Lecture 6

Solving SDPs using the matrix multiplicative weights algorithm

6.1 The Matrix Multiplicative Weights Algorithm

We will see how a large class of semidefinite programs can be solved very efficiently using an extension of the Multiplicative Weights Algorithm to the case of matrices. Recall the set-up of the algorithm:

- There are n experts.
- At each step $t = 1, \cdots, T$, the player chooses a distribution $p^{(t)}$ over experts.
- A cost vector $m^{(t)} \in [-1, 1]^n$ is supplied by the environment.
- The player suffers a loss of $p^{(t)} \bullet m^{(t)}$.

Now suppose there is a continuous set of experts, each associated with a unit vector $v \in \mathbb{R}^n, \|v\| = 1$. Let the distribution over experts be a distribution $\mathcal{D}$ on the set of all unit vectors, the unit sphere $\mathcal{S}^{n-1}$. Additionally let the loss be specified by a symmetric matrix $M \in \mathbb{R}^{n\times n}, 0 \preceq M \preceq \mathbb{I}$ (all eigenvalues of M are between 0 and 1) such that by definition the loss associated with expert v is given by $v^\top M v$. Our assumption on M implies this loss will always fall in the interval $[0, 1]$. The expected loss under distribution $\mathcal{D}$ is given by

$$\begin{aligned}\mathbb{E}_{v\sim\mathcal{D}}[v^\top M v] &= \mathbb{E}_{v\sim\mathcal{D}}[M \bullet vv^\top] \\ &= M \bullet \underbrace{\mathbb{E}_{v\sim\mathcal{D}}[vv^\top]}_{\rho\succeq 0},\end{aligned}$$

where $\rho \succeq 0$ satisfies $\mathrm{Tr}(\rho) = \mathbb{E}_{v\sim\mathcal{D}}\|v\|^2 = 1$. ρ is called a *density matrix*. (As a notation reminder, $A \bullet B = \mathrm{Tr}(A^\top B)$.)

We then have the following extension of MWA:

Definition 6.1. (**MMWA**) Let $\eta > 0$. Initialize $W^{(1)} = \mathbb{I}$. For $t = 1, \cdots, T$:

- Observe a loss matrix $0 \preceq M^{(t)} \preceq \mathbb{I}$.
- Set $X^{(t)} = \frac{W^{(t)}}{\mathrm{Tr}(W^{(t)})}$. Suffer a loss $X^{(t)} \bullet M^{(t)}$.
- Update $W^{(t+1)} = \exp(-\eta(M^{(1)} + \cdots + M^{(t)}))$.

Remark 6.2. Note that the normalization condition $0 \preceq M^{(t)} \preceq \mathbb{I}$ and $X^{(t)} \bullet \mathbb{I} = \mathrm{Tr}(X^{(t)}) = 1$ implies that the losses are always in $[0, 1]$.

Remark 6.3. The matrix algorithm is a strict generalization of MWA. We can recover the experts framework by considering the special case where all loss matrices are diagonal:

$$M^{(t)} = \begin{pmatrix} m_1^{(t)} & & & \\ & \ddots & 0 & \\ & 0 & \ddots & \\ & & & m_n^{(t)} \end{pmatrix}.$$

Indeed in this case all the $X^{(t)}$ matrices are diagonal as well, and their diagonal coefficients represent a probability distribution over the experts.

Remark 6.4. The algorithm calls for taking the exponential of a symmetric matrix. There are many equivalent ways in which this can be defined. The simplest is to diagonalize A as $A = UDU^\top$, where U is orthogonal and D is diagonal, and define $\exp(A) = U\exp(D)U^\top =$

$$U \begin{pmatrix} e^{D_{11}} & & & \\ & \ddots & 0 & \\ & 0 & \ddots & \\ & & & e^{D_{nn}} \end{pmatrix} U^\top.$$

More generally, for any function f we can define $f(A)$ in this way. Taking $f(x) = x^k$ for some integer k, you can verify that this definition agrees with the usual definition of the matrix product. For instance, $A^2 = (UDU^T)(UDU^T) = UD^2U^T$ since U is orthogonal.

In practice, to avoid explicit diagonalization one can use the Taylor expansion of the exponential: $\exp(A) = \mathbb{I} + A + \frac{A^2}{2!} + \cdots + \frac{A^k}{k!} + \cdots$, which converges exponentially fast. Since the exponential needs to be computed at every step of the algorithm it is important to implement it as efficiently as possible.

