New Approximation Algorithms for Minimum Enclosing Convex Shapes

Ankan Saha* S.V. N. Vishwanathan† Xinhua Zhang‡

Abstract

Given n points in a d dimensional Euclidean space, the Minimum Enclosing Ball (MEB) problem is to find the ball with the smallest radius which contains all n points. We give two approximation algorithms for producing an enclosing ball whose radius is at most ϵ away from the optimum. The first requires $O(nd\mathcal{L}/\sqrt{\epsilon})$ effort, where $\mathcal{L}$ is a constant that depends on the scaling of the data. The second is a $O^*(nd\mathcal{Q}/\sqrt{\epsilon})$ approximation algorithm, where $\mathcal{Q}$ is an upper bound on the norm of the points. This is in contrast with *coresets* based algorithms which yield a $O(nd/\epsilon)$ greedy algorithm. Finding the Minimum Enclosing Convex Polytope (MECP) is a related problem wherein a convex polytope of a fixed shape is given and the aim is to find the smallest magnification of the polytope which encloses the given points. For this problem we present $O(mnd\mathcal{L}/\epsilon)$ and $O^*(mnd\mathcal{Q}/\epsilon)$ approximation algorithms, where m is the number of faces of the polytope. Our algorithms borrow heavily from convex duality and recently developed techniques in non-smooth optimization, and are in contrast with existing methods which rely on geometric arguments. In particular, we specialize the excessive gap framework of Nesterov [19] to obtain our results.

1 Introduction

Given a set $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ of n points in $\mathbb{R}^d$, the minimum enclosing ball (MEB) is the ball with the smallest radius which contains all the points in S. The problem of finding a MEB arises in application areas as diverse as data mining, learning, statistics, computer graphics, and computational geometry [9]. Therefore efficient algorithms for this problem are not only of theoretical interest, but also have wide practical applicability.

Exact algorithms for finding the MEB typically have an exponential dependence on d [16, 25]. For example, the Welzl [25] algorithm runs in $O(n(d+1)(d+1)!)$ time which makes it inadmissible for many practical applications; in the case of linear support vector machines (SVMs) data may have a million or more dimensions. Therefore, there has been a significant interest in finding approximation algorithms for this problem.

State of the art approximation algorithms for the MEB problem extensively use the concept of coresets [1, 6, 20, 26]. Given an $\epsilon > 0$, an ϵ-coreset $S' \subset S$ has the property that if the smallest enclosing ball containing S' is expanded by a factor of $(1+\epsilon)$, then the resulting ball also contains S. Therefore, locating an ϵ-coreset is equivalent to finding an $(1+\epsilon)$ approximation algorithm to the MEB problem. The approximation guarantees of such algorithms are **multiplicative**. Briefly, a coreset is built incrementally in a greedy fashion [20, 26]. At every iteration, the MEB of the current candidate coreset is built. If every point in S lies in an $(1+\epsilon)$ ball of the current solution then the algorithm stops, otherwise the most *violated* point, that is, the point which is furthest away from the current MEB is included in the candidate coreset and the iterations continue. The best known algorithms in this family have a running time of $O(nd/\epsilon)$ [6, 20, 26].

In contrast, we present two algorithms which are derived by casting the problem of finding the MEB as a convex but non-smooth optimization problem. By specializing a general framework of Nesterov [19], our first algorithm achieves a running time of $O(nd\mathcal{L}/\sqrt{\epsilon})$ where $\mathcal{L}$ is an input dependent constant (see §3.1). Our second algorithm, on the other hand, requires $O^*(nd\mathcal{Q}/\sqrt{\epsilon})$ effort, where $\mathcal{Q}$ is an upper bound on the norm of the points. Also, the approximation guarantees of our algorithms are **additive**, that is, given a tolerance $\epsilon > 0$ and denoting the optimal radius by R^*, our algorithms produces a function whose value lies between R^{*2} and $R^{*2} + \epsilon$. Although additive and multiplicative approximation guarantees seem different, by a simple argument in §3.3, we show that our algorithms can also yield a traditional scale-invariant ϵ multiplicative approximation.

We extend our analysis to the closely related minimum enclosing convex polytope (MECP) problem, and

*Department of Computer Science, University of Chicago ankans@cs.uchicago.edu

†Departments of Statistics and Computer Science, Purdue University vishy@stat.purdue.edu

‡Department of of Computing Science, University of Alberta xinhua2@ualberta.ca

present two new algorithms. As before, given a set $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ of n points in $\mathbb{R}^d$, the task here is to find the smallest polytope of a given fixed shape which encloses these points. In our setting translations and magnifications are allowed but rotations are not allowed. We present $O(mnd\mathcal{L}/\epsilon)$ and $O^*(mnd\mathcal{Q}/\epsilon)$ approximation algorithms, where m denotes the number of faces of the polytope.

We apply our algorithms to two problems of interest in machine learning namely finding the maximum margin hyperplane and computing the distance of a polytope from the origin. A coreset algorithm for the first problem was proposed by Har-Peled et al. [11] while the second one was studied by Gärtner and Jaggi [10]. In both cases our algorithms improve the dependence on ϵ.

Our paper is structured as follows: In §2 we introduce notation, briefly review some results from convex duality, and present the general framework of Nesterov [19]. In §3 we address the MEB problem and in §4 the MECP problem, and present our algorithms and their analysis. We discuss some applications of our results to machine learning problems in §5. The paper then concludes with a discussion and outlook for the future in §6. Technical proofs can be found in Appendix A, B, and C, the applicability of our algorithms for training support vector machines (SVMs) is discussed in Appendix D, while preliminary experimental evaluation can be found in Appendix E.

2 Definitions and Preliminaries

In this paper, lower bold case letters (*e.g.*, $\mathbf{w}$, $\boldsymbol{\mu}$) denote vectors, while upper bold case letters (*e.g.*, $\mathbf{A}$) denote matrices or linear operators. We use w_i to denote the i-th component of $\mathbf{w}$, A_{ij} to denote the (i,j)-th entry of $\mathbf{A}$, and $\langle \mathbf{w}, \mathbf{w}' \rangle := \sum_i w_i w'_i$ to denote the Euclidean dot product between vectors $\mathbf{w}$ and $\mathbf{w}'$. Δ_k denotes the k dimensional simplex. Unless specified otherwise, $\|\cdot\|$ refers to the Euclidean norm $\|\mathbf{w}\| := \sqrt{\langle \mathbf{w}, \mathbf{w} \rangle} = \left(\sum_{i=1}^n w_i^2\right)^{\frac{1}{2}}$. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, we define the operator norm of $\mathbf{A}$ as

$$\|\mathbf{A}\| = \max \{\langle \mathbf{A}\mathbf{w}, \mathbf{u} \rangle : \|\mathbf{w}\| = 1, \|\mathbf{u}\| = 1\}.$$

We also denote $\overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$, and $[t] := \{1, \ldots, t\}$.

Definition 1. *Let $Q_1 \subseteq \mathbb{R}^n$, $f : Q_1 \to \overline{\mathbb{R}}$, and $f^* := \min_{\mathbf{w} \in Q_1} f(\mathbf{w}) < \infty$. A point $\mathbf{w}' \in Q_1$ such that*

$$f(\mathbf{w}') \leq f^* + \epsilon \tag{2.1}$$

is said to be an ϵ-accurate minimizer of f. We will also sometimes call $\mathbf{w}'$ an ϵ-accurate solution.

The following three standard concepts from convex analysis (see *e.g.* Hiriart-Urruty and Lemaréchal [12]) are extensively used in the sequel.

Definition 2. *A convex function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is strongly convex with respect to a norm $\|\cdot\|$ if there exists a constant $\rho > 0$ such that $f - \frac{\rho}{2}\|\cdot\|^2$ is convex. ρ is called the modulus of strong convexity of f, and for brevity we will call f ρ-strongly convex or ρ-s.c.*

Definition 3. *Suppose a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is differentiable on $Q \subseteq \mathbb{R}^n$. Then f is said to have Lipschitz continuous gradient (l.c.g) with respect to a norm $\|\cdot\|$ if there exists a constant L such that*

$$\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\| \forall\ \mathbf{w}, \mathbf{w}' \in Q. \tag{2.2}$$

For brevity, we will call f L-l.c.g.

Definition 4. *The Fenchel dual of a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is a function $f^\star : \mathbb{R}^n \to \overline{\mathbb{R}}$ defined by*

$$f^\star(\mathbf{w}^\star) = \sup_{\mathbf{w} \in \mathbb{R}^n} \{\langle \mathbf{w}, \mathbf{w}^\star \rangle - f(\mathbf{w})\} \tag{2.3}$$

Strong convexity and Lipschitz continuity of the gradient are related by Fenchel duality according to the following lemma:

Lemma 2.1. ([12, Theorem 4.2.1 and 4.2.2])

1. *If $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is ρ-s.c., then $f^\star$ is finite on $\mathbb{R}^n$ and $f^\star$ is $\frac{1}{\rho}$-l.c.g.*
2. *If $f : \mathbb{R}^n \to \mathbb{R}$ is convex, differentiable on $\mathbb{R}^n$, and L-l.c.g, then $f^\star$ is $\frac{1}{L}$-s.c.*

2.1 Nesterov's Framework In sections 3 and 4 we will show that the MEB and MECP problems respectively can be cast as minimizing convex non-smooth objective functions. In a series of papers, Nesterov [17–19] proposed a general framework for this task, which we now briefly review.

Let Q_1 and Q_2 be subsets of Euclidean spaces and $\mathbf{A}$ be a linear map from Q_1 to Q_2. Suppose f and g are convex functions defined on Q_1 and Q_2 respectively, and we are interested in the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w} \in Q_1} J(\mathbf{w}) \text{ where} \\ &J(\mathbf{w}) := f(\mathbf{w}) + g^\star(\mathbf{A}\mathbf{w}) \\ &\qquad = f(\mathbf{w}) + \max_{\mathbf{u} \in Q_2} \{\langle \mathbf{A}\mathbf{w}, \mathbf{u} \rangle - g(\mathbf{u})\}. \end{aligned} \tag{2.4}$$

We will make the following standard assumptions: a) Q_2 is compact; b) with respect to a certain norm on Q_1, the function f defined on Q_1 is ρ-s.c. but not necessarily

l.c.g, and c) with respect to a certain norm on Q_2, the function g defined on Q_2 is L_g-*l.c.g* and convex, but not necessarily strongly convex.

