ORIE 6334 Bridging Continuous and Discrete Optimization Nov. 18, 2019

Lecture 19

Lecturer: David P. Williamson *Scribe: Andrew Morgan*

We will finish our discussion of algorithms for spectral sparsifiers over the next two lectures. In this lecture, we cover a matrix-based analogue of the Multiplicative Weights Update algorithm (introduced in Lecture 3), as well as the matrix version of the related algorithm for solving "packing" problems (also introduced in Lecture 3). The latter will, in fact, directly provide us with a *deterministic* algorithm for producing a spectral sparsifier of a given graph on $O(n\ln(n)/\epsilon^2)$ edges (though the version we prove gives only $O(n\ln(n)/\epsilon^3)$), in contrast to the randomized one (also with $O(n\ln(n)/\epsilon^2)$ edges) given in the previous lecture.

Next lecture, the final lecture on spectral sparsifiers, will demonstrate a deterministic algorithm which will produce a *linear-sized* sparsifier—that is, one with $O(n/\epsilon^2)$ edges.

1 The Matrix Multiplicative Weights Update Algorithm

In the previous lecture, we gave a generalization of the well-known Chernoff bound to the case of symmetric matrices. Here, we will begin by providing a similar treatment of the Multiplicative Weights Update algorithm.

Recall that the (scalar version of the) Multiplicative Weights Update algorithm worked over time steps $t = 1, 2, \ldots, T$, and at each time step would choose some decision $i \in [n]$ according to internal weights, receive a value $v_t(i) \in [0, 1]$, and update the internal weights according to the observed values for each decision. The key feature of this algorithm is that it can achieve "almost" as much total value over the T time steps as the maximum attainable by any fixed decision i.

For the *matrix* version of the algorithm, we will instead let our decision space consist of vectors $u \in \mathbb{R}^n$ with $||u|| = 1$. The algorithm will, at each time step $t \in [T]$, choose such a vector u_t, observe a matrix $M_t \in \mathbb{R}^n$ with $0 \preccurlyeq M_t \preccurlyeq I$, and receive value $u^T M_t u \in [0, 1]$.

Notice in particular that, were M_t diagonal with entries $M_t(i, i) = v_t(i)$, this would in fact reduce to the scalar Multiplicative Weights Update algorithm, since u_t could be considered as a probability distribution over the choices $i \in [n]$. So the matrix version is in fact a strict generalization of the scalar version.

As in the scalar version, the algorithm will make a randomized decision based on weights; this time, the weights at time step t will be defined by a matrix $W_t \in \mathbb{R}^{n\times n}$ with $W_t \succcurlyeq 0$. If we define $P_t = W_t/\mathrm{tr}(W_t)$, then this gives a matrix $P_t \in \mathbb{R}^{n\times n}$ with $P_t \succcurlyeq 0$ and $\mathrm{tr}(P_t) = 1$. Letting λ_{it}, x_{it} be the respective eigenvalues and corresponding eigenvectors of P_t, then, we notice that $P_t = \sum_{i=1}^n \lambda_{it} x_{it} x_{it}^T$, where $\lambda_{it} \geq 0$ and $\sum_{i=1}^n \lambda_{it} = 1$. That is, the eigenvalues λ_{it} can be seen as defining a *probability distribution* over the eigenvectors, such that we will

[0]This lecture is drawn from Arora and Kale 2016 `http://dl.acm.org/citation.cfm?doid=2837020`; Kale's thesis `http://www.satyenkale.com/papers/thesis.pdf`; and de Carli Silva, Harvey, and Sato 2015 `https://www.cs.ubc.ca/~nickhar/Publications/SparsifierMMWUM/SparsifierMMWUM.pdf`.

pick x_{it} with probability λ_{it} at time step t. We present the formal algorithm; see Algorithm 1.

