# CS 270 Combinatorial Algorithms and Data Structures, Spring 2015

## Lecture 20: Matrix multiplicative update

The multiplicative weight update method can be generalized to a matrix setting, and this can be used to solving SDP.

---

## Matrix setting

There is a nice generalization of the multiplicative weights update method to a matrix setting, and we will see that it can be used to design faster primal-dual algorithms for solving SDPs.

The setting is as follows:

- There is a decision corresponding to each unit vector in $\mathbb{R}^n$.
- Each round, a symmetric cost matrix $M^{(t)} \in \mathbb{R}^{n \times n}$ is revealed, and the cost of each unit vector $v$ is $v^T M v$.
  We assume that $v^T M^{(t)} v \in [0,1]$ for all $v$; equivalently we assume the eigenvalues of $M^{(t)}$ are in $[0,1]$. As in the original setting, this is a bound on how big a mistake we can make, and is crucial to the convergence proof. We can write $0 \preceq M \preceq I$.
- In each round, we can choose a probability distribution over the set of unit vectors $p_v$ with $\sum_v p_v = 1$. Then, the expected cost of this probability distribution is $\sum_v p_v v^T M^{(t)} v = \sum_v M^{(t)} \bullet (p_v v v^T) = M^{(t)} \bullet \left( \sum_v p_v v v^T \right)$.
  Let $P^{(t)} = \sum_v p_v v v^T$. Since $P^{(t)}$ is a convex combination of PSD matrix of trace 1, $P^{(t)}$ is also a PSD matrix of trace 1, and such a matrix is called a density matrix. Conversely, given a density matrix $P$, we can write its eigendecomposition as $P = \sum_{i=1}^n \lambda_i v_i v_i^T$ with $\|v_i\| = 1$ and $\sum_{i=1}^n \lambda_i = tr(P) = 1$, and thus it corresponds to a probability distribution over unit vectors. Therefore, we will specify a probability distribution compactly as a density matrix.

We would like to generalize the multiplicative weight update theorem to the matrix setting: the total expected cost of the probability distributions is not much more than the total cost of a unit vector.

The main question is what would be the multiplicative update step?

In the previous setting, $w_i^{(t+1)} := \prod_{\ell=1}^{t} (1 - \eta m_i^{(\ell)}) \approx \prod_{\ell=1}^{t} (1-\eta)^{m_i^{(\ell)}} = (1-\eta)^{\sum_{\ell=1}^{t} m_i^{(\ell)}} = e^{-\eta' \sum_{\ell=1}^{t} m_i^{(\ell)}}$, where $\eta' = -\ln(1-\eta)$.

It turns out that there is an appropriate exponential function for matrix.

Given a matrix $A$, we define $e^A := \sum_{i \geq 0} \frac{A^i}{i!}$ as the matrix exponential of $A$. Some properties of $e^A$:

- Regardless of $A$, $\exp(A)$ is positive semidefinite, because $\exp(A) = \exp(\frac{1}{2}A)\exp(\frac{1}{2}A)$ (exercise).
- Suppose $A$ is real symmetric. Then $A = VDV^T$ as the eigendecomposition. Then $A^i = VD^iV^T$ since $VV^T = I$. Then $e^A = Ve^DV^T = \sum_{i=1}^{n} e^{\lambda_i} v_i v_i^T$ where $\lambda_i$ is the $i$-th eigenvalue and $v_i$ is the corresponding eigenvector. So, suppose we have an inequality that holds for all $\lambda_i$, then it also holds for the matrix. For example, if we know that $(1-\eta)^x \le 1-\eta x$ for all $x \in [0,1]$, then we can conclude that $(1-\eta)^A \preceq (I-\eta A)$ for $0 \preceq A \preceq I$ as all eigenvalues of $A$ are in $[0,1]$. To see it, first rewrite $(1-\eta)^x$ as $e^{-\eta' x}$ where $\eta' = -\ln(1-\eta)$ and $(1-\eta)^A = e^{-\eta' A}$. Then $I - \eta A - e^{-\eta' A} = VV^T - \eta VDV^T - Ve^{-\eta' D}V^T = V(I - \eta D - e^{-\eta' D})V^T$, and this is PSD iff $1 - \eta x - e^{-\eta' x} \ge 0$ holds for all eigenvalues of $A$ as $I - \eta D - e^{-\eta' D}$ is a diagonal matrix. So, the matrix exponential of a symmetric matrix behaves like the exponential of a number.

