[(PDF) Lecture Notes on Machine Learning: Minimum Enclosing Balls (researchgate.net)](https://www.researchgate.net/publication/331152293_Lecture_Notes_on_Machine_Learning_Minimum_Enclosing_Balls)

[(PDF) Lecture Notes on Machine Learning: Frank-Wolfe for Minimum Enclosing Balls (researchgate.net)](https://www.researchgate.net/publication/331315423_Lecture_Notes_on_Machine_Learning_Frank-Wolfe_for_Minimum_Enclosing_Balls)

[(PDF) NumPy / SciPy Recipes for Data Science: Frank-Wolfe for Minimum Enclosing Balls (researchgate.net)](https://www.researchgate.net/publication/344138310_NumPy_SciPy_Recipes_for_Data_Science_Frank-Wolfe_for_Minimum_Enclosing_Balls)

INTRODUCTION

Minimum enclosing balls (MEBs) provide a very versatile data representation for a wide range of learning and analysis tasks. Use cases include (but are not limited to) accelerated training of support vector machines or other classifiers, symbolic learning, identification of landmarks or prototypes, analysis of data streams, and anomaly detection. (**TODO**: ADD HERE THAT WE FOCUS ON THE LAST PART). MEBs are also at the heart of one class support vector machines, support vector clustering, or support vector data description. While there exists a whole spectrum of algorithms for estimating the MEB of a given data set, a particularly general approach is to formalize the problem in terms of constrained quadratic optimization and we, too, will adhere to this strategy.

Minimum enclosing balls have numerous important applications in clustering, nearest neighbor search, data classification, support vector machines, machine learning, facility location, collision detection, computer graphics, and military operations. We refer the reader to [24] and the references therein. In particular, many of these applications give rise to large-scale instances of the MEB problem, and a reasonably small accuracy suffices for such applications.

While the primal problem of estimating an MEB is rather unwieldy, we will see that the corresponding dual problem can be solved rather easily. Indeed, the dual MEB problem is of a form that allows for using simple solvers such as the basic Frank-Wolfe algorithm and its variations. We will discuss this in detail in this project. (**TODO**: change last sentence)

**The Primal MEB Problem**

Given a data set , the minimum enclosing ball problem is to determine the smallest n-ball:

that contains all the points in .

Since Euclidean balls are uniquely defined in terms of their center and radius , the basic problem therefore is to determine optimal choices and for these two parameters. In this report, we approach this problem from the point of view of constrained convex optimization. Indeed, noting the equivalencies:

we can immediately formalize the problem of finding and as an inequality constrained quadratic minimization problem, namely

Which we can rewrite as:

In plain(er) English, this means that we are simultaneously searching for an appropriate center and radius. The (squared) radius should be as small as possible but not smaller. That is, the (squared) distance between each of the given data points and the center of the ball must be less than or equal to the (squared) radius of the ball.

While there are algorithms that can solve (4) directly, a more general approach consists in working with the corresponding dual problem which we will derive next.

By setting , we obtain:

Our goal in this note is to show that the above primal (minimization) problem has the following dual (maximization) problem:

where is a vector of Lagrange multipliers.

**Deriving the Dual Problem**

To begin with, we write down the Lagrangian of the problem in (3). Letting denote the Lagrange multiplier for the i-th inequality in (3), we have

Observe that, when expanding the expression in (6) to the one in (7), we introduced the shorthand just to reduce notational clutter.

In fact, we can write our Lagrangian even more compactly. To this end, we note that

where is a data matrix whose columns correspond to the data points in . Hence, the Lagrangian in (7) can also be written as

Given this Lagrangian, we next evaluate the Karush-Kuhn-Tucker conditions. Considering the KKT 1 conditions (stationarity), we have

Plugging the result in (14) into the one in (13), the latter simplifies to

Moreover, plugging the expressions in (14) and (15) back into the Lagrangian in (12), we obtain the Lagrangian dual, because

is a function that only depends on the Lagrange multipliers .