Algorithm 1: Matrix Multiplicative Weights Update

$W_t \leftarrow I$
for $t \leftarrow 1$ **to** T **do**
  $P_t \leftarrow \frac{W_t}{tr(W_t)}$
  Make decision $u_t = x_{it}$ with prob. λ_{it} for x_{it}, λ_{it} eigenvectors/eigenvalues of P_t
  Get value $u_t^T M_t u_t$
  $W_{t+1} \leftarrow \exp^*(\epsilon \sum_{k=1}^{T} M_k)$.
end

To analyze this algorithm, we first introduce some new notation:

- Let $A \bullet B = \sum_{i,j} a_{ij} b_{ij}$ denote the "matrix inner product" of two matrices $A = (a_{ij})$ and $B = (b_{ij})$.
- Let $\exp^*(A)$ denote the "matrix exponential" of a matrix given by the Taylor series of the exponential:
$$\exp^*(A) = I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \dots$$
Defining the matrix exponential separately is an important distinction, since the matrix exponential does not satisfy certain properties of the scalar exponential. For instance, it is not always the case that $\exp^*(A + B) = \exp^*(A)\exp^*(B)$.

As with scalar Multiplicative Weights, we wish to show that, in expectation, the algorithm does not perform much worse than the best fixed decision $u \in \mathbb{R}^n$ (with $||u|| = 1$). First, since the algorithm chooses x_{it} with probability λ_{it}, the expected value of the algorithm is given by:

$$\sum_{t=1}^{T} \sum_{i=1}^{n} \lambda_{it} (x_{it}^T M_t x_{it}) = \sum_{t=1}^{T} \sum_{i=1}^{n} (\lambda_{it} x_{it}^T x_{it}) \bullet M_t = \sum_{t=1}^{T} P_t \bullet M_t$$

Next, consider the best fixed decision u; we can bound the value of this decision by:

$$\sum_{t=1}^{T} u^T M_t u = u^T \left(\sum_{t=1}^{T} M_t \right) u \leq \max_{u \in \mathbb{R}^n : ||u||=1} u^T \left(\sum_{t=1}^{T} M_t \right) u = \lambda_{\max} \left(\sum_{t=1}^{T} M_t \right)$$

using the observation that $\lambda_{\max}(A) = \max_{x \in \mathbb{R}^n : ||x||=1} x^T A x$. (As an aside, this implies that the best fixed decision u is in fact the eigenvector associated with the eigenvalue $\lambda_{\max}\left(\sum_{t=1}^{T} M_t\right)$.)

Hence, we set out to prove that $\sum_{t=1}^{T} P_t \bullet M_t$ is "not much less" than $\lambda_{\max}\left(\sum_{t=1}^{T} M_t\right)$. We will require the following facts about the matrix exponent and matrix inner product to conduct the analysis:

Lemma 1 (Golden-Thompson Inequality.) *For symmetric matrices A and B:*

$$\mathrm{tr}(\exp^*(A+B)) \leq \mathrm{tr}(\exp^*(A)\exp^*(B))$$

Fact 1 *For symmetric matrices A and B, $\mathrm{tr}(AB) = A \bullet B$.*

Fact 2 *$X \bullet A \leq X \bullet B$ if $A \preccurlyeq B$ and $X \succcurlyeq 0$.*

Fact 3 *$\exp^*(\epsilon A) \preccurlyeq I + (e^\epsilon - 1)A$ if $0 \preccurlyeq A \preccurlyeq I$.*

Fact 4 *$\lambda_{\max}(\exp^*(A)) = \exp(\lambda_{\max}(A))$.*

Facts 1 and 2 are straightforward from the definitions, and Fact 3 can be shown similarly to the analogous fact proven as part of Problem Set 4. Fact 4 follows from the fact that A^k (for $k > 1$) has the same eigenvectors as A and corresponding eigenvalues equal to λ^k (where λ is the corresponding eigenvalue of A); hence,

$$\lambda_{\max}(\exp^*(A)) = \lambda_{\max}(I + A + \frac{1}{2!}A^2 + \ldots) = 1 + \lambda_{\max}(A) + \frac{1}{2!}\lambda_{\max}(A)^2 + \ldots = \exp(\lambda_{\max}(A))$$

The above facts allow us to prove the following theorem:

Theorem 2 *Let $\epsilon \leq 1/2$. Then:*

$$\sum_{t=1}^{T} P_t \bullet M_t \geq \frac{1}{1+\epsilon}\lambda_{\max}\left(\sum_{t=1}^{T} M_t\right) - \frac{1}{\epsilon}\ln(n)$$

Proof: The proof proceeds analogously to that of the scalar Multiplicative Weights Update algorithm; we begin by upper-bounding and lower-bounding $\mathrm{tr}(W_{T+1})$, and combining the bounds will provide the desired conclusion.