After these discussions, it is then natural to update the weight as $W^{(t+1)} = (1-\eta)^{\sum_{\ell=1}^{t} M^{(\ell)}} = \exp(-\eta' \sum_{\ell=1}^{t} M^{(\ell)})$. The formal algorithm is as follows.

---

Fix an $\eta < \frac{1}{2}$. Let $\eta' = -\ln(1-\eta)$.

For $t = 1, 2, \ldots, T$ do

1) Compute $W^{(t)} = (1-\eta)^{\sum_{\ell=1}^{t-1} M^{(\ell)}} = \exp(-\eta' \sum_{\ell=1}^{t-1} M^{(\ell)})$.
2) Set $P^{(t)} = \frac{W^{(t)}}{\mathrm{Tr}(W^{(t)})}$ and observe the event $M^{(t)}$.

---

The scaling is to make sure that $P^{(t)}$ is a density matrix.

<u>Theorem</u> The matrix multiplicative weights update algorithm will produce density matrices such that $\sum_{t=1}^{T} M^{(t)} \bullet P^{(t)} \le (1+\eta) v^T (\sum_{t=1}^{T} M^{(t)}) v + \frac{\ln n}{\eta}$ for all unit vector $v \in \mathbb{R}^n$.

Equivalently, $\sum_{t=1}^{T} M^{(t)} \bullet P^{(t)} \le (1+\eta)\lambda_{\min}(\sum_{t=1}^{T} M^{(t)}) + \frac{\ln n}{\eta}$. We assume $0 \preceq M^{(t)} \preceq I$ for all $t$.

<u>Proof</u> In the previous setting, we use $\Phi^{(t)} = \sum_i w_i^{(t)}$ as a potential function.

In the matrix setting, we use $\Phi^{(t)} = \mathrm{Tr}(W^{(t)})$ as the potential function.

$$\begin{aligned}
\mathrm{Tr}(W^{(t+1)}) &= \mathrm{Tr}\left(\exp\left(-\eta' \sum_{\ell=1}^{t} M^{(\ell)}\right)\right) \\
&\le \mathrm{Tr}\left(\exp\left(-\eta' \sum_{\ell=1}^{t-1} M^{(\ell)}\right)\exp(-\eta' M^{(t)})\right) && \left(\text{Golden-Thompson inequality: } \mathrm{Tr}(\exp(A+B)) \le \mathrm{Tr}(\exp(A)\exp(B))\right) \\
&= W^{(t)} \bullet \exp(-\eta' M^{(t)}) && (\mathrm{Tr}(AB) = A \bullet B \text{ for symmetric } A) \\
&\le W^{(t)} \bullet (I - \eta M^{(t)}) && ((1-\eta)^A \preceq (I - \eta A) \text{ if } 0 \preceq A \preceq I \text{ as explained above})
\end{aligned}$$

$= W \bullet \exp(-\eta M^{(t)})$ ( $\mathrm{Tr}(AB) = A \bullet B$ for symmetric $A$ )

$\leq W^{(t)} \bullet (I - \eta M^{(t)})$ ( $(1-\eta)^A \preceq (I - \eta A)$ if $0 \preceq A \preceq I$ as explained above )

$= \mathrm{Tr}(W^{(t)})(1 - \eta M^{(t)} \bullet P^{(t)})$ ( use $W^{(t)} = P^{(t)} \cdot \mathrm{Tr}(W^{(t)})$ )

$\leq \mathrm{Tr}(W^{(t)}) e^{-\eta M^{(t)} \bullet P^{(t)}}$

By induction, since $\mathrm{Tr}(W^{(1)}) = \mathrm{Tr}(I) = n$, this implies that $\mathrm{Tr}(W^{T+1}) \leq n \exp\left(-\eta \sum_{t=1}^{T} M^{(t)} \bullet P^{(t)}\right)$.