Now, if we further note that the KKT 3 conditions (dual feasibility) demand , all our (intermediate) results and considerations so far reveal that the dual of (3) amounts to

Finally, we point out that is concave in so that is convex. The solution to the maximization problem in (6) can therefore also be found by considering an equivalent minimization problem, namely

**From Lagrange Multipliers to MEB Parameters**

If we could solve the dual problem in (7) we would obtain a vector of optimal Lagrange multipliers. Assuming that we could determine them, they would immediately allow us to compute the center of the sought after MEB because, according to (5), we have

But what about the corresponding radius ?

The radius of the minimum enclosing ball can be obtained from considering the KKT 4 conditions (complementary slackness). These dictate that, at a solution, we must have

Summing these equations over all i then yields

for which it is fairly easy to see (considering our computations above) that some additional algebra leads to

In other words, we find that the radius of the minimum enclosing ball for the given data amounts to

**Frank-Wolfe for Minimum Enclosing Balls**

Reiterating what we said above, we emphasize that the dual MEB problem in (7) asks for a minimizer of the convex function (8). Also, the non-negativity () and sum-to-one () constraints in (7) imply that any feasibly solution must reside in the standard simplex:

This insight allows us to write our problem in an even more compact manner, namely

Written like this, the dual MEB problem is now clearly recognizable as an instance of a convex minimization problem over a compact convex set. This is great because the Frank Wolfe algorithm provides a simple tool for solving this particular kind of problem.

Next, we show how to specialize the general form of the Frank-Wolfe computations in (5), (6), and (8) to the minimum enclosing ball problem in (12).

The Frank-Wolfe algorithm is an iterative solver. First, it makes an initial feasible guess for the solution. The midpoint of is contained in and therefore feasible. This means that we can use as a starting point, however for simplicity in our implementation we’ve decided to use the first row of the basis matrix. At each iteration, the algorithm determines which minimizes the inner product and apply a conditional gradient update where the step size [0, 1] decreases over time. Updates will thus never leave the feasible set and the efficiency of the algorithm is due to the fact that it turns a quadratic problem into a series of linear ones.

Finding the step direction involves the gradient of the objective function at the current estimate. In our specific case, this gradient is given by

Therefore, each Frank-Wolfe iteration has to solve

Note that the expression on the right-hand side is linear in and needs to be minimized over which is a compact convex set. A minimum of a linear function over a convex set is necessarily attained at a vertex of said set. As the vertices of correspond to the standard basis vectors of , we therefore only have to determine which of these minimizes the above inner product. Hence, the problem in (15) simplifies to

which, in order to explicate that the resulting will be a standard basis vector , can also be written as

This can be solved without computing inner products: we just need to determine the index of the smallest component of the gradient, namely

Given these considerations, the general update step for the Frank-Wolfe algorithm thus specializes to

In conclusion, solving the dual MEB problem can be fairly straightforward by applying the original Frank Wolfe algorithm and its variations. In this project, we have examined, implemented and compared three different versions of the original Frank Wolfe algorithm. In the following, we will introduce each of these three algorithms and subsequently present our results by applying them to address the MEB problem in the context of anomaly detection.

**1. Away-steps Frank-Wolfe algorithm:**

The convergence rate of the original Frank Wolfe algorithm is known to be slow (sublinear) when the solution lies at the boundary. A simple fix is to add the possibility to take ‘away steps’ during optimization, an operation that importantly does not require a feasibility oracle. The away-steps variant of the Frank-Wolfe algorithm, that can also remove weight from ‘bad’ atoms in the current active set, was proposed by P. Wolfe. The precise method is stated below in Algorithm 1.

When the optimal solution lies at the boundary of , the iterates of the classical FW algorithm start to zig-zag between the vertices defining the face containing the optimal solution, which makes the convergence rate of the iterates slow, i.e. sublinear.