First, consider any time step $t \in [t]$:

$$\begin{aligned}
\mathrm{tr}(W_{t+1}) &= \mathrm{tr}\left(\exp^*\left(\epsilon \sum_{k=1}^{t} M_k\right)\right) \\
&\leq \mathrm{tr}\left(\exp^*\left(\epsilon \sum_{k=1}^{t-1} M_k\right)\exp^*(\epsilon M_t)\right) && \text{(by Golden-Thompson)} \\
&= \exp^*\left(\epsilon \sum_{k=1}^{t-1} M_k\right) \bullet \exp^*(\epsilon M_t) && \text{(by Fact 1)} \\
&= W_t \bullet \exp^*(\epsilon M_t) \\
&= \mathrm{tr}(W_t) P_t \bullet \exp^*(\epsilon M_t) \\
&\leq \mathrm{tr}(W_t) P_t \bullet (I + (e^\epsilon - 1)M_t) && \text{(by Facts 2 and 3)} \\
&= \mathrm{tr}(W_t)(P_t \bullet I + (e^\epsilon - 1)P_t \bullet M_t) && \text{(by distributivity of } \bullet\text{)} \\
&= \mathrm{tr}(W_t)(1 + (e^\epsilon - 1)P_t \bullet M_t) && \text{(since } \mathrm{tr}(P_t) = 1\text{)} \\
&\leq \mathrm{tr}(W_t)\exp((e^\epsilon - 1)(P_t \bullet M_t)) && \text{(since } 1 + x \leq e^x\text{)}
\end{aligned}$$

Note that exp in the last step is scalar exponentiation, rather than the matrix exponent $\exp^*$. We continue by bounding W_{T+1} using a telescoping product, once again as in scalar Multiplicative Weights:

$$\mathrm{tr}(W_{T+1}) \leq \mathrm{tr}(W_T)\exp((e^\epsilon - 1)(P_T \bullet M_T)) \leq \dots$$

$$\dots \leq \mathrm{tr}(W_1)\exp\left((e^\epsilon - 1)\sum_{t=1}^{T}(P_t \bullet M_t)\right) = n\exp\left((e^\epsilon - 1)\sum_{t=1}^{T}(P_t \bullet M_t)\right)$$

Next, we prove a lower bound:

$$\mathrm{tr}(W_{T+1}) = \mathrm{tr}\left(\exp^*\left(\epsilon\sum_{t=1}^{T}M_t\right)\right) \geq \lambda_{\max}\left(\exp^*\left(\epsilon\sum_{t=1}^{T}M_t\right)\right) = \exp\left(\epsilon\lambda_{\max}\left(\sum_{t=1}^{T}M_t\right)\right)$$

which follows from the fact that $\mathrm{tr}(A) \geq \lambda_{\max}(A)$ for $A \succcurlyeq 0$ and from Fact 4. Finally, we combine the two bounds and take the natural log of both sides:

$$n\exp\left((e^\epsilon - 1)\sum_{t=1}^{T}(P_t \bullet M_t)\right) \geq \mathrm{tr}(W_{T+1}) \geq \exp\left(\epsilon\lambda_{\max}\left(\sum_{t=1}^{T}M_t\right)\right)$$

$$\ln(n) + (e^\epsilon - 1)\sum_{t=1}^{T}(P_t \bullet M_t) \geq \epsilon\lambda_{\max}\left(\sum_{t=1}^{T}M_t\right)$$

$$\sum_{t=1}^{T}(P_t \bullet M_t) \geq \frac{\epsilon}{e^\epsilon - 1}\lambda_{\max}\left(\sum_{t=1}^{T}M_t\right) - \frac{1}{e^\epsilon - 1}\ln(n) \geq \frac{1}{1+\epsilon}\lambda_{\max}\left(\sum_{t=1}^{T}M_t\right) - \frac{1}{\epsilon}\ln(n)$$

where the last step follows from $e^\epsilon - 1 \leq \epsilon(1+\epsilon)$ (for $0 \leq \epsilon \leq 1/2$) and from $e^\epsilon - 1 \geq \epsilon$. □

