MEB$ algorithm. In addition, for the Heart Disease dataset the ratio between the two times and the two iterations is much bigger than the one of the Arcene dataset. This is due to the two different conformation of the datasets ($m << n$ or $m >> n$); those results are almost in line with the ones of the original paper (Yildirim [3]). We investigated the performance of the two algorithms for a relatively small range of ϵ but we can already see that the two behave differently: Algorithm 4 appears to have a sub linear convergence while Algorithm 5 appears to have a linear convergence as expected.

3.6 Anomaly detection

Our aim was to find anomalies in the datasets presented in Section 3.3 and Section 3.4. For us, an anomaly is an element outside the radius of the computed MEB. In order to achieve this, we considered subsets regarding the labels we focused on. The Heart Disease dataset has 4 different positive classes (see Section 3.3); we run our MEB algorithms using $\epsilon = 0.001$ over the data with label greater than $1, 2, 3, 4$ (see Table 3). For each algorithm, we reported three columns:

- *r*: contains two values: the first one is the number of instances with distance from the center greater than the radius (r) of all dataset; the second one contains the same value but considering only the instances belonging to the class;
- *r + annulus*: contains two values: the first one is the number of instances with distance from the center greater than the r + annulus of all dataset, where with annulus we denote the ϵ-expansion of the ball; the second one is the same value but considering only the instances belonging to the class;
- *tot*: the cardinality of the class considered.

The Arcene dataset has 2 different classes: 1 for people with cancer and -1 otherwise; we run our MEB algorithms using $\epsilon = 0.0001$ over the data isolating the two classes (see Table 4). For each algorithm, we considered the same three columns as before.

	MEB1			**MEB2**		
class	r	r + annulus	tot	r	r + annulus	tot
≥ 1	$21,6$	$15,0$	98	$19,4$	$15,0$	98
≥ 2	$44,5$	$38,0$	63	$44,3$	$41,0$	63
≥ 3	$75,6$	$69,0$	40	$73,4$	$61,0$	40
4	$183,4$	$179,0$	15	$182,3$	$179,0$	15

Table 3: Heart disease anomaly detection

	MEB1			**MEB2**		
class	r	r + annulus	tot	r	r + annulus	tot
1	$36,9$	$27,0$	44	$32,5$	$27,0$	44
-1	$15,12$	$3,0$	56	$8,5$	$3,0$	56

Table 4: Arcene anomaly detection

From Table 3 and Table 4, we can see that in general the $second\_MEB$ gives a better approximation of the optimal MEB because there are less elements of the considered class outside the radius. As expected there are no elements outside the annulus for our chosen ϵ. On the other hand, $first\_MEB$ gives a larger set of elements that can be seen as anomalies. Looking at the results of the first column, the second number represents the number of anomalies found by our algorithms. We can see that MEB2 gives a better approximation since we have lower numbers. Obviously, as we shrink the class we will have more elements outside, but we focus only on the one of the same class. We can notice that while on the Table 3 the elements of the class considered outside the ball of radius r is much smaller than the complementary class elements, in Table 4 the class -1 has a different behaviour: this is due to the conformation of our dataset.

References

[1] Simon Lacoste-Julien, Martin Jaggi, *On the Global Linear Convergence of Frank-Wolfe Optimization Variants*, 2015

[2] Kenneth L.Clarkson, *Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm*, 2010

[3] E.Alper Yildirim, *Two Algorithms for the Minimum Enclosing Ball problem*, 2008

[4] Martin Jaggi, *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization*, 2013

[5] S. M. Robinson, *Generalized equations and their solutions, part ii: Applications to nonlinear programming*, 1982, pp. $200 - 221$