The key difficulty in solving (2.4) arises because $g^\star$ and hence J may be non-smooth. Our aim is to uniformly approximate $J(\mathbf{w})$ with a smooth and strongly convex function. Towards this end let d be a σ-s.c. smooth function with the following properties:

$$\min_{\mathbf{u}\in Q_2} d(\mathbf{u}) = 0, \quad \mathbf{u}_0 = \operatorname*{argmin}_{\mathbf{u}\in Q_2} d(\mathbf{u}), \text{ and } \mathcal{D} := \max_{\mathbf{u}\in Q_2} d(\mathbf{u}).$$

In optimization parlance d is called a prox-function. For a positive constant $\mu \in \mathbb{R}$ define

$$J_\mu(\mathbf{w}) := f(\mathbf{w}) + \max_{\mathbf{u}\in Q_2} \left\{ \langle \mathbf{A}\mathbf{w}, \mathbf{u}\rangle - g(\mathbf{u}) - \mu\, d(\mathbf{u}) \right\}. \tag{2.5}$$

It can be easily verified that J_μ is not only smooth and convex but also $\frac{1}{\sigma\mu}\|\mathbf{A}\|^2$-*l.c.g* [18]. Furthermore, if $\mathcal{D} < \infty$ then J_μ is uniformly close to J, that is,

$$J_\mu(\mathbf{w}) \le J(\mathbf{w}) \le J_\mu(\mathbf{w}) + \mu\,\mathcal{D}\ . \tag{2.6}$$

If some mild constraint qualifications hold [*e.g.* Theorem 3.3.5 of 4] one can write the dual $D(\mathbf{u})$ of $J(\mathbf{w})$ using $\mathbf{A}^\top$ (the transpose of $\mathbf{A}$) as

$$\begin{aligned} D(\mathbf{u}) &:= -g(\mathbf{u}) - f^\star(-\mathbf{A}^\top\mathbf{u}) \\ &= -g(\mathbf{u}) - \max_{\mathbf{w}\in Q_1} \left\{ \langle -\mathbf{A}\mathbf{w}, \mathbf{u}\rangle - f(\mathbf{w}) \right\}, \end{aligned} \tag{2.7}$$

and assert the following:

$$\begin{aligned} \inf_{\mathbf{w}\in Q_1} J(\mathbf{w}) &= \sup_{\mathbf{u}\in Q_2} D(\mathbf{u}), \text{ and} \\ J(\mathbf{w}) &\ge D(\mathbf{u}) \quad \forall\ \mathbf{w}\in Q_1, \mathbf{u}\in Q_2. \end{aligned} \tag{2.8}$$

The key idea of excessive gap minimization pioneered by Nesterov [19] is to maintain two estimation sequences $\{\mathbf{w}_k\}$ and $\{\mathbf{u}_k\}$, together with a diminishing sequence $\{\mu_k\}$ such that

$$\boxed{J_{\mu_k}(\mathbf{w}_k) \le D(\mathbf{u}_k), \text{ and } \lim_{k\to\infty} \mu_k = 0.} \tag{2.9}$$

The idea is illustrated in Figure 1. In conjunction with (2.8) and (2.6), it is not hard to see that $\{\mathbf{w}_k\}$ and $\{\mathbf{u}_k\}$ approach the solution of $\min_{\mathbf{w}} J(\mathbf{w}) = \max_{\mathbf{u}} D(\mathbf{u})$. Using (2.6), (2.5), and (2.9), we can derive the following bound on the duality gap:

$$J(\mathbf{w}_k) - D(\mathbf{u}_k) \le J_{\mu_k}(\mathbf{w}_k) + \mu_k\mathcal{D} - D(\mathbf{u}_k) \le \mu_k\mathcal{D}. \tag{2.10}$$

In other words, the duality gap is reduced at the same rate at which μ_k approaches 0. To turn this idea into an implementable algorithm we need to answer the following two questions:

1. How to efficiently find initial points $\mathbf{w}_1$, $\mathbf{u}_1$ and μ_1 that satisfy (2.9).
2. Given $\mathbf{w}_k$, $\mathbf{u}_k$, and μ_k, how to *efficiently* find iterates $\mathbf{w}_{k+1}$, $\mathbf{u}_{k+1}$, and μ_{k+1} which maintain (2.9). To achieve the best possible convergence rate it is desirable to anneal μ_k as fast as possible while still allowing $\mathbf{w}_k$ and $\mathbf{u}_k$ to be updated efficiently.

We will now show how the MEB and MECP problems can be cast as convex optimization problems and derive implementable algorithms by answering the above questions.

3 Minimum Enclosing Ball

Given a set of n points $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in a d dimensional space $\mathbb{R}^d$, a Euclidean ball $B(\mathbf{c}, R)$ of radius R centered at $\mathbf{c}$ is said to be an enclosing ball if $\forall i \in [n]$, $\mathbf{x}_i \in B(\mathbf{c}, R)$.

3.1 Formulation as an Optimization Problem

Clearly, $\mathbf{x}_i \in B(\mathbf{c}, R)$ if, and only if, $\|\mathbf{c} - \mathbf{x}_i\|^2 \le R^2$. Using this observation, the MEB problem can be cast as the following optimization problem:

$$\min_{R\in\mathbb{R}} R \qquad \text{s.t. } \|\mathbf{c} - \mathbf{x}_i\|^2 \le R^2 \quad \forall i,$$

which in turn can be reformulated as

$$\min_{\mathbf{c}\in\mathbb{R}^d} \max_{\mathbf{x}_i\in S} \|\mathbf{c} - \mathbf{x}_i\|^2. \tag{3.11}$$

Rearranging terms, we solve the following optimization problem

$$\begin{aligned} &\min_{\mathbf{c}\in\mathbb{R}^d} J(\mathbf{c}) \qquad \text{where} \\ J(\mathbf{c}) &= \|\mathbf{c}\|^2 + \max_{\mathbf{x}_i\in S} \left\{ -2\langle \mathbf{c}, \mathbf{x}_i\rangle + \|\mathbf{x}_i\|^2 \right\} \\ &= \|\mathbf{c}\|^2 + \max_{\mathbf{u}\in\Delta_n} \left\{ \langle \mathbf{A}\mathbf{c}, \mathbf{u}\rangle + \langle \mathbf{u}, \mathbf{b}\rangle \right\}, \end{aligned} \tag{3.12}$$

with $\mathbf{A} = -2[\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_n]^\top$, and $b_i = \|\mathbf{x}_i\|^2$. Clearly, $J(\mathbf{c})$ can be identified with (2.4) by setting $Q_1 = \mathbb{R}^d$, $Q_2 = \Delta_n$, $f(\mathbf{c}) = \|\mathbf{c}\|^2$, and $g(\mathbf{u}) = -\langle \mathbf{u}, \mathbf{b}\rangle$. It can be verified that g is 0-*l.c.g*, while f is 2-s.c. Therefore, one can employ Nesterov's framework (§2.1) to minimize $J(\mathbf{c})$. However, as we stated before, we need to specialize the framework to our setting to obtain an efficient and implementable algorithm. Towards this end note that the Fenchel dual of $\|\cdot\|^2$ is $\frac{1}{4}\|\cdot\|^2$, and use (2.7) to write the dual of (3.12) as

$$D(\mathbf{u}) = \langle \mathbf{u}, \mathbf{b}\rangle - \frac{1}{4}\mathbf{u}^\top\mathbf{A}\mathbf{A}^\top\mathbf{u}. \tag{3.13}$$

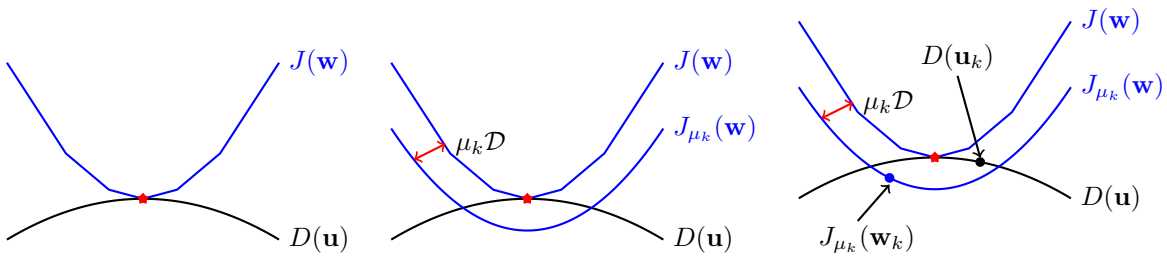

Figure 1: Illustration of excessive gap. By strong convexity, the dual $D(\mathbf{u})$ is always a lower bound to the primal $J(\mathbf{w})$ (left). We approximate $J(\mathbf{w})$ by a smooth lower bound $J_{\mu_k}(\mathbf{w})$ (middle). Since $D(\mathbf{u}_k)$ is sandwiched between $J_{\mu_k}(\mathbf{w})$ and $J(\mathbf{w})$, as $\mu_k \to 0$ we get closer and closer to the true optimum (right).

By using the definition of matrix norm, the gradient

$$\nabla D(\mathbf{u}) = \mathbf{b} - \frac{1}{2}\mathbf{A}\mathbf{A}^\top\mathbf{u}, \tag{3.14}$$

can be shown to satisfy

$$\begin{aligned}\|\nabla D(\mathbf{u}_1) - \nabla D(\mathbf{u}_2)\| &= \left\|-\frac{1}{2}\mathbf{A}\mathbf{A}^\top\mathbf{u}_1 + \frac{1}{2}\mathbf{A}\mathbf{A}^\top\mathbf{u}_2\right\| \\ &\leq \frac{1}{2}\left\|\mathbf{A}\mathbf{A}^\top\right\|\|\mathbf{u}_1 - \mathbf{u}_2\|, \end{aligned} \tag{3.15}$$

thus establishing that $D(\mathbf{u})$ is $\frac{1}{2}\left\|\mathbf{A}\mathbf{A}^\top\right\|$-*l.c.g.* We define $L = \frac{1}{2}\left\|\mathbf{A}\mathbf{A}^\top\right\|$.

Recall that $d(\cdot)$ denotes a prox-function, which we will specify shortly. For notational convenience, define the following three maps:

$$\mathbf{c}(\mathbf{u}) = \operatorname*{argmin}_{\mathbf{c}\in Q_1}\left\{\langle \mathbf{A}\mathbf{c}, \mathbf{u}\rangle + \|\mathbf{c}\|^2\right\} \tag{3.16}$$

$$\mathbf{u}_\mu(\mathbf{c}) = \operatorname*{argmax}_{\mathbf{u}\in Q_2}\left\{\langle \mathbf{A}\mathbf{c}, \mathbf{u}\rangle + \langle \mathbf{u}, \mathbf{b}\rangle - \mu d(\mathbf{u})\right\} \tag{3.17}$$

$$V(\mathbf{u}, \mathbf{g}) = \operatorname*{argmin}_{\mathbf{v}\in Q_2}\left\{d(\mathbf{v}) - \langle \nabla d(\mathbf{u}) - \mathbf{g}, \mathbf{v}\rangle\right\}. \tag{3.18}$$

With this notation in place, we describe our excessive gap minimization method in Algorithm 1. Unrolling the recursive update for μ_k yields

$$\begin{aligned}\mu_k &= (1-\tau_{k-1})\,\mu_{k-1} = \frac{k}{k+2}\,\mu_{k-1} \\ &= \frac{(k)(k-1)\ldots 2}{(k+2)(k+1)\ldots 4}\frac{L}{\sigma} \\ &= \frac{6}{(k+1)(k+2)}\frac{L}{\sigma}. \end{aligned} \tag{3.19}$$

Plugging this into (2.10) immediately yields the following theorem:

THEOREM 3.1. (DUALITY GAP) *The sequences* $\{\mathbf{c}_k\}$ *and* $\{\mathbf{u}_k\}$ *in Algorithm 1 satisfy*

$$J(\mathbf{c}_k) - D(\mathbf{u}_k) \leq \frac{6L\mathcal{D}}{\sigma(k+1)(k+2)}. \tag{3.20}$$

All that remains is to show that

THEOREM 3.2. *The update rule of Algorithm 1 guarantees that (2.9) is satisfied for all* $k \geq 1$.

