Read this first (quickly):
https://www.researchgate.net/publication/331315423_Lecture_Notes_on_Machine_Learning_Frank-Wolfe_for_Minimum_Enclosing_Balls

Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm

A geometric shape, like a triangle or pyramid, but in higher dimensions. It's like a container that holds all the points we're interested in. In this case, it's a container that holds points that satisfy certain conditions.

KENNETH L. CLARKSON

IBM Almaden Research Center

simplex = generalization of the notion of a triangle or tetrahedron to arbitrary dimensions.

greedy algorithm = makes the locally optimal choice at each stage.

k-dimensional subspace of the original n-dimensional space

Abstract. The problem of maximizing a concave function $f(x)$ in the unit simplex Δ can be solved approximately by a simple greedy algorithm. For given k, the algorithm can find a point $x_{(k)}$ on a k-dimensional face of Δ, such that $f(x_{(k)}) \geq f(x_*) - O(1/k)$. Here $f(x_*)$ is the maximum value of f in Δ, and the constant factor depends on f. This algorithm and analysis were known before, and related to problems of statistics and machine learning, such as boosting, regression, and density mixture estimation. In other work, coming from computational geometry, the existence of ϵ-*coresets* was shown for the minimum enclosing ball problem by means of a simple greedy algorithm. Similar greedy algorithms, which are special cases of the Frank-Wolfe algorithm, were described for other enclosure problems. Here these results are tied together, stronger convergence results are reviewed, and several coreset bounds are generalized or strengthened.

Categories and Subject Descriptors: G.1.6 [**Numerical Analysis**]: Optimization—*Convex programming; gradient methods; quadratic programming methods*; G.1.2 [**Numerical Analysis**]: Approximation; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Coresets, sparsity, boosting, regression, approximation, minimum enclosing ball

ACM Reference Format:

Clarkson, K. L. 2010. Coresets, sparse greedy approximation, and the Frank-Wolfe Algorithm. ACM Trans. Algor. 6, 4, Article 63 (August 2010), 30 pages.
DOI = 10.1145/1824777.1824783 http://doi.acm.org/10.1145/1824777.1824783

1. *Introduction*

For a given concave function $f : \mathbb{R}^n \to \mathbb{R}$, consider the optimization problem

for convex, we do min

$$\max_{x \in \Delta} f(x) \tag{1}$$

Some of this work was done at Bell Labs, Alcatel-Lucent.

Authors' address: K. L. Clarkson, Room B1-312, IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95301, e-mail: ken.clarkson@gmail.com.

DOI 10.1145/1824777.1824783 http://doi.acm.org/10.1145/1824777.1824783

ALGORITHM 1.1. *Given concave function* $f : \mathbb{R}^n \to \mathbb{R}$, *maximize* f *in* Δ *as follows.*

Pick as $x_{(0)}$ *the vertex of* Δ *with largest* f *value;*
For $k = 0 \ldots \infty$, *find* $x_{(k+1)}$ *from* $x_{(k)}$ *as follows:*
  $i' := \arg\max_i \nabla f(x_{(k)})_i$; gradient
  $\alpha' := \arg\max_{\alpha \in [0,1]} f(x_{(k)} + \alpha(e(i') - x_{(k)}))$; step size; line search
  $x_{(k+1)} := x_{(k)} + \alpha'(e(i') - x_{(k)})$; FW step
  that is, $x_{(k+1)}$ *is the point on the line segment from* $x_{(k)}$ *to* $e(i')$ *that maximizes* f;

FIG. 1. A version of the Frank-Wolfe algorithm

Convex hull (convex envelope) of a sample of points is the minimum convex set enclosing them all.

where $\Delta \subset \mathbb{R}^n$ is the unit simplex that is the convex hull of the unit basis vectors of $\mathbb{R}^n$. The vertices of Δ are the points $e(i)$, $i = 1 \ldots n$, where $e(i)$ has coordinate $e(i)_i = 1$, and all other coordinates zero.

Special cases of this problem include the problems of training support vector machines and other classifiers, approximating functions as convex combinations of other functions, finding D-optimal designs, estimating mixtures of probability densities, and finding the smallest balls, ellipsoids, or axis-aligned ellipsoids containing a given set of points.

For min problem, f(xk) is decreasing.

Algorithm 1.1, shown in Figure 1, generates a sequence $x_{(k)} \in \Delta$, $k = 0, \ldots \infty$, with increasing $f(x_{(k)})$. In practice the loop would exit when $x_{(k)}$ is an adequate solution by some application-specific criterion.

The procedure follows a limited form of gradient ascent: rather than optimize in the direction of the gradient, only the largest component of the gradient is used, so that $x_{(k)}$ has at most $k+1$ nonzero entries. Moreover, the search direction $e(i') - x_{(k)}$ is used, from $x_{(k)}$ toward $e(i')$, not the direction $e(i')$; this keeps the search within Δ.

The computational task of finding the maximizing α' can be done by solving a quadratic equation, if f is a quadratic (multivariate) function. In Section 7, provably good performance is shown for a variant in which the α' value used at step k is a predefined α_k. Otherwise, the determination of α' is left to be determined for a given problem class.

Instances of this procedure, and variations of it, have been proposed independently many times, but the oldest version seems to be due to Frank and Wolfe [1956]; they also proved a fundamental approximation result for general concave functions, described shortly. Similar algorithms and results have appeared in the machine learning literature, as *sparse greedy approximation*, as discussed in Section 3.3. In the computational geometry literature, similar algorithms have been proposed for the purpose of finding (or simply proving the existence of) coresets, which are discussed in Section 3.2, Section 8, and Section 9.

One of the contributions of this article is just to put together these lines of work.[1] However, there are some specific new results:

—a definition of coresets that applies in the general setting of Algorithm 1.1, with corresponding general existence results (Section 8), proven with algorithms that are variations of Algorithm 1.1. One construction is worst-case tight (Section 9), exactly;

[1] The Frank-Wolfe algorithm was applied as such in some of the cited applications (e.g., Ahipasaoglu et al. [2006] and Kumar and Yildirim [2005]).

—coreset results for support vector machines (SVM) that are significantly tighter, in constant factor, than known before (Section 11.2);

—sample complexity bounds for learning, based on the coreset results (Section 10); in the case of SVM, these are similar to those proven via perceptrons and Novikoff's mistake bound (Section 11.2);

—improvements in generality and provable speed of a practical SVM training algorithm based on coresets [Tsang et al. 2006], and a sharper analysis of a simple approximation algorithm for minimum enclosing balls [Bădoiu and Clarkson 2008];

—a quantitative approximation bound for Anyboost.L1 [Mason et al. 2000], since it is an instance of Algorithm 1.1;

—an algorithm for sparse greedy approximation that is a bit simpler than the one known in the machine learning literature [Zhang 2003];

—Coreset results for boosting (Section 11.3), previously unknown.

A look at Figure 3 in Section 5 gives some idea of the range of applications of Algorithm 1.1. (However, results here do not generally apply to the ellipsoid problems.)

The next section discusses fundamental properties of the optimization problem, and of the Frank-Wolfe algorithm as applied to it. Section 3 discusses motivation and related work, Section 4 discusses some particular cases of f, and Section 5 gives a summary of old and new results, and outlines the remainder of the article.

2. *Background*

Polytope = a geometric object with flat sides. The reason we need P is because we want to make sure that the solution we find is feasible.

2.1. The Dual, and Optimality Conditions.

The original Frank-Wolfe algorithm applies in the general context of maximizing a concave function f within a feasible polytope $\mathcal{P}$. At each iteration, the algorithm solves the linear programming problem of finding the optimum y' maximizing within P the local linear approximation $\tilde{f}_x(y) := f(x) + (y-x)^T \nabla f(x)$ to $f(y)$ near $x = x_{(k)}$. The approximation $\tilde{f}_x(y)$ is a linear function of y, and its graph is the hyperplane tangent to the graph of f at x. Having found y', the algorithm then solves the one-dimensional optimization problem of finding the largest value of f on the line segment $[x, y']$.

FW: Mininize the linear approximation of f in xk. The formula for f~ comes from first order convexity.

In this article, the feasible polytope $\mathcal{P}$ is the simplex Δ, and the solution of the maximization problem is the $y' \in \Delta$ that maximizes $y^T \nabla f(x)$; that optimum y' is simply $e(i')$, as used in Algorithm 1.1. That is, $\max_{y \in \Delta} y^T \nabla f(x) = \max_i \nabla f(x)_i$. The maximum value of the linear approximation $\tilde{f}_x(y)$ within Δ can be thus expressed as the function

$$w(x) := \max_{y \in \Delta} \tilde{f}_x(y) = f(x) + (e(i') - x)^T \nabla f(x)$$
$$= \max_i \nabla f(x)_i + f(x) - x^T \nabla f(x). \qquad (2)$$

Wolfe dual

The function $w(x)$ will be called here the *Wolfe dual* function. By the concavity of f, every hyperplane tangent to the graph of f is otherwise above it, and so for all $x, x' \in \Delta$, $w(x) \geq f(x')$. In particular, the minimum value of w is no smaller than $f(x_*)$, the maximum value of f in Δ.

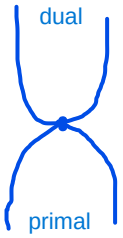
A key property needed here is that the minimum value of w occurs *at* x_*, and is equal to $f(x_*)$. That is, the *duality* conditions

$$\min_{x \in \mathbb{R}^n} w(x) = w(x_*) = f(x_*) \geq f(x) \qquad (3)$$

true maximum

x* is maximum

primal maximum = dual minimum (and vice versa)



hold. Thus, solving the *primal* problem of finding the maximum $f(x)$ in Δ can be done by solving the *Wolfe dual* problem of minimizing $w(x)$, and vice versa. The Wolfe dual can be stated with $\max_i \nabla f(x)_i$ expressed using explicit inequality constraints.

The complementary slackness conditions guarantee that the values of the primal and dual are the same.

from equations (2) and (3):

$$\min_{z\in\mathbb{R}, x\in\mathbb{R}^n} \underbrace{z}_{(3)} + \underbrace{f(x) - x^T \nabla f(x)}_{(2)} \quad \text{subject to } z \geq \max_i \nabla f(x)_i \tag{4}$$

The condition of *complementary slackness* also holds at the optimum x_*, and corresponding optimum $z_* = \max_i \nabla f(x_*)_i$. This condition is that for $i = 1 \ldots n$, either the i'th coordinate of x_* is zero, or $z_* - \nabla f(x_*)_i$ is zero.

A derivation of the Wolfe dual, using the Lagrangian dual, is given in Appendix 12.

2.1.1. *Minimum Enclosing Balls.* Some problems are better understood in their Wolfe dual formulation. For example, the Minimum Enclosing Ball (MEB, 1-center, smallest enclosing sphere) problem is as follows: given a set of points $P = \{p_1 \ldots p_n\}$ in d dimensions, find the smallest ball containing all points of P. This could be described as: find the vector c and value z such that $z + c^2 \geq (p_i - c)^2$ for all i, and $z + c^2$ is as small as possible. (Here for a vector v, v^2 denotes $\|v\|^2 = v^{Tv}$.) That is, $z + c^2$ bounds the squared distance of c to every one of the input points p_i, and this squared distance is minimized. The relation $z + c^2 \geq (p_i - c)^2$ is the same as $z \geq p_i^2 - 2p_i^{Tc}$, which is linear in z and c. Moreover, suppose c is a convex combination of the input points, that is, $c = Ax$ for $x \in \Delta$, where the columns of A are the input points. Then the function $f(x) = x^{Tb} - x^{TATAx}$, where $b_i = p_i^2$, has the gradient $b - 2A^{TAx} = b - 2A^{Tc}$, and the i'th coordinate of $\nabla f(x)$ is $p_i^2 - 2p_i^{Tc} = (p_i - c)^2 - c^2$. Also $z + f(x) - x^T \nabla f(x) = z + x^{TAATx} = z + c^2$, and so the MEB problem can be put in the form (4). (The assumption $c = Ax$ is no loss of generality: by starting with the formulation of the problem in terms of z and c, and deriving the dual via Lagrangian relaxation, the primal problem (1) emerges.)

c - center (vector)
z - radius (value)

$\|v\|^2 = v^T v$

$z \geq p_i^2 - 2p_i^T c$

bi is zi in our Report

$f(x) = x^T b - x^T A^T A x$

$\nabla_x f(x) = b - 2A^T Ax = b - 2A^T c$

$p_i^2 - 2p_i^T c = (p_i - c)^2 - c$

$z + f(x) - x^T \nabla f(x) = z + x^T A A^T x = z + c^2$

(*) Important for FW applied to MEB: Find the farthest point from the current center.

If the Frank-Wolfe algorithm is applied to maximizing the function $f(x)$ associated with the MEB problem, the coordinate i' maximizing $\nabla f(x)_i$ corresponds to the point $p_{i'}$ that is farthest from the current $c_{(k)} = Ax_{(k)}$. If the stepsize α' is chosen by the fixed schedule $\alpha' = \alpha_k = 2/(k+3)$ at step k, then the resulting algorithm is very close to the one proposed by Bădoiu and Clarkson [2003] (the "simple" algorithm), who thus reinvented a version of the Frank-Wolfe algorithm for the particular case of MEB. (Such a fixed schedule of stepsizes is discussed for the general algorithm in 7.)

there's a mistake here

$c^2 = c^T c = (Ax)^T Ax = x^T A^T Ax$

$z + x^T b - x^T A^T Ax - x^T b + 2x^T A^T c = z + (Ax)^T Ax = z + (Ax)^2$

The linear complementarity condition ensures that the complementary slackness conditions are satisfied at the optimal solution.

slackness

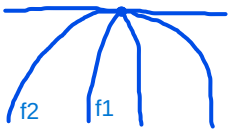
For the MEB problem, the linear complementarity condition implies that the i'th coordinate of x_* is nonzero exactly when $z_* + c_*^2 = (p_i - c_*)^2$, that is, p_i "contributes" to $c_* = Ax_*$ exactly when p_i is among the input points that realize the radius of the MEB.

pi lies on the boundary of the smallest enclosing ball.