Exercise 1. Show that if A and B are symmetric matrices that commute ($AB = BA$) then $\exp(A + B) = \exp(A)\exp(B)$. Find two real 2×2 symmetric matrices A and B such that $\exp(A + B) \neq \exp(A)\exp(B)$.

This shows that the definition of $W^{(t)}$ given in the matrix algorithm is *not* equivalent in general to the update rule $W^{(t)} := \exp(-\eta M^{(t)})W^{(t-1)}$ that we saw in the experts framework;

the two rules are identical only if all matrices $M^{(t)}$ commute. In general, we only know the inequality $\mathrm{Tr}(e^{A+B}) \leq \mathrm{Tr}(e^A e^B)$, which is known as the Golden-Thompson inequality and will be used in the analysis of the convergence time of the MMWA.

The following theorem is a direct extension of the experts theorem:

Theorem 6.5. *For any $M^{(1)}, \ldots, M^{(T)} \in \mathbb{R}^{n \times n}$ such that $0 \preceq M^{(t)} \preceq \mathbb{I}$ for all $t = 1, \ldots, T$, and $0 < \epsilon \leq 1/2$, let $\eta = -\ln(1-\epsilon)$. Then the following relation holds for $M^{(t)}$ and $X^{(t)}$ defined as in the Matrix Multiplicative Weights Algorithm with parameter η:*

$$\sum_{t=1}^{T} M^{(t)} \bullet X^{(t)} \leq (1+\epsilon) \inf_{\rho \in \mathrm{Pos}(\mathbb{R}^n),\, \mathrm{Tr}(\rho)=1} \Big(\sum_{t=1}^{T} M^{(t)} \bullet \rho \Big) + \frac{\ln n}{\epsilon}. \tag{6.1}$$

Proof. You will see the proof in your homework. □

6.2 Solving SDPs using the MMWA

We will see how the MMWA can be used to approximately solve an arbitrary semidefinite program. As we did for the case of LPs, we first reduce the optimization problem to a feasibility problem. Suppose we are given an SDP in standard form:

$$\begin{aligned}
(\mathcal{P}): \quad & \sup \quad B \bullet X \\
& \text{s.t.} \quad A_i \bullet X \leq c_i \quad \forall i = 1 \ldots m \\
& \qquad X \succeq 0 \\
(\mathcal{D}): \quad & \inf \quad c^\top y \\
& \text{s.t.} \quad \sum_i y_i A_i - B \succeq 0 \\
& \qquad y_i \geq 0
\end{aligned}$$

Assume $A_1 = \mathbb{I}, c_1 = R$, which gives the constraint that $\mathrm{Tr}(X) \leq R$, effectively imposing an a priori bound on the size of X. Assume also that the optimal α of the primal problem is non-negative (if not, we can always shift the optimum by adding a positive multiple of the identity to B to make it PSD). To perform a reduction from deciding optimality to deciding feasibility we perform a binary search over $\alpha \in [0, \|B\|R]$ (where $\|B\|$ is the largest eigenvalue of B, so $|B \bullet X| \leq \|B\|\mathrm{Tr}(X) \leq \|B\|R$), at each step deciding feasibility of the following problem:

$$\begin{aligned}
\exists ?\, X \quad \text{s.t.} \quad & B \bullet X > \alpha \\
& A_i \bullet X \leq c_i \\
& X \succeq 0.
\end{aligned} \tag{6.2}$$

We will use the MMWA presented in the previous section to answer this decision problem. The idea is to start with a "random" guess of X, such as $X = R\mathbb{I}/n$. Then we iteratively "improve" X to either turn it into a feasible solution, or somehow obtain a proof that the problem is *not* feasible, in which case we know the objective value of the original SDP is smaller than α. In the latter case we repeat the search between 0 and α; in the former we search between α and $\|B\|R$. The number of iterations required will be logarithmic in $\|B\|R/\delta$, where δ is the precision we'd like to achieve.

The following is the main claim that underlies the "improvement" subroutine for X that we will present afterwards:

Claim 6.6. *Let $X \in \mathbb{R}^{n\times n}$ be PSD. The following are equivalent:*

- *(i) $\exists y \in \mathbb{R}^n_+$ such that $c^\top y \leq \alpha$ and $X \bullet (\sum_i y_i A_i - B) \geq 0$;*
- *(ii) Either X is infeasible or $B \bullet X \leq \alpha$.*

Note that the claim is saying that, if we are not happy (the current X is either infeasible, or it is feasible but does not satisfy the condition on the objective value), then there is a "reason" for this: the vector y is a positive linear combination of the constraints such that X does not satisfy the combined constraint. We will use this y as an indication as to how X can be "improved".