Proof. See Appendix A.

Algorithm 1 Excessive gap minimization applied to MEB

Ensure: Sequences $\{\mathbf{c}_k\}$, $\{\mathbf{u}_k\}$, and $\{\mu_k\}$ that satisfy (2.9), with $\lim_{k\to\infty}\mu_k = 0$

1: Initialize: Let $\mathbf{u}_0 = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$, $\mu_1 = \frac{L}{\sigma}$, $\mathbf{c}_1 = \mathbf{c}(\mathbf{u}_0)$, $\mathbf{u}_1 = V\left(\mathbf{u}_0, -\frac{1}{\mu_1}\nabla D(\mathbf{u}_0)\right)$.

2: **for** $k = 1, 2, \ldots$ **do**

3: Let $\tau_k \leftarrow \frac{2}{k+3}$.
$\boldsymbol{\beta}_k \leftarrow (1-\tau_k)\mathbf{u}_k + \tau_k\mathbf{u}_{\mu_k}(\mathbf{c}_k)$.

4: Update: $\mathbf{c}_{k+1} \leftarrow (1-\tau_k)\mathbf{c}_k + \tau_k\mathbf{c}(\boldsymbol{\beta}_k)$.
$\boldsymbol{\xi}_k \leftarrow V\left(\mathbf{u}_{\mu_k}(\mathbf{c}_k), \frac{-\tau_k}{(1-\tau_k)\mu_k}\nabla D(\boldsymbol{\beta}_k)\right)$.
$\mathbf{u}_{k+1} \leftarrow (1-\tau_k)\mathbf{u}_k + \tau_k\boldsymbol{\xi}_k$.
$\mu_{k+1} \leftarrow (1-\tau_k)\mu_k$.

5: **end for**

Recall that Nesterov's framework requires a σ-s.c. prox-function on Q_2, which in our case is the n-dimensional simplex Δ_n. The values of L and $\mathcal{D}$ in (3.20) depend on the choice of the prox-function $d(\cdot)$ and the corresponding norm chosen for Q_2. We present two natural choices.

3.1.1 Q_2 endowed with ℓ_2 norm: We endow Q_2 with the ℓ_2 norm, and let

$$d(\mathbf{u}) = \frac{\sigma}{2}\|\mathbf{u} - \mathbf{u}_0\|^2 \tag{3.21}$$

with $\mathbf{u}_0 = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right) \in \mathbb{R}^n$ be the prox-function. Clearly d is strongly convex with respect to the ℓ_2 norm.

For this choice $\mathbf{u}_0 = \operatorname{argmin}_{\mathbf{u}\in\Delta_n} d(\mathbf{u})$, $d(\mathbf{u}_0) = 0$, and

$$\begin{aligned}\mathcal{D} &= \max_{\mathbf{u}\in\Delta_n} d(\mathbf{u}) \\ &= \frac{\sigma}{2}\max_{\mathbf{u}\in\Delta_n} \|\mathbf{u}-\mathbf{u}_0\|^2 = \frac{\sigma}{2}\left(1-\frac{1}{n}\right)^2 \leq \frac{\sigma}{2}.\end{aligned} \tag{3.22}$$

On the other hand,

$$\begin{aligned}L &= \frac{1}{2}\left\|\mathbf{A}\mathbf{A}^\top\right\| = \frac{1}{2}\max_{\|\mathbf{c}\|=\|\mathbf{u}\|=1} \mathbf{c}^\top\mathbf{A}\mathbf{A}^\top\mathbf{u} \\ &= \frac{1}{2}\lambda_{max}\left(\mathbf{A}\mathbf{A}^\top\right),\end{aligned} \tag{3.23}$$

where $\|\mathbf{c}\|$ and $\|\mathbf{u}\|$ use ℓ_2 norm, and $\lambda_{max}\left(\mathbf{A}\mathbf{A}^\top\right)$, which we denote by $\mathcal{L}$ in the sequel, denotes the maximum eigenvalue of the matrix $\mathbf{A}\mathbf{A}^\top$. Plugging the above $\mathcal{D}$ and L into (3.20) immediately yields:

$$J(\mathbf{c}_k) - D(\mathbf{u}_k) \leq \frac{3\mathcal{L}}{(k+1)(k+2)}. \tag{3.24}$$

Solving for k by setting $J(\mathbf{c}_k) - D(\mathbf{u}_k) \leq \epsilon$ yields a $O(\mathcal{L}/\sqrt{\epsilon})$ bound on the number of iterations.

3.1.2 Q_2 endowed with the ℓ_1 norm: If we endow the space Q_2 with the ℓ_1 norm, a natural choice for the strongly convex prox-function is the entropy function. We set

$$d(\mathbf{u}) = \frac{\sigma}{2}\sum_{i=1}^{n}\left(u_i \ln u_i - u_{0,i}\ln u_{0,i}\right) \tag{3.25}$$

with $\mathbf{u}_0$ defined as before, which gives $\mathbf{u}_0 = \operatorname{argmin}_{\mathbf{u}\in\Delta_n} d(\mathbf{u})$, $d(\mathbf{u}_0) = 0$. The strong convexity of the entropy with respect to the ℓ_1 norm follows from a fundamental inequality in information theory [14]. (Also see Proposition 5.1 in [2] for a simplified exposition). Noting that $\sum_i u_{0,i}\ln u_{0,i} = -\ln n$ gives

$$\begin{aligned}\mathcal{D} &= \max_{\mathbf{u}\in\Delta_n} d(\mathbf{u}) \\ &= \frac{\sigma}{2}\left(\max_{\mathbf{u}\in\Delta_n}\sum_{i=1}^{d} u_i\ln u_i + \ln n\right) = \frac{\sigma}{2}\ln n.\end{aligned} \tag{3.26}$$

As is standard [see *e.g.* 26], if we assume that the input data points lie inside a ball of radius $\mathcal{Q}$, that is, $\max_i \|\mathbf{x}_i\| \leq \mathcal{Q}$ then we can write

$$\begin{aligned}L &= \frac{1}{2}\left\|\mathbf{A}\mathbf{A}^\top\right\| = \frac{1}{2}\max_{\|\mathbf{v}\|_1=\|\mathbf{u}\|_1=1} \mathbf{v}^\top\mathbf{A}\mathbf{A}^\top\mathbf{u} \\ &= \max_i \max_{\|\mathbf{v}\|_1=1} \left|\mathbf{v}^\top\mathbf{A}\mathbf{x}_i\right| \\ &= 2\max_i\max_j \left|\mathbf{x}_j^\top\mathbf{x}_i\right| = 2\max_i\|\mathbf{x}_i\|^2 = 2\mathcal{Q}^2.\end{aligned} \tag{3.27}$$

Plugging this back into (3.20) and using (3.26) yields

$$J(\mathbf{c}_k) - D(\mathbf{u}_k) \leq \frac{6\mathcal{Q}^2\ln n}{(k+1)(k+2)}. \tag{3.28}$$

Therefore, to obtain an ϵ accurate solution of (3.12) it suffices to ensure that the above expression is $\leq \epsilon$. Solving for k yields a $O^*(\mathcal{Q}/\sqrt{\epsilon})$ bound on the number of iterations.

3.2 Time Complexity Per Iteration

Each iteration of Algorithm 1 requires computing $\mathbf{c}(\mathbf{u})$, $\mathbf{u}_\mu(\mathbf{c})$, and $V(\mathbf{u},\mathbf{g})$ (see (3.16), (3.17), and (3.18)). All other operations either require constant or linear time. We now show that each of these three maps can be computed in $O(nd)$ time.

By computing the gradient of $\langle\mathbf{A}\mathbf{c},\mathbf{u}\rangle + \|\mathbf{c}\|^2$ and setting it to zero we can show that

$$\mathbf{c}(\mathbf{u}) = -\frac{1}{2}\mathbf{A}^\top\mathbf{u}. \tag{3.29}$$

Since $\mathbf{A}$ is a $n\times d$ matrix computing $\mathbf{c}(\mathbf{u})$ takes $O(nd)$ time.

3.2.1 Computing $\mathbf{u}_\mu(\mathbf{c})$ and $V(\mathbf{u},\mathbf{g})$ for the ℓ_2 norm. When the prox-function (3.21) is used, computation of $\mathbf{u}_\mu(\mathbf{c})$ can be cast as the following Quadratic programming (QP) problem with linear constraints:

$$\begin{aligned}\min_{\mathbf{u}} \frac{\mu\sigma}{2}\|\mathbf{u}\|^2 - \langle\mathbf{A}\mathbf{c}+\mathbf{b}+\mu\sigma\mathbf{u}_0,\mathbf{u}\rangle \\ \text{s.t. } \sum_i u_i = 1 \text{ and } 0\leq u_i\leq 1.\end{aligned} \tag{3.30}$$

This is the problem of calculating Euclidean projections onto a unit simplex. Computing $\mathbf{A}\mathbf{c}+\mathbf{b}+\mu\sigma\mathbf{u}_0$ requires $O(nd)$ time. Given this, we present an algorithm in Appendix B that can be applied to solve (3.30) in $O(n)$ time. Similarly after some simple algebraic manipulation, the computation of $V(\mathbf{u},\mathbf{g})$ for

$$\mathbf{g} = \frac{-\tau_k}{(1-\tau_k)\mu_k}\nabla D(\boldsymbol{\beta}_k) = \frac{-\tau_k}{(1-\tau_k)\mu_k}(\mathbf{b} - \frac{1}{2}\mathbf{A}\mathbf{A}^\top\boldsymbol{\beta}_k)$$

can be cast as a RP with linear constraints as follows:

$$\begin{aligned}\min_{\mathbf{v}} \frac{\sigma}{2}\|\mathbf{v}\|^2 - \langle\sigma\mathbf{u}-\mathbf{g},\mathbf{v}\rangle \\ \text{s.t. } \sum_i v_i = 1 \text{ and } 0\leq v_i\leq 1.\end{aligned} \tag{3.31}$$

Computing $\mathbf{g}$ takes $O(nd)$ effort[1], and the algorithm in Appendix B can be used to solve the above QP in $O(n)$ time.

[1] To compute $\mathbf{A}\mathbf{A}^\top\boldsymbol{\beta}_k$ efficiently we first compute $\mathbf{a} = \mathbf{A}^\top\boldsymbol{\beta}_k$ and then compute $\mathbf{A}\mathbf{a}$.

We note that the linear time algorithm in Appendix B is key to obtaining our computational complexity bounds. It has been rediscovered independently by many authors (including us), with the earliest known reference to the best of our knowledge being Pardalos and Kovoor [21]. Some recent work by Dai and Fletcher [7] and Liu and Ye [15] has focused on improving the practical performance while Duchi et al. [8] give an expected linear time algorithm for the same problem.

