Sublinear Algorithms for Optimization and Machine Learning

Alexander Irpan
alexirpan@berkeley.edu

Ronald Kwan
rkwan@berkeley.edu

May 11, 2015

Contents

1 Introduction

Recently, there has been a surge of interest in machine learning algorithms. These have broad applications in industry, from spam detection to image recognition and natural language processing. This interest is fueled by the trend of big data. Statistical theory indicates that the increase in dataset sizes and processing power leads to greater and greater accuracy and capabilities.

However, improving results in this manner comes at a cost: as datasets grow in size, the time it takes for learning algorithms to run increases as well. With datasets that range in the size of terabytes, algorithms that run in linear or log-linear time can still take days of computation time. Luckily, the study of sublinear algorithms has also become a burgeoning field with the advent of the ability to collect and store these large data sets.

In the past few years, there have been several works discussing the use of sublinear algorithms to solve problems relevant to machine learning. In particular, the work of Elad Hazan and his collaborators have produced a sublinear approach that can be applied to several optimization problems that arise in machine learning.

In this survey, we explore the work of Hazan et al., starting with the original paper [1] in which they introduce the sublinear approach used throughout the three papers, first through the linear perceptron (Section 2), then by presenting a general primal-dual algorithm (Section 3). We then move on to the later works, where this approach is applied to the problems of solving SDPs [5] (Section 4) and training SVMs [4] (Section 5).

2 Preliminaries and the Perceptron

In Hazan et al's initial paper [1], the authors begin by describing a sublinear approach to classification by solving the perceptron learning problem. They then apply the intuition and methods to different and more general settings. Following their lead, we too begin by describing classification and their perceptron algorithm.

2.1 The Linear Classification Problem

In the linear classification problem, the learner is given a set of n labeled examples, where each is a d-dimensional vector. These form an $n \times d$ matrix A, where the ith row A_i is the ith example. The labels comprise a vector $y \in \{+1, -1\}^n$.

The goal is to find a separating hyperplane (represented by a normal vector x) such that for all i, $y(i) \cdot A_i x \geq 0$. Assume x is in the unit Euclidean ball $\mathbb{B}$.

We will assume throughout that $A_i \in \mathbb{B}$ for all $i \in [n]$, where $[m]$ denotes the set of integers $\{1, 2, \ldots, m\}$. Any linear classification can be reduced to this by appropriate scaling.

We also assume that the labels $y(i)$ are all 1, by taking $A_i \leftarrow -A_i$ for any i with $y(i) = -1$. The max-min formulation of the perceptron is finding the optimal x for

$$\max_{x \in \mathbb{B}} \min_i A_i x$$

Let the optimum be σ. A separating hyperplane exists iff $\sigma \geq 0$ while $x \neq 0$. This optimum σ is called the *margin*. Assuming there exists a separating hyperplane, any optimal solution has $\|x\| = 1$, since scaling x up increases the value if it is non-negative.

For an ϵ-approximate solution the goal is to find a vector $x_\varepsilon \in \mathbb{B}$ such that

$$\forall i' \;\; A_{i'} x_\varepsilon \geq \max_{x \in \mathbb{B}} \min_i A_i x - \varepsilon = \sigma - \varepsilon \tag{1}$$

Relax $\min_i A_i x$ to $\min_{p \in \Delta} p^\top A x$, where $\Delta \subset \mathbb{R}^n$ is the unit simplex $\{p \in \mathbb{R}^n \mid p_i \geq 0, \sum_i p_i = 1\}$. The optimal strategy is to put all weight on the minimal A_i, so the optimum is still σ. Thus we can regard the optimum as the outcome of a game to determine $p^\top A x$, between a minimizer choosing $p \in \Delta$, and a maximizer choosing $x \in \mathbb{B}$, yielding

$$\sigma \equiv \max_{x \in \mathbb{B}} \min_{p \in \Delta} p^\top A x,$$

From standard duality results, σ is also the optimum of the dual problem

$$\min_{p \in \Delta} \max_{x \in \mathbb{B}} p^\top A x,$$

and the optimum vectors p^* and x^* are the same for both problems.

This formulation changes the discrete optimization over $i \in [n]$ to a continuous optimization over $p \in \Delta$, which makes it much easier to find an approximate solution.

2.2 Algorithm: Sublinear Perceptron

The classical Perceptron Algorithm returns an ε-approximate solution to the linear classification problem in $\frac{1}{\varepsilon^2}$ iterations, and total time $O(\varepsilon^{-2}M)$. In contrast, the paper's version of the perceptron algorithm takes $O(\varepsilon^{-2}(n+d)(\log n))$ time to return an ε-approximate solution with probability at least $\frac{1}{2}$. This is optimal in the unit-cost RAM model up to polylog factors, but we will not prove this. See section 6 of [1] for the lower bound proofs.

Algorithm 1 Sublinear Perceptron

1: Input: $\varepsilon > 0$, $A \in \mathbb{R}^{n \times d}$ with $A_i \in \mathbb{B}$ for $i \in [n]$.
2: Let $T \leftarrow 200^2\varepsilon^{-2}\log n$, $y_1 \leftarrow 0$, $w_1 \leftarrow \mathbf{1}_n$, $\eta \leftarrow \sqrt{\frac{\log n}{T}}$.
3: **for** $t = 1$ to T **do**
4: $\quad p_t \leftarrow \frac{w_t}{\|w_t\|_1}$, $x_t \leftarrow \frac{y_t}{\max\{1, \|y_t\|\}}$.
5: $\quad$ Choose $i_t \in [n]$ by $i_t \leftarrow i$ with prob. $p_t(i)$.
6: $\quad y_{t+1} \leftarrow y_t + \frac{1}{\sqrt{2T}} A_{i_t}$
7: $\quad$ Choose $j_t \in [d]$ by $j_t \leftarrow j$ with probability $x_t(j)^2/\|x_t\|^2$.
8: $\quad$ **for** $i \in [n]$ **do**
9: $\quad\quad \tilde{v}_t(i) \leftarrow A_i(j_t)\|x_t\|^2/x_t(j_t)$
10: $\quad\quad v_t(i) \leftarrow \text{clip}(\tilde{v}_t(i), 1/\eta)$
11: $\quad\quad w_{t+1}(i) \leftarrow w_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$
12: $\quad$ **end for**
13: **end for**
14: **return** $\bar{x} = \frac{1}{T}\sum_t x_t$

The sublinear perceptron algorithm is presented in Figure 1. We explain the approach of the algorithm in more depth before proceeding.

To solve $\max_{x \in \mathbb{B}} \min_{p \in \Delta} p^\top Ax$, the algorithm does gradient descent. Each iteration, we update by computing the gradient with respect to x, then with respect to p. These are $p^\top A$ and Ax respectively. The first is used to update the current best hyperplane y. The second is used to update a weight vector w with a multiplicative weight update. In the psuedocode, lines 5-6 update y with the sampled gradient w.r.t. x, using randomized online gradient descent (OGD), and lines 7-12 update w with the sampled gradient w.r.t. p, using a version of multiplicative weights (MW). Given enough iterations, the normalized versions of both $p^\top A$ and Ax converge to solutions x, p, and the bulk of the following is proving bounds on the convergence rate.

2.3 Technical Lemmas

Before the main result, which gives the performance of this algorithm, we present several supporting technical results.

To get sublinear performance, instead of evaluating the aforementioned dot products in full, we can define random variables whose expected values are the dot products. Note that by the formulation $\max_{x \in ball} \min_{p \in \Delta} p^\top Ax$, all dot products we need to estimate are in one of two forms.

1. $p^\top v$, where p satisfies $\sum_i p(i) = 1$.
2. $u^\top x$, where x satisfies $\sum_i x(i)^2 = 1$.

For the first form, we can define a random variable V where $\Pr(V = v(i)) = p(i)$. A simple calculation gives $\mathbf{E}[V] = p^\top v$. For the second form, define U as $\Pr(U = u(i)/x(i)) = x(i)^2$. These are called the ℓ_1-sample and ℓ_2-sample.

The ℓ_2-sample can be problematic because $u(i)/x(i)$ could be very large if $x(i)$ is small. To deal with especially large values, the samples are clipped. Given sampled value s, $\text{clip}(s, a)$ gives $-a$ if

$s < -a$, s if $-a \le s \le a$, and a if $s > a$. This clipping biases the estimator, but it can be shown the effect is negligible.