2.2. The Measure C_f. For each function $f(x)$ for which the Frank-Wolfe algorithm might be applied to solve (1), there is an associated value $C_f \geq 0$ measuring the "nonlinearity" of $f(x)$; C_f might also be termed a measure of curvature or smoothness. The quantity C_f will appear below in approximation bounds for the Frank-Wolfe algorithm.

The purpose of Cf is to assess the suitability of the Frank-Wolfe algorithm for optimizing the given function. In general, the algorithm is well-suited for optimizing functions that have lower nonlinearity or curvature because it relies on linear approximations at each iteration. When Cf is small, the function is relatively close to being linear, and the linear approximations used in the Frank-Wolfe algorithm are likely to be accurate and converge quickly.

On the other hand, for functions with higher nonlinearity or curvature (larger Cf), the linear approximations may not capture the function's behavior well, and the algorithm might require more iterations to converge. In such cases, other optimization methods that handle more complex functions might be more efficient.



As mentioned, the concavity of f implies that the graph of $f(x)$ is below any of its tangent hyperplanes; the measure C_f quantifies how *much* below that is. Formally, the quantity C_f is defined as the largest quantity so that for all $x, x', y \in \Delta$, with y collinear with x and x', so that $y = x + \alpha(x' - x)$ for some α,

This follows from first order convexity. We can subtract alpha*Cf from that formula and it will still hold.

$$f(y) \geq f(x) + (y - x)^T \nabla f(x) - \alpha^2 C_f. \tag{5}$$

For example, when $f(x)$ is a linear function, C_f is zero. As discussed in Section 4.3, for the function $f(x) = x^T b + x^{TAA^T x}$ associated with the MEB problem, the value C_f is the squared diameter $\text{Diam}(A\Delta)^2$ of the polytope $A\Delta := \{Ax \mid x \in \Delta\}$, that is, within a constant factor of $\text{Rad}(A\Delta)^2$, the square of the MEB radius.

$f(x) = x^T b + x^T A A^T x$

A related quantity that has appeared in the functional analysis and machine learning literatures is the *modulus of convexity*, introduced by Clarkson [1936]. This quantity could be defined similarly to C_f, but is a lower bound on nonlinearity, instead of an upper bound.

2.3. PRIMAL AND DUAL APPROXIMATION. It is convenient to discuss approximation bounds in terms of measures that are scaled by C_f. Define $h(x)$ by

primal gap

$$h(x) := (f(x_*) - f(x))/4C_f, \tag{6}$$

the gap between the value of f at x, and its maximum value $f(x_*)$ within Δ. Similarly, define $g(x)$ by

(3) w(x) >= f(x)
=> g(x) >= h(x) >= 0

$$g(x) := (w(x) - f(x))/4C_f, \tag{7}$$

a scaled version of the gap between the dual value $w(x)$ and $f(x)$. By (3), $g(x) \geq h(x) \geq 0$ for all feasible x.

The following theorem is essentially due to Ahipasaoglu et al. [2006], and generalizes a lemma of Khachiyan [1996].

THEOREM 2.1. *For simplex Δ and continuously differentiable concave function f, one iteration of Algorithm* 1.1 *satisfies*

max

$$h(x_{(k+1)}) \leq h(x_{(k)}) - g(x_{(k)})^2. \quad \text{The new gap <= old gap - duality gap^2}$$

(3)

The slightly weaker bound $h(x_{(k+1)}) \leq h(x_{(k)}) - h(x_{(k)})^2$ is immediately implied, since $g(x) \geq h(x)$. This weaker bound was shown by Frank and Wolfe [1956] for Algorithm 1.1.

PROOF. For simpler notation, let $x := x_{(k)}$ and $y := x_{(k+1)} = x + \alpha(e(i') - x)$. By definition (5) of C_f and (2) of $w(x)$,

$$\begin{aligned} f(y) &= f(x + \alpha(e(i') - x)) \quad \text{FW step} \\ &\overset{(5)}{\geq} f(x) + \alpha(e(i') - x)^T \nabla f(x) - \alpha^2 C_f \\ &= f(x) + \alpha(\max_i \nabla f(x)_i - x^T \nabla f(x)] - \alpha^2 C_f \\ &\overset{(2)}{=} f(x) + \alpha(w(x) - f(x)) - \alpha^2 C_f. \end{aligned}$$

$w(x) = \max_i \nabla f(x)_i + f(x) - x^T \nabla f(x)$

This implies, using the definitions (6) and (7) of $h(x)$ and $g(x)$,

$$h(y) = (f(x_*) - f(y))/4C_f$$
$$\leq h(x) - \alpha(w(x) - f(x))/4C_f + \alpha^2/4 \quad \text{(From the PROOF above and (6):)}$$
$$\stackrel{(7)}{=} h(x) - \alpha g(x) + \alpha^2/4 \tag{8}$$
$$\leq h(x) - g(x)^2. \tag{9}$$

Here the last step assumes $g(x) \leq 1/2$, so that the minimizing α is less than one. Suppose on the contrary that the $\alpha \in [0, 1]$ that minimizes (8) is $\alpha = 1$, which occurs only when $g(x) \geq 1/2$. Then y is some vertex v of Δ, and $h(v) \leq h(x) - 1/2 + 1/4 < h(x)$, which implies $f(v) > f(x) = f(x_{(k)})$. But $f(x_{(k)})$ is nondecreasing in k, and by choice of $x_{(0)}$, $f(x_{(k)}) \geq f(x_{(0)}) \geq f(v)$, contradiction. Hence the minimizing α is less than one. □

THEOREM 2.2. *For simplex Δ and concave function f, Algorithm 1.1 finds a point $x_{(k)}$ on a k-dimensional face of Δ such that*

$$h(x_{(k)}) \stackrel{(6)}{=} (f(x_*) - f(x_{(k)}))/4C_f \leq 1/(k+3)$$

for $k > 0$.

Upper bound for the gap between the value of f at x(k), and its maximum value f(x*) within the simplex delta. => measure of error

PROOF. The vertices of the k-face will be the vertices of Δ (the $e(i')$ vectors) associated with each iterate $x_{(k)}$. It remains to bound the values $h(x_{(k)})$. Since $g(x) \geq h(x)$, we always have, by Theorem 2.1,

$$h\left(x_{(k+1)}\right) \leq h(x_{(k)}) - h\left(x_{(k)}\right)^2. \tag{10}$$

From the proof of the last theorem, $g(x_{(0)}) \leq 1/2$, and so $h(x_{(1)}) \leq 1/2 - (1/2)^2 = 1/4$. More generally, noting that $1 - \gamma \leq 1/(1+\gamma)$ for $\gamma > -1$, and letting $h_k := h(x_{(k)})$,

h(x(1)) <= 1/4
h(x(2)) <= 1 / (1 + 4) = 1/5
h(x(3)) <= 1 / (1 + 5) = 1/6
...
h(x(k)) <= 1 / (k + 3)

$$h\left(x_{(k+1)}\right) \leq h_k(1 - h_k) \leq \frac{h_k}{1+h_k} \stackrel{(/hk}{=} \frac{1}{1+1/h_k}$$

and so by induction, $h(x_{(k)}) \leq 1/(k+3)$ for $k > 0$, and the theorem follows. □

THEOREM 2.3. *For simplex Δ and continuously differentiable concave function f, and given $\epsilon > 0$, Algorithm 1.1 will have an iterate $x_{(\hat{k})}$ with $\hat{k} \leq 2K$, where $K := \lceil 1/\epsilon \rceil$, so that $g(x_{(\hat{k})}) \leq \epsilon$. That is, $w(x_{(\hat{k})}) - f(x_{(\hat{k})}) \leq 4\epsilon C_f$.* (from 7)

We use this as our stopping condition!

Of course, to determine which of the $x_{(k)}$ has $g(x_{(k)}) \leq \epsilon$ seems to need the knowledge of C_f; for an existence proof this knowledge can be assumed. However, since $w(x) - f(x) = z(x) - x^T \nabla f(x)$, where $z(x) := \max_i \nabla f(x)_i$, the point $x_{(\hat{k})}$, that minimizes $z(x_{(k)}) - x_{(k)}^T \nabla f(x_{(k)})$ over $k = K \ldots 2K$, will have both $h(x_{(k)}) \leq \epsilon$ and $g(x_{(k)}) \leq \epsilon$. (from (4))

PROOF. The previous theorem shows that $k \leq K$ iterations suffice to obtain $x_{(k)}$ with $h(x_{(k)}) \leq \epsilon$. During subsequent iterations, either iterate $x_{(j)}$ has $g(x_{(j)}) \leq \epsilon$, or $h(x_{(j+1)}) \leq h(x_{(j)}) - \epsilon^2$ by Theorem 2.1. Hence for some $\hat{k} \leq 2K$, $g(x_{(\hat{k})}) \leq \epsilon$, since otherwise $h(x_{(2K)}) < 0$. □

(*) IMPORTANT! MEB dual explained

For the MEB problem, the dual objective function $w(x)$ is the squared radius of the ball centered at $c = Ax$ that contains all the input points. If the duality gap

$w(x) - f(x)$ is small, then so is $w(x) - w(x_*)$. Hence Theorem 2.3 shows that the Frank-Wolfe algorithm needs $O(nd/\epsilon)$ time to return $x_{(\hat{k})}$ with

$$w\left(x_{(\hat{k})}\right) \leq f\left(x_{(\hat{k})}\right) + 4\epsilon C_f \leq R^2 + \kappa\epsilon R^2,$$

n - number of points in the input set.
d - dimension of the space.
epsilon - small positive constant that determines the desired approximation factor.

squared radius = dual w(x)
w(x) >= f(x), so: f(x) <= R^2
We also use: 4Cf = kR^2

for an absolute constant κ, and where $R = \mathrm{Rad}(A\Delta)$ is the MEB radius. That is, an enclosing ball is found with radius within a $1 + \epsilon$ factor of the minimum. This observation uses the fact that C_f for the MEB problem is proportional to the square of the MEB radius. This time bound sharpens the analysis of Bădoiu and Clarkson [2003] for their reinvented version, and matches the running time of Panigrahy's algorithm [2004], with an arguably simpler algorithm. Some other approximation algorithms for MEB [Bădoiu and Clarkson 2003; Kumar et al. 2003] have running time $O(nd/\epsilon + 1/\epsilon^{O(1)})$, and are somewhat more complicated.

3. *Related Work*

3.1. Sparse Approximation.

The convergence rates for Frank-Wolfe are slow compared to modern methods, but the simplicity of Algorithm 1.1 means that for many problems, the work per iteration is small; in particular, the solution of linear systems, as is often needed for faster-converging methods, is not needed by Algorithm 1.1. For some large-scale applications, such a trade-off is favorable to algorithms like Algorithm 1.1, as discussed by Platt [1999] and Tsang et al. [2005a, 2005b] for the particular case of the training of Support Vector Machines (SVM). Another example is semidefinite programming, where the number of variables can be quite large. Similar considerations also motivate some recent interest in more general gradient descent algorithms [Nemirovski 2005; Hazan 2006; Taskar et al. 2006].

Another motivation is that sometimes rough approximations are acceptable for an application. For example, when computing a statistical estimator involves an optimization problem, the noisiness of the data implies that the parameters given by the optimal solution, and the corresponding true values of those parameters, are only related by error upper bounds. In such a situation, an approximation within a small factor of the error upper bounds is likely to be quite acceptable. Such a point was observed most recently perhaps by Altun and Smola [2006]; they propose the use of Algorithm 3.1 (discussed in Section 3.3) for some general problems of inference; it's worth noting that Algorithm 1.1 is often applicable to these problems also.

(*) Sparsity of the solutions

Perhaps the main point of interest for Algorithm 1.1 is the sparsity of the solutions that it finds. The iterate $x_{(k)}$ has few (at most $k+1$) nonzero entries, and so is sparse in that sense. Thus the convergence results show that there are sparse solutions that are provably good approximations. In the extreme case when f is simply a linear function and $C_f = 0$, the optimum x_* is one of the vertices: a very sparse solution, with one nonzero entry. More generally, the smaller C_f is, the flatter f is, and the more effective a procedure based on local linear approximation by the gradient will be.

The MEB problem is an instance of quadratic programming, where the function $f(x)$ has the form $f(x) = a + x^T b + x^{TMx}$, where $a \in \mathbb{R}$, $b \in \mathbb{R}^n$, and M is a negative semidefinite $n \times n$ matrix. The convex approximation (Section 11.1),

margin estimation (Section 11.2), L_2-SVM, and L_2-SVR problems are all quadratic problems, and the Frank-Wolfe algorithm can be used effectively to obtain approximate solutions. Moreover, while for MEB, and for some instances of these problems, $M = -A^T A$ for a known matrix $n \times d$ matrix A, the Frank-Wolfe algorithm needs only the matrix M in its operations. This fact is helpful when d is large, where in the context of SVM it is called the *kernel trick*.

Similarly to MEB, the MEE (minimum enclosing ellipsoid) problem, to find the smallest ellipsoid enclosing a set of points, is dual to a concave maximization problem over Δ [Khachiyan 1996; Kumar and Yildirim 2005; Ahipasaoglu et al. 2006]. This maximization problem is called the D-optimal design problem, and an algorithm similar to Algorithm 1.1 was proposed for it by Wynn [1970] and by Fedorov [1972]. Not all results given here apply in an interesting way to this problem, however, because the C_f bound is too large when considered over all of Δ.