Proof. Assume (i) and suppose X is feasible; we need to show that $B \bullet X \leq \alpha$. From the assumption in (i), we have that

$$\begin{aligned} 0 &\leq \sum_i y_i A_i \bullet X - X \bullet B \\ &\leq y^\top c - X \bullet B \\ &\leq \alpha - X \bullet B, \end{aligned}$$

where the second inequality uses the assumption that X is feasible and the third the assumption on y. Thus $X \bullet B \leq \alpha$, as desired.

We prove the converse implication by contrapositive. Assume (i) does not hold; let's show that (ii) does not hold either. We can assume that $B \bullet X > 0$, as otherwise taking $y = 0$ shows that (i) in fact holds. Scale X so that we have $B \bullet X = \alpha$. For each $i \in \{1, \ldots, m\}$ consider a vector $y = (\alpha/c_i)e_i$ (y is 0 in all but the i-th coordinate). Then y satisfies $y \geq 0$, and $c^\top y = \alpha$. Using our assumption that (i) does not hold, it must be that $X \bullet (\sum_i y_i A_i - B) < 0$, thus $X \bullet A_i \frac{\alpha}{c_i} < X \bullet B = \alpha$. So $\forall i \in \{1, \ldots, m\}$ we have $X \bullet A_i < c_i$. Finally we can scale X a tiny bit more so that we still have $X \bullet A_i \leq c_i$ for all i, but now $B \bullet X > \alpha$. Hence X is feasible with objective value strictly larger than α, and (ii) does not hold, as required. □

The equivalence stated in the claim shows that the existence of a y such that condition (i) is satisfied is a proof that we have not yet reached our goal of finding a good feasible solution. The idea is that solving (i) is much easier than solving the full SDP. In particular y does not need to be dual feasible and in fact, (i) is an LP feasibility problem.

We will show how to solve the SDP feasibility problem under the following assumption:

Assumption: There exists an oracle $\mathcal{O}$ such that given a PSD matrix $X \in \mathbb{R}^{n \times n}$ as input, $\mathcal{O}$ returns either:

(a) A vector y such that (i) in Claim 6.6 above holds, or

(b) A statement that no such y exists.

The "quality" of $\mathcal{O}$ is measured through its "width" σ, which is the largest possible value that $\|\sum_i y_i A_i - B\|$ can take over all vectors y that the oracle might return. When designing an oracle, we want it to be fast and have small width. We will see how to do this in some special cases (such as the SDP for MAXCUT we saw in the previous lecture). Assuming the oracle is given, consider the following algorithm:

Algorithm for solving SDP: Let $X^{(1)} = \frac{R\mathbb{I}}{n}, \varepsilon = \frac{\delta\alpha}{2\sigma R}, \eta = -\ln(1-\varepsilon)$, where $\delta > 0$ is an accuracy parameter. For $t = 1, \cdots, T$, do:

(1) Execute $\mathcal{O}$ on $X^{(t)}$.

- If case (b) happens, then it follows that (i) does not hold, and by Claim 6.6 (ii) does not hold either, and X is feasible such that $B \bullet X > \alpha$: we are done. Stop and return $X^{(t)}$.

- If instead case (a) happens, let $y^{(t)}$ be the vector returned by $\mathcal{O}$. Define a loss matrix
$$M^{(t)} = \frac{\sum_i y_i^{(t)} A_i - B + \sigma\mathbb{I}}{2\sigma},$$
so that the assumption on the width of $\mathcal{O}$ implies $0 \leq M^{(t)} \leq \mathbb{I}$.

(2) Update $X^{(t+1)}$ as in MMWA:
$$W^{(t+1)} = e^{-\eta \sum_{i=1}^{t} M^{(i)}}, \qquad X^{(t+1)} = R\frac{W^{(t+1)}}{\mathrm{Tr}(W^{(t+1)})}.$$

The following theorem states the guarantee for this algorithm.

Theorem 6.7. *Assume case (b) does not happen for $T = \frac{8\sigma^2 R^2}{\delta^2\alpha^2}\ln(n)$ steps. Then $\bar{y} = \frac{\delta\alpha}{R}e_1 + \frac{1}{T}\sum_{t=1}^{T} y^{(t)}$ is a feasible solution to $(\mathcal{D})$ with objective value less than or equal to $(1+\delta)\alpha$.*

We can make the following remarks on the running time of this algorithm:

- The overall performance depends on the running time of the oracle, which needs to be executed once per iteration.