Lemma 2.1. *Let X be a random variable such that $|\mathbf{E}[X]| \le C/2$. Let $\bar{X} = \text{clip}(X, C)$. Then*

$$|\mathbf{E}[X] - \mathbf{E}[\bar{X}]| \le \frac{2Var(X)}{C}$$

Proof. For $x > C$, $x - \mathbf{E}[X] \ge C/2$, so

$$C(x - C) \le 2(x - \mathbf{E}[X])(x - C) \le 2(x - \mathbf{E}[X])^2$$

Then find the difference by direct evaluation:

$$\begin{aligned}
\mathbf{E}[X] - \mathbf{E}[\bar{X}] &= \int_{x<-C} (x + C)\, d\mu_X + \int_{x>C} (x - C)\, d\mu_X \\
&\le \int_{x>C} (x - C)\, d\mu_X \\
&\le \frac{2}{C} \int_{x<-C} (x - \mathbf{E}[X])^2\, d\mu_X \\
&\le \frac{2}{C} \operatorname{Var}[X]
\end{aligned}$$

Similarly, one can show $\mathbf{E}[\bar{X}] - \mathbf{E}[X] \ge -\frac{2}{C} \operatorname{Var}[X]$, completing the proof. □

Later, we show $\operatorname{Var}[X]$ is small enough to make the bias negligible.

The multiplicative weight update used is different from the one given in class. In the following, let q_t represent the vector of estimates, where $q_t(i)$ is an estimate of $A_i x$.

Definition 1 (Variance MW Algorithm)**.** *Let $q_1, q_2, \cdots q_T$ be a sequence of vectors in $\mathbb{R}^n$. Initialize weights w to the all-one vector, and set a learning parameter $\eta = \sqrt{\frac{\log n}{T}}$. For each $t \in [T]$, set*

$$p_t = w_t / \|w_t\|_1, \quad w_{t+1}(i) = w_t(i)(1 - \eta q_t(i) + \eta^2 q_t(i)^2)$$

Lemma 2.2 (Variance MW Lemma)**.** *The MW algorithm satisfies*

$$\sum_{t \in [T]} p_t^\top q_t \le \min_{i \in [n]} \sum_{t \in [T]} \max\{q_t(i), -1/\eta\} + \frac{\log n}{\eta} + \eta \sum_{t \in [T]} p_t^\top q_t^2.$$

where q_t^2 is the vector such that $q_t^2(i) = (q_t(i))^2$.

Proof of Lemma 2.2, Weak Regret. The proof is similar to the proof from class. We first show an upper bound on $\log \|w_{T+1}\|_1$, then a lower bound, and then relate the two.

From the weight update and $p_t = w_t / \|w_t\|_1$,

$$\begin{aligned}
\|w_{t+1}\|_1 &= \sum_{i \in [n]} w_{t+1}(i) \\
&= \sum_{i \in [n]} p_t(i) \|w_t\|_1 (1 - \eta q_t(i) + \eta^2 q_t(i)^2) \\
&= \|w_t\|_1 (1 - \eta p_t^\top q_t + \eta^2 p_t^\top q_t^2).
\end{aligned}$$

Since $\|w_1\|_1 = n$, we can use $1 + z \le \exp(z) \forall z \in \mathbb{R}$ and induction on t to get

$$\log \|w_{T+1}\|_1 = \log n + \sum_{t \in [T]} \log(1 - \eta p_t^\top q_t + \eta^2 p_t^\top q_t^2) \le \log n - \sum_{t \in [T]} \eta p_t^\top q_t + \eta^2 p_t^\top q_t^2. \tag{2}$$

Now for the lower bound. From the weight update and induction on t,

$$\begin{aligned}
w_{T+1}(i) &= \prod_{t\in[T]} (1-\eta q_t(i)+\eta^2 q_t(i)^2) \\
\log\|w_{T+1}\|_1 &= \log\left[\sum_{i\in[n]}\prod_{t\in[T]} (1-\eta q_t(i)+\eta^2 q_t(i)^2)\right] \\
&\geq \log\left[\max_{i\in[n]}\prod_{t\in[T]} (1-\eta q_t(i)+\eta^2 q_t(i)^2)\right] \\
&= \max_{i\in[n]}\sum_{t\in[T]} \log(1-\eta q_t(i)+\eta^2 q_t(i)^2) \\
&\geq \max_{i\in[n]}\sum_{t\in[T]} [\min\{-\eta q_t(i),1\}],
\end{aligned}$$

where we use the fact that $1+z+z^2 \geq \exp(\min\{z,1\})$ for all $z\in\mathbb{R}$. Putting this together with the upper bound (2),

$$\begin{aligned}
\max_{i\in[n]}\sum_{t\in[T]} [\min\{-\eta q_t(i),1\}] &\leq \log n - \sum_{t\in[T]} \eta p_t^\top q_t + \eta^2 p_t^\top q_t^2 \\
\sum_{t\in[T]} \eta p_t^\top q_t &\leq -\left(\max_{i\in[n]}\sum_{t\in[T]} [\min\{-\eta q_t(i),1\}]\right) + \log n + \eta^2 p_t^\top q_t^2, \\
&= \min_{i\in[n]}\sum_{t\in[T]} [\max\{\eta q_t(i),-1\}] + \log n + \eta^2 p_t^\top q_t^2,
\end{aligned}$$

and the lemma follows, dividing through by η. □

Next, we introduce a general purpose concentration inequality. The Bernstein inequality states that, for random variables $Z_t : t\in[T]$, where each is independent and $E[Z_t]=0, E[Z_t^2]\leq s, |Z_t|\leq V$,

$$\text{Prob}\left\{\sum_{t\in[T]} Z_t \geq \alpha\right\} \leq \exp\left(-\frac{\alpha^2}{2(Ts+\alpha V/3)}\right)$$

This can be proved in a similar way to the Chernoff bound.

For this setting, we want to bound how much $v_t(i)$ deviate from the mean, but the random variables v_t are not independent. However, they are a martingale under a certain filtration, and this weaker condition is good enough to prove Bernstein's inequality. The following extension to Bernstein's inequality is stated without proof; see Lemma B.3 from [1] for the explicit details.

Lemma 2.3. *Let $\{Z_t\}$ be a martingale difference sequence with respect to filtration $\{S_t\}$, such that $\mathbf{E}[Z_t|S_1,...,S_t]=0$. Assume the filtration $\{S_t\}$ is such that the values in S_t are determined using only those in S_{t-1}, and not any previous history, meaning the joint probability distribution is*

$$\text{Prob}\left\{S_1=s_1, S_2=s_2,\ldots,S_T=s_t\right\} = \prod_{t\in[T-1]} \text{Prob}\left\{S_{t+1}=s_{t+1} \mid S_t=s_t\right\},$$

In addition, assume for all t, $\mathbf{E}[Z_t^2|S_1,...,S_t]\leq s$, and $|Z_t|\leq V$. Then

$$\text{Prob}\left\{\sum_{t\in T} Z_t \geq \alpha\right\} \leq \exp\left(-\frac{\alpha^2}{2(Ts+\alpha V/3)}\right)$$

With this lemma in hand, we can now bound the error between the entries of v_t (generated by only sampling 1 entry of the vector) and the true value of the entry $A_i x_t$.

Lemma 2.4. *For $\eta \geq \sqrt{\frac{\log n}{T}}$, with probability at least $1 - O(1/n)$,*

$$\max_i \sum_{t\in[T]} [v_t(i) - A_i x_t] \leq 4\eta T.$$

Proof. Let $\bar{v_t}(i)$ be the unclipped $v_t(i)$. Recall that as defined, $E[\bar{v_t}(i)] = A_i x_t$, and more importantly $E[v_t(i)] \neq A_i x_t$ because $v_t(i)$ is biased from the clipping.

We assume the expectation's absolute value is bounded above by $\frac{1}{\eta}$.

First, bound the variance of the unclipped $\bar{v_t}(i)$. Recall that A_i is from the unit ball and each x_t is inside the unit ball, so

$$Var(\bar{v_t}(i)) \leq E[\bar{v_t}(i)^2] \leq \|A_i\|^2 \|x_t\|^2 \leq 1$$

By the clipping lemma, $|E[v_t(i)] - A_i x_t| \leq 2\eta$.