This primal/dual approximation property is not far from the specification of a *coreset*, discussed next.

3.2. CORESETS. Coresets were first explicitly described for the MEB problem. In this context, for $\epsilon > 0$, an *ϵ-coreset* $P' \subset P$ has the property that if the smallest ball containing P' is expanded by a factor of $1 + \epsilon$, then the resulting ball contains P. Thus, P' gives both a good approximate solution and proves that no solution is much better.

A fundamental property of the MEB problem is that, as shown by Bădoiu et al. [2002], there are ϵ-coresets whose size (number of points) depends only on ϵ, and not on the dimension d, or the number n of points. Algorithms are known for finding ϵ-coresets of size $O(1/\epsilon)$ [Kumar et al. 2003; Bădoiu and Clarkson 2003], and size $\lceil 1/\epsilon \rceil$ is worst-case optimal [Bădoiu and Clarkson 2008]. Coresets for MEB have been applied to the k-center problem [Bădoiu et al. 2002], to computational biology [Bar-Joseph et al. 2003], and to machine learning [Nielsen and Nock 2006], including support vector regression [Tsang et al. 2005b].

The results of Ben-David et al. [2002] also imply the existence of coresets, for MEB and a few other problems. Their work relies on the existence result attributed to Maurey, as mentioned in Section 3.3; also Section 6 gives an argument like Maurey's. Their application was the *densest-ball problem*, discussed shortly.

As mentioned, the MEB problem is the Wolfe dual of an instance of the optimization problem (1). Moreover, a known algorithm for finding MEB coresets [Bădoiu and Clarkson 2003] is similar to Algorithm 1.1. This is not a coincidence: Section 8 shows via an adaptation of Algorithm 1.1 that ϵ-coresets exist for the dual problem of (1), with a size that depends only on ϵ and C_f. Here the idea of a coreset is generalized from the MEB problem to the more general setting.

As defined technically (Definition 8.1), a coreset here is a subset $N \subset \{1, \ldots, n\}$, or equivalently the face Δ_N of Δ specified by N, where Δ_N is the convex hull of $\{e(i) \mid i \in N\}$. In particular, a coreset is such a subset (or face) with the property that $\arg\max_{x \in \Delta_N} f(x)$ is a good approximate solution to the full problem (1), both primally and dually. In other words, N is a *combinatorial* specification of an approximate solution, which is also a certificate of the solution's near-optimality.

The technical definition includes a factor of 2 and of a variation C_f^* of C_f, so that as specialized to the MEB problem, the definition matches the previous one in the literature.

Choose a coordinate to make nonzero = choosing a specific dimension (or variable) that will be updated in the current iteration of the algorithm. => selecting a direction in which to move the current solution.

The coreset existence proof includes an algorithm, Algorithm 8.2, that builds a coreset; the algorithm is very similar to Algorithm 1.1, but does a little more work: having chosen a coordinate to make nonzero, it finds the point x that maximizes $f(x)$ over all points with the same set of nonzero entries. That is, it solves some small optimization subproblems. It deals with fewer variables.

By iteratively selecting coordinates and solving these small optimization subproblems, Algorithm 8.2 constructs a coreset by gradually adding points to it.

The coreset size, for a given quality of additive approximation ϵ, is at most $4C_f/\epsilon$, as shown in Theorem 8.3. A simpler general algorithm, given in Theorem 8.4, gives a poorer quality coreset, while a slower algorithm, Algorithm 9.1, gives a coreset of roughly half the size obtainable via the faster algorithm, with the same approximation quality. Algorithm 9.1 uses an "away" step within each iteration, in which a nonzero coordinate is set to zero. Such a step was considered by Todd and Yildirim [2005] as a heuristic improvement for optimization, and within the algorithm for optimal MEB coresets by Bădoiu and Clarkson [2008]. The results here generalize the latter, and are asymptotically optimal for MEB coresets, as $\epsilon \to 0$.

densest-ball problem

The exact size ℓ of a coreset of given quality is significant, because some algorithms exploiting coresets involve enumerating, for an input set of n points, all $\binom{n}{\ell}$ subsets of size ℓ. One such application is the densest-ball problem: given a set of points, find the smallest ball containing half the points. The existence of a small ϵ-coreset for the points inside that smallest ball implies that enumeration of all subsets of that size will allow the densest-ball problem to be solved approximately. A similar application of coresets in the convex approximation setting is described in Section 11.1; for the densest-ball problem, such algorithms were proposed by Ben-David et al. [2002], as mentioned.

Several previous papers have shown the effectiveness of coreset techniques for support vector machine (SVM) training and regression [Tsang et al. 2005a; Chu et al. 2004; Tsang et al. 2006], and the latter generalizes from the MEB problem to a more general quadratic objective function. The work here is inspired by and would seem to include that prior generalization. Har-Peled et al. [2007] give an algorithm for hard-margin SVM training that is not far from Algorithm 8.2. Their algorithm solves small subproblems optimally at each step, as in Algorithm 8.2, but unlike Algorithm 1.1, and is perhaps slower than Algorithm 1.1 as specialized to SVM training. (However, a variant of their algorithm avoids such subproblem computations [Har-Peled 2007].) Margin estimation, as in hard-margin SVM training, is discussed in Section 11.2.

Coresets for other classes of problems have seen wide application in computational geometry: see Agarwal et al. [2005] for a survey.

Mathematically, we can say: min f(x) = - max(-f(x))

3.3. SPARSE GREEDY APPROXIMATION. The problem (1) is often given in the equivalent formulation of minimizing a convex function, and in that form is considered in statistics, approximation theory, and machine learning. (Here we minimize a convex function by maximizing its concave negation.) For example, for an appropriate $d \times n$ matrix A and point p, maximizing $f(x) := -\|Ax - p\|_2^2$ for $x \in \Delta$ would correspond to the problem of finding the convex combination of the columns of A that is closest to p in Euclidean norm. Similarly, if a collection of functions $p_i(t)$ and a target function $p(t)$ is given, then $f(x) := -\|\sum_i x_i p_i(t) - p(t)\|_2^2$, corresponds to the problem of finding the closest (in L_2) convex combination of the $p_i()$ functions to match p. The case where maximizing f minimizes the Kullback-Leibler divergence was analyzed by Li and Barron [2000], and by Zhang [2002] , and is discussed in Section 11.5.

ALGORITHM 1.1:
Pick as x(0) the vertex of Delta with largest f value;
For k = 0 ...Inf, find x(k+1) from x(k) as follows:
i':= arg maxi grad f (x(k))i;
alpha':= arg max alpha e [0,1] f(x(k) + alpha(e(i') - x(k)));
x(k+1) := x(k) + alpha'(e(i') - x(k));
that is, x(k+1) is the point on the line segment from x(k) to e(i') that maximizes f;

unit simplex

ALGORITHM 3.1. *Given concave function* $f : \mathbb{R}^n \to \mathbb{R}$, *maximize* f *in* Δ *as follows.*

Pick as $x_{(0)}$ *the vertex of* Δ *with largest* f *value;*
For $k = 0 \ldots \infty$, *find* $x_{(k+1)}$ *from* $x_{(k)}$ *as follows:*
$(i', \alpha') := \arg\max_{(i,\alpha)} f(x_{(k)} + \alpha(e(i) - x_{(k)}))$;
$x_{(k+1)} := x_{(k)} + \alpha'(e(i') - x_{(k)})$;
that is, $x_{(k+1)}$ *maximizes* $f(x)$ *over all line segments from* $x_{(k)}$ *to the vertices of* Δ;

Find the coordinate i and step size alpha that maximize the function f when moving from the current point x(k) in the direction of the coordinate i. Specifically, we're considering the line segment from x(k) to the vertex e(i) of the unit simplex delta.

Move along the line segment from x(k) to the vertex e(i') with the step size alpha'.

FIG. 2. Sparse greedy approximation

Algorithm 1.1 can be viewed generically as finding, at each step, a good coordinate in which to change x, and then adjusting this coordinate to maximize $f(x)$. One such algorithm in particular, described in Figure 2, has been analyzed before. Plainly this algorithm will find an iterate $x_{(k+1)}$ for which $f(x_{(k+1)})$ is at least as large as for Algorithm 1.1. Sparsity results for Algorithm 3.1 were shown in generality by Zhang [2003], whose proof has the same general idea as analyses by Li and Barron [2000] and Jones [1992]. Maurey proved the existence of a sparse x' with $f(x') \geq f(x_*) - O(1/(k+3))$, for a class of functions $f(x)$ arising in convex approximation, using a probabilistic argument ([Pisier 1981]; see also Barron [1993]), and Jones [1992] showed that a greedy algorithm such as Algorithm 3.1 yields a similar output.

Algorithm 3.1 tends to yield solutions that are concentrated on a subset of the dimensions while other dimensions remain zero or close to zero.

Algorithm 3.1, and variations, has been applied to boosting, regression, convex approximation, and estimation of mixture models [Zhang 2003] (applicability to SVM training is also mentioned by Zhang, but no specific results are given). Algorithm 1.1 can thus be similarly applied, for those $f(x)$ whose gradients can be computed; this includes all the specific applications considered by Zhang. When a single gradient computation is faster than n function evaluations, as may be true, Algorithm 1.1 and its variants will be faster than Algorithm 3.1 and comparable variants. (However, it's also true that the n evaluations of a function at closely related arguments may be sped up.) Also, since the problems discussed by Zhang involve optimization over convex hulls, the coreset results of Section 8 apply. The existence of coresets for these problems may have some useful applications; the case of convex approximation is discussed in Section 11.1.

Algorithm 1.1 vs 3.1

Algorithm 1.1 is also closely related to Anyboost.L1 [Mason et al. 2000], which can roughly be viewed as the specialization of Algorithm 1.1 to boosting. Although convergence results were shown for Anyboost, it doesn't seem that quantitative results have been, and so the results here are a new contribution in that respect.

4. *Important Cases of* $f(x)$ *and* C_f

When $f(x)$ satisfies some additional conditions beyond differentiability and concavity, we can say more about the dual and C_f.

4.1. TWICE DIFFERENTIABLE.

When f is twice differentiable, the Taylor expansion at $\alpha = 0$ of $f(x + \alpha(x' - x))$ as a function of α is.

$$f(x + \alpha(z - x)) = f(x) + \alpha(x' - x)^T \nabla f(x) + \frac{1}{2}\alpha^2 (x' - x)^T \nabla^2 f(\tilde{x})(x' - x).$$

Here the last term is the Lagrange remainder, where $\tilde{x}$ is a point on the line segment $[x, x + \alpha(x' - x)]$. (Everywhere but in Section 9, the values of α considered are between 0 and 1, so that $\tilde{x}$ in on the line segment $[x, x']$.)

Since f is concave, $\nabla^2 f(\tilde{x})$ is negative semidefinite, and so the last term is always no more than zero. Rearranging (5), (5) + Taylor's expansion

Cf is less than or equal to the sup (least upper bound of a set of values).

$$C_f \le \sup_{x,x' \in \Delta} -\frac{1}{2}(x' - x)^T \nabla^2 f(\tilde{x})(x' - x)$$
$$\le \sup_{x,x' \in \Delta, \tilde{x}} -\frac{1}{2}(x' - x)^T \nabla^2 f(\tilde{x})(x' - x), \quad (11)$$

where $\tilde{x}$ is constrained to lie in Δ, and on the line through x and x'.

4.2. OPTIMIZATION WITHIN A CONVEX HULL. For functions of the form $f(x) = \hat{f}(Ax)$, the gradient

reparameterization

$$\nabla f(x) = A^T \nabla \hat{f}(\overset{c}{Ax}),$$

so that

$$f(x) - x^T \nabla f(x) = \hat{f}(Ax) - x^T (A^T \nabla \hat{f}(\overset{c}{Ax}))$$
$$= \hat{f}(Ax) - x^{TAT} \nabla \hat{f}(\overset{c}{Ax})$$
$$= \hat{f}(c) - c^T \nabla \hat{f}(c),$$

for $c := Ax$. The dual problem (4) becomes

$$\min_{z \in \mathbb{R}, c \in \mathbb{R}^d} z + \hat{f}(c) - c^T \nabla \hat{f}(c) \quad (12)$$
$$\text{subject to } z\mathbf{e} \ge A^T \nabla \hat{f}(c)).$$

We can write this as $\min_c \hat{w}(c)$, with $\hat{w}(c) := \hat{z}(c) + \hat{f}(c) - c^T \nabla \hat{f}(c)$, and $\hat{z}(c) := \max_i (A^T \nabla \hat{f}(c))_i$. Note that $\hat{w}(Ax) = w(x)$, just as $\hat{f}(Ax) = f(x)$. (this is given)

Here the bound (11) for C_f can be expressed as follows. With $a := Ax \in A\Delta$, and similarly $b := Az$, $\tilde{a} := A\tilde{x}$, and observing that $\nabla^2 f(x) = A^T \nabla^2 \hat{f}(Ax) A$,

$$C_f \le \sup_{x,z \in \Delta, \tilde{x}} -\frac{1}{2}(z - x)^T \nabla^2 f(\tilde{x})(z - x)$$
$$= \sup_{x,z \in \Delta, \tilde{x}} -\frac{1}{2}(z - x)^T (A^T \nabla^2 \hat{f}(A\tilde{x}) A)(z - x)$$
$$= \sup_{a,b \in A\Delta, \tilde{a}} -\frac{1}{2}(b - a)^T \nabla^2 \hat{f}(\tilde{a})(b - a) \quad (13)$$

b - a = Az - Ax = A(z-x)
(b - a)^T = (A(z-x)))^T = (z-x)^T * A^T

convex hull

The preceding results imply approximation results for $\hat{f}$, over $A\Delta := \{Ax \mid x \in \Delta\}$, that is, the convex hull of the columns of A.