3.2.2 Computing $\mathbf{u}_\mu(\mathbf{c})$ and $V(\mathbf{u},\mathbf{g})$ for the ℓ_1 norm. For the ℓ_1 prox-function (3.25), the computation of the minimizer $\mathbf{u}_\mu(\mathbf{c})$ of (3.17) can be cast as the following optimization problem

$$\min_{\mathbf{u}} \sum_{i=1}^{n} u_i \ln u_i - \sum_{i=1}^{n} r_i u_i \quad \text{s.t.} \quad \sum_i u_i = 1 \text{ and } 0 \le u_i \le 1. \tag{3.32}$$

where $r_i = \frac{2}{\mu\sigma}[\mathbf{A}\mathbf{c}+\mathbf{b}]_i$. The corresponding minimizer can be calculated in close form (lemma 4 in [18], also see Appendix C):

$$u_i = \frac{\exp(r_i)}{\sum_{i=1}^{n}\exp(r_i)} \qquad i = 1,\ldots,n. \tag{3.33}$$

The complexity of calculating $\mathbf{u}_\mu(\mathbf{c})$ is thus only dependent on that of computing $\mathbf{A}\mathbf{c}+\mathbf{b}$ and therefore takes $O(nd)$ time as described before.

Similarly, the computation of $V(\mathbf{s},\mathbf{g})$ can be cast in the same form as (3.32), but with $r_i = \ln s_i + 1 - \frac{2}{\sigma}g_i$. So its solution also has the form (3.33), and the only computational cost is to compute $\mathbf{g}$ which is $O(nd)$ as is discussed in §3.2.1.

3.3 Multiplicative versus Additive Approximation: Scale Invariance

Existing approximation algorithms for the MEB problem based on coresets provide a multiplicative approximation. Given a set of points, the coreset algorithms output a center $\mathbf{c}$ and radius R such that all the given points lie inside the ball of radius $R(1+\epsilon)$ centered at $\mathbf{c}$. In contrast, our algorithm produces a center and a radius R such that $R^{*2} \le R^2 \le R^{*2} + \epsilon'$, where R^* denotes the radius of the optimal minimum enclosing ball.

At first glance the two types of guarantees do not look directly comparable since the additive guarantees seem to vary with change of scale. We show how our algorithms can produce a scale invariant multiplicative approximation. For brevity, we will only discuss the case when Q_2 is endowed with the ℓ_1 norm. Exactly the same arguments hold when Q_2 is endowed with the ℓ_2 norm.

To produce an ϵ_M *scale invariant* multiplicative approximation with our algorithm, set $\epsilon' = \epsilon_M R^{*2}$. In view of (3.28) it follows that $R^{*2} \le R^2 \le R^{*2}(1+\epsilon_M)$ whenever[2]

$$\frac{6\mathcal{Q}^2 \ln n}{(k+1)(k+2)} \le R^{*2}\epsilon_M. \tag{3.34}$$

Solving for k obtains

$$\frac{\mathcal{Q}}{R^*}\sqrt{\frac{6\ln n}{\epsilon_M}} \le k. \tag{3.35}$$

However, since R^* is unknown this bound cannot be used as a practical stopping criterion. Instead, one can use the following observation: select a arbitrary pair of points $\mathbf{x}_i$ and $\mathbf{x}_j$ from S and compute $\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|$. Denote this quantity by $\mathcal{P}$, and let $\mathbf{c}^*$ be the center of the optimal MEB. Clearly

$$\begin{aligned}\mathcal{P} = \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\| &\le \frac{1}{2}\|\mathbf{x}_i - \mathbf{c}^*\| + \frac{1}{2}\|\mathbf{x}_j - \mathbf{c}^*\| \\ &= \frac{1}{2}R^* + \frac{1}{2}R^* = R^*.\end{aligned} \tag{3.36}$$

Therefore replacing R^* by $\mathcal{P}$ in (3.35) yields the following computable upper bound on the number of iterations:

$$\frac{\mathcal{Q}}{\mathcal{P}}\sqrt{\frac{6\ln n}{\epsilon_M}} \le k. \tag{3.37}$$

This shows that $O^*(1/\sqrt{\epsilon_M})$ iterations of our algorithm suffice to produce a ϵ_M-multiplicative approximation.

Note that just like in the case of coreset based algorithms this bound is scaling invariant, that is, if all the $\mathbf{x}_i$ in S are scaled by a factor $\alpha > 0$ the bound still holds. To see this one merely has to observe that after scaling $\mathcal{P}$ becomes $\alpha\mathcal{P}$ and $\mathcal{Q}$ becomes $\alpha\mathcal{Q}$, but the ratio $\mathcal{Q}/\mathcal{P}$ which appears in (3.37) remains unchanged. Thus while our algorithm is modeled as a generic optimization formulation, it is possible to obtain scale invariance by appropriately choosing ϵ' to be $\epsilon_M\mathcal{P}^2$.

It should be noted that the set of points can be easily bounded in a ball of size $\mathcal{Q}$ by arbitrarily choosing a point $\mathbf{x}_i$, finding the point $\mathbf{x}_j$ farthest from it and then shifting the axes so that $\mathbf{x}_i$ became the origin and scale all points by $\frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|}$. The resulting input points then have norm at most 1.

[2] We prove our bounds in terms of R^2 but it is trivial to convert this to a bound in terms of R.

4 Minimum Enclosing Convex Polytope

In the Minimum Enclosing Convex Polytope (MECP) problem we are given a polytope of fixed shape which can be translated and magnified but rotations are not allowed. Furthermore, we assume that the polytope has a finite number of faces and hence it can be expressed as an intersection of a finite number of halfspaces:

$$\langle \mathbf{w}_i, \mathbf{x} - \mathbf{c} \rangle \leq t_i \qquad i = 1, 2, ...m,$$

where $\mathbf{c}$ is the *center* of the polytope about which it is magnified. Given a set of points $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ we want to find the minimum magnification of the convex polytope that encloses all the points in S.

4.1 Formulation as an Optimization problem Clearly an enclosing polytope is one for which $\langle \mathbf{w}_i, \mathbf{x}_j - \mathbf{c} \rangle \leq t_i$ for all i and j. This observation helps us to cast the MECP problem as the following optimization problem

$$\min_{R \in \mathbb{R}, \mathbf{c} \in Q_1} R \qquad \text{s.t. } \langle \mathbf{w}_i, \mathbf{x}_j - \mathbf{c} \rangle \leq R t_i$$

or equivalently as

$$\min_{R \in \mathbb{R}, \mathbf{c} \in Q_1} R \qquad \text{s.t. } \left\langle \frac{\mathbf{w}_i}{t_i}, \mathbf{x}_j - \mathbf{c} \right\rangle \leq R \quad \forall i, j.$$

Here R is the scale of magnification of the polygon and $Q_1 \subset \mathbb{R}^d$ is a bounded set, for example, a ball of certain fixed radius $\mathcal{Q}$ centered at the origin which is assumed to contain the solution $\mathbf{c}$. Usually Q_1 is taken to be a ball which contains all the points in S but this need not always be the case. Also we will assume that $\mathbf{w}_i/t_i$ lies inside a ball of radius $\mathcal{W}$. Writing $\tilde{\mathbf{w}}_i = \mathbf{w}_i/t_i$ the problem can be expressed as,

(4.38)
$$\min_{\mathbf{c} \in Q_1} \max_{i,j} \langle \tilde{\mathbf{w}}_i, \mathbf{x}_j - \mathbf{c} \rangle = \min_{\mathbf{c} \in Q_1} \max_{\mathbf{u} \in \Delta_{mn}} \sum_{i,j} u_{ij} (\langle \tilde{\mathbf{w}}_i, \mathbf{x}_j - \mathbf{c} \rangle).$$

Notice that, as before, we have replaced the maximization over a finite set by a maximization over the simplex. Clearly this problem can be rewritten as

(4.39)
$$\min_{\mathbf{c} \in Q_1} J(\mathbf{c}) \text{ where}$$
$$J(\mathbf{c}) = \max_{\mathbf{u} \in \Delta_{mn}} \sum_{i,j} u_{ij} (\langle \tilde{\mathbf{w}}_i, \mathbf{x}_j - \mathbf{c} \rangle)$$
$$= \max_{\mathbf{u} \in \Delta_{mn}} \{ \langle \mathbf{A}\mathbf{c}, \mathbf{u} \rangle + \langle \mathbf{u}, \mathbf{b} \rangle \}.$$

We used $\mathbf{A} = [\underbrace{-\tilde{\mathbf{w}}_1, \ldots}_{n \text{ times}} \underbrace{-\tilde{\mathbf{w}}_2, \ldots}_{n \text{ times}} \ldots - \tilde{\mathbf{w}}_m]^\top$, a $mn \times d$ matrix with each $\tilde{\mathbf{w}}_i$ repeated n times as the rows and $\mathbf{b}$ a mn dimensional vector with $b_{ij} = \langle \mathbf{w}_i, \mathbf{x}_j \rangle$ to write the above expression. Clearly $J(\mathbf{c})$ can be identified with (2.4) by setting $Q_2 = \Delta_{mn}$, $f(\mathbf{c}) = 0$ and $g(\mathbf{u}) = -\langle \mathbf{b}, \mathbf{u} \rangle$. Here, g is 0-*l.c.g*, however, f is no longer strongly convex. Therefore, we will work with the following function

$$J_\eta(\mathbf{c}) = \eta \|\mathbf{c}\|^2 + \max_{\mathbf{u} \in \Delta_{mn}} \{ \langle \mathbf{A}\mathbf{c}, \mathbf{u} \rangle - g(\mathbf{u}) \}.$$

Let $J_\eta^* = \min_{\mathbf{c} \in Q_1} J_\eta(\mathbf{c})$ and $J^* = \min_{\mathbf{c} \in Q_1} J(\mathbf{c})$. Since $\|\mathbf{c}\|^2 \leq \mathcal{Q}^2$ for all $c \in Q_1$ we have that

$$J_\eta^* \leq J^* + \eta \mathcal{Q}^2.$$

Suppose we can minimize J_η to $\epsilon/2$ precision, that is, find a $\mathbf{c}$ such that $J_\eta(\mathbf{c}) \leq J_\eta^* + \epsilon/2$ then the above observation allows us to write the following series of inequalities

$$J(\mathbf{c}) \leq J_\eta(\mathbf{c}) \leq J_\eta^* + \frac{\epsilon}{2} \leq J^* + \frac{\epsilon}{2} + \eta \mathcal{Q}^2.$$

In other words, every $\epsilon/2$ accurate solution of J_η is a $\epsilon/2 + \eta\mathcal{Q}^2$ accurate solution of J. In particular, if we set $\eta = \epsilon/2\mathcal{Q}^2$ then every $\epsilon/2$ accurate solution of J_η is an ϵ accurate solution of J. Furthermore, J_η is nearly identical to (3.12) except that $\|\mathbf{c}\|^2$ is now replaced by $\frac{\epsilon}{2\mathcal{Q}^2} \|\mathbf{c}\|^2$. Arguments analogous to the ones we made in §3.1 can be used to show that $D_\eta(\mathbf{u})$ is $L_\eta := \frac{\mathcal{Q}^2}{2\epsilon} \|\mathbf{A}\mathbf{A}^\top\|$-*l.c.g*.

Note that $\tilde{\mathbf{w}}_i$ are assumed to lie inside a ball of radius $\mathcal{W}$. When Q_2 is endowed with the ℓ_1 norm, by an argument analogous to (3.27), it follows that $L_\eta = \mathcal{Q}^2\mathcal{W}^2/2\epsilon$. Plugging this into (3.20) and using (3.26) obtains

(4.40)
$$J(\mathbf{c}_k) - D(\mathbf{u}_k) \leq \frac{3\mathcal{Q}^2\mathcal{W}^2 \ln mn}{2\epsilon(k+1)(k+2)}.$$

In order to obtain an $\epsilon/2$ accurate solution we need to solve for k by setting $\frac{3\mathcal{Q}^2\mathcal{W}^2 \ln mn}{2\epsilon(k+1)(k+2)} \leq \epsilon/2$. This yields an $O^*(\mathcal{Q}/\epsilon)$ iteration bound as claimed. Extending the arguments for MEB to the MECP case, the per iteration complexity of $O(mnd)$ can readily be established.