To address the zig-zagging problem of FW, Wolfe proposed to add the possibility to move away from an active atom in (see middle of Figure 1); this simple modification is sufficient to make the algorithm linearly convergent for strongly convex functions. We describe the away-steps variant of Frank-Wolfe in Algorithm 1. The away direction is defined in line 4 by finding the atom in that maximizes the potential of descent given by . Note that this search is over the (typically small) active set , and is fundamentally easier than the linear oracle . The maximum step-size as defined on line 9 ensures that the new iterate stays in conv() ⊆ , which guarantees that the convex representation is maintained. Computing the true maximum feasible step-size would require the ability to know when we cross the boundary of along a specific line, which is not possible for general . Using the conservative maximum step-size of line 9 ensures that we do not need this more powerful oracle. This is why Algorithm 1 requires to maintain , (unlike standard FW). Finally, as in classical FW, the FW gap is an upper bound on the unknown suboptimality, and can be used as a stopping criterion:

(by convexity)

If , then we call this step a drop step, as it fully removes the atom from the currently active set of atoms , (by settings its weight to zero). We cannot guarantee sufficient progress in this case, but the drop step decreases the active set size by one, and thus they cannot happen too often (not more than half the time). The weight updates for lines 12 and 13 are of the following form: For a FW step, we have if ; otherwise . Also, we have and for . For an away step, we have if (a drop step); otherwise .Also, we have and for .

**Algorithm 1: Away-steps Frank-Wolfe algorithm**

Let , and so that for and 0 otherwise)

**For** **do**

Let  and (the FW direction)

Let and (the away direction)

**if** **then** **return**

**if** **then**

and (choose the FW direction)

**else**

and (choose away direction; maximum feasible step-size)

**end if**

Line-search

Update (and accordingly for the weights)

Update (and accordingly for the weights)

**end for**

**Blended Pairwise Conditional Gradients (BPCG)**

The Pairwise Conditional Gradients algorithm is a natural generalization of the Away-step Frank-Wolfe algorithm. Moreover, it uses the blending criterion from (Braun et al., 2019a) to efficiently blend together local PCG steps with global Frank-Wolfe steps. Here we analyze a simpler modification of the PCG algorithm, obtained by combining it with the blending criterion from the Blended Conditional Gradients Algorithm. This new algorithm is called Blended Pairwise Conditional Gradients. BPCG does not exhibit swap steps anymore and the convergence rates are that which the original PCG algorithm would achieve if swap steps would not occur. As such it improves the convergence rates of PCG.

Same as we stated above, we focus on the general smooth convex case and the strongly convex case over polytopes.

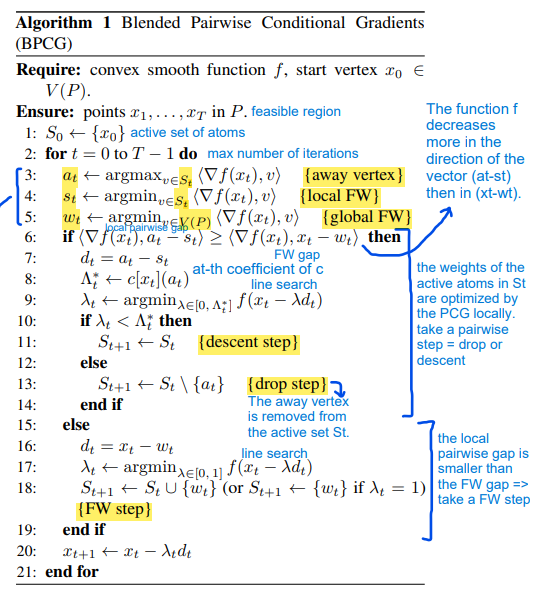
Additionally, the numerical experiments suggest that the BPCG algorithm outputs fairly sparse solutions practically.

Blended Pairwise Conditional Gradients algorithm (BPCG) is shown in Algorithm 1.

In Algorithm 1, if the local pairwise gap h∇f(xt), at − sti is smaller than the Frank-Wolfe gap h∇f(xt), xt − wti, a FW step (line 16-18) is taken. Otherwise, the weights of the active atoms in St are optimized by the Pairwise Conditional Gradients (PCG) locally. If the step size λt is larger than Λ ∗ t , the away vertex is removed from the active set St and we call the step drop step (line 13). Otherwise we call the step descent step (line 11). Descent step and drop step are referred to as pairwise step all together.

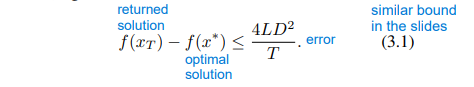
By the structure of the BPCG algorithm, the sparsity of the solutions is expected since the new atoms are not added to St until the local pairwise gap decreases sufficiently. Moreover, since the PCG is implemented locally, BPCG does not exhibit swap steps in which weight is shifting from the away atom to the Frank-Wolfe atom.