The following is a corollary of Theorem 2.2, and is the main part of Theorem 3.1 of Zhang [2003].

THEOREM 4.1. *If $f(x) = \hat{f}(Ax)$, where $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ is a twice differentiable concave function, then with $C_f = -\frac{1}{2} \sup_{a,b \in A\Delta, \tilde{a} \in [a,b]} (b - a)^T \nabla^2 \hat{f}(\tilde{a})(b - a)$, there is a k-face Δ' of Δ and a point $a \in A\Delta'$ such that*

b = Az

$$\hat{f}(a) - \inf_{b \in A\Delta} \hat{f}(b) \le 4C_f/(k + 3).$$

From Theorem 2.2:

$$[f(x^1) - f(x_{(k)})]/4C_f \le \frac{1}{k+3}$$
$$f(x^1) - f(x_{(k)}) \le \frac{4C_f}{k+3}$$

PROOF. The function $f(x) := \hat{f}(Ax)$ is concave if $\hat{f}$ is, and the theorem follows from Theorem 2.2 and the bound (13) for C_f. □

4.3. QUADRATIC FUNCTIONS. If $f(x)$ is a quadratic concave function, that is, has the form

$$f(x) = a + x^T b + x^T M x, \tag{14}$$

where $a \in \mathbb{R}$, $b \in \mathbb{R}^n$, and M is a negative semidefinite $n \times n$ matrix, then $f(x) = \hat{f}(Ax)$, where

$$\begin{aligned} \hat{f}(c) &:= a + c_d - \tilde{c}^T \tilde{c} \\ \tilde{c} &:= [c_1, \ldots, c_{d-1}, 0]^T \text{ (a column vector)} \\ A &:= \begin{bmatrix} \tilde{A} \\ b^T \end{bmatrix} \\ M &= -\tilde{A}^T \tilde{A} \\ \tilde{A} &\text{ is } (d-1) \times n, \text{ for some } d. \end{aligned}$$

Note that such a $\tilde{A}$ can always be found, and the given $\hat{f}$ is concave.

The dual problem here is

$$\begin{aligned} &\min_{z \in \mathbb{R}, x \in \mathbb{R}^d} z - x^T M x \\ &\text{subject to } z \geq \max_i (b + 2Mx)_i. \end{aligned} \tag{15}$$

For quadratic f, $\nabla^2 f(x) = 2M$. Moreover, there is a factorization $-M = A^T A$ for a matrix A, and so

$$\begin{aligned} C_f &\leq \sup_{x,z \in \Delta} -\frac{1}{2}(z-x)^T \nabla^2 f(x)(z-x) \\ &\leq \sup_{x,z \in \Delta} (z-x)^T A^T A (z-x) \\ &= \sup_{a,b \in A\Delta} \|b-a\|_2^2. \end{aligned} \tag{16}$$

That is, C_f is no more than the square of the diameter of the polytope $A\Delta$, and the latter is bounded by the square of the diameter of the minimum enclosing ball of $A\Delta$, and hence by four times the square of the radius of that ball.

4.4. ASYMPTOTIC C_f^*. It is of interest to bound a more restricted set of values than is used for C_f, namely, where x is near the optimal point x_*. This models the use of C_f in analyzing the algorithms discussed here, as they converge to the optimum. Given region $\Delta' \subset \Delta$, consider $C_f(\Delta')$, defined as in (5), but where x is restricted to Δ'. For given $\gamma \geq 0$, consider Δ_γ to be the subset of Δ that contains all x such that $f(x) \geq f(x^*) - \gamma$. For large enough $k = \Omega(C_f/\gamma)$, as $\gamma \to 0$, all $x_{(k)}$ are in Δ_γ, and so for $C_f(\Delta\gamma) < C_f = C_f(\Delta)$, the additive bound of Theorem 2.2 can be tightened for large k. We will assume at times that $C_f(\Delta_\gamma) = C_f^*(1 + o(1))$ as $\gamma \to 0$, where $C_f^* := C_f(\{x_*\})$. This is true for quadratic functions, for which C_f^* is the maximum squared distance of x_* to a point in $A\Delta$. For MEB in particular,

C_f^* is the square of the radius of the MEB. That is, the asymptotic C_f^* for MEB is one quarter the general bound. This distinction is not significant in general, but by showing results with respect to C_f^*, it will be possible to show that some constructions for MEB are asymptotically optimal, including the constant factor in the leading term.

5. *Outline*

A probabilistic argument for the existence of sparse approximate solutions is reviewed in Section 6, with a particular example related to k-means clustering. Section 7 gives some variations on Algorithm 1.1 that do less work per iteration by using a predefined series of α' values. Section 8 considers the case of optimization within the convex hull of a set of points, and then gives a construction for coresets of functions $f(x)$ of the general form $\hat{f}(Ax)$ where A is an $n \times d$ matrix; as noted, this includes all quadratic concave $f(x)$. Tighter bounds for coresets using an algorithm with "away" steps is discussed in Section 9. In Section 10, some tail estimates for random data are shown, and finally some specific applications are considered in Section 11: convex approximation, margin estimation, Adaboost, approximation in L_v, and mixture density estimation. The results for margin estimation include an error probability estimate based on the tail estimates of Section 10.

Figure 3 lists some of the problems that can be described as instances of (1). It is intended as a summary and as a list of pointers to applications and results here and elsewhere: the first-time reader need not try to understand everything in it.

In the table, the conditions that x is an n-vector and A is $d \times n$ imply the dimensions of the remaining values. As elsewhere in this article, the notation c^2 for a vector c denotes $c^T c$, and **e** is the n-vector with all coordinates equal to one. Also the i'th column of A is p_i, or instead $y_i p_i$, when a given n-vector y is associated with the problem. As usual in this article, $c = Ax$, and $M := -A^T A$, a negative-semidefinite matrix. The notation $(A \circ A)_j$ denotes the n-vector whose i'th coordinate is p_{ij}^2, and $H \succeq 0$ denotes the condition that matrix H is positive semidefinite.

(*) Important: The dual of MEB is a quadratic function!

The three problems listed after the general quadratic are themselves quadratic, so that the duals of MEB, convex approximation, and L_2-SVM can be read off from the dual for the general quadratic. At the risk of some confusion, the "M" of the general quadratic is instantiated for L_2-SVM as the matrix with block structure

$$-[A^T \; y \; I/C][A^T \; y \; I/C]^T,$$

and so the "c" of the general quadratic is for L_2-SVM the $(d + 1 + n)$-vector $[c^T \; q \; w^T]^T$. Along similar lines, the "A" of the general quadratic is not quite the same as the "A" for the general expression for maximizing functions within a convex hull of points.

of sparse iterates

6. *Existence via a Probabilistic Argument*

A probabilistic proof can be given for the existence of a sparse x, that has K nonzero entries, such that $f(x)$ is not far from $f(x_*)$. (Not a coreset, however.) approximate solution For simplicity, consider here only the quadratic case $f(x) = a + x^T b + x^T Mx$,

Problem	$f(x)$ References	Dual problem Notes
General Function	$f(x)$ §2.1 for dual	$\min_{z,x}$ $z + f(x) - x^T \nabla f(x)$ s.t. $z\mathbf{e} \geq \nabla f(x)$
Functions Within Convex Hull	$f(x) := \hat{f}(Ax)$ §8	$\min_{z,c}$ $z + \hat{f}(c) - c^T \nabla \hat{f}(c)$ s.t. $z\mathbf{e} \geq A^T \nabla \hat{f}(c)$
General Quadratic	$a + x^T b + x^T Mx$ §4.3	$\min_{z,c}$ $a + z + c^2$ s.t. $z\mathbf{e} \geq b - 2A^T c$ $c = Ax$; $C_f \leq \text{Diam}(A\Delta)^2$
Minimum Enclosing Ball SVDD	$x^T b(A) + x^T Mx$ §1, §4.3	$\min_{z,c}$ $z + c^2$ s.t. $z\mathbf{e} \geq b(A) - 2A^T c$ $b(A)_i := p_i^2$, $C_f^* \leq \text{Rad}(A\Delta)^2$
Convex Approximation, Margin Estimation Hard Margin SVM	$-(p - Ax)^2$ §11.1,§11.2	$\min_{z,c}$ $-p^2 + z + c^2$ s.t. $z\mathbf{e} \geq 2A^T p - 2A^T c$
L_2-SVM (two-class)	$-x^T[M + yy^T + I/C^2]x$ $= -([A^T\ y\ I/C]^T x)^2$ [Tsang et al. 2006]	$\min_{z,c,q,w}$ $z + c^2 + q^2 + w^2$ s.t. $z\mathbf{e} \geq -2(A^T c + qy + w/C)$ $C_f \leq (\text{Diam}(A\Delta) + 1 + 1/C)^2$; $c = Ax$, $q = y^T x$, $w = x/C$; $y_i = \pm 1$
L_2-SVR	(see [Tsang et al. 2006])	
Adaboost	$-\log \sum_j \exp(-\eta(Ax)_j r_j)$ §11.3 [Zhang 2003]	(not given here) A_{ji} = predictor i, datapoint j; $r_j = \pm 1$; $C_f \leq 4\eta^2$;
L_v Regression, $1 < v < \infty$	$-\|p - Ax\|_v^{v'}$, $v' := \min\{v, 2\}$ §11.4,[Zhang 2003]	(not given here) $C_f \leq 2(v-1)\,\text{Diam}(A\Delta)^2$, $v \geq 2$;
Mixture Density Estimation	$\sum_j p_j \log(Ax)_j$ §11.5,[Zhang 2002] [Li and Barron 2000]	(not given here) $C_f \leq \sup_{i,i'} \sum_j p_j A_{ji}^2 / A_{ji'}^2$
ME Ellipsoid, D-Optimal Design	$\log\det AXA^T$ [Ahipasaoglu et al. 2006] [Khachiyan 1996]	$\min_{H \succeq 0}$ $-\log\det H$ s.t. $p_i^T H p_i \leq d$, $i = 1 \ldots n$ diagonal X, $X_{ii} = x_i$;
ME Axis-Aligned Ellipsoid	$(1/2)[d \log d + \sum_{1 \leq j \leq d} \log(x^T(A \circ A)_j - (Ax)_j^2)]$ [Kumar and Yildirim 2005]	$\min_{\gamma,\mu}$ $-\sum_j \log \gamma_j$ s.t. $\sum_j (\gamma_j p_{ij} - \mu_j)^2 \leq 1$ $i = 1 \ldots n$

FIG. 3. Some instances of optimization problem reference **??**plain]eq:primal gen.

also quadratic

The dual formulations of these can be derived directly from the dual formulation of a more general quadratic optimization problem.

with M negative semidefinite as usual. Let $Y \in \mathbb{R}^d$ be a random variable defined as follows: a coordinate i is chosen with probability x_i^* (here writing x^* for the optimum), and the i'th coordinate $Y_i := 1/K$, and all other coordinates are zero. Consider a collection Y^k, $k = 1 \ldots K$, of such random variables, independently distributed. Then for $Z := \sum_k Y^k \in \Delta$, $\mathrm{E}[Z] = x^*$, and at most K coordinates of

Y has a non-zero value of 1/K in one randomly chosen coordinate, and all other coordinates are zero:
0 0 0 ... 1/K ... 0 0

Y1 + Y2 + ... + Y_K → E[Z] = (1/K) * (xi*) * K = xi*

Z are nonzero. Also, for $i \neq j$,

$$\begin{aligned}
\mathrm{E}[Z_i Z_j] &= \mathrm{E}\left[\sum_k Y_i^k \sum_{k'} Y_j^{k'}\right] \\
&= \sum_{k \neq k'} \mathrm{E}\left[Y_i^k Y_j^{k'}\right] + \sum_k \mathrm{E}\left[Y_i^k Y_j^k\right] \\
&= \sum_{k \neq k'} \mathrm{E}\left[Y_i^k\right] \mathrm{E}\left[Y_j^{k'}\right] + 0 \\
&= (K^2 - K) x_i^* x_j^* / K^2 = x_i^* x_j^* (1 - 1/K),
\end{aligned}$$

Didn't go into detail of the math here.

and

$$\begin{aligned}
\mathrm{E}\left[Z_i^2\right] &= \mathrm{E}\left[\sum_k Y_i^k \sum_{k'} Y_i^{k'}\right] \\
&= \sum_k x_i^* / K^2 + \sum_{k \neq k'} (x_i^*)^2 / K^2 \\
&= x_i^* / K + (x_i^*)^2 (1 - 1/K),
\end{aligned}$$

and so

$$\mathrm{E}[Z^T M Z] = \sum_{i,j} m_{ij} \mathrm{E}[Z_i Z_j] = (x^*)^T M x^* (1 - 1/K) + \sum_i m_{ii} x_i^* / K.$$

Using these observations, and with as usual $M = -A^T A$ where A has columns p_i,

f(x*) f(z)

$$\begin{aligned}
&a + (x^*)^T b + (x^*)^T M x^* - \mathrm{E}[a + Z^T b + Z^T M Z] \\
&\quad = (x^*)^T M x^* - \mathrm{E}[Z^T M Z] \\
&\quad = (x^*)^T M x^* - \left[(x^*)^T M x^* (1 - 1/K) + \sum_i m_{ii} x_i^* / K\right] \\
&\quad = (x^*)^T M x^* / K + \sum_i p_i^2 x_i^* / K \qquad \text{mii = -pi^2} \\
&\quad \leq \sum_i p_i^2 x_i^* / K \qquad (17)
\end{aligned}$$

Since $f(x^*) - \mathrm{E}[f(Z)]$ is bounded in this way, there must exist some z with K nonzero entries that meets this bound. The preceding satisfies

xi* is a prob., so sum of all probabilities = 1

$$\sum_i p_i^2 x_i^* / K \leq [\max_i p_i^2] \sum_i x_i^* / K = \max_i p_i^2 / K. \qquad (18)$$

Any quadratic problem can be translated, without loss of generality for this construction, so that the origin is the center of the MEB of the p_i, implying $\max_i p_i^2 = \mathrm{Rad}(A\Delta)^2$, and so $\mathrm{Rad}(A\Delta)^2/K$ is a bound on the additive error. As mentioned, for the special case of convex approximation, such a bound was shown by Maurey [Pisier 1981] in a similar way. The bound (18) is slightly sharper, with respect to the constant, than the bounds shown for Algorithm 1.1 and Algorithm 3.1.