On the other hand, we have $\mathrm{Tr}(W^{(T+1)}) = \mathrm{Tr}\left(\exp\left(-\eta' \sum_{t=1}^{T} M^{(t)}\right)\right) \geq \exp\left(-\eta' \lambda_{\min}\left(\sum_{t=1}^{T} M^{(t)}\right)\right)$,

because $\mathrm{Tr}(e^A) = \mathrm{Tr}(V e^D V^T) = \mathrm{Tr}(e^D) = \sum_{i=1}^{n} e^{\lambda_i(A)} \geq e^{\lambda_{\min}(A)}$.

Therefore, we have $\exp\left(-\eta' \lambda_{\min}\left(\sum_{t=1}^{T} M^{(t)}\right)\right) \leq n \exp\left(-\eta \sum_{t=1}^{T} M^{(t)} \bullet P^{(t)}\right)$

$\Rightarrow \eta \sum_{t=1}^{T} M^{(t)} \bullet P^{(t)} \leq \ln\left(\frac{1}{1-\eta}\right) \lambda_{\min}\left(\sum_{t=1}^{T} M^{(t)}\right) + \ln n$

$\Rightarrow \eta \sum_{t=1}^{T} M^{(t)} \bullet P^{(t)} \leq (\eta + \eta^2) \lambda_{\min}\left(\sum_{t=1}^{T} M^{(t)}\right) + \ln n$ (using $\ln\left(\frac{1}{1-\eta}\right) \leq \eta + \eta^2$ for $\eta < \frac{1}{2}$)

$\Rightarrow \sum_{t=1}^{T} M^{(t)} \bullet P^{(t)} \leq (1+\eta) \lambda_{\min}\left(\sum_{t=1}^{T} M^{(t)}\right) + \frac{\ln n}{\eta}$. □

---

## Primal-dual SDP

The matrix multiplicative update method can be used nicely to design a fast algorithm for solving SDP.

Consider the following primal dual pair.

(primal)
$$\max C \bullet X \quad A_i \bullet X \leq b_i \text{ for } 1 \leq i \leq m \quad \mathrm{Tr}(X) \leq 1 \quad X \succeq 0$$

$\Leftrightarrow$

$$\max C \bullet X \quad A_i \bullet X \leq b_i \text{ for } 0 \leq i \leq m \quad (\text{where } A_0 = I, b_0 = 1) \quad X \succeq 0$$

(dual)
$$\min b^T y \quad \sum_{i=0}^{m} y_i A_i - C \succeq 0 \quad y_i \geq 0 \text{ for } 0 \leq i \leq m$$

Focus on solving the dual program. Reduce it to a decision problem.

$$\sum_{i=0}^{m} y_i A_i - C \succeq 0 \quad b^T y \leq \alpha \quad y \geq 0$$

$\Leftrightarrow$

$$\left(\sum_{i=0}^{m} y_i A_i - C\right) \bullet X \geq 0 \text{ for all } X \succeq 0 \quad b^T y \leq \alpha \quad y \geq 0.$$

As in the previous approach, we identify some constraints as hard and use multiplicative update to combine them, while some constraints as easy and directly implement an oracle to solve them.

In this case, the easy constraints are $\{b^T y \leq \alpha, y \geq 0\}$ and the hard constraints as the PSD constraint.

To satisfy the hard constraint, we need to find $y$ that satisfies all possible density matrices $X$.

Equivalently, we need to find $y$ that satisfies $v^T\left(\sum_{i=0}^{m} y_i A_i - C\right)v \geq 0$ for all unit vectors $v \in \mathbb{R}^n$.

There are infinitely many unit vectors to consider. It seems too complicated.