Note that Algorithm 1 uses line search here to simplify the presentation. However it can be run alternatively with the short-step rule (t/t+2).



Convergence analysis

Theorem 3.1. Let P be a convex feasible domain of diameter D. Assume that f is convex and L-smooth. Let {xi} T i=0 ⊂ P be the sequence given by the BPCG algorithm (Algorithm 1). Then, it holds that



As shown in Theorem 3.1, the BPCG algorithm achieves a convergence rate which is no worse than that of the PCG. In addition, since BPCG does not exhibit swap steps, the convergence rate of BPCG including (possibly dimension-dependent) constant factors is considered to be better than that of the PCG especially when the dimension of the feasible region is high.

As such BPCG’s convergence rate is much more in line with that of the AwayStep Frank-Wolfe algorithm (see (Lacoste-Julien and Jaggi, 2015)). Moreover, since the iterations of BPCG include many local updates in which no new atoms are added, it is expected that the BPCG algorithm outputs sparser solutions than the PCG algorithm in terms of the support size of the supporting convex combination.

Remark 3.1. We clarify the advantages of the BPCG algorithm in comparison to other CG variants.

Compared to the vanilla CG: The BPCG algorithm ensures linear convergence for the strongly convex polyhedral case, whereas the vanilla CG (Frank and Wolfe, 1956; Levitin and Polyak, 1966) does not. Although both algorithms guarantee a O(1/k) convergence for infinite-dimensional cases, we expect the BPCG algorithm to output much sparser solutions due to the aggressive dropping of unnecessary vertices from the convex combination and adding new vertices only when they are needed for primal progress.

Compared to the Away CG algorithm: BPCG and Away CG (Wolfe, 1970; Guélat and Marcotte, 1986) have the same convergence guarantee for the general smooth convex case and strongly convex polyhedral case however BPCG empirically produces significantly sparser solutions.

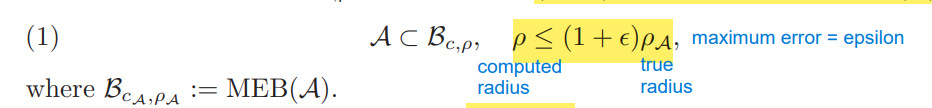
BPCG also applies to the infinite-dimensional setting and its (empirical) sparsity is very high making it a prime candidate for problems such as sparse signal recovery, matrix completion, and numerical integration.

**(1 + epsilon)-approximation to MEB(A)**

(1 + epsilon)-approximation to the radius of the minimum enclosing ball of A.

The first algorithm is closely related to the Frank–Wolfe algorithm with a proper initialization applied to the dual formulation of the minimum enclosing ball problem. We establish that this algorithm converges in O(1/epsilon) iterations with an overall complexity bound of O(mn/epsilon) arithmetic operations. In addition, the algorithm returns a “core set” of size O(1/epsilon), which is independent of both m and n.