To prove the lemma, we show that for all i, $\sum_{t\in[T]} [v_t(i) - E[v_t(i)]] \leq 4\eta T$ with probability $1 - O(\frac{1}{n^2})$. By triangle inequality and union bound, this would prove the desired result.

For a fixed i, let $Z_t^i \equiv v_t(i) - E[v_t(i)] = v_t(i) - A_i x_t$. This has expectation 0. Consider the filtration given by

$$S_t \equiv (x_t, p_t, w_t, y_t, v_{t-1}, i_{t-1}, j_{t-1}, v_{t-1} - \mathbf{E}[v_{t-1}]),$$

Denote $\mathbf{E}_t[\cdot] = \mathbf{E}[\cdot|S_t]$. Observe that

1. $\forall t$. $\mathbf{E}_t[(Z_t^i)^2] = Var(v_t(i)) \leq \mathrm{Var}(\bar{v_t}(i)) \leq 1$ (follows from expectation abs value bounded above by $1/\eta$, so clipping must reduce the variance.)
2. $|Z_t^i| \leq 2/\eta$. This holds since by clipping, $|v_t(i)| \leq 1/\eta$, and hence

$$|Z_t^i| = |v_t(i) - E[v_t(i)]| \leq |v_t(i)| + |E[v_t(i)]| \leq \frac{2}{\eta}$$

These conditions are strict enough to use the extended Bernstein inequality, with $s = 1$ and $V = 2/\eta$. This gives

$$\begin{aligned}\mathrm{Prob}\left\{\sum_{t\in[T]} Z_t \geq 4\eta T\right\} &\leq \exp\left(-\frac{16\eta^2 T^2}{2(T + \frac{8T}{3})}\right) \\ &\leq \exp\left(-\frac{48}{22}\eta^2 T\right) \leq \exp(-2\eta^2 T)\end{aligned}$$

and for $\eta \geq \sqrt{\frac{\log n}{T}}$ this is $\leq \frac{1}{n^2}$. □

For the following proofs, $\mu_t(i) = A_i x_t$ represents the expectation of the unclipped estimator $\bar{v_t}$.

Lemma 2.5. *For $\eta \geq \sqrt{\frac{\log n}{T}}$, with probability at least $1 - O(1/n)$, it holds that*

$$\left|\sum_{t\in[T]} \mu_t(i_t) - \sum_t p_t^\top v_t\right| \leq 10\eta T$$

The motivation here is A_{i_t} is the row selected on iteration t. We want a way to bound the regret over the distribution given by multiplicative weights.

Proof. Prove this by proving 2 other inequalities. For the same η w.p. $1 - O(1/n)$,

$$\left| \sum_{t \in [T]} p_t^\top v_t - \sum_t p_t^\top \mu_t \right| \leq 4\eta T.$$

$$\left| \sum_{t \in [T]} \mu_t(i_t) - \sum_t p_t \mu_t \right| \leq 6\kappa\eta T.$$

where κ is some constant such that $|\mu_t(i)| \leq \kappa$. For the perceptron algorithm, $\kappa = 1$. Combining with triangle inequality gives the lemma.

The proof for the first one is essentially the same as the previous lemma. Lemma 2.1 implies that $|\operatorname{\mathbf{E}}[v_t(i)] - \mu_t(i)| \leq 2\eta$ as shown in the previous proof. Since p_t is a distribution, it follows that $|\operatorname{\mathbf{E}}[p_t^\top v_t] - p_t^\top \mu_t| \leq 2\eta$

Let $Z_t \equiv p_t^\top v_t - \operatorname{\mathbf{E}}[p_t^\top v_t] = \sum_i p_t(i) Z_t^i$, where $Z_t^i = v_t(i) - \operatorname{\mathbf{E}}[v_t(i)]$. Consider the filtration given by

$$S_t \equiv (x_t, p_t, w_t, y_t, v_{t-1}, i_{t-1}, j_{t-1}, v_{t-1} - \operatorname{\mathbf{E}}[v_{t-1}]),$$

Using the notation $\operatorname{\mathbf{E}}_t[\cdot] = \operatorname{\mathbf{E}}[\cdot|S_t]$, the quantities $|Z_t|$ and $\operatorname{\mathbf{E}}_t[Z_t^2]$ can be bounded as follows:

$$|Z_t| = \left| \sum_i p_t(i) Z_t^i \right| \leq \sum_i p_t(i) |Z_t^i| \leq 2\eta^{-1} \qquad \text{using } |Z_t^i| \leq 2\eta^{-1} \text{ as shown in Lemma 2.4.}$$

By properties of variance, we have

$$\operatorname{\mathbf{E}}[Z_t^2] = Var[p_t^\top v_t] = \sum_i p_t(i)^2 Var(v_t(i)) \leq \max_i Var[v_t(i)] \leq 1.$$

Apply Bernstein's inequality to get the same result.

The proof of the second also follows from Bernstein's. Let $Z_t \equiv E[v_t(i_t)] - p_t \mu_t$, where now μ_t is a constant vector and i_t is the random variable, and consider the filtration given by

$$S_t \equiv (x_t, p_t, w_t, y_t, v_{t-1}, i_{t-1}, j_{t-1}, Z_{t-1}),$$

The expectation of $\mu_t(i_t)$, conditioning on S_t with respect to the random choice $r(i_t)$, is $p_t \mu_t$. Hence $\operatorname{\mathbf{E}}_t[Z_t] = 0$, where $\operatorname{\mathbf{E}}_t[\cdot]$ denotes $\operatorname{\mathbf{E}}[\cdot|S_t]$. The parameters $|Z_t|$ and $\operatorname{\mathbf{E}}[Z_t^2]$ can be bounded as follows:

$$\begin{aligned} |Z_t| &\leq |\mu_t(i)| + |p_t \mu_t| \leq 2\kappa \\ \operatorname{\mathbf{E}}[Z_t^2] &= \operatorname{\mathbf{E}}[(\mu_t(i) - p_t^\top \mu_t)^2] \leq 2 \operatorname{\mathbf{E}}[\mu_t(i)^2] + 2(p_t^\top \mu_t)^2 \leq 4\kappa^2 \end{aligned}$$

Applying Lemma 2.3 to $Z \equiv \sum_{t \in T} Z_t$, with $s = 4\kappa^2 \, V \leq 2\kappa$, we obtain

$$\text{Prob}\left\{ \sum_{t \in [T]} Z_t \geq \alpha \right\} \leq \exp\left(-\frac{\alpha^2}{4\kappa^2 T + 2\kappa\alpha} \right),$$

With $\alpha = 6\kappa\eta T$, and assuming $1 > \eta \geq \sqrt{\frac{\log n}{T}}$, we obtain

$$\begin{aligned}\text{Prob}\left\{\sum_{t\in[T]} Z_t \geq 6\kappa\eta T\right\} &\leq \exp\left(-\frac{36\eta^2\kappa^2T^2}{4\kappa^2T + 12\kappa^2\eta T}\right)\\ &\leq \exp\left(-\frac{36\eta^2\kappa^2T^2}{17\kappa^2T}\right)\\ &\leq \exp(-2\eta^2T) \leq \frac{1}{n^2}\end{aligned}$$

Combining both results completes the proof. □

Lemma 2.6. *With probability at least $1 - \frac{1}{4}$, it holds that $\sum_t p_t^\top v_t^2 \leq 8\kappa^2 T$.*

Proof. Again let $\bar{v_t}(i)$ be the unclipped $v_t(i)$.

$$\mathbf{E}[\bar{v_t}^2(i)] = \text{Var}(\bar{v_t}(i)) + \mathbf{E}[\bar{v_t}(i)]^2 \leq 1 + \kappa^2 \leq 2\kappa^2$$

under the assumption $\kappa \geq 1$. Clipping can only reduce the second moment, so $\mathbf{E}[v_t^2(i)] \leq \mathbf{E}[\bar{v_t}^2(i)] \leq 2\kappa^2$. Since p_t is a distribution, by linearity of expectation $\mathbf{E}[\sum_{t\in[T]} p_t^\top v_t^2] \leq 2\kappa^2 T$. Since $v_t^2 \geq 0$, Markov's inequality gives $\mathbf{E}[\sum_{t\in[T]} p_t^\top v_t^2] \geq 8\kappa^2 T$ w.p. $\frac{1}{4}$. □

Lemma 2.7 (Lazy Projection OGD)**.** *Consider a set of vectors $q_1, \ldots, q_T \in \mathbb{R}^d$ such that $\|q_i\|_2 \leq 1$. Let*

$$x_{t+1} \leftarrow \arg\min_{x\in\mathbb{B}} \left\{\sum_{\tau=1}^{t} q_\tau^\top \cdot x + \sqrt{2T}\|x\|_2^2\right\}$$

Then

$$\max_{x\in\mathbb{B}} \sum_{t=1}^{T} q_t^\top x \leq \sum_{t=1}^{T} q_t^\top x_t + 2\sqrt{2T}$$

This is true even if each q_t is dependent on $x_1, \ldots, x_{t-1}$.