- We need to compute a matrix exponential to update X and it is important to do this efficiently. In general, diagonalizing the loss matrices will take $O(n^3)$ time. Often it is possible to do much better by observing that the only information the oracle requires is the inner product of $X^{(t)}$ with the constraint matrices, but not the full matrix $X^{(t)}$ itself. So it is often the case that it is possible to do the update much more efficiently by keeping only a "low-dimensional sketch" of the matrix exponential. This is based on the Johnson-Lindenstrauss lemma for dimensionality reduction that we will see in a couple weeks.

- We would like to have R and the width of the oracle to be not too large, otherwise the algorithm will require lots of iterations. We will see how to do this for the special case of the MAXCUT SDP in the next lecture.

- The algorithm depends inverse polynomially on the approximation error $\delta\alpha$. This is not great, and in general we could hope for an algorithm that depends only polylogarithmically on δ^{-1}. For example this is the case for the ellipsoid algorithm.

- If all these parameters, $R, \sigma, \delta\alpha$ are constants, or such that $\sigma R/(\delta\alpha)$ is not too large, then the overhead $\ln(n)$ in the number of iterations for the algorithm is very small. This is the main strength of the MMWA, and we will see how to take advantage of it for the case of MAXCUT.

Proof. First let's check the claim on the objective value:

$$\begin{aligned} c^T y &= \frac{\delta\alpha}{R} c^T e_1 + \frac{1}{T}\sum_i c^T y^{(t)} \\ &\leq \delta\alpha + \alpha = (1+\delta)\alpha, \end{aligned}$$

where we used $c_1 = R$ and the guarantee $c^T y^{(t)} \leq \alpha$ for any $y^{(t)}$ returned by the oracle.

Next we need to verify that the returned solution is feasible. From condition (a) we know that $y \geq 0$, so it remains to check that $\sum_i y_i A_i - B \geq 0$. From the MMWA theorem we know that for any unit vector v, and in particular the eigenvector of $\sum_t M^{(t)}$ associated with its smallest eigenvalue λ_n,

$$\begin{aligned} \frac{T}{2} \leq& \sum_{t=1}^{T} M^{(t)} \bullet X^{(t)} \\ \leq& (1+\epsilon)\sum_{t=1}^{T} v^T M^{(t)} v + \frac{\log n}{\epsilon} \\ =& (1+\epsilon)\lambda_n\Big(\sum_{t=1}^{T} \frac{\sum_i y_i^{(t)} A_i - B + \sigma I}{2\sigma}\Big) + \frac{\log n}{\epsilon} \\ =& \frac{1+\epsilon}{2\sigma}\lambda_n\Big(\sum_i \Big(\sum_t y_i^{(t)}\Big) A_i - TB\Big) + (1+\varepsilon)\frac{T\sigma}{2\sigma} + \frac{\log n}{\epsilon}. \end{aligned}$$

Here the first line holds by definition of $M^{(t)}$ and using guarantee (i) for the oracle, the third line holds by our choice of v, and the fourth uses that for any PSD A, $\lambda_i((A + b\mathbb{I})/c) = (\lambda_i(A) + b)/c$ for any b and any $c > 0$ (where λ_i is the i-th largest eigenvalue). Rearranging terms,

$$\begin{aligned}\Big(-\frac{\varepsilon(1+\varepsilon)T}{2} - \frac{\log n}{\epsilon}\Big)\frac{2\sigma}{(1+\epsilon)T} &\leq \lambda_n\Big(\sum_i \Big(\frac{1}{T}\sum_t y_i^{(t)}\Big)A_i - B\Big) \\ &= \lambda_n\Big(\sum_i \bar{y}_i A_i - B\Big) - \frac{\delta\alpha}{R},\end{aligned}$$

where the equality follows from the definition of $\bar{y}$. Using the definition of T and ε it follows that

$$\frac{-4\sigma\log n}{(1+\epsilon)T} \leq \lambda_n\Big(\sum_i \bar{y}_i A_i - B\Big) - \frac{\delta\alpha}{R}.$$

Given the choice of parameters made in the theorem you can check that $\frac{\delta\alpha}{R} - \frac{4\sigma\log n}{\epsilon(1+\epsilon)T} > 0$, so the smallest eigenvalue of $\sum_i \bar{y}_i A_i - B$ is positive, meaning this is a PSD matrix and $\bar{y}$ is feasible, as required. □