For some problems, it could be that (17), the average of the p_i^2 as weighted by x_i^*, gives a better bound than simply using $\max_i p_i^2$. A specific useful example of this is given next; heuristically, this implies that sparse solutions exist that are significantly better than the greedy constructions here can find. However, for MEB the x_i are nonzero only for the p_i with largest p_i^2, and so $\sum_i p_i^2 x_i^*$ for MEB will be $\mathrm{Rad}(A\Delta)^2$, implying that the greedy algorithms do pretty well for MEB.

As an example of a problem where (17) can be helpful, consider the quadratic problem where the objective function is

$$-\sum_i (p_i - Ax)^2/n = -\left[\sum_i p_i^2/n - 2x^T A^T p_i/n + x^T A^T Ax\right]$$
$$= -\left[\left[\sum_i p_i^2/n\right] - 2x^T A^T A\mathbf{e}/n + x^T A^T Ax\right],$$

for which the optimum x is $\mathbf{e}/n$. That is, the problem is to minimize the sum of squares of Euclidean distances to the p_i, and the solution is the center of mass, the coordinate-wise mean. The additive error, from (17), is $\sum_i p_i^2/nK$. We can assume, without loss of generality, that $\sum_i p_i = 0$, and so the optimum value of the objective function is $-\sum_i p_i^2/n$, and the *relative* error is $1/K$. This observation is due to Inaba et al. [1994], and has been applied to k-means clustering. The probabilistic choice in the preceding construction is for this problem simply a uniform random sample of the p_i.

Is there a deterministic construction that finds a sample of comparable quality?

7. *Variations*

A few variations of Algorithm 1.1 or Algorithm 3.1 suggest themselves. For one, since $h(x_{(k)}) \le 1/(k+3)$ by Theorem 2.2, and the optimal α minimizing (8) is $2g(x_{(k)}) \le 2h(x_{(k)})$, we might avoid searching for α simply by using $\alpha_k := 2/(k+3)$ at step k.

With this multiplier, (8): h(y) <= h(x) - alpha*g(x) + (alpha^2)/4

$$h(x_{(k+1)}) \le h(x_{(k)}) - \alpha_k h(x_{(k)}) + \alpha_k^2/4,$$

(3) w(x) >= f(x) => g(x) >= h(x)

by (8), and using $g(x) \ge h(x)$. Since for $\alpha_k < 1$ the right-hand side is increasing in $h(x_{(k)})$, it is maximized when $h(x_{(k)})$ is at its maximum, which inductively is no more than $1/(k+3)$. We have

$$h(x_{(k+1)}) \le \frac{1}{k+3} - \frac{2}{k+3}\frac{1}{k+3} + \frac{1}{4}\left(\frac{2}{k+3}\right)^2$$
$$= \frac{1}{k+4} - \frac{1}{(k+4)(k+3)^2}$$
$$< 1/(k+4).$$

Theorem 2.2: h(x(k)) <= 1/(k + 3)

Thus searching for α is not necessary for the bounds to hold. Note that with this approach, the function $f(x)$ need only be evaluated n times at the beginning, to determine $x_{(0)}$; only gradient evaluations are needed thereafter.

Such lazier algorithms have already been proposed for special cases, for example by Li and Barron [2000], and by Bădoiu and Clarkson [2003]. (It may only be

coincidental that subgradient and stochastic gradient algorithms also often are described using a step size that decreases as $1/k$.)

The primal-dual approximation bounds of Theorem 2.3 can also be attained using a fixed α_k sequence: choose $\alpha_k = 2/(k+3)$ as before, for $k \leq K := \lceil 1/\epsilon \rceil$, and then use $\alpha_k = 2\epsilon$ for $k > K$. If $g(x_{(k)}) \geq \epsilon$, for all $k = K \ldots 2K$, then $h(x_{(2K)}) < 0$, since $h(x_{(k+1)}) \leq h(x_{(k)}) - \epsilon^2$ for $k > K$. Therefore some $k \leq 2K$ must have $g(x_{(k)}) < \epsilon$.

Another variation is to work harder within the current face: that is, optimize the coordinates that are currently nonzero, before making another coordinate nonzero. For example, the sparse greedy algorithm itself could be used for several "minor" steps, choosing among the current set of nonzero coordinates, obtaining a solution that is close to optimal, subject to the restriction of being on the current k-face. Heuristically, this suggests that more "bang for the buck" would be obtained at each major step, but improvements in provable bounds don't seem to have been obtained. Such a harder-working algorithm is helpful, however, for showing the existence of coresets, as discussed next. (This variant algorithm is not far from orthogonal matching pursuit, as discussed in Section 12.)

8. *Coresets*

The term "coreset" has many different technical meanings; the most natural definition for this article encompasses some, but not all of them. To give that definition, first we need to give notation for some restricted versions of the primal and dual problems.

As in Section 3.2, for $N \subset \{1, 2, \ldots n\}$ let Δ_N denote the face of Δ that is the convex hull of $\{e(i) \mid i \in N\}$. Then the Wolfe dual of the restricted primal problem

$$\max_{x \in \Delta_N} f(x)$$

is $\min_x w_N(x)$, where

$$w_N(x) := z_N(x) + f(x) - x^T \nabla f(x), \tag{2}$$

and $z_N(x) := \max_{i \in N} \nabla f(x)_i$. We let x_N denote the optimum for the restricted primal; this is also the optimum for the restricted dual. (by strong duality primal solution = dual solution)

The dual of the restricted primal has fewer constraints than the dual to the full problem, and so $w_N(x) \leq w(x)$, and in general $w_N(x) \leq w_{N'}(x)$ for $N \subset N'$. This is consistent with an analogous relation for the primal problem: the more restricted the domain over which f is maximized, the smaller that maximum can be. (*) Important!

Theorem 2.3 shows that Algorithm 1.1 can be used to find a point x' with few nonzero entries, and with primal and dual values close to optimal. That is, x' is a kind of sparse certificate regarding the optimum value of the optimization problem. A coreset is combinatorial version of such a certificate.

Definition 8.1. Given a concave function $f(x)$, an ϵ-coreset for the problem $\max_{x \in \Delta} f(x)$ is a subset N of the coordinate indices so that $w(x_N) - f(x_N) \leq 2\epsilon C_f^*$.

(Note that the asymptotic nonlinearity measure C_f^* is used, not the global version C_f. This, with the factor of 2, implies that an ϵ-coreset for MEB by this definition is a $(1/(1+1/\epsilon))$-coreset by prior definitions [Bădoiu and Clarkson 2008], that

ALGORITHM 8.2. *Given concave function* $f : \mathbb{R}^n \to \mathbb{R}$ *and* $\epsilon > 0$, *find a coreset* N_k *as follows.*

Let $i' := \arg\max_i f(e(i))$ *and* $N_0 := \{i'\}$;
For $k = 0 \ldots \infty$, *find* N_{k+1} *from* N_k *as follows:*
  if $g(x_{N_k}) < \epsilon$ *return* N_k;
  $i' := \arg\max_i \nabla f(x_{N_k})_i$;
  $N_{k+1} := N_k \cup \{i'\}$;

Each iterate x(k) is the optimum for its corresponding k-face of Delta. This ensures that xN is a "canonical" approximate solution, determined by N.

FIG. 4. An algorithm for general coresets

is, nearly the same. As discussed in Section 4.4, $C_f^* \le C_f \le 4C_f^*$ for quadratic problems, and for the MEB problem, C_f^* is equal to the square of the radius of the MEB.)

For MEB: Cf* = r^2

(*) Important! The active set in general is not a coreset!

It might be thought that the set N of indices of the nonzero coordinates of a sparse primal/dual approximate solution x' is a coreset. This is not necessarily true, as it may be that $w(x_N) \gg w(x')$. A patch for this is to use a variant of Algorithm 1.1, for which each iterate $x_{(k)}$ is the optimum for its corresponding k-face of Δ. This ensures that x_N is a "canonical" approximate solution, determined by N.

The variant algorithm is shown in Figure 4. Here is a theorem whose proof uses it.

THEOREM 8.3. *Let* $f(x)$ *be a concave function. Algorithm* 8.2 *returns a set* N_k *indexing a* $(2\epsilon C_f/C_f^*)$*-coreset, for* $k \le 2K$, *where* $K := \lceil 1/\epsilon \rceil$. *This requires* $O(ndK + K\,Q(f, K))$ *time, assuming evaluation of* $\nabla f(x)$ *needs* $O(nd)$ *time, for some* d, *and where where* $Q(f, K)$ *is the time needed to find the restricted maxima* x_{N_k} *for* $k \le 2K$. *The returned set is a* $\epsilon(2 + o(1))$*-coreset, as* $\epsilon \to 0$, *assuming* $C_f(\Delta_\gamma) = (1 + o(1))C_f^*$ *as* $\gamma \to 0$.

PROOF. Algorithm 8.2 computes an iterate x_{N_k} such that $f(x_{N_k})$ is at least as large as that for Algorithm 3.1, and hence also for Algorithm 1.1. Since Algorithm 8.2 uses the same choice of new coordinate i' as Algorithm 1.1 it follows that Theorem 2.2 and Theorem 2.3 apply to it also, so that some N_k for $k \le 2K$ has $g(x_{N_k}) \le \epsilon$. Since each iterate x_{N_k} of Algorithm 8.2 is an optimum for N_k, it follows that the N_k with $g(x_{N_k}) \le \epsilon$ is a $(2\epsilon C_f/C_f^*)$-coreset, using the definitions (7) and Definition 8.1 of g and coresets. The time bound follows from Theorem 2.3 and the structure of Algorithm 8.2. The last statement uses the convergence of x_{N_k} and the asymptotic relation discussed in Section 4.4. □

Algorithm 1.1 and Algorithm 3.1 can also be used to create coresets in the following more direct way, but the bounds are worse.

THEOREM 8.4. *After* K^2 *iterations of Algorithm* 1.1 *(or Algorithm* 3.1*), where* $K := 1/\epsilon$, *the coordinate set* N_{K^2} *indexing nonzero entries of* $x_{(K^2)}$ *corresponds to a* $(2\epsilon C_f/C_f^*)$*-coreset.*

We need (1/epsilon)^2 iterations to create a coreset. => O((1/epsilon)^2)

PROOF. From Theorem 2.1, since $h(x) \ge 0$ for all x, it must hold that $g(x) \le \sqrt{h(x)}$ for all $x \in \Delta$. Moreover, by definition $f(x_{N_k}) \ge f(x_{(k)})$, and so $h(x_{N_k}) \le h(x_{(k)})$. (Here again, x_{N_k} denotes the optimum in Δ_{N_k}.) Since by Theorem 2.2, the value $h(x_{(K^2)}) \le 1/(K + 3)^2$, it follows that

Theorem 2.1: h(x(k+1)) <= h(x(k)) - g(x(k))^2

$$g(x_{N_{K^2}}) \le \sqrt{h(x_{N_{K^2}})} \le \sqrt{h(x_{(K^2)})} \le 1/(K + 3) < \epsilon,$$

Theorem 2.1 — h(x_Nk) <= h(x_k) — Theorem 2.2 — K = 1/epsilon

Theorem 2.2: h(x(k)) = [f(x*) - f (x(k))] / 4Cf <= 1/(k + 3)

As we saw on the previous page: The more restricted the domain over which f is maximized, the smaller that maximum can be.

and so N_{K^2} is an $(4\epsilon C_f/C_f^*)$-coreset, using the definitions (7) and Definition 8.1 of g and coresets. □

The main case of interest for these coreset results is for functions $f(x)$ that have the form $f(x) = \hat{f}(Ax)$, where A is a $d \times n$ matrix for some d, and $\hat{f} : \mathbb{R}^d \to \mathbb{R}$. Here a subset N of the indices corresponds to a subset of the columns of A.

As discussed in Section 4, the nonlinearity measure C_f is bounded by the squared diameter of $A\Delta$, which is bounded by the squared diameter of the MEB of the columns of A, and when f corresponds to the MEB problem, C_f^* is the squared radius of the MEB. Thus Theorem 8.3 implies that an ϵ-coreset exists for MEB, of size close to $4/\epsilon$.

Coreset for the MEB problem.

It was claimed near Definition 8.1 that the general definition of an ϵ-coreset specializes for MEB to something close to the standard definitions for MEB. For example, one definition ("alternate" in Bădoiu and Clarkson [2008]) is that $r(c_N) \le r_N(c_N)/(1-\epsilon)$, where $r(c_N) = \sqrt{\hat{w}(c_N)}$ is the distance of c_N to the farthest input point p_i, and $r_N(c_N) = \sqrt{\hat{w}_N(c_N)}$ is the distance of c_N to the farthest point indexed by N; that is, if the smallest ball containing the points indexed by N is expanded by a factor of $1/(1-\epsilon)$, the resulting ball contains all the points.