The final algorithm returns the average solution x_t over all iterations. Each x_t is a projection of y_t to the unit ball.

See Theorem 2.1 in [3] for the proof. Notice that the solution of the above optimization problem is simply:

$$x_{t+1} = \frac{y_{t+1}}{\max\{1, \|y_{t+1}\|\}} \ , \ y_{t+1} = \frac{-\sum_{\tau=1}^{t} q_\tau}{\sqrt{2T}}$$

2.4 Main Theorem

Theorem 2.8 (Perceptron)**.** *With probability 1/2, the sublinear perceptron returns a solution $\bar{x}$ that is an ε-approximation.*

Proof. First we use the regret bounds for lazy gradient descent to lower bound $\sum_{t\in[T]} A_{i_t} x_t$, next we get an upper bound for that quantity using the Weak Regret lemma above, and then we combine the two.

By definition, $A_i x^* \geq \sigma$ for all $i \in [n]$, and so, using the bound of Lemma 2.7,

$$T\sigma \leq \max_{x\in\mathbb{B}} \sum_{t\in[T]} A_{i_t} x \leq \sum_{t\in[T]} A_{i_t} x_t + 2\sqrt{2T}, \tag{3}$$

or rearranging,

$$\sum_{t\in[T]} A_{i_t} x_t \geq T\sigma - 2\sqrt{2T}. \tag{4}$$

Now we turn to the MW part of our algorithm. By the Weak Regret Lemma 2.2, and using the clipping of $v_t(i)$,

$$\sum_{t\in[T]} p_t^\top v_t \leq \min_{i\in[n]} \sum_{t\in[T]} v_t(i) + (\log n)/\eta + \eta \sum_{t\in[T]} p_t^\top v_t^2.$$

By Lemma 2.4 above, with high probability, for any $i \in [n]$,

$$\sum_{t\in[T]} A_i x_t \geq \sum_{t\in[T]} v_t(i) - 4\eta T,$$

so that with high probability

$$\sum_{t\in[T]} p_t^\top v_t \leq \min_{i\in[n]} \sum_{t\in[T]} A_i x_t + \frac{\log n}{/\eta} + \eta \sum_{t\in[T]} p_t^\top v_t^2 + 4\eta T$$

Combining (4) and (2.4) we get

$$\begin{aligned}\min_{i\in[n]} \sum_{t\in[T]} A_i x_t \geq & -\frac{\log n}{\eta} - \eta \sum_{t\in[T]} p_t^\top v_t^2 - 4\eta T \\ & + T\sigma - 2\sqrt{2T} - \left| \sum_{t\in[T]} p_t^\top v_t - \sum_{t\in[T]} A_{i_t} x_t \right|\end{aligned}$$

By Lemmas 2.5, 2.6 we have w.p at least $\frac{3}{4} - O(\frac{1}{n}) \geq \frac{1}{2}$

$$\begin{aligned}\min_{i\in[n]} \sum_{t\in[T]} A_i x_t &\geq -\frac{\log n}{\eta} - 8\eta T - 4\eta T + T\sigma - 2\sqrt{2T} - 10\eta T \\ &\geq T\sigma - \frac{\log n}{\eta} - 22\eta T.\end{aligned}$$

Dividing through by T, and using our choice of $\eta = \sqrt{\frac{\log n}{T}} = \frac{\epsilon}{200}$ gives

$$\begin{aligned}\min_{i\in[n]} A_i \left(\sum_{t\in[T]} x_t/T \right) &\geq \sigma - \frac{\log n}{200^2 \epsilon^{-2} \log n \frac{\epsilon}{200}} - 22\frac{\epsilon}{200}. \\ &= \sigma - \frac{\epsilon}{200} - 22\frac{\epsilon}{200} \geq \sigma - \frac{\epsilon}{2}\end{aligned}$$

So $\min_i A_i \bar{x} \geq \sigma - \varepsilon/2$ w.p. at least $1/2$ as claimed. □

It seems like the constant for T can be made much smaller, something like $T = 30^2 \epsilon^{-2} \log n$ should work. In any case, we need at least $O((\log n)\epsilon^{-2})$ iterations.

For run time, sampling and updating y, w each iteration takes $\tilde{O}(n+d)$ time, and it overall runs in sublinear $\tilde{O}(\epsilon^{-2}(n+d))$.

3 Generic Primal-Dual Algorithm

Hazan et al. [1] follow their analysis of the perceptron algorithm by applying similar methods to related problems, including Minimum Enclosing Ball (MEB) and the Kernelized Perceptron. We focus on their derivation of a general primal-dual method in sublinear time for a subset of problems.

In this scenario, we have a constrained optimization problem for which low-regret algorithms exist and low-variance sampling can be applied efficiently. Consider the problem

$$\max_{x \in \mathcal{K}} \min_{i} c_i(x) = \sigma$$

with optimum σ. Suppose that, for $\mathcal{K}$ and cost functions $c_i(x)$, there exists an iterative low-regret algorithm (LRA) with regret $R(T) = o(T)$. Let

$$T_\varepsilon(LRA) = \min\{T : R(T)/T \leq \varepsilon\}$$

Also, suppose we have a procedure **Sample**(x, c) that returns an unbiased estimate of $c(x)$ with variance ≤ 1 and runs in constant time. Finally, assume that $|c_i(x)| \leq 1$ for all $x \in \mathcal{K},\ i \in [n]$. An algorithm for solving this problem is given in Figure 2.

Algorithm 2 Generic Sublinear Primal-Dual Algorithm

Let $T \leftarrow \max\left\{T_\varepsilon(LRA), \frac{\log n}{\varepsilon^2}\right\}$, $x_1 \leftarrow LRA(\text{initial})$, $w_1 \leftarrow \mathbf{1}_n$, $\eta \leftarrow \frac{1}{100}\sqrt{\frac{\log n}{T}}$.
for $t = 1$ to T **do**
  for $i \in [n]$ **do**
    $v_t(i) \leftarrow$ **Sample**(x_t, c_i)
    $v_t(i) \leftarrow \text{clip}(\tilde{v}_t(i), 1/\eta)$
    $w_{t+1}(i) \leftarrow w_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$
  end for
  $p_t \leftarrow \frac{w_t}{\|w_t\|_1}$,
  Choose $i_t \in [n]$ by $i_t \leftarrow i$ with probability $p_t(i)$.
  $x_t \leftarrow LRA(x_{t-1}, c_{i_t})$
end for
return $\bar{x} = \frac{1}{T}\sum_t x_t$

The proof of the performance is very similar to that of the sublinear perceptron.

Theorem 3.1. *Algorithm 2 returns, with probability at least $\frac{1}{2}$, an ε-approximate solution, in $\max\{T_\varepsilon(LRA), \log(n)/\varepsilon^2\}$ iterations.*

Proof. We bound $\sum_{t\in[T]} c_{i_t}(x_t)$ above and below and combine these bounds.

