Matrix Multiplicative Weight Updates for Solving SDPs

Vaishakh Ravindrakumar

April 13, 2017

Abstract

In this write-up, we elucidate an algorithm based on matrix multiplicative weight updates to near-optimally solve SDPs.

1 Introduction

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{binary_search.png}
\caption{Binary search procedure illustrated. In each case, the interval shrinks as shown above.}
\end{figure}

At a high level, the algorithm is based on binary search, where at each step, a primal-dual method based on multiplicative updates is used to refine our search interval.

In each step, we start with a guess for our primal objective ($\alpha$) and invoke an Oracle to either give us a primal feasible $X^{(t)}$ or a dual feasible $y^*$. If the former occurs (Case 1), we have a guarantee that the primal objective is at least $\alpha$ and if the latter occurs (Case 2), the dual objective is at most $(1 + \delta)\alpha$ for some arbitrarily small $\delta$. This is illustrated in the Figure 1. This allows us to shrink our search interval and by repeating this procedure multiple times, we obtain an approximate optimal value to our SDP to any degree of accuracy.

The rest of the writeup is organized as follows. In Section 2, we formally define an SDP. In Section 3, we describe the Oracle. In Section 4, we describe the procedure used to update the ‘density’ matrix. In Section 5, using the Oracle and ‘density’ updating procedures as black-boxes, we describe our main algorithm. Finally in Section 6, we describe the proof of correctness.
2 Definitions

The primal (P) of an SDP is given by:

\[
\begin{align*}
\max & \quad C \cdot X \\
\text{subject to:} & \quad \forall j \in [m] : A_j \cdot X \leq b_j \\
& \quad X \succeq 0
\end{align*}
\]

The dual (D) for the same is:

\[
\begin{align*}
\min & \quad b^T y \\
\text{subject to:} & \quad \sum_{j=1}^m y_j A_j - C \succeq 0 \\
& \quad y \geq 0
\end{align*}
\]

Here \(X, C, A_j\) are real matrices of dimension \(n \times n\) and \(b, y\) are vectors in \(\mathbb{R}^n\). Further, we assume that \(A_1 = I\) and \(b_1 = R\). This ensures that \(tr(X) \leq R\).

3 The Oracle

The Oracle takes \(X^{(t)}\) as its argument and searches for a \(y\) in the polytope \(D_\alpha = \{y : y \geq 0, b \cdot y \leq \alpha\}\) such that

\[
\sum_{j=1}^m (A_j \cdot X^{(t)}) y_j - C \cdot X^{(t)} \geq 0.
\]

Then, depending on if the Oracle succeeds or fails in finding such a \(y\), we have the following cases:

**Case 1 (Success):** If the Oracle finds such a \(y\), either \(X^{(t)}\) is infeasible or \(C \cdot X^{(t)} \leq \alpha\).

**Proof.** To the contrary, assume \(X^{(t)}\) is feasible and \(C \cdot X^{(t)} > \alpha\). Then,

\[
\sum_{j=1}^m (A_j \cdot X^{(t)}) y_j - C \cdot X^{(t)} \leq \sum_{j=1}^m b_j y_j - C \cdot X^{(t)} < \alpha - \alpha = 0
\]

which is a contradiction. Here, the first inequality follows since \(X^{(t)}\) is feasible and the second strict inequality follows since \(y \in D_\alpha\) and \(C \cdot X^{(t)} > \alpha\). □

**Case 2 (Failure):** If the Oracle fails, then a suitably scaled version of \(X^{(t)}\) is a primal feasible solution with objective value at least \(\alpha\).
Proof. For this, consider the following linear program:

\[
\begin{align*}
\max & \quad \sum_{j=1}^{m} (A_j \cdot X^{(t)}) y_j \\
\text{subject to} & \quad b_y \leq \alpha \\
& \quad y \geq 0
\end{align*}
\]

and its dual:

\[
\begin{align*}
\min & \quad \alpha \phi \\
\forall j & \quad b_j \phi \geq (A_j \cdot X^{(t)}) \\
& \quad \phi \geq 0
\end{align*}
\]

Since there is no \( y \in D_\alpha \) which satisfies (1), the optimum of this primal is less than \( C \cdot X^{(t)} \).