(*) Important!

To show this relation: suppose N is an ϵ-coreset for MEB, by Definition 8.1. Then $w(c_N) - w_N(c_N) = w(c_N) - f(c_N) \le 2\epsilon C_f^*$. Since here $C_f^* = w(c_*) = r(c_*)^2$, we have

this holds for MEB

r(cN)^2 = w_hat(cN)

$$r(c_N)^2 - r_N(c_N)^2 \le 2\epsilon r(c_*)^2 \le 2\epsilon r(c_N),$$

There is a mistake here, should be: r(cN)^2/(1-epsilon).

or

$$r(c_N)^2 \le r_N(c_N)^2/(1-2\epsilon),$$

Also, probably a mistake here, should be: (1 - epsilon).

or

$$r(c_N) \le r_N(c_N)(1+o(1))/(1-\epsilon),$$

as $\epsilon \to 0$, since

Algebra:

$$\frac{1}{1-2\epsilon} = \frac{1}{(1-\epsilon)^2}\left(1 + \frac{\epsilon^2}{1-2\epsilon}\right) \le \frac{1}{(1-\epsilon)^2}\left(1 + \frac{\epsilon^2}{2(1-2\epsilon)}\right)^2.$$

9. *Coresets via "Away" Steps*

Once we add an atom to the active set, it remains in the active set (no dropping).

The algorithms presented so far have been monotone in the sense that once a coordinate of x becomes nonzero, it remains so, or at least is not specifically made to be zero again. However, a few known algorithms have been nonmonotone: an algorithm for optimal MEB coreset construction [Bădoiu and Clarkson 2008] preserves the number of nonzero entries: the number of nonzero entries is some K, and in a single step, a new coordinate is made nonzero, and then another coordinate is set to zero, preserving the number of nonzero coordinates. It is shown that progress can be made in this way, by showing that a quantity similar to $g(x)$ is large enough.

ALGORITHM 9.1. *Given concave function $f : \mathbb{R}^n \to \mathbb{R}$ and $\epsilon > 0$, find a coreset N as follows.*

Let $i' := \arg\max_i f(e(i))$ *and* $N := \{i'\}$;
Repeatedly update N as follows:
  if $g(x_N) \le \epsilon/2$ *return* N;
  $i' := \arg\max_i \nabla f(x_N)_i$;
  $N := N^* := N \cup \{i'\}$;
  if $|N^*| > \lceil 1/\epsilon \rceil$:
    $x := x_{N^*}$;
    $i'' := \arg\min_{i \in N^*} x_i$;
    $N := N^{**} := N^* \setminus \{i''\}$;

away step

This algorithm preserves the number of nonzero entries in the active set.

After some time (see if condition): in a single step, a new coordinate is made nonzero, and then another coordinate is set to zero, preserving the number of nonzero coordinates.

FIG. 5. A coreset algorithm with "away" steps

Another algorithm, by Todd and Yildirim [2005], discusses a version of Algorithm 1.1 with an additional possible "away" step: it considers whether reducing a nonzero variable, not necessarily to zero, might improve the objective function. If so, the reduction is done. No provable improvement is shown, however, by including such a step.

It is shown next that a constant-factor improvement in provable coreset size is obtainable, by including a step that reduces a nonzero variable to zero. The improvement is almost as sharp as the optimal MEB coreset result [Bădoiu and Clarkson 2008], and gives an improvement over Theorem 8.3 in the general setting.

The algorithm, shown in Figure 5, is simply Algorithm 8.2 with an additional possible "away" step, where a coordinate is set to zero. Since the points $x_{(k)}$ are always the optimum points x_N of some simplex Δ_N, with vertices $\{e(j) \mid j \in N\}$, the algorithm is given in terms of maintenance of the set N. The sets N^* and N^{**} are defined only to aid discussion of N at different points in the algorithm.

For $|N| \le \lceil 1/\epsilon \rceil$, the algorithm proceeds as in Algorithm 8.2. When $|N| > \lceil 1/\epsilon \rceil$, the "away" step finds an index i'' to remove from N. Note that before finding i'', the optimum for the current N is found; this is helpful in the analysis.

THEOREM 9.2. *For $\epsilon > 0$ and concave $f(x)$, Algorithm 9.1 returns an $(\epsilon C_f / C_f^*)$-coreset. The returned set is an $\epsilon(1 + o(1))$-coreset, as $\epsilon \to 0$, assuming $C_f(\Delta_\gamma) = (1 + o(1))C_f^*$ as $\gamma \to 0$.*

PROOF. Since Algorithm 9.1 exits with a similar termination condition as Algorithm 8.2, similar conditions as in Algorithm 8.3 hold for the index set N that it returns. It remains to show that the algorithm terminates.

Theorem 2.1 shows that addition of i' to N results in $x = x_{N^*}$ with

Theorem 2.1

$$h(x) \le h(x_N) - g(x_N)^2, \tag{19}$$

for $|N| \le K$, so inductively $h(x_{N^*}) \le 1/|N^*|$ for $|N^*| \le K$.

maximum

When $|N^*| > K$, the "away" step is also done. Since $f(x_{N^{**}}) \ge f(y)$ for any $y \in \Delta_{N^{**}}$, this inequality holds in particular when $y = x - \alpha''(e(i'') - x)$, where $\alpha'' := x_{i''}/(1 - x_{i''})$.

away step

For $x = x_{N^*}$ as in the algorithm, since x has at least $K+1$ nonzero entries, $\sum_j x_j = 1$, and $x \geq 0$, it must hold that $x_{i''} \leq 1/(K+1)$, and so

$$\alpha'' = \frac{x_{i''}}{1 - x_{i''}} \leq \frac{1/(K+1)}{1 - 1/(K+1)} = 1/K. \tag{20}$$

Also, since $x = x_{N^*}$ is optimal for N^*, its duality gap is zero.

$$0 = w_{N^*}(x) - f(x) = z_{N^*}(x) - x^T \nabla f(x) = \max_{i \in N^*} \nabla f(x)_i - x^T \nabla f(x)$$

Since $x^T \nabla f(x)$ is a convex combination of the coordinates of $\nabla f(x)$, this can be true only if all coordinates $\nabla f(x)_i$ with $i \in N^*$ must be equal to $\max_{i \in N^*} \nabla f(x)_i = x^T \nabla f(x)$. In particular,

$$e(i'')^T \nabla f(x) = \nabla f(x)_{i''} = x^T \nabla f(x). \tag{21}$$

Thus

>= f(y) (by definition of maximum)

$$\begin{aligned}
f(x_{N^{**}}) &\geq f(x - \alpha''(e(i'') - x)) \\
&\geq f(x) - \alpha''(e(i'') - x)^T \nabla f(x) - (\alpha'')^2 C_f && \text{def. of } C_f \\
&= f(x) - (\alpha'')^2 C_f && \text{by (21)} \\
&\geq f(x) - C_f/K^2. && \text{by (20)}
\end{aligned}$$

Putting together this relation and (19), and referring to N at the beginning of the loop, we have

$$\begin{aligned}
h(x_{N^{**}}) &\leq h(x) + 1/4K^2 && \text{def. of } h, \text{ and preceding} \\
\text{(19)} &\leq h(x_N) - g(x_N)^2 + 1/4K^2 && \text{by (19)} \\
&< h(x_N) - \epsilon^2/4 + 1/4K^2, && \text{by test on } g(x_N) \\
&\leq h(x_N), && K > 1/\epsilon
\end{aligned}$$

and so an iteration reduces the value of h. Since the value of $h(x_N)$ (at the beginning of each iteration) is decreasing, any given set N is seen only once, and so eventually the loop terminates with $g(x_N) \leq \epsilon/2$, and N of size at most K specifies a coreset with the properties as claimed. □

As remarked in Section 1, as applied to MEB this result is best possible in the leading term [Bădoiu and Clarkson 2008].

10. *Tail Estimates for Random Data*

In the case of optimizing within a convex hull, so $f(x) = \hat{f}(Ax)$ for a concave function $\hat{f}$ and $d \times n$ matrix A, suppose the columns p_i, $i = 1 \ldots n$ of A are random variables, independently and identically distributed. Consider an ϵ-coreset N, so that

The columns of A are the points (and the rows are the features).

$$w(x_N) - f(x_N) = z(x_N) - x_N^T \nabla f(x_N) \leq 2\epsilon C_f^*.$$

g(xN)<= epsilon/2

Since $z(x_N) \geq \nabla f(x_N)_i = p_i^T \nabla \hat{f}(Ax_N)$ for all i, this condition implies that

z(x) := max_i [grad(f(x))]i

$$p_i^T \nabla \hat{f}(Ax_N) \leq x_N^T \nabla f(x_N) + 2\epsilon C_f^*,$$

Follows from above's equation.

that is, all the p_i lie in a particular halfspace with normal vector $\nabla \hat{f}(Ax_N)$. Let

$$H(x) := \{p \in \mathbb{R}^d \mid p^T \nabla \hat{f}(Ax) \leq x^T \nabla f(x) + 2\epsilon C_f^*\},$$

I think this is a MEB.

so that all the p_i lie in $H(x_N)$. For any fixed $x \in \mathbb{R}^n$, the fact that all the $p_i \in H(x)$ would suggest that $H(x)$ contains most of the probability mass of the distribution of the p_i: if that mass is $1 - m$, then the probability is $(1 - m)^n \leq \exp(-mn)$ that all the points p_i appear in $H(x)$. So, unless the mass m in the complement of $H(x)$ is small, it is unlikely that all p_i will be in $H(x)$. Similarly, for any given choice of N, and letting $s := |N|$, the probability is $(1 - m)^{n-s} \leq \exp(-m(n - s))$ that all $p_i \in H(x_N)$, for all $i \notin N$, and when $H(x_N) \geq 1 - m$. Since there are only $\binom{n}{s}$ possible N, we have given most of the proof of the following theorem.

We have n, d-dimensional, points.

THEOREM 10.1. *For $f(x) = \hat{f}(Ax)$, where the columns of A are independently and identically distributed, for a coreset N of size s, with probability $1 - \delta$ the probability mass in the complement of $H(x_N)$ is no more than* $(\log(1/\delta) + s \log(ne/s))/(n - s)$.

Skipped this part.

PROOF. We use the union bound, so that the probability of failure is at most $\binom{n}{s}(1 - m)^{n-s}$, together with the bounds $\binom{n}{s} \leq (ne/s)^s$ and as before

$$(1 - m)^{n-s} \leq \exp(-m(n - s)).$$

It is thus enough to pick a value of m large enough that

$$(ne/s)^s \exp(-m(n - s)) \leq \delta,$$

which on solving for m yields the bound of the theorem. □

A similar result, in a setting where regions are "defined" by a small number of objects, implies the existence of probabilistic algorithms for a variety of geometric problems [Clarkson 1988, 1987], and is called the *compression lemma* in the learning theory literature [Littlestone and Warmuth 1986; Floyd and Warmuth 1995].

11. *Specific Cases*

The following subsections discuss some specific applications; the discussions of convex approximation, Adaboost, and L_v approximation follow Zhang [2003], and serve simply to translate results of that article for Algorithm 3.1 into results for Algorithm 1.1 in the notation of this article. The discussion of kernel methods follows that of Tsang et al. [2005a] to some degree. The discussion of Kullback-Leibler divergence gives a bound like that of Zhang [2002].

11.1. CONVEX APPROXIMATION. For suitable matrix A and point p, consider the primal problem (1) where $f(x) := \hat{f}(c) := -(c - p)^2$ with $c = Ax$; that is, the problem is to find the convex combination of the columns of A of minimum distance to p. Via (12), and using $\nabla \hat{f}(c) = -2(c - p)$, the dual objective function is

$$z + \hat{f}(c) - c^T \nabla \hat{f}(c) = z - (c - p)^2 + 2c^T(c - p) = z + c^2 - p^2$$

(12):
min z + f_hat(c) - c^T*grad(f_hat(c))
subject to: ze >= A^T*grad(f_hat(c))

so the dual problem is

$$\begin{aligned} &\min_{z\in\mathbb{R},c\in\mathbb{R}^d} z + c^2 - p^2 \\ &\text{subject to } z \geq \max_i -2[A^T(c-p)]_i. \end{aligned} \tag{22}$$

A^T*grad(f_hat(c))

Although the preceding considers only the finite-dimensional case, and not more general functional approximation as discussed by Zhang [2003], the barriers to such a generalization are not enormous.

As shown by Zhang [2003], and also discussed in Section 4.2, the measure C_f here is the square of the diameter of the MEB of the columns of A, and the asymptotic version C_f^* is the square of the radius. It is clear that Algorithm 1.1 as well as Algorithm 3.1 can be used to obtain an approximate solution, and that Algorithm 9.1 can be used to show that coresets exist.

(*) Important:
For MEB:
Cf = diameter^2
Cf* = radius^2

In contrast to some other problems considered in this article, here the main problem of interest is the primal problem, and the nature and significance of coresets for the dual problem is perhaps mysterious: what do they mean here, and what are they good for?