Since $c_i(x^*) \geq \sigma$ for all $i \in [n]$, the LRA regret bound gives

$$T\sigma \leq \max_{x\in\mathbb{B}} \sum_{t\in[T]} c_{i_t}(x) \leq \sum_{t\in[T]} c_{i_t}(x_t) + R(T)$$
$$\sum_{t\in[T]} c_{i_t}(x_t) \geq T\sigma - R(T)$$

For MW, we refer to Lemma 2.2, and since v_t are clipped,

$$\sum_{t\in[T]} p_t^\top v_t \leq \min_{i\in[n]} \sum_{t\in[T]} v_t(i) + \frac{\log(n)}{\eta} + \eta \sum_{t\in[T]} p_t^\top v_t^2$$

From Lemmas 2.4, 2.5, since **Sample** is unbiased with variance ≤ 1, with high probability

$$\forall i \in [n] : \sum_{t\in[T]} v_t(i) \leq \sum_{t\in[T]} c_i(x_t) + O(\eta T)$$

$$\left| \sum_{t\in[T]} c_{i_t}(x_t) - \sum_{t\in[T]} p_t^\top v_t \right| = O(\eta T)$$

It follows from the previous inequality that with high probability,

$$\sum_{t\in[T]} c_i(x_t) \leq \min_{i\in[n]} \sum_{t\in[T]} c_i(x_t) + O\left(\frac{\log(n)}{\eta} + \eta \sum_{t\in[T]} p_t^\top v_t^2 + \eta T\right)$$

Combining this upper bound with the lower bound, with high probability,

$$\min_{i\in[n]} \sum_{t\in[T]} c_i(x_t) \geq -O\left(\frac{\log n}{\eta} + \eta T + \eta \sum_{t\in[T]} p_t^\top v_t^2\right) - R(T)$$

And by Lemma 2.6, with probability at least $\frac{1}{2}$,

$$\min_{i\in[n]} \sum_{t\in[T]} c_i(x_t) \geq -O\left(\frac{\log n}{\eta} + \eta T\right) - R(T)$$

Dividing by T, $\min_i c_i \bar{x} \geq \sigma - \varepsilon/2$, as desired. □

4 Approximating Semidefinite Programs

4.1 Problem

Semidefinite programming (SDP) is a field growing in importance, with applications in combinatorial optimization and machine learning. We investigate sublinear algorithms for solving general SDPs. As with the previous sections, we assume the SDP is in memory and we can access an arbitrary entry of the matrices in constant time (e.g. as if they were in an array.)

The SDP is assumed to be of the form

$$\begin{aligned} &\text{maximize } C \circ X \\ &\text{subject to } A_i \circ X \geq 0, \;\; i = 1, \ldots, m \\ &X \succeq 0 \end{aligned}$$

Additionally, we require the Frobenius norm of the constraints $\left(\|A\| = \sqrt{\sum_{i,j} A_{ij}^2}\right)$ to be at most 1. This is analogous to requiring the examples from the perceptron to have norm at most 1. All SDPs can be reduced to this form, and these assumptions are most convenient to proving the desired bounds.

Instead of optimizing with respect to the constraints, we only try to find a feasible solution in sublinear time. Given a feasibility algorithm, we can do binary search by adding a $C \circ X \geq c$ constraint to get an optimum.

4.2 Algorithm: Sublinear SDP

In the previous section, it was shown that there is a general purpose sublinear algorithm for problems of the form

$$\max_x \min_i c_i(x) = \sigma$$

with a low regret algorithm (LRA) that gives regret $o(T)$ in T iterations. The paper reduces SDP feasibility to this form and gives such an LRA.

The feasibility problem is written as the min-max formulation

$$\max_{X \succeq 0} \min_{i \in [m]} A_i \circ X$$

which has a non-negative optimum iff the SDP is feasible. Let the optimum be σ. An ϵ additive approximation algorithm is an algorithm that returns a feasible X such that $\min_{i \in [m]} A_i \circ X \geq \sigma - \epsilon$. Additionally, we restrict X to be from the bounded semidefinite cone $\mathcal{K} = \{X : X \succeq 0, Tr(X) \leq 1\}$. If $\sigma > 0$, then allowing unbounded X can make $A_i \circ X$ arbitrarily large. Again, a feasible solution exists iff a feasible solution in $\mathcal{K}$ exists, so this loses no generality.

To create a low regret algorithm, we loosen the min-max problem to

$$\max_x \min_{p \in \Delta_m} \sum_{i \in [m]} p(i) A_i \circ X$$

where Δ_m is the unit simplex. This still has the same optimum, since the best solution sets $p(i) = 1$ at the best constraint.

Similarly to the perceptron, we treat the optimization problem as a game. One player maximizes the objective over X by keeping p fixed. The other minimizes the objective over p by keeping X fixed. Both do updates by gradient descent. Finding a sublinear regret implementation of both will give sublinear performance.

The two players can be thought of as the primal player and the dual player. The primal maximizes x by adding a vector in the direction of the gradient $\sum_i p(i) A_i$. To get sublinear performance, use the ℓ_1 sample, which samples A_i with prob. $p(i)$.

The dual player updates p with a MW algorithm, where weight $w(i)$ is updated in direction of gradient $A_i \circ X$. Again, to get sublinear performance, $A_i \circ X$ is sampled by picking only a specific entry of A_i based on X. This is done with the ℓ_2-sample, and the sample is clipped to deal with extreme values. Since $\|A_i\| \leq 1$, the variance of the unclipped estimator is at most 1.

We store a weight vector $w \in \mathbb{R}^m$ and a matrix Y that is the running sum of sampled constraints. Like the perceptron, at each iteration we need to project the running total Y onto an X in the space of valid solutions $\mathcal{K}$. For the perceptron, we assume the valid solutions are in the unit ball, and the projection is easy. In the SDP problem, projection onto the positive semidefinite cone $\mathcal{K}$ is much harder. To do so, we use Hazan's algorithm, which has the following guarantee, proved in [2].

Lemma 4.1. *Given matrix $Y \in \mathbb{R}^{n \times n}, \epsilon > 0$, let $f(X) = -\|Y - X\|^2$, and $X^* = \operatorname{argmax}_{X \in \mathcal{K}} f(X)$. Hazan's algorithm returns a solution $\tilde{X} \in \mathcal{K}$ of rank at most ϵ^{-1} such that $f(X^*) - f(\tilde{X}) \leq \epsilon$ in $\tilde{O}\left(\frac{n^2}{\epsilon^{1.5}}\right)$ time*

The full algorithm is in Figure 3. $0_{n \times n}$ is the zero matrix, $\mathbf{1}_m$ is the all one vector.

4.3 Analysis

To analyze the performance, we first state the corresponding regret lemmas without proof.

Algorithm 3 Sublinear SDP

Input: $\varepsilon > 0$, $A_i \in \mathbb{R}^{n\times n}$ for $i \in [m]$.
Let $Y_1 \leftarrow 0_{n\times n}$, $w_1 \leftarrow \mathbf{1}_m, \eta \leftarrow \sqrt{\frac{\log m}{T}}, \epsilon_P = \epsilon/2$.
for $t = 1$ to T **do**
    $p_t \leftarrow \frac{w_t}{\|w_t\|_1}$, $X_t \leftarrow \text{ApproxProject}(Y_t, \epsilon_P^2)$
    Choose $i_t \in [m]$ by $i_t \leftarrow i$ with prob. $p_t(i)$.
    $Y_{t+1} \leftarrow Y_t + \frac{1}{\sqrt{2T}} A_{i_t}$
    Choose $(j_t, l_t) \in [n] \times [n]$ by $(j_t, l_t) \leftarrow (j, l)$ with probability $X_t(j,l)^2/\|X_t\|^2$.
    for $i \in [m]$ **do**
        $\tilde{v}_t(i) \leftarrow A_i(j_t, l_t)\|X_t\|^2/X_t(j_t, l_t)$
        $v_t(i) \leftarrow \text{clip}(\tilde{v}_t(i), 1/\eta)$
        $w_{t+1}(i) \leftarrow w_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$
    end for
end for
return $\bar{X} = \frac{1}{T}\sum_t X_t$

Lemma 4.2. *For $1/4 \geq \eta \geq \sqrt{\frac{\log m}{T}}$, with probability at least $1 - O(1/m)$,*

$$\max_{i\in[m]} \sum_{t\in[T]} (v_t(i) - A_i \circ X_t) \leq 4\eta T.$$

Lemma 4.3. *For $1/4 \geq \eta \geq \sqrt{\frac{\log m}{T}}$, with probability at least $1 - O(1/m)$,*

$$\left| \sum_{t\in[T]} A_{i_t} \circ X_t - \sum_{t\in[T]} p_t^T v_t \right| \leq 10\eta T.$$

Lemma 4.4. *With probability at least $1 - \frac{1}{4}$, $\sum_t p_t^\top v_t^2 \leq 8\kappa^2 T$. For the SDP problem, $\kappa = 1$.*

Lemma 4.5. *Consider matrices $A_1, \ldots, A_T \in \mathbb{R}^{n\times n}$ such that $\|A_i\| \leq 1$. Let $X_0 = 0_{n\times n}, X_{t+1} \leftarrow \arg\min_{X\in\mathcal{K}} \|\frac{1}{\sqrt{2T}} \sum_{\tau=1}^t A_\tau - X\|$ Then*

$$\max_{x\in\mathcal{K}} \sum_{t=1}^T A_t \circ X - \sum_{t=1}^T A_t \circ X_t \leq 2\sqrt{2T}.$$

All follow from the perception regret lemmas since $A_i \circ X_t$ is no different from a dot product of vectors in higher dimensional space. The performance result and proof of it are also similar to those for the perceptron.