Thus, the primal optimum is finite, implying that the dual is feasible and has the same optimum. Let \( \phi^* \) be the optimum solution of the dual. Then \( \alpha \phi^* \leq C \cdot X^{(t)} \).

Further, since \( b_1 = R \) and \( A_1 X^{(t)} = tr(X^{(t)}) = R \), we conclude that \( \phi^* \geq 1 \). Let \( X^* = \frac{1}{\phi^*} X^{(t)} \). Then \( \forall j, A_j \cdot X^* \leq b_j \) and \( C \cdot X^* \geq \alpha \). So \( X^* \) is primal feasible (for \( (P) \)) with objective value at least \( \alpha \).

Although our Oracle, if it succeeds, gives us a \( y \in D_\alpha \), it is infeasible for the dual \( (D) \). Thus, we define a stronger \((\ell, \rho)\)-bounded Oracle as:

**Definition 1.** An \((\ell, \rho)\)-bounded Oracle for parameters \( 0 \leq \ell \leq \rho \) finds a vector \( y \in D_\alpha \) that additionally satisfies \( \sum_j A_j y_j - C \in [-\ell, \rho] \) or \( \sum_j A_j y_j - C \in [-\rho, \ell] \).

We invoke this stronger Oracle in our main algorithm. Then, if the Oracle never fails, it allows us to derive from a collection of \( y^{(t)} \), a dual \( (D) \) feasible \( y^* \). It is also stated without proof that if the stronger Oracle fails, a suitably scaled version of \( X^{(t)} \) is still primal feasible with objective value at least \( \alpha \).

### 4 Matrix Multiplicative Update (MMU)

The Matrix Multiplicative Update takes as its argument the penalty matrices \( M^{(i)}, i \in [1 : t] \) and for a fixed \( \epsilon \leq 1/2 \), computes the ‘density’ matrix \( P^{(t)} \). It does this by maintaining a ‘weight’ matrix \( W^{(t)} \), initialized as \( W^{(1)} = I_n \). Then, at each time \( t \in [1 : T] \), the following steps are carried out:

1. Compute the ‘density’ matrix \( P^{(t)} = W^{(t)}/tr(W^{(t)}) \).
2. Update the ‘weight’ matrix \( W^{(t+1)} = \exp(-\epsilon \sum_{i=1}^{t} M^{(i)}) \).

Step 1 guarantees that the matrix \( P^{(t)} \) has eigenvalues that sum to 1. In step 2, since the matrix \( -\epsilon \sum_{i=1}^{t} M^{(i)} \) is symmetric, \( W^{(t+1)} \geq 0 \). This follows from the following lemma:

**Lemma 1.** If a matrix \( Q \) is symmetric, \( \exp(Q) \geq 0 \).
Proof.

Since \( Q \) is symmetric, \( Q = UDU^T \) for some diagonal matrix \( D \) and orthonormal matrix \( U \). Then we have,

\[
\exp(Q) = 1 + Q + \frac{Q^2}{2!} + \frac{Q^3}{3!} + ... = U(1 + D + \frac{D^2}{2!} + \frac{D^3}{3!} + ...)U^T
\]

Since for any \( x \in \mathbb{R} \), \( 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... = \exp(x) \geq 0 \), the diagonal matrix \( D' \equiv 1 + D + \frac{D^2}{2!} + \frac{D^3}{3!} + ... \succeq 0 \). Thus, \( \exp(Q) = UD'U^T \succeq 0 \), proving our lemma. \( \square \)

**Lemma 2.** Then, the matrix multiplicative weight updates algorithm guarantees that

\[
(1 - \epsilon) \sum_{M^{(t)} \succeq 0} M^{(t)} \cdot P^{(t)} + (1 + \epsilon) \sum_{M^{(t)} \preceq 0} M^{(t)} \cdot P^{(t)} \leq \lambda_n \left( \sum_{t=1}^{T} M^{(t)} \right) + \ln \frac{n}{\epsilon}. \tag{2}
\]

Proof of this lemma may be found in Satyen Kale’s thesis [1], Chapter 3 and will be used later in proving correctness of our main algorithm.