Given epsilon > 0, a ball Bc,ρ is said to be a (1 + )-approximation to MEB(A) if

****

A subset X ⊆A is said to be an -core set (or a core set) of A if

****

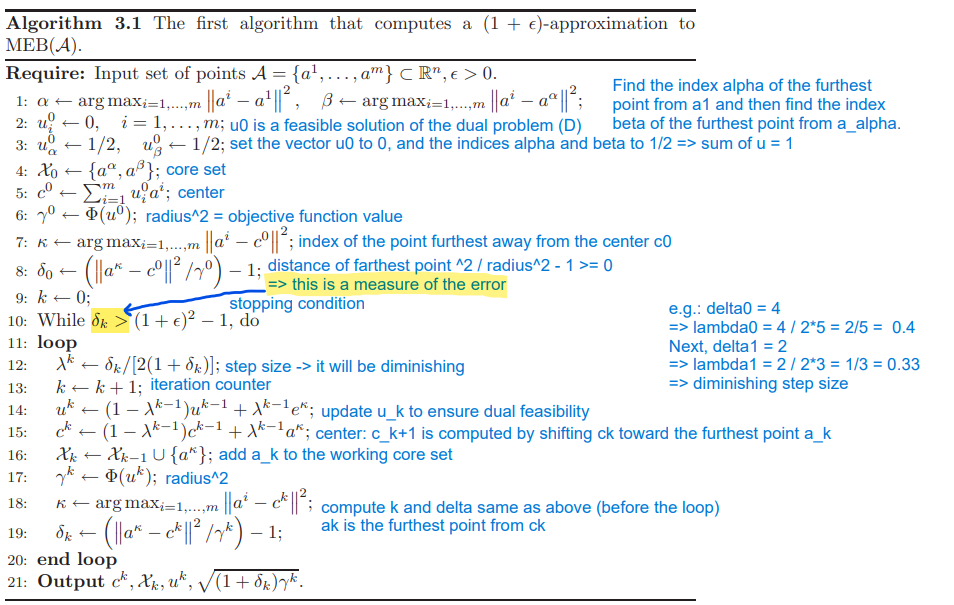
where BcX ,ρX := MEB(X). Small core sets play an important role in designing efficient algorithms for large-scale problems, since they provide a compact representation of the input set A. If a small -core set X is available, then solving the problem on X already yields a good approximation to MEB(A). Since the center cA of MEB(A) lies in the convex hull of A (cf. section 2), it follows from Carath´eodory’s theorem that there always exists a 0-core set of size at most n + 1.

Our first algorithm is closely related to the Frank–Wolfe algorithm [15] applied to the dual formulation of the problem. At each iteration, the algorithm can only add points to the working core set.

This algorithm computes a (1 + epsilon)-approximation to MEB(A) in O(mn/epsilon) operations, which matches the currently best known dependence on epsilon. In addition, it explicitly computes an epsilon-core set of size O(1/ epsilon).

We first compare our algorithms with the one proposed by Panigrahy [33], which computes a (1 + )-approximation to the minimum enclosing ball of a finite set of points in O(mn/) arithmetic operations. Panigrahy’s algorithm starts with a ball whose radius is known to be smaller than that of the minimum enclosing ball and maintains an upper bound ζ on the difference between these two radii. At each iteration, the algorithm moves the current ball toward the furthest point from the center until the ball touches that particular point without changing the radius of the ball. After repeating such iterations O(1/ζ) times, the algorithm either provides a certificate that an approximate solution has been computed or decides that either the radius can be increased or the error bound ζ can be decreased. The whole procedure is then repeated using the new parameters for the radius and the error bound. Similarly to Panigrahy’s algorithm, each of our algorithms also constructs a sequence of balls, and our first algorithm moves the center toward the furthest point from the center of the current ball at each iteration. However, the center moves by only a fraction of this distance.

Algorithm 1 constructs balls of strictly increasing radii in each iteration, and the radius and the error bound are updated at each iteration. We check the termination criterion in each iteration, which has the potential advantage of earlier termination than that predicted by the theoretical worst-case estimate.



We now describe Algorithm 3.1 in more detail. In step 1, the algorithm computes the furthest point aα ∈ A from a1 ∈ A and then computes the furthest point aβ ∈ A from aα. Steps 2 and 3 initialize the vector u0 ∈ Rm. Note that u0 is a feasible solution of the dual problem (D). The core set X0 is initialized at step 4. At each iteration, the algorithm implicitly constructs a “trial” ball with center ck and radius (γk)1/2. This ball coincides with MEB(A) if and only if uk is an optimal solution of (D). Otherwise, at least one point in A lies outside of this ball. Note that δk satisfies ||aκ – ck|| 2 = (1 + δk)γk, where aκ ∈ A is the furthest point from ck. It follows that the trial ball encloses A if its radius is expanded by a factor of (1 + δk)1/2, i.e., Φ(uk) ≤ Φ(u∗) ≤ (1 + δk)Φ(uk). Unless the termination criterion is satisfied, the new center ck+1 is computed by shifting ck toward the furthest point aκ, which is added to the working core set Xk+1, and uk+1 is updated accordingly to ensure that dual feasibility is maintained. The algorithm continues in an iterative manner by computing a new trial ball corresponding to uk+1.