On the other hand, if we endow Q_2 with the ℓ_2 ball, using arguments similar to (3.23), we have $L_\eta = Q^2\mathcal{L}/2\epsilon$ where $\mathcal{L}$ is again used to denote the maximum eigenvalue of $\mathbf{A}\mathbf{A}^\top$. Plugging it back into (3.20) and using (3.22) yields

(4.41)
$$J(\mathbf{c}_k) - D(\mathbf{u}_k) \leq \frac{3\mathcal{Q}^2\mathcal{L}}{2\epsilon(k+1)(k+2)}.$$

Just like in the ℓ_1 case, obtaining an $\epsilon/2$ accurate solution requires us to solve for k by setting $\frac{3\mathcal{Q}^2\mathcal{L}}{2\epsilon(k+1)(k+2)} \leq$

$\epsilon/2$ which yields an $O(\mathcal{Q}\sqrt{\mathcal{L}}/\epsilon)$ iteration bound. The per iteration complexity of $O(mnd)$ follows in the same way by extending the arguments for MEB (Q_2 endowed with the ℓ_2 norm) to the MECP problem.

5 Applications to Machine Learning

The connection between the MEB problem and SVMs has been discussed in a number of publications [6, 10, 11]. Practical algorithms using coresets were also proposed in Tsang et al. [24] and Tsang et al. [23]. In all these cases, our MEB algorithm can be plugged in as a subroutine and will yield corresponding speedups. We describe these kernel based algorithms and their connection with MEB in appendix D. In this section, we present two machine learning problems wherein our algorithms lead to algorithms with better dependence on ϵ than existing coreset based approaches.

5.1 Finding Large Margin Classifiers In Har-Peled et al. [11] a coreset algorithm (Coreset SVM) for finding the maximum margin hyperplane was described. It turns out that our MECP algorithm can be specialized to their setting, and yields improved bounds. Briefly, given m labeled data points $(\mathbf{z}_i, y_i)$ with $\mathbf{z}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ the maximum margin hyperplane can be found by solving[3] $\operatorname{argmax}_{\mathbf{c}\in\mathbb{R}^d,\|\mathbf{c}\|=1} \min_i y_i \langle \mathbf{z}_i, \mathbf{c}\rangle$. Equivalently, by defining $\tilde{\mathbf{w}}_i = y_i \cdot \mathbf{z}_i$ one can solve

$$\operatorname*{argmin}_{\mathbf{c}\in\mathbb{R}^d,\|\mathbf{c}\|=1} \max_i \langle \tilde{\mathbf{w}}_i, -\mathbf{c}\rangle . \tag{5.42}$$

The above problem can be identified with (4.38) if we set S to be the empty set and Q_1 to be $\left\{\mathbf{c} \in \mathbb{R}^d \text{ s.t. } \|\mathbf{c}\| = 1\right\}$. With this substitution our MECP algorithm can directly be applied to solve (5.42). In this case $\mathcal{Q} = 1$ and the bound (4.40) yields

$$\frac{3\mathcal{W}^2 \ln m}{2\epsilon(k+1)(k+2)} \leq \frac{\epsilon}{2}. \tag{5.43}$$

Solving for k shows that $O^*(\mathcal{W}/\epsilon)$ iterations suffice to obtain an ϵ accurate solution of (5.42).

The Coreset SVM algorithm produces a multiplicative approximation. To compare with the bounds given by Har-Peled et al. [11] we follow the same scheme described in §3.3. Let ρ^* denote the optimal margin, that is, $\min_{\mathbf{c}\in\mathbb{R}^d,\|\mathbf{c}\|=1} \max_i \langle \tilde{\mathbf{w}}_i, -\mathbf{c}\rangle$, and set $\epsilon = \rho^* \epsilon_M$. Substituting into (5.43) and solving for k shows that to produce a ϵ_M multiplicative approximation with our algorithm, it suffices to take

$$k \geq \sqrt{3}\left(\frac{\mathcal{W}\ln m}{\rho^*}\right)\frac{1}{\epsilon_M} \tag{5.44}$$

steps. In contrast, Har-Peled et al. [11] compute a Coreset SVM C of size

$$|C| = O\left(\left(\frac{\mathcal{W}}{\rho^*}\right)^2 \frac{1}{\epsilon_M}\right) \tag{5.45}$$

in $|C|$ iterations. Furthermore, the computational complexity of Coreset SVM is $O(md|C| + |C|T(|C|))$, where $T(|C|)$ is the cost of training a SVM on $|C|$ points. Since each iteration requires only $O(md)$ effort, our algorithm has a computational complexity of $O^*\left(md\frac{\mathcal{W}}{\rho^*}\frac{1}{\epsilon_M}\right)$. However, there is one notable difference between the two algorithms. Coreset SVM produces a sparse solution in terms of the number of support vectors, while our algorithm comes with no such guarantees.

5.2 Computing Polytope Distance Given a set of points $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ the polytope distance is defined as the shortest distance ρ of any point inside the convex hull of S, $\operatorname{conv}(S)$, to the origin [10]. Equivalently, we are looking for the vector with the smallest norm in $\operatorname{conv}(S)$. A variant is to compute the distance between two polytopes given by the points $S_+ = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$ and $S_- = \{\mathbf{z}'_1, \mathbf{z}'_2, \ldots, \mathbf{z}'_{n'}\}$. This problem can be solved by finding the polytope distance of the Minkowski difference of $\operatorname{conv}(S_+)$ and $\operatorname{conv}(S_-)$ [3]. Since the arguments for both cases are by and large very similar we will stick with the simpler formulation in this paper. The polytope distance problem has a number of applications in machine learning. A partial (and by no means exhaustive) list of relevant publications includes Bennett and Bredensteiner [3], Gärtner and Jaggi [10], and Keerthi et al. [13].

To analyze this problem in our setting we start with a technical lemma. A version of this lemma also appears as Theorem A.2 in Bennett and Bredensteiner [3].

LEMMA 5.1. *The following two problems are duals of each other:*

$$\min_{\mathbf{c}} J(\mathbf{c}) := \max_i \langle \mathbf{c} - \mathbf{x}_i, \mathbf{c}\rangle \tag{5.46}$$

$$\max_{\mathbf{u}\in\Delta_n} D(\mathbf{u}) := -\frac{1}{4}\mathbf{u}^\top \mathbf{A}\mathbf{A}^\top \mathbf{u} = -\frac{1}{4}\left\|\mathbf{A}^\top \mathbf{u}\right\|^2, \tag{5.47}$$

where $\mathbf{A} = -[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top$.

[3] Har-Peled et al. [11] use $\mathbf{x}_i$ for the training data and $\mathbf{w}$ for the hyperplane. Here we use $\mathbf{z}_i$ and $\mathbf{c}$ respectively to be consistent with our notation.

Proof. First rewrite the objective function in (5.46) as

$$J(\mathbf{c}) = \|\mathbf{c}\|^2 + \max_i \langle -\mathbf{x}_i, \mathbf{c} \rangle = \|\mathbf{c}\|^2 + \max_{\mathbf{u} \in \Delta_n} \langle \mathbf{u}, \mathbf{A}\mathbf{c} \rangle .$$

This can be identified with (2.4) by setting $Q_1 = \mathbb{R}^d$, $Q_2 = \Delta_n$, $f(\mathbf{c}) = \|\mathbf{c}\|^2$, and $g(\mathbf{u}) = 0$. The dual problem (5.47) can directly be read off from (2.7) by noting that that the Fenchel dual of $\|\cdot\|^2$ is $\frac{1}{4}\|\cdot\|^2$.

Clearly (5.47) computes the vector with the smallest norm in $\mathrm{conv}(S)$, which is equivalent to the polytope distance problem. Furthermore, (5.46) and (5.47) are identical to (3.12) and (3.13) respectively with $\mathbf{b} = \mathbf{0}$. Therefore the algorithm we described in §3 can be applied with minor modifications to yield a $O^*(nd\mathcal{Q}/\sqrt{\epsilon})$ algorithm for this problem also. Since the Gärtner and Jaggi [10] algorithm selects coresets, at most $O(1/\epsilon)$ components of $\mathbf{u}$ are non-zero. However, in our case no such guarantees hold.

6 Conclusions and Future Work

We presented two approximation algorithms for the MEB problem whose running times are $O(nd\mathcal{L}/\sqrt{\epsilon})$ and $O^*(nd\mathcal{Q}/\sqrt{\epsilon})$ respectively. In general the two algorithms are incomparable because the constants $\mathcal{Q}$ and $\mathcal{L}$ depend on different aspects of the data. In some pathological cases $\mathcal{L}$ can be as large as $O(n)$ while $\mathcal{Q}$ is independent of the number of points. These aspects must be borne in mind when selecting an appropriate algorithm for the task on hand.

Unlike existing algorithms, which rely heavily on geometric properties, our algorithms are motivated and derived purely from a convex optimization viewpoint. We extended our analysis to the MECP problem and obtained $O(mnd\mathcal{L}/\epsilon)$ and $O^*(mnd\mathcal{Q}/\epsilon)$ algorithms. Our treatment yields algorithms with better dependence on ϵ. Preliminary experiments, which can be found Appendix E, suggest that our algorithms are also competitive on problems with a large number of data points.

The nd term in our bounds can be replaced by $\mathrm{nnz}(\mathbf{A})$, where $\mathrm{nnz}(\cdot)$ denotes the number of non-zero entries in the matrix $\mathbf{A} = [\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_n]^\top$. In some cases, when the dataset is sparse, $\mathrm{nnz}(\mathbf{A})$ may be significantly smaller than nd.

A more general version of the MECP problem was studied by Panigrahy [20]. In his setting the convex polytope is allowed translations, magnifications, and rotations. A simple greedy approach (very reminiscent of coresets) yields an $(1+\epsilon)$ multiplicative approximation algorithm which takes $O(1/\epsilon^2)$ iterations to converge. Please consult Panigrahy [20] for details.

Recently, Clarkson [6] significantly expanded the scope of the coreset based algorithms and showed that they can be viewed as instances of a sparse greedy algorithm which minimizes a convex function over a unit simplex. This interpretation opens up the exciting possibility that some of our methods can be carried over to the setting discussed by Clarkson [6] and hence will lead to speedups for a number of other problems.

A natural question to ask is the following: Can our algorithms be extended to deal with arbitrary convex shapes? In other words, given an arbitrary convex shape can we find the optimal magnification and translation that is required to enclose the set of points $\{\mathbf{x}_j\}_{j=1}^n$ at hand. Unfortunately, a straightforward extension seems rather difficult. Even though it is well known that every convex shape can be described as an intersection of half planes, the number of half planes need not be finite. In such a case our MECP algorithm, which crucially relies on the number of half planes being finite, is clearly not applicable. Somewhat surprisingly, we are able to obtain a $O(1/\sqrt{\epsilon})$ algorithm for the MEB problem, even though a ball is made up of an intersection of infinitely many half planes. Clearly, the strong convexity of the objective function plays an important role in this context. We are currently trying to characterize such problems in the hope that this investigation will lead to efficient algorithms for a number of other related problems.