Remark. At one point in class, it was suggested that we might be able to prove a similar bound to the above by updating $W_{t+1} \leftarrow (I+\epsilon M_t)W_t$ rather than $W_{t+1} \leftarrow \exp^*(\epsilon\sum_{k=1}^{T}M_k)$. Whether this worked was unclear, though the upper bound appeared to hold mostly as written.

2 Packing Problems and Deterministic Sparsifiers

Now recall the application of the scalar Multiplicative Weights Update algorithm to solving "packing problems"—specifically, given $A \in \mathbb{R}^{m\times n}$ and a convex set $Q \subseteq \mathbb{R}^n$, we could use a variant of the algorithm to find $x \in Q$ such that $Ax \leq (1+\epsilon)e$, as long as we have some *oracle* that, given $p \geq 0$ in $\mathbb{R}^m$, will find $x \in Q$ such that $p^TAx \leq p^Te$ if such an x exists.

We shall now show the analogue of this application to the matrix Multiplicative Weights algorithm. Suppose we have some set of matrices $B_i \succcurlyeq 0$ for $i \in [m]$ such that $\sum_{i=1}^{m}B_i = I$. We would like to obtain some "sparse" vector $y \geq 0$ in $\mathbb{R}^m$ such that $(1-\epsilon)I \preccurlyeq \sum_{i=1}^{m}y(i)B_i \preccurlyeq (1+\epsilon)I$, where by "sparse" we mean that the number of non-zero entries of y is very small (specifically, we will attain $O(n\ln(n)/\epsilon^3)$ non-zeroes).

To do this, we will need an oracle analogous to that of the scalar version, but with some added slack ϵ and an additional lower bound: specifically, given some "probability matrices" $P, \tilde{P} \succcurlyeq 0$ such that $\text{tr}(P) = \text{tr}(\tilde{P}) = 1$, the oracle must return a y such that $y(i) = \alpha > 0$ for exactly one $i \in [m]$ (that is, y has exactly *one* non-zero), and furthermore such that $\alpha P \bullet B_i \leq (1+\epsilon) P \bullet I = 1 + \epsilon$ and $\alpha \tilde{P} \bullet B_i \geq (1-\epsilon)\tilde{P} \bullet I = 1 - \epsilon$.

We also require a notion of the *width* of the oracle; we define this as

$$\rho = \max_{i \in [m]} \max_{y \in Y} \alpha \text{tr}(B_i)$$

over the set Y of y that can be returned by the oracle. (Recall that the analogous width for the scalar oracle was $\max_{i \in [m]} \max_{x \in X} (Ax)(i)$ over X returnable by the oracle; notice that the matrix version gives $\frac{1}{\rho} \sum_{i=1}^{m} y(i) B_i \preccurlyeq I$, whereas the scalar version gives $\frac{1}{\rho} Ax \leq e$.)

Before presenting and proving the algorithm, let's see how we can apply this algorithm to finding spectral sparsifiers. We can define one matrix for every edge of the graph:

$$B_{(i,j)} = L_G^{\dagger/2} (e_i - e_j)(e_i - e_j)^T L_G^{\dagger/2}$$

Then:

$$\sum_{(i,j) \in E} B_{(i,j)} = \sum_{(i,j) \in E} L_G^{\dagger/2} (e_i - e_j)(e_i - e_j)^T L_G^{\dagger/2} = L_G^{\dagger/2} \left(\sum_{(i,j) \in E} (e_i - e_j)(e_i - e_j)^T \right) L_G^{\dagger/2}$$

$$= L_G^{\dagger/2} L_G L_G^{\dagger/2} = I^*$$

where, as before, I^* is the identity with respect to vectors orthogonal to e. If we run the algorithm, we obtain a sparse vector y such that:

$$(1-\epsilon) I \preccurlyeq \sum_{(i,j) \in E} y(i,j) B_{i,j} \preccurlyeq (1+\epsilon) I$$

$$(1-\epsilon) I \preccurlyeq L_G^{\dagger/2} \left(\sum_{(i,j) \in E} y(i,j)(e_i - e_j)(e_i - e_j)^T \right) L_G^{\dagger/2} \preccurlyeq (1+\epsilon) I$$

$$(1-\epsilon) I \preccurlyeq L_G^{\dagger/2} L_H L_G^{\dagger/2} \preccurlyeq (1+\epsilon) I$$

$$(1-\epsilon) L_G \preccurlyeq L_H \preccurlyeq (1+\epsilon) L_G$$

where H is a weighted subgraph of G with weights $y(i,j)$ for each edge. Specifically, this means that H will be a spectral sparsifier (on $O(n \ln(n)/\epsilon^3)$ edges, though this can be improved to $O(n \ln(n)/\epsilon^2)$ by adjusting the algorithm); furthermore, the following algorithm will be deterministic, and so we will have constructed a sparsifier deterministically. We next present and prove the algorithm; see Algorithm 2.

To explain why this works, first notice that $\frac{1}{\rho} \sum_{i=1}^{m} y_t(i) B_i$ effectively plays the role of M_t in the matrix Multiplicative Weights Update algorithm, since

$$\text{tr}(B_i) \alpha_t \leq \rho \implies 0 \preceq \frac{1}{\rho} \sum_{i=1}^{m} y_t(i) B_i \preceq I$$

Algorithm 2: Algorithm for Feasibility

$W_1 \leftarrow I, \tilde{W}_1 \leftarrow I$
for $t \leftarrow 1$ **to** T **do**
  $P_t \leftarrow \frac{W_t}{\text{tr}(W_t)}, \tilde{P}_t \leftarrow \frac{\tilde{W}_t}{\text{tr}(\tilde{W}_t)}$
  Run oracle to find y_t s.t. $\exists$ only one i s.t. $y_t(i) = \alpha_t > 0$, $\alpha_t P_t \bullet B_{it} \leq (1+\epsilon)$, $\alpha \tilde{P}_t \bullet B_{it} \geq (1-\epsilon)$
  $W_t \leftarrow \exp^*(\frac{\epsilon}{\rho} \sum_{k=1}^{t} \sum_{i=1}^{m} y_k(i) B_i)$
  $\tilde{W}_t \leftarrow \exp^*(-\frac{\epsilon}{\rho} \sum_{k=1}^{t} \sum_{i=1}^{m} y_k(i) B_i)$
end
return $\bar{y} = \frac{1}{T} \sum_{t=1}^{T} y_t$

Also, we have:

$$\sum_{t=1}^{T} P_t \bullet \left(\frac{1}{\rho} \sum_{i=1}^{m} y_t(i) B_i \right) = \frac{1}{\rho} \sum_{t=1}^{T} \alpha_t B_{it} \bullet P_t \leq (1+\epsilon) \frac{T}{\rho}$$

So, by Theorem 2 (the matrix Multiplicative Weights Update algorithm):

$$\begin{aligned}
(1+\epsilon)\frac{T}{\rho} &\geq \sum_{t=1}^{T} P_t \bullet \left(\frac{1}{\rho} \sum_{i=1}^{m} y_t(i) B_i \right) \\
&\geq \frac{1}{1+\epsilon} \lambda_{\max} \left(\frac{1}{\rho} \sum_{i=1}^{m} y_t(i) B_i \right) - \frac{1}{\epsilon} \ln(n) \\
&= \frac{1}{1+\epsilon} \frac{T}{\rho} \lambda_{\max} \left(\sum_{i=1}^{m} \bar{y}(i) B_i \right) - \frac{1}{\epsilon} \ln(n)
\end{aligned}$$

Rearranging terms gives:

$$\lambda_{\max} \left(\sum_{i=1}^{m} \bar{y}(i) B_i \right) \leq (1+\epsilon)^2 + \frac{(1+\epsilon)\rho}{\epsilon T} \ln(n)$$