## 5 Main Algorithm

We run the algorithm for \( t = 1 \) to \( T \), where \( T = \frac{8\ell\rho R^2 \ln(n)}{\delta^2 \alpha^2} \). We set \( \epsilon = \frac{\delta \alpha}{2\ell R} \).

1. Run step 1 of MMU to compute \( P^{(t)} \).
2. Invoke **Oracle** using \( X^{(t)} = RP^{(t)} \).
3. If the **Oracle** fails, stop and output \( X^{(t)} \).
4. Else, let \( y^{(t)} \) be the vector generated by the **Oracle**.
5. Compute the loss matrix to be provided to MMU as :

\[
M^{(t)} = \frac{1}{\ell + \rho} \left[ \sum_{j=1}^{m} A_j y_j^{(t)} - C + \ell^{(t)} I \right],
\]

where

\[
\ell^{(t)} = \begin{cases}
\ell & \text{if } \sum_{j=1}^{m} A_j y_j^{(t)} - C \in [-\ell, \rho] \\
-\ell & \text{if } \sum_{j=1}^{m} A_j y_j^{(t)} - C \in [-\rho, \ell].
\end{cases}
\]

6. Run step 2 of MMU to compute \( W^{(t+1)} \).
7. Increment \( t \) and goto Step 1.
6 Correctness

At any $t = 1$ to $T$, if the Oracle fails, then since a suitably scaled version of $X^{(t)}$ is a primal feasible solution with objective value at least $\alpha$, we move to the greater interval in the Binary Search. If $\forall t = 1$ to $T$ the Oracle doesn’t fail, we move to the lesser interval in the Binary Search and follows from the following theorem. Note that the above procedure is exactly as described in Section 1 and illustrated in Figure 1.

**Theorem 1.** Assume that the Oracle never fails for $T$ iterations. Let $\overline{y} = \frac{1}{T} \sum_{t=1}^{T} y^{(t)}$ and define $y^*$ as $y^*_1 = \overline{y}_1 + \frac{\delta}{R}$ and $y^*_j = \overline{y}_j$ for $j \geq 2$. Then $y^*$ is a feasible dual solution with objective value at most $(1 + \delta)\alpha$.

**Proof.** We provide a proof sketch, the complete proof may be found in Satyen Kale’s thesis [1], page 37-39. Now, during any round $t$ of the MMU algorithm, (1) is satisfied by $X^{(t)}$ and thus we have

$$M^{(t)} \cdot P^{(t)} = \frac{1}{\ell + \rho} \left( \sum_{j=1}^{m} A_{j} y_{j}^{(t)} - C + \ell^{(t)} I \right) \cdot \frac{1}{R} X^{(t)} \geq \frac{\ell^{(t)}}{\ell + \rho}.$$ 

Plugging this into (2), and equivalently writing $M^{(t)} \succeq 0$ as $\ell^{(t)} \geq 0$, $M^{(t)} \preceq 0$ as $\ell^{(t)} \leq 0$ we have

$$\sum_{t: \ell^{(t)} \geq 0} \frac{(1 - \epsilon)\ell^{(t)}}{\ell + \rho} + \sum_{t: \ell^{(t)} \leq 0} \frac{(1 + \epsilon)\ell^{(t)}}{\ell + \rho} \leq \lambda \left( \frac{\sum_{t=1}^{T} \left( \sum_{j=1}^{m} A_{j} y_{j}^{(t)} - C + \ell^{(t)} I \right) \right) + \frac{\ln n}{\epsilon}.$$

Simplifying the above expression, dividing by $T$ and plugging in the specified values of $\epsilon, T$ and using that $\overline{y} = \sum_{t=1}^{T} y^{(t)}$ gives us the desired result: $(1 + \delta)\alpha \geq \frac{\delta}{R} b_1 + b.\overline{y} = b.y^*$.  

This completes the analysis of our algorithm.

7 Acknowledgements

I’d like to declare that I had a discussion with another student in the course, Anant Dhayal and that it’s not a coincidence if our write-ups bear similarities. I’d also like to state that this report is almost completely based on Satyen Kale’s thesis [1].

References