Rotations are a natural concept when working with geometric algorithms. This does not naturally carry over to our setting where we use convex optimization. In order to introduce rotations one has to work with orthogonal matrices, which significantly complicates the optimization strategy. A fruitful pursuit would be to investigate if the insights gained from coresets can be used to solve complicated optimization problems which involve orthogonal matrices efficiently.

Even though we are only beginning to scratch the surface on exploring connections between optimization and computational geometry, we firmly believe that this cross pollination will lead to exciting new algorithms in both areas.

Acknowledgements

The authors would like to thank Ambuj Tewari for various useful discussions. We also thank the anonymous referees from SODA 2010, 2011 and SoCG 2010 who provided excellent feedback on earlier drafts of this manuscript and pointed us to the work of Gärtner and Jaggi [10].

References

[1] M. Badoiu and K. L. Clarkson. Optimal Core-Sets for Balls. In *Computational Geometry: Theory and Applications*, 2002.

[2] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[3] K. P. Bennett and E. J. Bredensteiner. Geometry in learning. In C. Gorini, E. Hart, W. Meyer, and T. Phillips, editors, *Geometry at Work*, Washington, D.C., 1998. Mathematical Association of America. Available http://www.math.rpi.edu/~bennek/geometry2.ps.

[4] J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. CMS books in Mathematics. Canadian Mathematical Society, 2000.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

[6] K. L. Clarkson. Coresets, Sparse Greedy Approximation, and the Frank-Wolfe algorithm. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 922–931. Society for Industrial and Applied Mathematics, 2008.

[7] Y.-H. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming: Series A and B archive*, 106(3):403–421, 2006.

[8] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the International Conference on Machine Learning*, 2008.

[9] D. J. Elzinga and D. W. Hearn. The minimum covering sphere problem. *Management Science*, 19: 96–104, 1972.

[10] B. Gärtner and M. Jaggi. Coresets for polytope distance. In *Annual Symposium on Computational Geometry*, 2009.

[11] S. Har-Peled, D. Roth, and D. Zimak. Maximum margin coresets for active and noise tolerance learning. In *International Joint Conference on Artificial Intelligence*, pages 836–841, 2007.

[12] J. B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms, I and II*, volume 305 and 306. Springer-Verlag, 1993.

[13] S.S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, January 2000.

[14] J. H. B. Kemperman. On the optimum rate of transmitting information. *Annals of Mathematical Statistics*, 40(6):2156–2177, 1969.

[15] J. Liu and J. Ye. Efficient euclidean projections in linear time. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, 2009.

[16] N. Megiddo. Linear Programming in Linear Time when the Dimension is fixed. *Journal of ACM*, 31 (1):114–127, 1984.

[17] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Soviet Mathematics Doklady*, 269:543–547, 1983.

[18] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[19] Y. Nesterov. Excessive Gap Technique in Nonsmooth Convex Minimization. *SIAM Journal on Optimization*, 16(1):235–249, 2005. ISSN 1052-6234.

[20] R. Panigrahy. Minimum Enclosing Polytope in High Dimensions. *CoRR*, cs.CG/0407020, 2004.

[21] P. M. Pardalos and N. Kovoor. An Algorithm for Singly Constrained Class of Quadratic Programs Subject to Upper and Lower Bounds. *Mathematical Programming*, 46:321–328, 1990.

[22] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[23] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005. ISSN 1532-4435.

[24] I. W. Tsang, A. Kocsor, and J. T. Kwok. Simpler core vector machines with enclosing balls. In *Proceedings of the International Conference on Machine Learning*, pages 911–918, 2007.

[25] E. Welzl. Minimum Enclosing Disks (balls and ellipsoids). *Lecture Notes in Computer Science*, 555:359–370, 1991.

[26] E. A. Yildirim. Two algorithms for the minimum enclosing ball problem. In *SIAM Journal of Optimization*, pages 1368–1391, 2008.

Appendix

A Proof of theorem 3.2

Our proof is by and large derived using results from Nesterov [19]. We begin with a technical lemma.

LEMMA A.1. *(Lemma 7.2 of Nesterov [19]) For any* $\mathbf{u}$ *and* $\bar{\mathbf{u}}$*, we have*

$$D(\mathbf{u}) + \langle \nabla D(\mathbf{u}), \bar{\mathbf{u}} - \mathbf{u} \rangle = \langle \mathbf{A}\mathbf{c}(\mathbf{u}) + \mathbf{b}, \bar{\mathbf{u}} \rangle + \|\mathbf{c}(\mathbf{u})\|^2 .$$

Proof. Direct calculation by using (3.13), (3.14), and (3.29) yields

$$\begin{aligned}
&D(\mathbf{u}) + \langle \nabla D(\mathbf{u}), \bar{\mathbf{u}} - \mathbf{u} \rangle \\
&= \langle \mathbf{u}, \mathbf{b} \rangle - \frac{1}{4}\mathbf{u}^\top \mathbf{A}\mathbf{A}^\top \mathbf{u} + \left\langle \mathbf{b} - \frac{1}{2}\mathbf{A}\mathbf{A}^\top \mathbf{u}, \bar{\mathbf{u}} - \mathbf{u} \right\rangle \\
&= \langle \bar{\mathbf{u}}, \mathbf{b} \rangle - \left\langle \frac{1}{2}\mathbf{A}\mathbf{A}^\top \mathbf{u}, \bar{\mathbf{u}} \right\rangle + \frac{1}{4}\mathbf{u}^\top \mathbf{A}\mathbf{A}^\top \mathbf{u} \\
&= \langle \mathbf{A}\mathbf{c}(\mathbf{u}) + \mathbf{b}, \bar{\mathbf{u}} \rangle + \|\mathbf{c}(\mathbf{u})\|^2 .
\end{aligned}$$

For convenience, we introduce the Bregman divergence and rewrite the map $V(\mathbf{s}, \mathbf{g})$. Define the Bregman divergence

$$\Delta(\mathbf{u}', \mathbf{u}) := d(\mathbf{u}') - d(\mathbf{u}) - \langle \nabla d(\mathbf{u}), \mathbf{u}' - \mathbf{u} \rangle .$$

Then it is easy to see that $V(\mathbf{s}, \mathbf{g})$ can be equivalently defined as

$$V(\mathbf{s}, \mathbf{g}) := \operatorname*{argmin}_{\mathbf{u} \in Q_2} \langle \mathbf{g}, \mathbf{u} - \mathbf{s} \rangle + \Delta(\mathbf{u}, \mathbf{s}).$$

Since d is σ-s.c., so

$$\Delta(\mathbf{u}', \mathbf{u}) \geq \frac{\sigma}{2} \|\mathbf{u}' - \mathbf{u}\|^2 . \tag{A.1}$$

As $\mathbf{u}_0$ minimizes d over Q_2, we have

$$\langle \nabla d(\mathbf{u}_0), \mathbf{u} - \mathbf{u}_0 \rangle \geq 0. \qquad \mathbf{u} \in Q_2. \tag{A.2}$$

We first show that the initial $\mathbf{w}_1$ and $\boldsymbol{\alpha}_1$ satisfy the excessive gap condition (2.9). Since $-D$ is L-*l.c.g* (from (3.15)), so

$$\begin{aligned}
&D(\mathbf{u}_1) \geq D(\mathbf{u}_0) + \langle \nabla D(\mathbf{u}_0), \mathbf{u}_1 - \mathbf{u}_0 \rangle - \frac{L}{2} \|\mathbf{u}_1 - \mathbf{u}_0\|^2 \\
&\text{(using defn. of } \mu_1 \text{ and (A.1))} \\
&\geq D(\mathbf{u}_0) + \langle \nabla D(\mathbf{u}_0), \mathbf{u}_1 - \mathbf{u}_0 \rangle - \mu_1 \Delta(\mathbf{u}_1, \mathbf{u}_0) \\
&\text{(using defn. of } \mathbf{u}_1) \\
&= D(\mathbf{u}_0) - \mu_1 \min_{\mathbf{u} \in Q_2} \left\{ -\frac{1}{\mu_1} \langle \nabla D(\mathbf{u}_0), \mathbf{u} - \mathbf{u}_0 \rangle + \Delta(\mathbf{u}, \mathbf{u}_0) \right\} \\
&\text{(using (A.2) and } d(\mathbf{u}_0) = 0) \\
&\geq D(\mathbf{u}_0) - \mu_1 \min_{\mathbf{u} \in Q_2} \left\{ -\frac{1}{\mu_1} \langle \nabla D(\mathbf{u}_0), \mathbf{u} - \mathbf{u}_0 \rangle + d(\mathbf{u}) \right\}
\end{aligned}$$

$$= \max_{\mathbf{u}\in Q_2} \{D(\mathbf{u}_0) + \langle \nabla D(\mathbf{u}_0), \mathbf{u} - \mathbf{u}_0\rangle - \mu_1 d(\mathbf{u})\}$$

(using Lemma A.1)

$$\geq \max_{\mathbf{u}\in Q_2} \left\{ \langle \mathbf{A}\mathbf{c}(\mathbf{u}_0) + \mathbf{b}, \mathbf{u}\rangle + \|\mathbf{c}(\mathbf{u}_0)\|^2 - \mu_1 d(\mathbf{u}) \right\}$$

$$= J_{\mu_1}(\mathbf{c}_1).$$

This shows that our initialization indeed satisfies (2.9). Second, we prove by induction that the updates in Algorithm 1 maintain (2.9). We begin with two useful observations. Using (3.19) and the definition of τ_k, one can bound

$$\mu_{k+1} = (1-\tau_k)\mu_k = \frac{6}{(k+3)(k+2)}\frac{L}{\sigma} \geq \tau_k^2 \frac{L}{\sigma}. \tag{A.3}$$

Let $\boldsymbol{\gamma} := \mathbf{u}_{\mu_k}(\mathbf{c}_k)$. The optimality conditions for (3.17) imply $\langle \mu_k \nabla d(\boldsymbol{\gamma}) - \mathbf{A}\mathbf{c}_k - \mathbf{b}, \mathbf{u} - \boldsymbol{\gamma}\rangle \geq 0$ and hence

$$\mu_k \langle \nabla d(\boldsymbol{\gamma}), \mathbf{u} - \boldsymbol{\gamma}\rangle \geq \langle \mathbf{A}\mathbf{c}_k + \mathbf{b}, \mathbf{u} - \boldsymbol{\gamma}\rangle . \tag{A.4}$$