Theorem 4.6. *Sublinear SDP (Algorithm 3) returns an ε-additive approximation with high probability.*

Proof. Start with the OGD lemma. Consider the sequence of sampled matrices $A_{i_1}, A_{i_2}, A_{i_3}, \cdots$ and assume we compute the exact projection instead of the approximate one. Let $\tilde{X}_t$ be the exact projection. By Lemma 4.5

$$\max_{X\in\mathcal{K}} \sum_{t\in[T]} A_{i_t} \circ X - \sum_{t\in[T]} A_{i_t} \circ \tilde{X}_t \leq 2\sqrt{2T}$$

The guarantee from Hazan's algorithm and some manipulation with law of cosines gives

$$\|X_t - \tilde{X}_t\|^2 = \|(Y_t - X_t) - (Y_t - \tilde{X}_t)\|^2 \leq \|Y_t - X_t\|^2 - \|Y_t - \tilde{X}_t\|^2 \leq \epsilon_P^2$$

Rewriting the lemma bound to use X_t instead of $\tilde{X}_t$ and some manipulation gives

$$\max_{X \in \mathcal{K}} \sum_{t \in [T]} A_{i_t} \circ X - \sum_{t \in [T]} A_{i_t} \circ X_t \leq 2\sqrt{2T} + \sum_{t \in [T]} A_{i_t} \circ (\tilde{X}_t - X_t)$$

Recall $\|A_i\| \leq 1$. By Cauchy-Schwarz, $A_{i_t} \circ (\tilde{X}_t - X_t) \leq \|A_i\| \|\tilde{X}_t - X_t\| \leq \epsilon_P$. Some rearrangement and substitution of $\sigma = \max_{X \in \mathcal{K}} \min_i A_i \circ X$ gives

$$\sum_{t \in [T]} A_{i_t} \circ X_t \geq \max_{X \in \mathcal{K}} \sum_{t \in [T]} A_{i_t} \circ X - 2\sqrt{2T} - T\epsilon_P \geq T\sigma - 2\sqrt{2T} - T\epsilon_P$$

We need to turn this bound on the sum over all iterations to a bound on the largest difference of $A_i \circ X$ from the optimal. For this, we use the variance MW regret bound (Lemma 2.2). Let q_t be the v_t generated each iteration. Since each v_t is clipped to $1/\eta$, we have

$$\sum_{t \in [T]} p_t^\top v_t \leq \min_{i \in [m]} \left(\sum_{t \in [T]} v_t(i) \right) + \frac{\log m}{\eta} + \eta \sum_{t \in [T]} p_t^\top v_t^2$$

By Lemma 4.2, $\sum_{t \in [T]} v(i) \leq \sum_{t \in [T]} A_i \circ X_t + 4\eta T$ w.h.p. for all i. So, w.h.p.

$$\sum_{t \in [T]} p_t^\top v_t \leq \min_{i \in [m]} \left(\sum_{t \in [T]} A_i \circ X_t \right) + 4\eta T + \frac{\log m}{\eta} + \eta \sum_{t \in [T]} p_t^\top v_t^2$$

Now, we massage this expression to let us use the bound on $\sum_t A_{i_t} \circ X_t$.

$$\begin{aligned}
\min_{i \in [m]} \sum_{t \in [T]} A_i \circ X_t &\geq \sum_{t \in [T]} p_t^\top v_t - 4\eta T - \frac{\log m}{\eta} - \eta \sum_{t \in [T]} p_t^\top v_t^2 \\
&= -4\eta T - \frac{\log m}{\eta} - \eta \sum_{t \in [T]} p_t^\top v_t^2 + \sum_{t \in [T]} A_{i_t} \circ X_t - \left(\sum_{t \in [T]} A_{i_t} \circ X_t - \sum_{t \in [T]} p_t^\top v_t \right) \\
&\geq -4\eta T - \frac{\log m}{\eta} - \eta \sum_{t \in [T]} p_t^\top v_t^2 + T\sigma - 2\sqrt{2T} - T\epsilon_P - \left| \sum_{t \in [T]} A_{i_t} \circ X_t - \sum_{t \in [T]} p_t^\top v_t \right|
\end{aligned}$$

Lemma 4.3 lets us replace the last term with $-10\eta T$, and Lemma 4.4 gives $\sum_{t \in [T]} p_t^\top v_t^2 \leq 8T$ w.p. $\geq 3/4$.

$$\min_{i \in [m]} \sum_{t \in [T]} A_i \circ X_t \geq T\sigma - T\epsilon_P - 22\eta T - \frac{\log m}{\eta} - 2\sqrt{2T}$$

Dividing through by T, and substituting $T = 60^2 \epsilon^{-2} \log m, \eta = \sqrt{\frac{\log m}{T}} = \frac{\epsilon}{60}$, gives

$$\begin{aligned}
\min_{i \in [m]} A_i \circ \left(\sum_{t \in T} X_t / T \right) &\geq \sigma - \epsilon_P - 22\eta - \frac{\log m}{\eta T} - \frac{2\sqrt{2}}{\sqrt{T}} \\
&= \sigma - \frac{\epsilon}{2} - \frac{22\epsilon}{60} - \frac{\epsilon}{60} - \frac{2\sqrt{2}\epsilon}{60\sqrt{\log m}} \\
&\geq \sigma - \epsilon
\end{aligned}$$

giving the desired approximation result. □

For run time analysis, the algorithm runs for $O(\epsilon^{-2} \log m) = \tilde{O}(\epsilon^{-2})$ iterations. It takes $O(m+n^2)$ time to sample i_t and (j_t, l_t), and it also takes $O(m+n^2)$ time to update w and Y_t. The runtime is dominated by the projection step, and the overall run time is $\tilde{O}\left(\frac{m}{\epsilon^2} + \frac{n^2}{\epsilon^5}\right)$

Similarly to the perceptron lower bound, any sublinear ϵ-approximation takes $\Omega\left(\frac{m}{\epsilon^2} + \frac{n^2}{\epsilon^2}\right)$. The gap between the algorithm's runtime and the optimal runtime comes entirely from the projection step. In fact, further work in this area has reduced the runtime to $\tilde{O}\left(\frac{m}{\epsilon^2} + \frac{n^2}{\epsilon^{2.5}}\right)$ by avoiding this projection. Instead, at each iteration we compute a $Z \in \mathcal{K}$ that is approximately orthogonal to the current X_t, and use it to update running total Y. This can be shown to give the same asymptotic performance, and computing this Z is faster than computing the projection. See [6] for details.

For further improvement, the main approach should find a more efficient way to update Y_t and/or generate $X_t \in \mathcal{K}$ from each Y_t.

5 Training Support Vector Machines (SVMs)

5.1 Problem

Support Vector Machines (SVMs) are a widely used machine learning technique for classification tasks. In order to solve the optimization problems used to train SVMs, it is common to use online stochastic optimization approaches, such as stochastic gradient descent, that are optimal with regard to generalization error with only using one pass over the data.