Here is a geometric interpretation. The corresponding optimum point c_N has a small duality gap: the gap is $\hat{z}(c_N) + c_N^2 - p^2 - (-(p-c_N)^2) \leq 4\epsilon \operatorname{Diam}(A\Delta)^2$, or

w(c) f_hat(c) Cf

$$\begin{aligned} 4\epsilon \operatorname{Diam}(A\Delta)^2 &\geq (\max_i -2[A^T(c_N-p)]_i) + 2c_N^2 - 2c_N^T p \\ &= 2[c_N^T(c_N-p) + \max_i -p_i^T(c_N-p)] \\ &= 2[(c_N-p)^T(c_N-p) + \max_i -(p_i-p)^T(c_N-p))] \\ &= 2[(c_N-p)^2 - \min_i (p_i-p)^T(c_N-p))] \end{aligned}$$

The left part (with max) is z_hat(cN) and the underscored part is the remaining from above's equation.

or

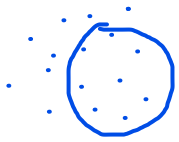
$$(p_i-p)^T(c_N-p)) \geq (c_N-p)^2 - 2\epsilon \operatorname{Diam}(A\Delta)^2,$$

for all i, or $(p_i-c_N)^T(c_N-p) \geq -2\epsilon \operatorname{Diam}(A\Delta)^2$. The boundary of the halfspace $H(c_N,0) := \{q \mid (q-c_N)^T(c_N-p) \geq 0\}$ is the hyperplane passing through c_N, and normal to $c_N - p$. The given halfspace is on the side away from p. The points p_i thus are all in a halfspace $H(c_N,\beta)$, bounded by a hyperplane parallel to $H(c_N,0)$, but translated by $\beta(c_N-p)$, where

$$\beta := -2\epsilon \operatorname{Diam}(A\Delta)^2/(c_N-p)^2 = 2\epsilon \operatorname{Diam}(A\Delta)^2/\hat{f}(c_N).$$

That is, c_N provides an immediate proof that the p_i cannot have a convex combination much closer to p than c_N is. A use of MEB coresets is the detection of outliers, the densest-ball problem discussed in Section 3.2; here, they could be used to detect inliers. That is, suppose we want to find the 10% of the points p_i such that deleting them makes the solution as much worse as possible, that is, decreases $f(x)$ the most. We could simply try all $\binom{n}{n/10}$ subsets, checking f after removing each, but this is very slow. An approximate solution, for some given $\epsilon > 0$, would be to try all subsets N' of size $1/\epsilon$, checking if at least 10% of the points p_i are not in $H(c_{N'})$, choosing the $c_{N'}$ that has minimum distance to p among all such $c_{N'}$.

(*) Important:
MEB for outliers!



We choose a subset of points and construct a MEB for it. Then we check whether 10% of the points are outside of the constructed MEB

This is for SVM, so we can skip this section. Also, the next sections, up until Appendix can be skipped.

11.2. MARGIN ESTIMATION. The margin estimation problem is the following: given a dataset comprising $p_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$, for $i = 1 \ldots n$, find

$$\begin{aligned} &\min_{c \in \mathbb{R}^d, \rho \in \mathbb{R}} \quad \rho + c^T c/2 \\ &\text{subject to } \rho \geq -y_i c^T p_i, i = 1 \ldots n, \end{aligned} \tag{23}$$

if this problem is feasible. If $\rho < 0$ and vector c are feasible, then they specify a hyperplane $\{x \mid c_*^T x = 0\}$, that separates the p_i with $y_i = +1$ from those with $y_i = -1$. The problem is to make the *margin* as large as possible, where the margin is the minimum distance $G(c) := -\rho/\|c\|$ of any point p_i to that hyperplane.

This problem is a scaled version of Wolfe dual of the convex approximation of Section 11.1, with $p = 0$, and indeed, the optimum solutions are the same. The difference from convex approximation is that good approximation bounds for margin estimation seem to require a small duality gap $g(x)$.

In the setting of support vector machines, a version of this problem (with the points given implicitly, in the form of an M matrix), is called *the training problem for hard margin SVM*.

While many formulations of this problem fix ρ and minimize $c^2 := c^T c$, or fix $c^2 = 1$ and maximize ρ, the same margin is obtained in the formulation here: for any given feasible pair c, ρ, and value $\beta > 0$, the pair $\beta c, \beta\rho$ is also feasible, and the value of β that minimizes $\beta\rho + \beta^2 c^2/2$ is $\tilde{\beta} = -\rho/c^2$, yielding the β-scaling-invariant $\tilde{\beta}\rho + \tilde{\beta}^2 c^2/2 = -\rho^2/2c^2$, proportional to the negative square of the margin. That is, the optimal value of the training problem is one half the negative square of the margin.

The dual to (23) is

$$\max_{x \in \Delta} -x^T A^T A x/2 \tag{24}$$

where A is the $d \times n$ matrix whose columns are the vectors $y_i p_i$. (This can be read off from (22), for example.)

(Indeed, at some cost in n, it is well-known that a broader class of SVM training problems are dual to a problem of the same form: use a $d \times n'$ matrix A', whose columns are all those possible of the form $p_i - p_j$, where $y_i = 1$ and $p_j = -1$. The polytope $A'\Delta$ is the Minkowski difference of the convex hulls of the "+" points and the "−" points, and the length of the shortest vector in $A'\Delta$ is the minimum distance between those two polytopes. Although n' may be as large as $n^2/4$, this does not affect results regarding coreset sizes; also, other formulations do not involve such a blow-up in A. Recently Gärtner and Jaggi [2009] have shown that Frank-Wolfe may be applied here with a running time that remains linear in n; they also analyze Gilbert's classical algorithm for SVM training, which is an instance of Frank-Wolfe.)

The duality gap $w(x) - f(x)$ is here $\rho + c^2$, so that at the optimum $c_* = Ax_*$ and ρ_*, we have $-\rho_* = c_*^2$, so that the margin $G(x_*) = -\rho_*/\|c_*\| = \|c_*\|$. By Theorem 2.3, Algorithms 1.1 and 3.1 find $x_{(\hat{k})}$ and corresponding $\rho_{(\hat{k})}$ and $c_{(\hat{k})}$ such that

$$\rho_{(\hat{k})} + c_{(\hat{k})}^2 = w(x_{(\hat{k})}) - f(x_{(\hat{k})}) \leq 4\epsilon C_f,$$

or $-\rho_{(\hat{k})} \geq c^2_{(\hat{k})} - 4\epsilon C_f$. We have, using also $c^2_{(\hat{k})} \geq c^2_*$,

$$\begin{aligned}\frac{G\left(x_{(\hat{k})}\right)^2}{G(x_*)^2} &= \frac{\rho^2_{(\hat{k})}/\|c_{(\hat{k})}\|^2}{c^2_*} \\ &\geq \frac{\left(c^2_{(\hat{k})} - 4\epsilon C_f\right)^2}{c^2_{(\hat{k})} c^2_*} \\ &\geq (1 - 4\epsilon C_f/c^2_*)^2,\end{aligned}$$

or, using $C_f \leq \operatorname{Diam}(A\Delta)^2/2$,

$$G\left(x_{(\hat{k})}\right) \geq G(x_*)(1 - 2\epsilon \operatorname{Diam}(A\Delta)^2/G(x_*)^2).$$

A similar argument using Theorem 9.2 shows that there is a set of points indexed by N, of size $(1 + o(1))/\epsilon$, so that

$$G(x_N) \geq G(x_*)(1 - 2\epsilon \operatorname{Rad}(A\Delta)^2/G(x_*)^2)$$

as $\epsilon \to 0$. The previous bound on the size for comparable quality was $64/\epsilon$ [Har-Peled et al. 2007].

Putting this condition together with Theorem 10.1, we can say the following: if there is coreset N with $G(x_N) > 0$ and if the points p_i are independently and identically distributed, then with probability $1 - \delta$, the probability mass of the region $H(c_N) := \{p \in \mathbb{R}^d \mid \rho < -y_i c_N^T p\}$ is at most $\log(1/\delta) + \frac{s}{n-s}\log(\frac{ne}{s})$, where $s := |N|$ need be no more than $(1 + o(1))\operatorname{Diam}(A\Delta)^2/G(x_*)^2$. Since for a classifier based on x_N, only the points in $H(c_N)$ can be misclassified, this gives an upper bound on the error probability of such a classifier. This result is similar to that derived by Graepel et al. [2000] for perceptrons. This is not surprising: the perceptron algorithm is a greedy procedure akin to Algorithm 1.1, and Novikoff's mistake bound (see Graepel et al. [2000] for example) implies the existence of sparse solutions to an optimization problem.

Tsang et al. [2005a] have shown that variations like hard-margin SVDD, one- and two-class L2-SVM, and L2-SVR can also be put into the framework here. As discussed in the Introduction, the approximation algorithms given here do not require, as for coresets, the exact solution of small problem instances. It remains to be seen, however, whether such simplification is helpful in practice.

11.3. ADABOOST. *Boosting* refers to the improvement of a collection of classifiers by using a linear combination of them. A collection of n classifiers is given, and d datapoints, such that classifier i gives a value $a_{ji} \in [-1, 1]$ for datapoint j, which has actual classification $r_j \in \{-1, 1\}$. The training problem is: for a given *loss function* $L(c, r)$, that gives the cost of making a prediction c for a classification r, find x such that $L(Ax, r)$ is as small as possible. It is no loss of generality to assume that the data is symmetric about the origin, that is, for every prediction a_{ji} for r_j, there is a j^- with $a_{j^-i} = -a_{ji}$ and $r_{j^-} = -r_j$. This implies that it is enough to consider $x \geq 0$, putting the problem nearly into the framework of this article.

The "ideal" loss function is perhaps $L(c, r) = \frac{1}{d}\sum_j I_{c_j r_j < 0}$, where the indicator function $I_s = 1$ when s is true, and $I_s = 0$ otherwise. That is, there is no loss if c_j and r_j agree in sign, and a loss $1/d$ otherwise. This function is difficult to

work with, however, so proxies are often used. A popular one, for Adaboost, is $\frac{1}{d}\sum_j \exp(-\eta c_j r_j)$, where η is a tunable parameter.

The training problem is then to minimize the loss, or equivalently to maximize $f(x) := -\log \sum_j \exp(-\eta(Ax)_j r_j)/d$, over $x \in \Delta$.

(Training an Adaboost classifier is an instance of *geometric programming* [Boyd and Vandenberghe 2004], often done with interior-point methods. However, it is also amenable to the simpler approach here.)

The restriction to nonnegative x is no loss of generality, as mentioned, and the restriction $x^T\mathbf{e} = 1$ simply amounts to an adjustment (or normalization) of the parameter η. With $c = Ax$, as usual, $f(x)$ can also be written as $f(x) = \hat{f}(c) = -\log \sum_j \exp(-\eta c_j r_j)/d$.

The gradient of this $\hat{f}$ is, using $V_j := \exp(-\eta c_j r_j)/d$,

$$\nabla \hat{f}(x)_j = \frac{V_j \eta r_j}{\sum_j V_j},$$

and $\nabla f(x) = A^T \nabla \hat{f}(Ax)$, as usual. The i' maximizing $\nabla f(x)_i$ is thus readily found.

To apply the results here, the value of C_f for this function f needs to be bounded. We will apply (13), which requires a bound on $-(b-a)^T \nabla^2 \hat{f}(\tilde{a})(b-a)/2$, for $a, b \in A\Delta$ and $\tilde{a}$ on a line through a and b, and in $A\Delta$.

The following lemma will be helpful.

LEMMA 11.1. *Suppose function $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ has the form $-\gamma(\tilde{f}(x))$, where $\tilde{f} : \mathbb{R}^d \to \mathbb{R}$, like $\hat{f}$, and $\gamma : \mathbb{R} \to \mathbb{R}$ with $\gamma''(x) \le 0$. Then for $c, \tilde{a} \in \mathbb{R}^d$,*

$$-c^T \nabla^2 \hat{f}(\tilde{a}) c \le \gamma'(\tilde{f}(\tilde{a})) c^T \nabla^2 \tilde{f}(\tilde{a}) c.$$

PROOF. We have

$$\nabla \hat{f}(\tilde{a}) = -\gamma'(\tilde{f}(\tilde{a})) \nabla \tilde{f}(\tilde{a}),$$

and

$$\nabla^2 \hat{f}(\tilde{a}) = -\gamma''(\tilde{f}(\tilde{a})) \nabla \tilde{f}(\tilde{a}) \nabla \hat{f}(\tilde{a})^T - \gamma'(\tilde{f}(\tilde{a})) \nabla^2 \tilde{f}(\tilde{a}).$$

Thus here

$$\begin{aligned} -c^T \nabla^2 \hat{f}(\tilde{a}) c &= \gamma''(\tilde{f}(\tilde{a}))(c^T \nabla \tilde{f}(\tilde{a}))^2 + c^T \gamma'(\tilde{f}(\tilde{a})) \nabla^2 \tilde{f}(\tilde{a}) c \\ &\le \gamma'(\tilde{f}(\tilde{a})) c^T \nabla^2 \tilde{f}(\tilde{a}) c, \end{aligned}$$

as claimed, since the first term is always no more than zero. □

Applying this lemma to $\gamma(x) = \log x$, and $\tilde{f}(\tilde{a}) = \sum_j \exp(-\eta \tilde{a}_j r_j) = \sum_j V_j$, we have

$$\begin{aligned} -(b-a)^T \nabla^2 \hat{f}(\tilde{a})(b-a) &\le \frac{(b-a)^T \nabla^2 \tilde{f}(\tilde{a})(b-a)}{\sum_j V_j} \\ &= \frac{(b-a)^T (\eta^2 \operatorname{diag}(V))(b-a)}{\sum_j V_j} && \text{using } r_j^2 = 1 \\ &\le 4\eta^2 && \text{using } |a_{ji}| \le 1. \end{aligned}$$

Here $\operatorname{diag}(V)$ is the diagonal matrix with $\operatorname{diag}(V)_{jj} = V_j$.