By using the update equation for $\mathbf{c}_{k+1}$ and the convexity of $\|\cdot\|^2$

$$\begin{aligned}
&J_{\mu_{k+1}}(\mathbf{c}_{k+1}) \\
&= \|\mathbf{c}_{k+1}\|^2 + \max_{\mathbf{u}\in Q_2}\{\langle \mathbf{A}\mathbf{c}_{k+1}, \mathbf{u}\rangle + \langle \mathbf{u}, \mathbf{b}\rangle - \mu_{k+1} d(\mathbf{u})\} \\
&= \|(1-\tau_k)\mathbf{c}_k + \tau_k \mathbf{c}(\boldsymbol{\beta}_k)\|^2 \\
&\quad + \max_{\mathbf{u}\in Q_2}\{(1-\tau_k)\langle \mathbf{A}\mathbf{c}_k, \mathbf{u}\rangle + \tau_k \langle \mathbf{A}\mathbf{c}(\boldsymbol{\beta}_k), \mathbf{u}\rangle \\
&\qquad\qquad + \langle \mathbf{u}, \mathbf{b}\rangle - (1-\tau_k)\mu_k d(\mathbf{u})\} \\
&\leq \max_{\mathbf{u}\in Q_2} \{(1-\tau_k)T_1 + \tau_k T_2\},
\end{aligned}$$

where

$$T_1 = -\mu_k d(\mathbf{u}) + \langle \mathbf{A}\mathbf{c}_k + \mathbf{b}, \mathbf{u}\rangle + \|\mathbf{c}_k\|^2 \qquad \text{and}$$
$$T_2 = \langle \mathbf{A}\mathbf{c}(\boldsymbol{\beta}_k) + \mathbf{b}, \mathbf{u}\rangle + \|\mathbf{c}(\boldsymbol{\beta}_k)\|^2 .$$

T_1 can be bounded as follows.

$$\begin{aligned}
T_1 &= -\mu_k[\Delta(\mathbf{u}, \boldsymbol{\gamma}) + d(\boldsymbol{\gamma}) + \langle \nabla d(\boldsymbol{\gamma}), \mathbf{u} - \boldsymbol{\gamma}\rangle)] \\
&\quad + \langle \mathbf{A}\mathbf{c}_k + \mathbf{b}, \mathbf{u}\rangle + \|\mathbf{c}_k\|^2 \\
&\text{(using (A.4))} \\
&\leq -\mu_k \Delta(\mathbf{u}, \boldsymbol{\gamma}) - \mu_k d(\boldsymbol{\gamma}) - \langle \mathbf{A}\mathbf{c}_k + \mathbf{b}, \mathbf{u} - \boldsymbol{\gamma}\rangle \\
&\quad + \langle \mathbf{A}\mathbf{c}_k + \mathbf{b}, \mathbf{u}\rangle + \|\mathbf{c}_k\|^2 \\
&= -\mu_k \Delta(\mathbf{u}, \boldsymbol{\gamma}) - \mu_k d(\boldsymbol{\gamma}) + \langle \mathbf{A}\mathbf{c}_k + \mathbf{b}, \boldsymbol{\gamma}\rangle + \|\mathbf{c}_k\|^2 \\
&\text{(using defn. of } \boldsymbol{\gamma}) \\
&= -\mu_k \Delta(\mathbf{u}, \boldsymbol{\gamma}) + J_{\mu_k}(\mathbf{w}_k) \\
&\text{(using induction assumption)} \\
&\leq -\mu_k \Delta(\mathbf{u}, \boldsymbol{\gamma}) + D(\mathbf{u}_k) \\
&\text{(using concavity of } D) \\
&\leq -\mu_k \Delta(\mathbf{u}, \boldsymbol{\gamma}) + D(\boldsymbol{\beta}_k) + \langle \nabla D(\boldsymbol{\beta}_k), \mathbf{u}_k - \boldsymbol{\beta}_k\rangle ,
\end{aligned}$$

while T_2 can be simplified by using Lemma A.1:

$$T_2 = D(\boldsymbol{\beta}_k) + \langle \nabla D(\boldsymbol{\beta}_k), \mathbf{u} - \boldsymbol{\beta}_k\rangle .$$

Putting the upper bounds on T_1 and T_2 together, and using (A.3) we obtain the following result.

$$\begin{aligned}
&J_{\mu_{k+1}}(\mathbf{c}_{k+1}) \\
&\leq \max_{\mathbf{u}\in Q_2}\{(1-\tau_k)[-\mu_k \Delta(\mathbf{u}, \boldsymbol{\gamma}) + D(\boldsymbol{\beta}_k) \\
&\qquad + \langle \nabla D(\boldsymbol{\beta}_k), \mathbf{u}_k - \boldsymbol{\beta}_k\rangle] \\
&\qquad + \tau_k [D(\boldsymbol{\beta}_k) + \langle \nabla D(\boldsymbol{\beta}_k), \mathbf{u} - \boldsymbol{\beta}_k\rangle]\} \\
&\leq \max_{\mathbf{u}\in Q_2}\{-\mu_{k+1}\Delta(\mathbf{u}, \boldsymbol{\gamma}) + D(\boldsymbol{\beta}_k) \\
&\qquad\qquad + \langle \nabla D(\boldsymbol{\beta}_k), (1-\tau_k)\mathbf{u}_k + \tau_k \mathbf{u} - \boldsymbol{\beta}_k\rangle\} \\
&\text{(using defn of } \boldsymbol{\beta}_k) \\
&= \max_{\mathbf{u}\in Q_2}\{-\mu_{k+1}\Delta(\mathbf{u}, \boldsymbol{\gamma}) + D(\boldsymbol{\beta}_k) + \tau_k \langle \nabla D(\boldsymbol{\beta}_k), \mathbf{u} - \boldsymbol{\gamma}\rangle\} \\
&= -\min_{\mathbf{u}\in Q_2}\{\mu_{k+1}\Delta(\mathbf{u}, \boldsymbol{\gamma}) - D(\boldsymbol{\beta}_k) - \tau_k \langle \nabla D(\boldsymbol{\beta}_k), \mathbf{u} - \boldsymbol{\gamma}\rangle\} \\
&\text{(using defn of } \boldsymbol{\xi}_k) \\
&= -\mu_{k+1}\Delta(\boldsymbol{\xi}_k, \boldsymbol{\gamma}) + D(\boldsymbol{\beta}_k) + \tau_k \langle \nabla D(\boldsymbol{\beta}_k), \boldsymbol{\xi}_k - \boldsymbol{\gamma}\rangle \\
&\text{(using (A.1))} \\
&\leq -\frac{\sigma}{2}\mu_{k+1}\|\boldsymbol{\xi}_k - \boldsymbol{\gamma}\|^2 + D(\boldsymbol{\beta}_k) + \tau_k \langle \nabla D(\boldsymbol{\beta}_k), \boldsymbol{\xi}_k - \boldsymbol{\gamma}\rangle \\
&\text{(using (A.3))} \\
&\leq -\frac{1}{2}\tau_k^2 L\|\boldsymbol{\xi}_k - \boldsymbol{\gamma}\|^2 + D(\boldsymbol{\beta}_k) + \tau_k \langle \nabla D(\boldsymbol{\beta}_k), \boldsymbol{\xi}_k - \boldsymbol{\gamma}\rangle \\
&\text{(using defn of } \mathbf{u}_{k+1}) \\
&= -\frac{L}{2}\|\mathbf{u}_{k+1} - \boldsymbol{\beta}_k\|^2 + D(\boldsymbol{\beta}_k) + \langle \nabla D(\boldsymbol{\beta}_k), \mathbf{u}_{k+1} - \boldsymbol{\beta}_k\rangle \\
&\text{(using } L\text{-}l.c.g \text{ of } -D) \\
&\leq D(\mathbf{u}_{k+1}).
\end{aligned}$$

B A linear time algorithm for a box constrained diagonal QP with a single linear equality constraint

In this section, we focus on the following simple QP:

$$\begin{aligned}
&\min \frac{1}{2}\sum_{i=1}^n d_i^2(\alpha_i - m_i)^2 \\
&s.t. \quad l_i \leq \alpha_i \leq u_i \quad \forall i \in [n]; \\
&\sum_{i=1}^n \sigma_i \alpha_i = z.
\end{aligned} \tag{B.5}$$

Without loss of generality, we assume $l_i < u_i$ and $d_i \neq 0$ for all i. Also assume $\sigma_i \neq 0$ because otherwise α_i can

be solved independently. To make the feasible region nonempty, we also assume

$$\begin{aligned}&\sum_i \sigma_i(\delta(\sigma_i>0)l_i+\delta(\sigma_i<0)u_i)\\ &\le z\\ &\le \sum_i \sigma_i(\delta(\sigma_i>0)u_i+\delta(\sigma_i<0)l_i).\end{aligned}$$

The algorithm we describe below stems from [21] and finds the exact optimal solution in $O(n)$ time.

With a simple change of variable $\beta_i=\sigma_i(\alpha_i-m_i)$, the problem is simplified as

$$\begin{aligned}\min\quad & \frac{1}{2}\sum_{i=1}^n \bar{d}_i^2\beta_i^2\\ s.t.\quad & l_i'\le\beta_i\le u_i' \quad \forall i\in[n];\\ & \sum_{i=1}^n\beta_i=z',\end{aligned}\qquad \text{where}$$

$$l_i'=\begin{cases}\sigma_i(l_i-m_i) & \text{if } \sigma_i>0\\ \sigma_i(u_i-m_i) & \text{if } \sigma_i<0\end{cases},$$

$$u_i'=\begin{cases}\sigma_i(u_i-m_i) & \text{if } \sigma_i>0\\ \sigma_i(l_i-m_i) & \text{if } \sigma_i<0\end{cases},$$

$$\bar{d}_i^2=\frac{d_i^2}{\sigma_i^2},\quad z'=z-\sum_i\sigma_i m_i.$$

We derive its dual via the standard Lagrangian.

$$\begin{aligned}L=&\frac{1}{2}\sum_i\bar{d}_i^2\beta_i^2-\sum_i\rho_i^+(\beta_i-l_i')+\sum_i\rho_i^-(\beta_i-u_i')\\ &-\lambda\left(\sum_i\beta_i-z'\right).\end{aligned}$$

Taking derivative:

$$\begin{aligned}\frac{\partial L}{\partial\beta_i}&=\bar{d}_i^2\beta_i-\rho_i^++\rho_i^--\lambda=0\\ &\Rightarrow\beta_i=\bar{d}_i^{-2}(\rho_i^+-\rho_i^-+\lambda).\end{aligned}\tag{B.6}$$

Substituting into L, we get the dual optimization problem

$$\begin{aligned}\min\ & D(\lambda,\rho_i^+,\rho_i^-)\\ &=\frac{1}{2}\sum_i\bar{d}_i^{-2}(\rho_i^+-\rho_i^-+\lambda)^2\\ &\quad-\sum_i\rho_i^+l_i'+\sum_i\rho_i^+u_i'-\lambda z'\\ s.t.\quad & \rho_i^+\ge0,\quad \rho_i^-\ge0\quad\forall i\in[n].\end{aligned}$$

Taking derivative of D with respect to λ, we get:

$$\sum_i\bar{d}_i^{-2}(\rho_i^+-\rho_i^-+\lambda)-z'=0.\tag{B.7}$$

The KKT condition gives:

$$\rho_i^+(\beta_i-l_i')=0,\tag{B.8a}$$

$$\rho_i^-(\beta_i-u_i')=0.\tag{B.8b}$$

Now we enumerate four cases.

1. $\rho_i^+>0$, $\rho_i^->0$. This implies that $l_i'=\beta_i=u_i'$, which contradicts our assumption.

2. $\rho_i^+=0$, $\rho_i^-=0$. Then by (B.6), $\beta_i=\bar{d}_i^{-2}\lambda\in[l_i',u_i']$, hence $\lambda\in[\bar{d}_i^2l_i',\bar{d}_i^2u_i']$.