Despite the apparent optimality of existing methods, Hazan et al. [4] have shown that an even better runtime can be achieved, albeit with a non-online approach. By making use of the sublinear primal-dual method described above to read only a subset of the features of the training data, allowing for fewer overall feature accesses, their algorithm, SVM-SIMBA, outperforms the existing online approaches.

We formally describe the linear SVM binary classification problem, and massage it to work with our previous approach. Let $\{\mathbf{x}_i, y_i : i = 1, \dots n\}$ be n labeled points, where $\mathbf{x}_i \in \mathbb{R}^d$, and $y_i \in \{\pm 1\}$. If needed, we normalize the $\mathbf{x}_i$ such that $\|\mathbf{x}_i\| \leq 1$. We define a predictor by a vector $\mathbf{w} \in \mathbb{R}^d$ and a bias $b \in \mathbb{R}$.

The goal for training an SVM can be written as an optimization problem in which we want to minimize the error of our predictor and the norm of $\mathbf{w}$. The error is defined as the average hinge loss, so that our problem is

$$\min_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}} \|\mathbf{w}\|, \hat{R}_{\text{hinge}}(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \left(1 - y\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)\right)_+ \tag{5}$$

This doesn't play nice with our framework, so we use a parameterization of the Pareto optimal points for this problem, introducing slack variables ξ_i and a parameter ν to write this as another optimization problem.

$$\begin{aligned} \max_{\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}, \xi_i \geq 0} \ & \min_{i \in [n]} \ y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) + \xi_i \\ & \|w\| \leq 1 \\ & \sum_{i=1}^{n} \xi_i \leq n\nu \end{aligned} \tag{6}$$

We can obtain all the Pareto optimal solutions of (5) by solving (6) exactly and varying ν. If we solve (6) approximately, we obtain a Pareto sub-optimal solution. The following lemma describes this formally.

Lemma 5.1. *For $\mathbf{w} \neq 0, b \in \mathbb{R}$, let $\nu = \hat{R}_{hinge}(\mathbf{w}, b)/\|w\|$. With this ν, let an ε-suboptimal solution to (6) be $\mathbf{w}^{(\varepsilon)}, b^{(\varepsilon)}, \xi^{(\varepsilon)}$, with value $\gamma^{(\varepsilon)}$. Consider a scaled solution $\tilde{\mathbf{w}} = \mathbf{w}^{(\varepsilon)}/\gamma^{(\varepsilon)}, \tilde{b} == b^{(\varepsilon)}/\gamma^{(\varepsilon)}$. Then*

$$\|\tilde{\mathbf{w}}\| \leq \frac{1}{1 - \varepsilon\|\mathbf{w}\|}\|\mathbf{w}\|$$
$$\hat{R}_{hinge}(\tilde{\mathbf{w}}, \tilde{b}) \leq \frac{1}{1 - \varepsilon\|\mathbf{w}\|}\hat{R}_{hinge}(\mathbf{w}, b)$$

Without affecting optimality, we can place some additional constraints on these problems: $\|\mathbf{w}\| \geq 1$, $0 \leq \nu \leq 1$, and $0 \leq \xi_i \leq 2$.

5.2 Algorithm: SVM-SIMBA

To introduce the sublinear approach to this problem, note that the primal-dual method described in Section 3 can be applied to saddle-point problems of the form

$$\max_{\mathbf{z} \in \mathcal{K}} \min_{p \in \Delta_n} \sum_{i=1}^{n} p_i c_i(\mathbf{z})$$

The SVM problem can be formulated in terms of this structure. Let $\mathbf{z} = (\mathbf{w}, b, \xi)$, and $\mathcal{K} = \mathbb{B}_d \times \mathbb{R} \times \Xi_\nu$, where $\Xi_\nu = \{\xi \in \mathbb{R}^n : 0 \leq \xi_i \leq 2, \|\xi\|_1 \leq n\nu\}$.

A straightforward application of the previous approach will work, but it yields a slow convergence rate, since the regret grows very quickly. To combat this, a hybrid approach is proposed, using a technique from the work of Plotkin, Shmoys, and Tardos (PST) [7]. In the PST primal-dual method, $\mathbf{z}$ is updated by fixing p and solving the "easy" problem

$$\max_{\mathbf{z} \in \mathcal{K}} \sum_{i \in [n]} p_i c_i(\mathbf{z})$$

This can be applied fruitfully to small-sized problems, and we use PST in the algorithm to update ξ and b, giving an update cost of $O(n)$ time for those terms.

The algorithm, called SIMBA (short for "Sublinear IMportance-sampling Bi-stochastic Algorithm"), is a sublinear-time approximation algorithm for the modified SVM binary classification problem (6). In the authors' presentation, the bias term b is omitted for a smoother presentation, and can easily be added back in with no consequence for the analysis. The labels y_i are then made redundant by setting $\mathbf{x}_i \leftarrow -\mathbf{x}_i$ for i such that $y_i = -1$.

5.3 Analysis

The analysis of this algorithm proceeds in a similar fashion to that of the previous algorithms, using the variance MW algorithm and the convergence lemma described at the beginning. The proof is sketched here, as in the paper.

Theorem 5.2. *SVM-SIMBA returns an ε-approximate solution to (6) with probability at least $1/2$. It can be implemented to run in time $\tilde{O}(\varepsilon^{-2}(n + d))$.*

Proof. Let the optimal solution to (6) be $(\mathbf{w}^*, b^*)$ with objective value γ^*. To relate the objective value of our returned solution to γ^*, we bound the average objective value above and below.

The upper bound follows from the regret bound for the MW algorithm: with probability $3/4 - O(1/n)$,

$$\frac{1}{T}\sum_{t \in [T]} p_t^\top (\mathbf{X}\mathbf{w}_t + \xi_t) \leq \frac{1}{T}\min_{i \in [n]} \sum_{t \in [T]} \left(\mathbf{x}_i^\top \mathbf{w}_t + \xi_t(i)\right) + O\left(\sqrt{\frac{\log(n)}{T}}\right)$$

Algorithm 4 SVM-SIMBA

Input: $\varepsilon > 0$, $0 \le \nu \le 1$, $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $\mathbf{x}_i \in \mathbb{B}_d$ for $i \in [n]$.
Let $T \leftarrow 100^2 \varepsilon^{-2} \log n$, $u_1 \leftarrow 0$, $q_1 \leftarrow \mathbf{1}_n$, $\eta \leftarrow \sqrt{\frac{\log(n)}{T}}$
for $t = 1$ to T **do**
    Choose $i_t \leftarrow i$ with probability $p_t(i)$
    Let $u_{t+1} \leftarrow u_t + \frac{1}{\sqrt{2T}} \mathbf{x}_{i_t}$, $\xi_t \leftarrow \arg\max_{\xi \in \Xi_\nu} \left(p_t^\top \xi\right)$
    $\mathbf{w}_t \leftarrow \frac{u_t}{\max\{1, \|u_t\|\}}$
    Choose $j_t \leftarrow j$ with probability $\mathbf{w}_t(j)^2 / \|\mathbf{w}_t\|^2$
    for $i = 1$ to n **do**
        $\tilde{v}_t(i) \leftarrow \mathbf{x}_i(j_t) \|\mathbf{w}_t\|^2 / \mathbf{w}_t(j_t) + \xi_t(i)$
        $v_t(i) \leftarrow \mathrm{clip}(\tilde{v}_t(i), 1/\eta)$
        $q_{t+1}(i) \leftarrow q_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$
    end for
    $p_t \leftarrow \frac{q_t}{\|q_t\|_1}$
end for
return $\bar{\mathbf{w}} = \frac{1}{T} \sum_t \mathbf{w}_t$, $\bar{\xi} = \frac{1}{T} \sum_t \xi_t$

The lower bound is a little more involved. The PST step gives us the bound $\sum_{t \in [T]} p_t^\top \xi_t \ge \sum_{t \in [T]} p_t^\top \xi^*$. Using a standard regret bound for the OGD update, we have, with probability at least $1 - O(1/n)$,

$$\frac{1}{T} \sum_{t \in [T]} p_t^\top (\mathbf{X} \mathbf{w}_t + \xi_t) \ge \gamma^* - O\left(\sqrt{\frac{\log(n)}{T}}\right)$$

Combining these bounds gives the following relation with probability at least $1/2$, and using $T = 100^2 \varepsilon^{-2} \log n$ gives the desired result.