But this is exactly what the matrix multiplicative update algorithm does!

Define the cost matrix $M^{(t)} = \sum_{i=0}^{m} y_i A_i - C$. (It does not satisfy $0 \preceq M^{(t)} \preceq I$. We will fix it.)

Then the cost of a unit vector $v$ is $\left(\sum_{i=0}^{m} y_i A_i - C\right) \bullet (vv^T)$. It is smaller if its constraint is violated more.

The algorithm will generate density matrices $P^{(1)}, P^{(2)}, \ldots, P^{(T)}$ such that if we can satisfy them, then we almost satisfy all unit vectors.

So, in each round, we ask the multiplicative algorithm to provide a density matrix $P^{(t)}$ to us.

Then, we find $y^{(t)}$ such that $\left(\sum_{i=0}^{m} y_i^{(t)} A_i - C\right) \bullet P^{(t)} \geq 0$, and return $M^{(t)} = \sum_{i=0}^{m} y_i^{(t)} A_i - C$ as the cost matrix to the multiplicative update algorithm as a feedback (to tell them what unit vectors are satisfied and what unit vectors are violated).

Then, the multiplicative algorithm will cleverly update its density matrix (to put more weight on the constraints that are violated more), and send us another matrix $P^{(t+1)}$, and the procedure is repeated.

Now, if we can find $y^{(1)}, y^{(2)}, \ldots, y^{(T)}$ that satisfy all the density matrices $P^{(1)}, P^{(2)}, \ldots, P^{(T)}$, by the matrix multiplicative update theorem above, they will almost satisfy all the unit vectors, and thus $\bar{y} = \frac{1}{T}\sum_{t=1}^{T} y^{(t)}$ is an almost feasible dual solution.

This is the idea of this approach, very similar to the previous setting, just that different constraints are considered hard.

Now, we formally describe the algorithm. First, we specify the oracle.

ORACLE: Given $X$, return $y \in \mathbb{R}^{m+1}$ such that $\left(\sum_{i=0}^{m} y_i A_i - C\right) \bullet X \geq 0$ and $b^T y \leq \alpha$ and $y \geq 0$.

Note that, given $X$, the first constraint is just a linear constraint $\sum_{i=0}^{m} y_i (A_i \bullet X) \geq C \bullet X$

So, the oracle just needs to solve a linear program with two non-trivial constraints.

As it turned out, for many combinatorial problems, the oracle can be implemented by combinatorial techniques.

Clearly, if there is a dual feasible solution with value $\leq \alpha$, then this LP is solvable.

Indeed, if this LP is not solvable, then one can prove by LP duality that the solution $X$ is primal feasible with value at least $\alpha$ (exercise).

Width : We assume the oracle always returns $y$ with $\| \sum_{i=0}^{m} y_i A_i - C \| \le w$. Again, this is to bound the maximum violation of a unit vector, and this parameter is crucial in bounding the convergence rate.

---

ALGORITHM

Set $P^{(1)} = I/n$.

For $t = 1, 2, \ldots, T$ do

1) Run the oracle with $P^{(t)}$. If the oracle fails, return $P^{(t)}$ as a primal feasible solution with value $\ge \alpha$. Otherwise, let $y^{(t)}$ be the solution that the oracle returns.
2) Set $M^{(t)} = \frac{1}{2w}\left( \sum_{i=0}^{m} y_i^{(t)} A_i - C + wI \right)$. This guarantees that $0 \le M^{(t)} \le I$ by our assumption on the width.
3) Use $M^{(t)}$ as the outcome to the matrix multiplicative algorithm, and use its output as $P^{(t+1)}$.

---

All the ideas are discussed. The following details are just formal execution of the ideas.

Analysis: By the matrix multiplicative update theorem, we have

$$\sum_{t=1}^{T} M^{(t)} \cdot P^{(t)} \le (1+\eta)\, \lambda_{\min}\left( \sum_{t=1}^{T} M^{(t)} \right) + \frac{\ln n}{\eta}$$ , where $\eta$ is the parameter used in MMU algo.