3. $\rho_i^+>0$, $\rho_i^-=0$. Now by (B.8) and (B.6), we have $l_i'=\beta_i=\bar{d}_i^{-2}(\rho_i^++\lambda)>\bar{d}_i^{-2}\lambda$, hence $\lambda<\bar{d}_i^2l_i'$ and $\rho_i^+=\bar{d}_i^2l_i'-\lambda$.

4. $\rho_i^+=0$, $\rho_i^->0$. Now by (B.8) and (B.6), we have $u_i'=\beta_i=\bar{d}_i^{-2}(-\rho_i^-+\lambda)<\bar{d}_i^{-2}\lambda$, hence $\lambda>\bar{d}_i^2u_i'$ and $\rho_i^-=-\bar{d}_i^2u_i'+\lambda$.

In summary, we have $\rho_i^+=[\bar{d}_i^2l_i'-\lambda]_+$ and $\rho_i^-=[\lambda-\bar{d}_i^2u_i']_+$. Now (B.7) turns into

$$f(\lambda):=\sum_i\underbrace{\bar{d}_i^{-2}([\bar{d}_i^2l_i'-\lambda]_+-[\lambda-\bar{d}_i^2u_i']_++\lambda)}_{=:h_i(\lambda)}-z'=0.\tag{B.9}$$

In other words, we only need to find the root of $f(\lambda)$ in (B.9). $h_i(\lambda)$ is given by

$$h_i(\lambda)=\begin{cases}l_i' & \text{if } \lambda<\bar{d}_i^2l_i'\\ \lambda\bar{d}_i^{-2} & \text{if } \bar{d}_i^2l_i'\le\lambda\le\bar{d}_i^2u_i'\\ u_i' & \text{if } \lambda>\bar{d}_i^2u_i'\end{cases}\tag{B.10}$$

Note that $h_i(\lambda)$ is a monotonically increasing function of λ, so the whole $f(\lambda)$ is monotonically increasing in λ. Since $f(\infty)\ge0$ by $z'\le\sum_i u_i'$ and $f(-\infty)\le0$ by $z'\ge\sum_i l_i'$, the root must exist. Considering that f has at most $2n$ kinks (nonsmooth points) and is linear between two adjacent kinks, the simplest idea is to sort $\{\bar{d}_i^2l_i',\bar{d}_i^2u_i':i\in[n]\}$ into $s^{(1)}\le\ldots\le s^{(2n)}$. If $f(s^{(i)})$ and $f(s^{(i+1)})$ have different signs, then the root must lie between them and can be easily found because f is linear in $[s^{(i)},s^{(i+1)}]$. This algorithm takes at least $O(n\log n)$ time because of sorting.

However, this complexity can be reduced to $O(n)$ by making use of the fact that the median of n (unsorted) elements can be found in $O(n)$ time. Notice that due to the monotonicity of f, the median of a set S gives exactly the median of function values, *i.e.*, $f(\text{MED}(S))=\text{MED}(\{f(x):x\in S\})$. Algorithm 2 sketches the idea of binary search. The while loop

terminates in $\log_2(2n)$ iterations because the set S is halved in each iteration. And in each iteration, the time complexity is linear to $|S|$, the size of current S. So the total complexity is $O(n)$. Note the evaluation of $f(m)$ potentially involves summing up n terms as in (B.9). However by some clever aggregation of slope and offset, this can be reduced to $O(|S|)$.

Algorithm 2 $O(n)$ algorithm to find the root of $f(\lambda)$. Ignoring boundary condition checks.

Require: Function f .
Initialize: Set kink set $S \leftarrow \left\{\bar{d}_i^2 l_i' : i \in [n]\right\} \cup \left\{\bar{d}_i^2 u_i' : i \in [n]\right\}$.
while $|S| > 2$ **do**
  Find median of S: $m \leftarrow \text{MED}(S)$.
  if $f(m) \geq 0$ **then**
    $S \leftarrow \{x \in S : x \leq m\}$.
  else
    $S \leftarrow \{x \in S : x \geq m\}$.
  end if
end while
return root $\frac{lf(u)-uf(l)}{f(u)-f(l)}$ where $S = \{l, u\}$.

C Calculation of $\mathbf{u}_\mu(\mathbf{c})$ when Q_2 is endowed with ℓ_1 norm

Representing $\mathbf{Ac} + \mathbf{b}$ by $\mathbf{r}$ and using concepts of convex optimization, the Lagrangian corresponding to the constrained optimization problem (3.32) can be set up as

$$Lag(\mathbf{u}, \lambda) = \frac{\mu\sigma}{2} \sum u_i \ln u_i - \sum_i u_i r_i - \lambda(\sum_i u_i - 1)$$

Taking gradient of the Lagrangian with respect to u_i and setting it to zero, we get

$$\frac{\mu\sigma}{2}(\ln u_i + 1) - r_i = \lambda \qquad \forall i = 1, \ldots, n$$

which yields

$$u_i = \exp\left(\frac{2(\lambda + r_i)}{\mu\sigma} - 1\right) \tag{C.11}$$

Plugging this back into the simplex constraint $\sum_i u_i = 1$, gives

$$\lambda = \frac{\mu\sigma}{2}\left(1 - \ln \sum_i \exp\left(\frac{2r_i}{\mu\sigma}\right)\right)$$

Replacing the value of λ in (C.11) yields the value of u_i in (3.33).

D Learning SVMs with MEB

Kernel methods in general and support vector machines (SVMs) in particular have received significant recent research interest in machine learning [22]. Underlying a SVM is a simple geometric idea. Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of n points $\mathbf{x}_i$ labeled by $y_i \in \{\pm 1\}$ the aim is to find the hyperplane which maximizes the margin of separation between points from different classes. This is compactly written as the following optimization problem (see Schölkopf and Smola [22] for details):

$$\min_{\mathbf{w} \in \mathbb{R}^d} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i \tag{D.12a}$$

$$\text{s.t.} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \text{ for all } i. \tag{D.12b}$$

Standard duality arguments (see *e.g.* Boyd and Vandenberghe [5]) yield the following dual Quadratic Programming (QP) problem

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m} \quad \langle \boldsymbol{\alpha}, \mathbf{e} \rangle - \frac{1}{2}\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} \tag{D.13a}$$

$$\text{s.t.} \quad \sum_{i=1}^n \alpha_i y_i = 0 \tag{D.13b}$$

$$0 \leq \alpha_i \leq C \text{ for all } i. \tag{D.13c}$$

Here K is a $m \times m$ matrix whose entries are given by $y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ and $\mathbf{e}$ denotes the vector of all ones. Since the dual only depends on $\mathbf{x}$ via the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ one can employ the kernel trick: map ξ_i into a feature space via $\phi(\mathbf{x}_i)$ and compute the dot product in the feature space by using a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) := \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ [22]. The kernel trick makes SVM rather powerful because simple linear decision boundaries in feature space map into non-linear decision boundaries in the original space where the datapoints live.

A number of different techniques have been proposed for solving the quadratic problem associated with SVMs. Of particular interest in our context is the Core Vector Machine (CVM) [23, 24]. The key idea of the CVM is the observation that solving (D.13) is equivalent to finding the MEB of the feature vectors. The MEB problem in feature space can be written as (see [24] for details)

$$\min_{\mathbf{c} \in \mathbb{R}^d, R \in \mathbb{R}} \quad R^2 \tag{D.14a}$$

$$\|\mathbf{c} - \phi(\mathbf{x}_i)\|^2 \leq R^2 \text{ for all } i. \tag{D.14b}$$

The dual of the above minimization problem then

becomes (also see (3.13))

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i K_{i,i} - \boldsymbol{\alpha}^\top K \boldsymbol{\alpha} \tag{D.15a}$$

$$\text{s.t.} \quad \alpha_i \geq 0, \qquad \sum_i \alpha_i = 1 \tag{D.15b}$$

where $K_{i,j} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ as before. In particular, if each $K_{i,i}$ equals a constant c then it can be shown that by a simple transformation that the standard SVM dual (D.13) and the CVM dual (D.15) can be identified. Therefore, every iteration of the CVM algorithm [23] identifies an active set of points, and computes the MEB of this active set. This is done via the coreset algorithm of [20], hence the name core vector machine. In fact, our algorithm for the MEB can directly be plugged into the CVM, and changes the rates of convergence of the inner iteration from $O(nd/\epsilon)$ to $O^*(nd/\sqrt{\epsilon})$ which is significantly better for small values of ϵ. We hope that our algorithm with newer convergence rates can also be used in similar machine learning algorithms to speed up their convergence.

E Experimental Results

The aim of our experiments is to demonstrate the efficacy of Algorithm 1 and compare its performance with existing MEB algorithms in terms of running time and number of iterations required for convergence. We perform preliminary experiments for the case when Q_2 is endowed with the ℓ_2 norm and see promising results. Following [26] we generate data using a random multivariate Gaussian distribution and vary n the number of data points and d the dimensions. For each fixed n and d we generate 5 random datasets and report the average performance of our algorithm with the multiplicative guarantee $\epsilon = 10^{-3}$ in Table 1 and 2. Recall from §3.3 that the multiplicative guarantee ϵ is equivalent to the additive tolerance $\epsilon\mathcal{P}^2$, where $\mathcal{P}$ is chosen via

$$\mathcal{P} = \frac{1}{2} \max_{\mathbf{x}_i, \mathbf{x}_j} \|\mathbf{x}_i - \mathbf{x}_j\| .$$

For reference we also reproduce the results reported in Table 1 of [26]. Although not a fair comparison, the CPU times gives an indication of the relative performance of various algorithms. In the table BC refers to the coreset algorithm of Badoiu and Clarkson [1] while A1 and A2 are MEB algorithms implemented in Yildirim [26]. Note that there is no efficient algorithmic implementation of Panigrahy's algorithm [20] while A1 and A2 are very closely related to the approach in [6] making use of the Frank Wolfe algorithm and returning a coreset with "away" steps.

n	d	Time			
		A1	A2	BC	Ours
500	10	0.06	0.03	0.12	0.04
1000	10	0.15	0.03	0.14	0.10
5000	20	1.7	0.36	3.11	1.08
10000	20	4.46	0.58	4.65	3.40
30000	30	27	6.45	24.59	10.43
50000	50	71.62	16.87	68.78	18.69
100000	100	287.99	77.74	268.11	83.18

Table 1: Convergence time for $n >> d$ ($\epsilon = 10^{-3}$)

n	d	Iterations			
		A1	A2	BC	Ours
500	10	168.7	44.5	435.5	44.2
1000	10	330.7	41.6	344.4	54.5
5000	20	246.8	46	464.2	69.7
10000	20	319.2	36.3	334.4	105.2
30000	30	446.4	103.6	409	77.8
50000	50	429.8	98.4	415.1	54.5
100000	100	451.7	119	422.6	63

Table 2: Iterations for convergence for $n >> d$ ($\epsilon = 10^{-3}$)

Our algorithm performs significantly better than BC and A1, while being comparable to A2. Furthermore, our algorithm usually takes smaller number of iterations and particularly shines when the number of points is large. Our implementation is preliminary, and we believe that our algorithm will benefit from practical speed ups in much the same way that algorithm A2 was obtained by improving A1 to get rid of redundancies.