$$\min_{i \in [n]} \sum_{t \in [T]} \left(\mathbf{x}_i^\top \mathbf{w}_t + \xi_t(i)\right) \ge \gamma^* - O\left(\sqrt{\frac{\log(n)}{T}}\right)$$

Regarding the runtime, note that the algorithm performs $O(\varepsilon^{-2} \log(n))$ iterations. With an appropriate implementation, an iteration updates $\mathbf{w}_t$ (taking $O(d)$ time), p_t (taking $O(n)$ time), and ξ_t (taking $O(n)$ time). The complete runtime is $\tilde{O}(\varepsilon^{-2}(n + d))$. □

To rework the bias term b into this algorithm, only a couple of lines need to be changed, where ξ_t is involved. The PST step is changed to

$$(\xi_t, b_t) \leftarrow \operatorname*{argmax}_{\xi \in \Xi_\nu, b \in [-1,1]} p_t^\top (\xi + b \cdot y)$$

and the dual update is changed to

$$\tilde{v}_t(i) \leftarrow \mathbf{x}_i(j_t) \|\mathbf{w}_t\|^2 / \mathbf{w}_t(j_t) + \xi_t(i) + y_i b_t$$

The algorithm returns the average bias, $\bar{b} = \sum_{t \in [T]} b_t / T$, and the analysis can be modified with a few minor technical changes to produce the same runtime.

5.4 Performance Comparison

With the runtime analysis in hand, SVM-SIMBA can be compared to previous methods for training SVMs, by both theoretical and practical methods.

To do the theoretical comparison, it is noted that the goal of learning is to find a predictor that minimizes the *generalization error* $\mathbf{R}_{\text{err}}(\mathbf{w}) = \text{Prob}\{y \langle \mathbf{w}, \mathbf{x} \rangle \leq 0\}$, with $\mathbf{x}, y$ distributed according to a source distribution. The goal of the analysis is to find the runtime to find a predictor $\mathbf{w}$ with generalization error $\mathbf{R}_{\text{err}}(\mathbf{w}) \leq \mathbb{R}^* + \delta$, where

$$\mathbf{R}^* = \mathbf{R}_{\text{hinge}}(\mathbf{w}^*) = \mathbf{E}\left[(1 - y \langle \mathbf{w}^*, \mathbf{x} \rangle)_+\right]$$

is the expected hinge loss for a predictor $\mathbf{w}^*$ with $\|\mathbf{w}^*\| \leq B$, given some B.

From [9], with high probability over a sample of size n, for all predictors $\mathbf{w}$,

$$\mathbf{R}_{\text{err}}(\mathbf{w}) \leq \hat{R}_{\text{hinge}}(\mathbf{w}) + O\left(\frac{\|\mathbf{w}\|^2}{n} + \sqrt{\frac{\|\mathbf{w}\|^2 \hat{R}_{\text{hinge}}(\mathbf{w})}{n}}\right)$$

An online perceptron algorithm can find $\mathbf{w} : \mathbf{R}_{\text{err}}(\mathbf{w}) \leq \mathbf{R}^* + \delta$ in time

$$O\left(\frac{B^2}{\delta} d \cdot \frac{\mathbf{R}^* + \delta}{\delta}\right)$$

Manipulating the runtime for SVM-SIMBA gives us the following statement.

Corollary 5.3. *For any $B \geq 1$ and $\delta > 0$, with high probability over a training set of size $n = \tilde{O}(B^2/\delta \cdot (\delta + \mathbf{R}^*)/\delta)$, SVM-SIMBA outputs $\mathbf{w}$ with $\mathbf{R}_{err}(\mathbf{w}) \leq \mathbf{R}^* + \delta$, where $\mathbf{R}^* = \inf_{\|\mathbf{w}^*\| \leq B} \mathbf{R}_{hinge}(\mathbf{w}^*)$, in time*

$$\tilde{O}\left(\left(B^2 d + \frac{B^4}{\delta} \cdot \frac{\delta + \mathbf{R}^*}{\delta}\right) \cdot \left(\frac{\delta + \mathbf{R}^*}{\delta}\right)^2\right)$$

When comparing this to the online runtime, the learning regime is a situation in which $\mathbf{R}^*$ is small and $\delta = \Omega(\mathbf{R}^*)$, so that $\frac{\mathbf{R}^* + \delta}{\delta} = O(1)$. It is also assumed that in using a norm-regularized approach like SVM, $d \gg B^2$, so that $d \gg B^2 \left(\frac{\mathbf{R}^* + \delta}{\delta}\right)^2$.

After massaging this expression, it is shown that the runtime of SVM-SIMBA is smaller, by a factor of $(\mathbf{R}^* + \delta) \leq 1$, than the runtime of the online approach, or more generally, any approach which considers entire sample vectors. This is a significant improvement when the error rate is small, or $(\mathbf{R}^* + \delta) = O(1/\log(B^2/\delta))$.

In addition to the theoretical result, SVM-SIMBA was implemented and tested against the competition. SVM-SIMBA and the Pegasos [8] algorithm (based on stochastic gradient descent) were both run on two datasets, then evaluated based on test error relative to the number of feature accesses. After tuning both algorithms' parameters to minimize test error, the plots indicate that SVM-SIMBA performed as well or better as Pegasos; SVM-SIMBA can obtain the same optimal test error as Pegasos with 100 times fewer feature accesses.

Therefore, SVM-SIMBA represents a significant improvement over the methods that preceded it, as shown by the runtime analysis and by practical results. This is accomplished by building on the sublinear primal-dual method explained earlier as well as the PST method.

6 Conclusion

In this survey, we have presented and analyzed the performance of several sublinear algorithms, given by Hazan et al., for optimization of constrained convex programs. The approach running through this survey was first discussed for the perceptron, then used for a more general primal-dual algorithm, which was applied to solving SDPs and training SVMs.

Although most of the results discussed here are specific to machine learning, there are some important takeaways for other problems as well. The generic primal-dual algorithm gives a potential

approach for any optimization problem that can be formalized as a max-min game, and it is worth trying to reduce other problems to this form.

This method also shows the power of multiplicative weights. By relaxing the problem to minimize over a probability distribution instead of the best index, we can efficiently compute a distribution that converges to putting almost all weight on the optimal index, which can be faster than computing the optimal index directly. The results discussed here give bounds even when the expert outcomes are not in $[-1, +1]$, as long as the outcomes have constant variance.

Finally, for general sublinear algorithms, the idea of defining an appropriate random variable whose expectation is the value we want to approximate gives a way to perform faster than runtime bounds that seem optimal, as long as the approximations are managed correctly.

References

[1] Clarkson, Kenneth L., Elad Hazan, and David P. Woodruff. "Sublinear optimization for machine learning." Journal of the ACM (JACM) 59.5 (2012): 23.

[2] Hazan, Elad. "Sparse approximate solutions to semidefinite programs." LATIN 2008: Theoretical Informatics. Springer Berlin Heidelberg, 2008. 306-316.

[3] Hazan, Elad. "The Convex Optimization Approach to Regret Minimization." Optimization for machine learning (2012): 287-303.

[4] Hazan, Elad, Tomer Koren, and Nati Srebro. "Beating SGD: Learning SVMs in sublinear time." Advances in Neural Information Processing Systems. 2011.

[5] Garber, Dan, and Elad Hazan. "Approximating semidefinite programs in sublinear time." Advances in Neural Information Processing Systems. 2011.

[6] Garber, Dan, and Elad Hazan. "Almost Optimal Sublinear Time Algorithm for Semidefinite Programming." arXiv preprint arXiv:1208.5211 (2012).

[7] Plotkin, Serge A., David B. Shmoys, and Éva Tardos. "Fast approximation algorithms for fractional packing and covering problems." Mathematics of Operations Research 20.2 (1995): 257-301.

[8] Shalev-Shwartz, Shai, Yoram Singer, Nathan Srebro, and Andrew Cotter. "Pegasos: Primal estimated sub-gradient solver for SVM." Mathematical programming 127.1 (2011): 3-30.

[9] Srebro, Nathan, Karthik Sridharan, and Ambuj Tewari. "Smoothness, low noise and fast rates." Advances in Neural Information Processing Systems. 2010.