So, if we set $T = \frac{(1+\epsilon)\rho}{\epsilon^2} \ln(n)$, this becomes:

$$\lambda_{\max} \left(\sum_{i=1}^{m} \bar{y}(i) B_i \right) \leq (1+\epsilon)^2 + \epsilon \leq 1 + 4\epsilon$$

We can symmetrically show (using $\tilde{P}_t$) that

$$\lambda_{\min} \left(\sum_{i=1}^{m} \bar{y}(i) B_i \right) \geq 1 - 4\epsilon$$

(where we let $\lambda_{\min}$ denote the minimum non-zero eigenvalue), which allows us to conclude that

$$(1-4\epsilon)I \preccurlyeq \sum_{i=1}^{m} \bar{y}(i)B_i \preccurlyeq (1+4\epsilon)I$$

Furthermore, since each y_t has one non-zero entry, the number of non-zeroes in $\bar{y}$ is at most the number of iterations, $T = \frac{(1+\epsilon)\rho}{\epsilon^2}\ln(n)$. By the following lemma, we can conclude that $\bar{y}$ has $O(n\ln(n)/\epsilon^3)$ non-zeroes, though we note that the algorithm can be modified to improve this to $O(n\ln(n)/\epsilon^2)$.

Lemma 3 *There is an oracle with width $\rho = O\left(\frac{(1+\epsilon)n}{\epsilon}\right)$.*

Proof: Recall that the oracle needs to find i and α such that $\alpha P \bullet B_i \leq 1+\epsilon$, $\alpha\tilde{P} \bullet B_i \geq 1-\epsilon$, and $\alpha\text{tr}(B_i) \leq \rho = (1+\epsilon)n/\epsilon$. Define $\tilde{p}_i = B_i \bullet \tilde{P}$. Then $\tilde{p}_i \geq 0$ since $\tilde{P} \succeq 0$ and $B_i \succeq 0$. Also

$$\sum_{i=1}^{n} \tilde{p}_i = \tilde{P} \bullet \left(\sum_{i=1}^{n} B_i\right) = \tilde{P} \bullet I = \text{tr}(\tilde{P}) = 1.$$

So $\tilde{p}_i$ is a probability distribution. Then

$$E_i\left[\frac{\text{tr}(B_i)}{\tilde{p}_i}\right] = \sum_{i=1}^{m} \text{tr}(B_i) = \text{tr}(I) = n,$$

so that

$$\Pr\left[\frac{\text{tr}(B_i)}{\tilde{p}_i} \leq \frac{(1+\epsilon)n}{\epsilon}\right] = 1 - \Pr\left[\frac{\text{tr}(B_i)}{\tilde{p}_i} > \frac{(1+\epsilon)n}{\epsilon}\right] > 1 - \frac{\epsilon}{1+\epsilon} = \frac{1}{1+\epsilon},$$

by Markov's inequality. Similarly,

$$E_i\left[\frac{P \bullet B_i}{\tilde{p}_i}\right] = \sum_{i=1}^{m} P \bullet B_i = P \bullet I = \text{tr}(P) = 1,$$

so that

$$\Pr\left[\frac{P \bullet B_i}{\tilde{p}_i} \leq 1+\epsilon\right] = 1 - \Pr\left[\frac{P \bullet B_i}{\tilde{p}_i} > 1+\epsilon\right] > 1 - \frac{1}{1+\epsilon},$$

again by Markov's inequality. So there must exist an index i such that both

$$\frac{P \bullet B_i}{\tilde{p}_i} \leq 1+\epsilon \text{ and } \frac{\text{tr}(B_i)}{\tilde{p}_i} \leq \frac{(1+\epsilon)n}{\epsilon} \equiv \rho.$$

Thus if we set $\alpha = 1/\tilde{p}_i$, we get that $\alpha P \bullet B_i \leq 1+\epsilon$, $\alpha\text{tr}(B_i) \leq \rho$, and

$$\alpha\tilde{P} \bullet B_i = \frac{1}{\tilde{p}_i}\tilde{P} \bullet B_i = 1 \geq 1-\epsilon,$$

where the final equation follows by the definition of $\tilde{p}_i$. □