Assuming the oracle never fails, then $M^{(t)} \cdot P^{(t)} = \frac{1}{2w}\left( \sum_{i=0}^{m} y_i^{(t)} A_i - C \right) \cdot P^{(t)} + \frac{1}{2} I \cdot P^{(t)} \ge \frac{1}{2}$,

since the first term $\ge 0$ if the oracle succeeds and the second term $= \frac{1}{2}$ as $P^{(t)}$ is a density matrix.

Therefore, the LHS of the inequality is at least $\frac{T}{2}$, and thus we have

$$\frac{T}{2} - \frac{\ln n}{\eta} \le (1+\eta)\, \lambda_{\min}\left( \sum_{t=1}^{T} \frac{1}{2w}\left( \sum_{i=0}^{m} y_i^{(t)} A_i - C + wI \right) \right)$$

$$\Rightarrow \quad \lambda_{\min}\left( \sum_{t=1}^{T} \left( \sum_{i=0}^{m} y_i^{(t)} A_i - C + wI \right) \right) \ge \frac{2w}{(1+\eta)} \left( \frac{T}{2} - \frac{\ln n}{\eta} \right)$$

$$\Rightarrow \quad \lambda_{\min}\left( \sum_{i=0}^{m} \left( \frac{\sum_{t=1}^{T} y_i^{(t)}}{T} \right) A_i - C \right) \ge \frac{-\eta w}{1+\eta} - \frac{2w \ln n}{(1+\eta)\eta T} \ge -\eta w - \frac{2w \ln n}{\eta T}$$

For an error parameter $\delta > 0$, set $\eta = \frac{\delta \alpha}{2w}$ and then set $T = \frac{8 w^2 \ln n}{\alpha^2 \delta^2}$, also let $\bar{y} = \frac{\sum_{t=1}^{T} y_i^{(t)}}{T}$,

we have $\lambda_{\min}\left( \sum_{i=0}^{m} \bar{y}_i A_i - C \right) \ge -\delta \alpha$.

Finally, recall that $A_0 = I$ and $b_0 = 1$ by the trace constraint.

Set $y^* = \bar{y} + \alpha\delta e_0$. Then, we have $\sum_{i=0}^{m} y_i^* A_i - C \succeq 0$, and is a dual feasible solution.

The value of $y^*$ is $b^T y^* = b^T \bar{y} + b^T \alpha\delta e_0 = b^T \left( \frac{\sum_{t=1}^{T} y^{(t)}}{T} \right) + \delta\alpha \leq \alpha + \delta\alpha = (1+\delta)\alpha$,

while the inequality holds because $b^T y^{(t)} \leq \alpha$ by the definition of the oracle.

To conclude, we have the following theorem.

Theorem In $O\left(\frac{w^2 \ln n}{\alpha^2 \delta^2}\right)$ iterations, the algorithm returns a dual feasible solution with value at most $(1+\delta)\alpha$, or returns a primal feasible solution with value at least $\alpha$.

Remark: If $\mathrm{tr}(X) \leq R$ in the primal program, by the same scaling argument we've seen before, we can solve it with an $O(R^2)$ blowup in the number of iterations.

The advantage of this method over the previous SDP solver is that there are no violations of the constraints, the violations are only in the objective function. This avoids further modification of the solution which may significantly change the objective value.

This approach is used successfully to get an $\tilde{O}(m)$ time algorithm to solve the max-cut SDP for regular graphs. We won't discuss this result.

The matrix multiplicative update algorithm has applications in machine learning problems, and is recently used to prove an important result in quantum computing (QIP=PSPACE).

---

## References


The material is taken from the survey paper "The multiplicative weights update method: a meta-algorithm and applications" by Arora, Hazan and Kale, and the paper "A combinatorial, primal-dual approach to semidefinite programs" by Arora and Kale.

For the proof of the Golden-Thompson inequality, you are referred to the blog post of Terrence Tao or the paper "Golden-Thompson from Davis" by Rivin.