11.4. CONVEX APPROXIMATION IN L_v. When approximating a point p by a convex combination Ax, for $x \in \Delta$, it may be of interest to use a different measure of distance than the Euclidean norm; a more general setting is to use L_v norms, $\|p\|_v := (\sum_j |p_j|^v)^{1/v}$. (Usually L_p norms are discussed, but this collides with our notation a bit, so here v is used for the numerical parameter instead.) The distance $\|Ax - p\|_v$ has the same minimum as the maximum of the function $f(x) := -\|Ax - p\|_v^{v'}$, where $v' := \min\{2, v\}$, and the latter functions are considered here. As usual, consider $f(x) = \hat{f}(c) = -\|c - p\|_v^{v'}$, where $c = Ax$.

For $v \geq 2$, apply Lemma 11.1 with $\gamma(x) = x^{2/v}$, and $\tilde{f}(\tilde{a}) = \|\tilde{a}\|_v^v$, obtaining

$$\begin{aligned}
-(b-a)^T \nabla^2 \hat{f}(\tilde{a})(b-a) &\leq \gamma'(\tilde{f}(\tilde{a}))(b-a)^T \nabla^2 \tilde{f}(\tilde{a})(b-a) \\
&= (2/v)(\|\tilde{a}\|_v^v)^{2/v-1} \sum_j (b-a)_j^2 v(v-1)|\tilde{a}_j - p_j|^{v-2} \\
&= 2(v-1) \sum_j (b-a)_j^2 \frac{|\tilde{a}_j - p_j|^{v-2}}{\|\tilde{a}\|_v^{v-2}} \\
&= 2(v-1) \sum_j (b-a)_j^2 \left(\frac{|\tilde{a}_j - p_j|^v}{\sum_j |c_j - p_j|^v} \right)^{1-2/v} \\
&\leq 2(v-1)\,\mathrm{Diam}(A\Delta)^2.
\end{aligned}$$

Thus Algorithm 1.1 can be used for L_v approximation, with similar bounds, for $\infty > v \geq 2$.

For $1 < v < 2$, unfortunately C_f cannot be found; however, an analog can be obtained, via an analog of (5), with $1/\alpha^v$ instead of $1/\alpha^2$. This allows an analog of (10) in which $h(x_{(k)})^2$ is replaced by $h(x_{(k)})^v$, which leads to an additive error of $O(1/\epsilon^{v-1})$ instead of $O(1/\epsilon^v)$. As shown by Donahue et al. [1997], an incremental approach like Algorithm 1.1 or Algorithm 3.1 cannot work for the $v = 1$ case.

11.5. MIXTURE DENSITY ESTIMATION. Another kind of convex approximation problem is mixture density estimation, where in the discrete version, a point p is a probability vector, that is $p \in \Delta$, as are the columns of a matrix A, comprising a "dictionary" of probability vectors, and we want to approximate p by the convex combination Ax. A common norm for such a purpose is the Kullback-Leibler divergence $D(p||Ax) := \sum_j p_j \log(p_j/(Ax)_j)$, and minimizing that divergence is equivalent to maximizing $f(x) := \sum_j p_j \log(Ax)_j$, or for $c := Ax$, $\hat{f}(c) := \sum_j p_j \log c_j$.

Here $\nabla^2 \hat{f}(c)_{jj'} = 0$, if $j \neq j'$, and $\nabla^2 \hat{f}(c)_{jj} = -p_j/c_j^2$. Thus $-(b-a)^T \nabla^2 \hat{f}(\tilde{a})(b-a)/2$ is here $\sum_j p_j (b_j - a_j)^2/2\tilde{a}_j^2$, and so C_f is here bounded by, using (13),

$$\sup_{a,b,\tilde{a} \in A\Delta, \tilde{a}} \sum_j p_j (b_j - a_j)^2/2\tilde{a}_j^2 \leq \sup_{a,b \in A\Delta} \sum_j p_j b_j^2/a_j^2,$$

where again $\tilde{a}$ is collinear with a and b.

12. *Conclusions*

A related line of research is concerned with finding *best m-term approximations*, finding sparse x minimizing $(p - Ax)^2$, where $x \in \mathbb{R}^n$ need not be in Δ. The orthogonal matching pursuit algorithm, which is very close to Algorithm 1.1 (in particular, the "harder working" version described in Section 7), has been shown to yield x that are good *relative* approximations, compared to the best x with the same sparsity [Tropp 2004], for suitable A. A sufficient condition for A to be suitable is that $A^T A = I + E$, where I is the identity matrix and E has entries that are all small in magnitude.

Skip to here.

Appendix: The Wolfe Dual via the Lagrangian

A familiar form of the Lagrangian relaxation [Boyd and Vandenberghe 2004] for a convex function $-f(x)$ and for the equivalent optimization problem $-\min_{x\in\Delta} -f(x)$, is

$$- \max_{\substack{z\in\mathbb{R} \\ \lambda\in\mathbb{R}^n, \lambda\geq 0}} \min_{x\in\mathbb{R}^n} -f(x) - x^T\lambda + z(x^T\mathbf{e} - 1),$$

1) Derivative of this = 0.
2) Lambda >= 0.
3) Go back to the original equation (the one on the left).

for which the minimum with respect to x is achieved when all derivatives with respect to x are zero, that is, $-\nabla f(x) - \lambda + z\mathbf{e} = 0$, or $\lambda = z\mathbf{e} - \nabla f(x)$. The condition $\lambda \geq 0$ implies $z\mathbf{e} \geq \nabla f(x)$, and plugging this expression for λ into the Lagrangian, the problem becomes

$$\begin{aligned} &- \max_{z\in\mathbb{R}} \min_{x\in\mathbb{R}^n} -f(x) - x^T(z\mathbf{e} - \nabla f(x)) + z(x^T\mathbf{e} - 1) \\ &\quad = - \max_{z\in\mathbb{R}, x\in\mathbb{R}^n} -f(x) + x^T\nabla f(x) - z \\ &\quad = \min_{z\in\mathbb{R}, x\in\mathbb{R}^n} z + f(x) - x^T\nabla f(x), \end{aligned}$$

subject to $z\mathbf{e} \geq \nabla f(x)$, as claimed.

The Slater condition, which is that a feasible point exists that satisfies all inequality constraints strictly, is easily seen to be satisfied, and so strong duality holds, by Slater's theorem [Boyd and Vandenberghe 2004]. Thus an optimum point x_*, for which $f(x_*)$ is maximum, is also a dual optimal point. For any feasible point $x \in \Delta$ we have

because h(x*) = 0

$$w(x) \geq w(x_*) = f(x_*) \geq f(x),$$

and also complementary slackness.

ACKNOWLEDGMENTS. I am grateful to Kasturi Varadarajan for pointing out an error in the probabilistic proof of Section 6, and other helpful remarks. I am also grateful to the referees for their many helpful comments, both general and specific.

REFERENCES

AGARWAL, P. K., HAR-PELED, S., AND VARADARAJAN, K. R. 2005. Geometric approximation via coresets. In *Combinatorial and Computational Geometry*, MSRI Publications, Cambridge University Press, 1–30. 2005.

AHIPASAOGLU, D., SUN, P., AND TODD, M. J. 2006. Linear convergence of a modified Frank-Wolfe algorithm for computing minimum volume ellipsoids. Tech. rep. 1452, Cornell University.

ALTUN, Y., AND SMOLA, A. 2006. Unifying divergence minimization and statistical inference via convex duality. In *The Proceedings of the 19th Annual Conference on Learning Theory (COLT'06)*. Springer, Berlin, 139–153.

BAR-JOSEPH, Z., GERBER, G. K., LEE, T. I., RINALDI, N. J., YOO, J. Y., ROBERT, F., GORDON, D. B., FRAENKEL, E., JAAKKOLA, T. S., YOUNG, R. A., AND GIFFORD, D. K. 2003. Computational discovery of gene modules and regulatory networks. *Nature Biotechnol. 21*, 1337–1342.

BARRON, A. R. 1993. Univeral approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. Theory 39,* 3, 930–945.

BEN-DAVID, S., EIRON, N., AND SIMON, H.-U. 2002. The computational complexity of densest region detection. *J. Comput. Syst. Sci. 64*, 22–47.

BOYD, S., AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge University Press, UK.

BĂDOIU, M., AND CLARKSON, K. L. 2003. Smaller core-sets for balls. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*. SIAM, 801–802.

BĂDOIU, M., AND CLARKSON, K. L. 2008. Optimal core-sets for balls. *Comput. Geom. Theory Appl. 40,* 1, 14–22.

BĂDOIU, M., HAR-PELED, S., AND INDYK, P. 2002. Approximate clustering via core-sets. In *Proceedings of the 34th Annual ACM Symposium on Theory of computing (STOC'02)*. ACM, New York, 250–257.

CHU, C. S., TSANG, I. W., AND KWOK, J. T. 2004. Scaling up support vector data description by using core-sets. In *Proceedings of the International Joint Conference on Neural Networks*. 425–430.

CLARKSON, K. L. 1936. Uniformaly convex spaces. *Trans. Amer. Math. Soc. 40*, 3, 396–414.

CLARKSON, K. L. 1987. New applications of random sampling to computational geometry. *Discr. Comput. Geom. 2*, 195–222.

CLARKSON, K. L. 1988. A randomized algorithm for closest-point queries. *SIAM J. Comput. 17*, 830–847.

DONAHUE, M. J., GURVITS, L., DARKEN, C., AND SONTAG, E. 1997. Rates of convex approximation in non-Hilbert spaces. *Construct. Approx. 13*, 187–220.

FEDOROV, V. V. 1972. *Theory of Optimal Experiments*. Academic Press, New York.

FLOYD, S., AND WARMUTH, M. 1995. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Mach. Learn. 21,* 3, 269–304.

FRANK, M., AND WOLFE, P. 1956. An algorithm for quadratic programming. *Naval Res. Logist. Quart. 3*, 95110.

GÄRTNER, B., AND JAGGI, M. 2009. Coresets for polytope distance. In *Proceedings of the Symposium on Computational Geometry*. ACM, New York, 33–42.

GRAEPEL, T., HERBRICH, R., AND WILLIAMSON, R. C. 2000. From margin to sparsity. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, 210–216.

HAR-PELED, S. 2007. Personal communication.

HAR-PELED, S., ROTH, D., AND ZIMAK, D. 2007. Maximum margin coresets for active and noise tolerant learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'07)*.

HAZAN, E. 2006. Approximate convex optimization by online game playing. http://arxiv.org/abs/cs.DS/0610119.

INABA, M., KATOH, N., AND IMAI, H. 1994. Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). In *Proceedings of the 10th Annual Symposium on Computational Geometry (SCG'94)*. ACM Press, New York, 332–339.

JONES, L. K. 1992. A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Ann. Statist. 20,* 1, 608–613.

KHACHIYAN, L. G. 1996. Rounding of polytopes in the real number model of computation. *Math. Oper. Res. 21,* 2, 307–320.

KUMAR, P., MITCHELL, J., AND YILDIRIM, E. A. 2003. Approximate minimum enclosing balls in high dimensions using core sets. *ACM J. Exper. Algor. 8*.

KUMAR, P., AND YILDIRIM, E. A. 2005. Minimum-volume enclosing ellipsoids and core sets. *J. Optimiz. Theory Appl. 126*, 1–21.

LI, J. Q., AND BARRON, A. R. 2000. Mixture density estimation. In *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, Cambridge, MA, 279–285.

LITTLESTONE, N., AND WARMUTH, M. 1986. Relating data compression and learnability. Tech. rep., Department of Computer and Information Sciences, Santa Cruz, CA. June.

MASON, L., BAXTER, J., BARTLETT, P., AND FREAN., M. 2000. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing System*. Vol. 12. MIT Press, Cambridge, MA.

NEMIROVSKI, A. 2005. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Optimiz 15*, 1, 229–251.

NIELSEN, F., AND NOCK, R. 2006. On approximating the smallest enclosing Bregman balls. In *Proceedings of the 22nd Annual Symposium on Computational Geometry (SCG'06)*. ACM Press, New York, 485–486.

PANIGRAHY, R. 2004. Minimum enclosing polytope in high dimensions. http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0407020.

PISIER, G. 1981. Remarques sur un resultat non publié de B. Maurey. *Séminaire d'Analyze Fonctionelle Palaiseau 1*, 12, 1980–1981.

PLATT, J. C. 1999. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*. MIT Press, Cambridge, MA, 185–208.

TASKAR, B., LACOSTE-JULIEN, S., AND JORDAN, M. I. 2006. Structured prediction, dual extragradient and Bregman projections. *J. Mach. Learn. Res. 7*, 1627–1653.

TODD, M. J., AND YILDIRIM, E. A. 2005. On Khachiyan's algorithm for the computation of minimum volume enclosing ellipsoids. Tech. rep., Cornell University.

TROPP, J. A. 2004. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inf. Theory 50,* 10, 2231–2242.

TSANG, I. W., KWOK, J. T., AND CHEUNG, P.-M. 2005a. Core vector machines: Fast SVM training on very large data sets. *J. Mach. Learn. Res. 6*, 363–392.

TSANG, I. W., KWOK, J. T., AND LAI, K. T. 2005b. Core vector regression for very large regression problems. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. ACM Press, New York, 912–919.

TSANG, I. W., KWOK, J. T., AND ZURADA, J. M. 2006. Generalized core vector machines. *IEEE Trans. Neural Netw. 17*, 5, 1126–1140.

WYNN, H. P. 1970. The sequential generation of D-optimum experimental design. *Ann. Math. Statist. 41*, 1655–1664.

ZHANG, T. 2002. A general greedy approximation algorithm with applications. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA.

ZHANG, T. 2003. Sequential greedy approximation for certain convex optimization problems. *IEEE Trans. Inf. Theory 49*, 682–691.

RECEIVED OCTOBER 2009; ACCEPTED OCTOBER 2009