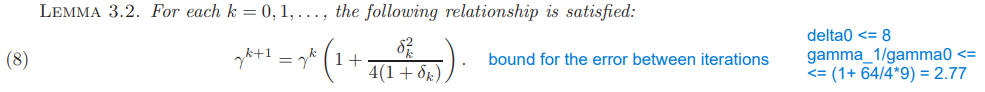
Algorithm 3.1 is the adaptation of the Frank–Wolfe algorithm to the dual problem (D). At each iteration, the quadratic objective function Φ(u) of (D) is linearized at the current feasible solution uk. Since the feasible region of (D) is the unit simplex, the unit vector eκ, where κ is the index of the furthest point in A from ck, solves the linearized subproblem. It is easy to verify that



We remark that Algorithm 3.1 uses only the first-order approximation to the objective function Φ. As such, each iteration is fairly cheap, but the number of iterations is usually significantly higher than other algorithms that use second-order information such as interior-point methods. However, such general-purpose algorithms become computationally infeasible for larger problems, since each iteration is usually much more expensive. This observation provides one of our motivations to develop a specialized algorithm for this problem.

Analysis of Algorithm 3.1

Lemma 3.1. u0 ∈ Rm satisfies γ0 = Φ(u0) ≥ (1/3)Φ(u∗) = (1/3)γA, where u∗ ∈ Rm and γA are the optimal solution and the optimal value of (D), respectively. Furthermore, δ0 ≤ 8.



Theorem 3.1. Given A := {a1,...,am} ⊂ Rn and epsilon ∈ (0, 1), Algorithm 3.1 computes a (1 + epsilon)-approximation to MEB(A) in at most 9 + 25/ epsilon iterations.

Theorem 3.1 establishes that Algorithm 3.1 converges in O(1/epsilon) iterations.

Theorem 3.2. Given A := {a1,...,am} ⊂ Rn and epsilon ∈ (0, 1), Algorithm 3.1 computes a (1 + epsilon)-approximation to MEB(A) in at most O(mn/epsilon) arithmetic operations

At each iteration, the dominating work is the computation of the furthest point from the center of the current trial ball, which also requires O(mn) operations

Theorem 3.3. Given A := {a1,...,am} ⊂ Rn and ∈ (0, 1), let η denote the index of the final iterate computed by Algorithm 3.1. Then, Xη ⊆ A is an -core set of A. Furthermore, |Xη| = O(1/).

Additional info that we can use:

The main idea is to approximate the given input set using only a carefully selected finite subset of points and then to refine this approximation by adding more points if necessary. This leads to an approximation of the primal formulation with only a finite number of constraints, and this approximation is refined by adding more constraints. In the dual formulation, we therefore start with a finite number of variables and add more variables if necessary.

The initial working core set X0 provides the first approximation to the given input set with only two points. Let Φ0(·) denote the objective function of the dual formulation of the minimum enclosing ball problem for X0, and let γA denote the optimal value of the aforementioned semi-infinite primal formulation.

Similarly, let Φk(·) denote the objective function of the dual formulation of the minimum enclosing ball problem for Xk ⊂ A. At iteration k in each algorithm, Xk provides the current finite approximation to A. Let ck ∈ Rn denote the current center. Each algorithm computes the furthest point in A from ck. In Algorithm 3.1, Xk+1 is obtained by adding this point to Xk. Unless the furthest point in A already belongs to Xk, the dual formulation for Xk+1 differs from that for Xk in only one variable. Therefore, [(uk)T , 0]T is a feasible solution for the new dual formulation that satisfies Φk+1([(uk)T , 0]T )=Φk(uk), which implies that the improvement in each iteration still obeys the relation given by Lemma 3.2, with γk+1 replaced by Φk+1(uk+1) and γk by Φk(uk). Note that the dimension of uk+1 is one more than that of uk in this case.

Both algorithms exploit the special structure of the dual formulation of the problem and can geometrically be viewed as generating a sequence of trial balls until a ball with desired properties is computed. Each of the two algorithms is especially well-suited for the large-scale instances of the minimum enclosing ball problem for which a moderate approximation suffices. Both algorithms can compute a small core set whose size depends only on the approximation parameter.

Testing: