Optimization II **Winter 2009/10**

Lecture 5 — November 9

Lecturer: Khaled Elbassioni

5.1 The multiplicative weights update method

In this lecture we present a procedure which can be thought of as a derandomization of the randomized fictitious play described in the previous lecture. Freund and Schapire [FS99] used this method, which was originally developed by Littlestone and Warmuth [LW94] from the machine learning community, to give a procedure for computing ε-saddle points for matrix games. A number of similar algorithms have also been developed for approximately solving special optimization problems, such as general linear programs [PST91], multicommodity flows problems [GK98], packing and covering linear programs [PST91, GK98, GK04, KY07, You01], some class of convex programs [Kha04], and semidefinite programs [AHK05, AK07]. Arora, Hazan and Kale [AHK06] gave a meta algorithm that puts many of these results under one umbrella. We will partly follow their presentation in this lecture.

5.1.1 The experts problem

Suppose that we are dealing with stocks. We have access to the advice of n experts: at the start of the day, each expert informs us of his/her prediction of the increase or decrease of the type of stock we are interested in. We would like to devise a procedure which, if we follow it, would guarantee that we do not make many more mistakes than the best expert. Is there such a procedure at all? Here is one.

Algorithm 1 RANDOMIZED WEIGHTED MAJORITY

```
1. $m_i(0) := 0$ for all $i \in [n]$
2. for $t = 1, 2, \ldots$ do
3.   At the beginning of day $t$:
4.     Pick an expert $i \in [n]$ with probability $\frac{p_i(t)}{|p(t)|}$, where $p_i(t) = e^{-\frac{\varepsilon m_i(t-1)}{2}}$
       and follow her advice
5.   At the end of day $t$:
6.     $m_i(t) := m_i(t-1) + 1$ for every expert $i$ who has made a mistake
```

In the above procedure, $m_i(t)$ stands for the number of mistakes made by expert i upto time (or day) t. The algorithm can be derandomized by thinking of $\{p_i(t)\}_i$ as weights, and replacing the random choice in step 4 by following the advice of the weighted majority (i.e., if experts of weight at least $\frac{\sum_i p_i(t)}{2}$ say the stock will increase, then follow them); if an expert makes a mistake, decrease his weight by a factor of $e^{-\frac{\varepsilon}{2}}$ (or $(1-\varepsilon)$); hence the name "multiplicative weights update".

We will show now that this intuitive procedure, if run for enough time, does actually what we want.

Theorem 5.1. *Let $m(t)$ be the number of mistakes made by procedure* RANDOMIZED WEIGHTED MAJORITY *upto time t. Then for $t \geq \frac{3\ln(n)}{\varepsilon^2}$, we have*

$$\mathbb{E}[\frac{m(t)}{t}] \leq \frac{m_i(t)}{t} + \varepsilon, \text{ for all } i \in [n]. \tag{5.1}$$

Proof: As in the case of matrix games, we analyze the change in the potential function $\Phi(t) \overset{\text{def}}{=} |p(t+1)| = \sum_{i=1}^{n} p_i(t+1)$, from one iteration to the next. Let

$$I(t) = \begin{cases} 1 & \text{if the procedure makes a mistake at time } t \\ 0 & \text{otherwise.} \end{cases}$$

Let also $S(t) = \{i \in [n] \ : \ \text{expert } i \text{ makes a mistake at time } t\}$. Then $\mathbb{E}[I(t)] = \sum_{i \in S(t)} \frac{p_i(t)}{|p(t)|}$ (since the procedure makes a mistake at time t if it picks an expert who makes a mistake at time t), and by linearity of expectation, $\mathbb{E}[m(t)] = \sum_{t'=1}^{t} \mathbb{E}[I(t')]$. Now we can write

$$\begin{aligned}
\Phi(t) &= \sum_{i\in[n]} p_i(t+1) = \sum_{i\in[n]} e^{-\frac{\varepsilon m_i(t)}{2}} = \sum_{i\in S(t)} e^{-\frac{\varepsilon(m_i(t-1)+1)}{2}} + \sum_{i\notin S(t)} e^{-\frac{\varepsilon m_i(t-1)}{2}} \\
&= \sum_{i\in S(t)} p_i(t) e^{-\frac{\varepsilon}{2}} + \sum_{i\notin S(t)} p_i(t) \\
&\leq (1 - \frac{\varepsilon}{2} + \frac{\varepsilon^2}{6}) \sum_{i\in S(t)} p_i(t) + \sum_{i\notin S(t)} p_i(t) \quad \text{(By Exercise 4 of Lecture 2)} \\
&\leq (1 + \frac{\varepsilon^2}{6}) \sum_{i\in[n]} p_i(t) - \frac{\varepsilon}{2} \sum_{i\in S(t)} p_i(t) = |p(t)| \left(1 + \frac{\varepsilon^2}{6} - \frac{\varepsilon}{2}\mathbb{E}[I(t)]\right) \\
&\leq |p(t)| e^{\frac{\varepsilon^2}{6} - \frac{\varepsilon}{2}\mathbb{E}[I(t)]} \quad \text{(Using the inequality } 1 + x \leq e^x\text{).}
\end{aligned}$$

Thus $\Phi(t) \leq \Phi(t-1) e^{\frac{\varepsilon^2}{6} - \frac{\varepsilon}{2}\mathbb{E}[I(t)]}$, and iterating we get that $\Phi(t) \leq \Phi(0) e^{\frac{\varepsilon^2}{6}t - \frac{\varepsilon}{2}\sum_{t'=1}^{t}\mathbb{E}[I(t')]} = \Phi(0) e^{\frac{\varepsilon^2}{6}t - \frac{\varepsilon}{2}\mathbb{E}[m(t)]}$.

Now we use $\Phi(0) = n$ and $\Phi(t) \geq p_i(t+1)$ for all i to conclude that

$$e^{-\frac{\varepsilon}{2}m_i(t)} \leq n e^{\frac{\varepsilon^2}{6}t - \frac{\varepsilon}{2}\mathbb{E}[m(t)]} \text{ for all } i \in [n],$$

and by taking logs, dividing by $\frac{\varepsilon t}{2}$, and rearranging

$$\mathbb{E}[\frac{m(t)}{t}] \leq \frac{2\ln n}{\varepsilon t} + \frac{\varepsilon}{3} + \frac{m_i(t)}{t} \text{ for all } i \in [n].$$

The theorem follows from the last inequality if we set $t \geq \frac{3\ln(n)}{\varepsilon^2}$. □

5.1.2 A generalization

Consider now a more general scenario where we have a payoff matrix $M \in \mathbb{R}^{[n]\times\mathcal{P}}$, where $\mathcal{P}$ is a (possibly infinite) set of outcomes. If we follow the advice of expert $i \in [n]$ at the beginning of the day, and the output at the end of the day turns out to be $x \in \mathcal{P}$, then we pay $M(i,x)$. Again, our objective is to devise a procedure that makes us pay not much more that what the best expert would pay.

The procedure is almost the same as in the previous section. The difference now is that $m_i(t)$ stands for the payment expert i would pay for her decisions upto time t, that is, $m_i(t) = \sum_{t'=1}^{t} M(i, x(t'))$, where $x(t')$ is the output that materializes at time t'. Since the entries of the matrix M are arbitrary, we have now to do some scaling by the *width parameter* $\rho \overset{\text{def}}{=} \max\{\max_{i\in[n],x\in\mathcal{P}} |M(i,x)|, 1\}$. Precisely, we set $p_i(t) = e^{-\frac{\varepsilon m_i(t-1)}{2\rho^2}}$, and change the update step in 6 to $m_i(t) := m_i(t-1) + M(i, x(t))$ for every expert $i \in [n]$.

The analysis can be carried out almost word-by-word as before. We leave it as an exercise.

Exercise 1. *Consider the generalized expert problem.*

(i) Show how to describe the original (simpler) expert problem as a special case of this generalized version above.

(ii) Show that for the generalized strategy given above, the expected payoff after $t = \frac{3\rho^2 \ln n}{\varepsilon^2}$ time steps satisfies

$$\mathbb{E}[\frac{m(t)}{t}] \le \frac{m_i(t)}{t} + \varepsilon. \tag{5.2}$$

Remark 1. *It will be important for the applications below to note that the above analysis remains valid, even if the output at time t* depends *on the weights $p_i(t)$.*

5.1.3 Application to linear programming

Consider the LP

$$z^* = \min\{c^T x : \ Ax \ge b, \ x \ge 0\}, \tag{5.3}$$

where $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n_+$, and $b \in \mathbb{R}^m$ are given matrices. In general the non-negativity constraint $x \ge 0$, can be replaced by $x \in \mathcal{P}$ for some polyhedron $\mathcal{P}$ as long as we have an *oracle* for answering the following feasibility question:

Oracle$(\mathcal{P}, w, \beta)$: Given a polyhedron $\mathcal{P} \subseteq \mathbb{R}^n$, a vector $w \in \mathbb{R}^n_+$, and a scalar $\beta > 0$, return a vector $x \in \mathcal{P}$ such that $w^T x \ge \beta$, if one exists.

An ε-approximate solution for (5.3) is an $x \in \mathbb{R}^n$ such that $c^T x = z^*$, and $A_i x \ge b_i - \varepsilon$, where A_i is the ith row of A. We will apply the framework in the previous section to approximately solving (5.3) as follows. We associate an expert i with each constraint $A_i x \ge b_i$, and let the set of outputs be

$$\mathcal{P} \overset{\text{def}}{=} \{x \in \mathbb{R}^n : \ c^T x = z^*, \ x \ge 0\}. \tag{5.4}$$

We assume that z^* is guessed correctly by binary search. The payoff matrix is defined as $M(i, x) = A_i x - b_i$ for $i \in [m]$ and $x \in \mathcal{P}$.

Remark 2. *Note that, if $\mathcal{P}$ is defined by (5.4), then given $w \in \mathbb{R}^n$ and β, Oracle$(\mathcal{P}, w, \beta)$ can be implemented by simply checking if $\max_{i\in[n]:c_i>0} \frac{w_i z^*}{c_i} \ge \beta$.*

As before let $\rho = \max\{\max_{i\in[n],x\in\mathcal{P}} |A_i x - b_i|, 1\}$ be the width. We will use the following *deterministic* version of the procedure. We assume that the LP has a feasible solution (the procedure can be modified easily to deal with the case when this is not true). Note that this assumption implies that the oracle will always return a vector in step 4.

Algorithm 2 APPROXIMATE LP SOLUTION

```
1. $m_i(0) := 0$ for all $i \in [m]$
2. for $t = 1, 2, \ldots, T \overset{\text{def}}{=} \frac{3\rho^2 \ln m}{\varepsilon^2}$ do
3.     $p_i(t) := e^{-\frac{\varepsilon m_i(t-1)}{2}}$ for $i = 1, \ldots, m$
4.     $x(t) :=$Oracle$(\mathcal{P}, p(t)^T A, p(t)^T b)$
5.     $m_i(t) := m_i(t-1) + M(i, x(t))$ for $i = 1, \ldots, m$
6. return $\bar{x}(t) = \frac{\sum_{t'=1}^{t} x(t')}{t}$
```

The following lemma states that procedure APPROXIMATE LP SOLUTION outputs an ε-approximate solution for LP (5.3).

Lemma 5.2. *After $t = \frac{3\rho^2 \ln n}{\varepsilon^2}$, $A_i\bar{x}(t) \geq b_i - \varepsilon$ for all $i \in [m]$.*

Proof: Although an independent analysis can be obtained by following exactly the same lines used for proving 5.2, it is also possible to derive the lemma directly from (5.2), by "simulating a generalized expert problem in the background", where the output at each time t is given by step 4 (and recalling Remark 1). Thus 5.2 will hold after $\frac{3\rho^2 \ln n}{\varepsilon^2}$ iterations. Note that the expected payoff we are guaranteed by the procedure at time t is (interpreting j as a random variable)

$$\mathbb{E}[M(j, x(t))] = \sum_{i=1}^{m} \frac{p_i(t)}{|p(t)|} M(i, x(t)) = \sum_{i=1}^{m} \frac{p_i(t)(A_i x(t) - b_i)}{|p(t)|} = \frac{p(t)^T A x(t) - p(t)^T b}{|p(t)|} \geq 0,$$

where the inequality follows form the fact that $x(t)$ is the output of the oracle. It follows that $\mathbb{E}[\frac{m(t)}{t}] = \frac{1}{t}\sum_{t'=1}^{t} \mathbb{E}[M(j, x(t))] \geq 0$, and hence by (5.2),

$$-\varepsilon \leq \frac{m_i(t)}{t} = \frac{\sum_{t'=1}^{t} M(i, x(t'))}{t} = \frac{\sum_{t'=1}^{t}(A_i x(t') - b_i)}{t} = A_i\bar{x}(t) - b_i,$$

for all $i \in [m]$, and the lemma follows. □

5.1.4 Application: approximating maximum multicommodity flows

As a special case of the application presented in the previous section, and to motivate the next lecture, let us consider the *maximum multicommodity flow problem*: Given a graph $G = (V, E)$, with edge capacities $c_e > 0$ for $e \in E$, and k source-sink pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ of vertices, the objective is to maximize the total flow between these pairs. The following is an LP formulation for the problem:

$$z^* = \max\{\sum_{I \in \mathcal{I}} f_I : \sum_{I \in \mathcal{I},\ e \in I} f_I \leq c_e \ \ \forall e \in E,\ f_I \geq 0 \ \ \forall I \in \mathcal{I}\},$$

where $\mathcal{I} = \cup_{i=1}^{k} \mathcal{I}_i$ is the set of paths between a source-sink pair (that is $\mathcal{I} = \cup_{i=1}^{k} \mathcal{I}_i$, where $\mathcal{I}_i$ is the set of paths from s_i to t_i), and f_I is a variable representing the flow on path I. To apply the previous framework, we define $\mathcal{P} = \{f \in \mathbb{R}^{\mathcal{I}} : \ \sum_{I \in \mathcal{I}} f_I = z^*, f_I \geq 0\}$. Note that the experts (or constraints) now correspond to edges. Given the weights $\{p_e(t)\}_{e\in E}$ at time t, the oracle in step 4 finds an $x \in \mathcal{P}$ such that

$$\sum_{e\in E} \frac{p_e(t)}{|p(t)|} \sum_{I\in\mathcal{I}:e\in I} \frac{f_I}{c_e} \leq 1,$$

which is equivalent to

$$\sum_{I\in\mathcal{I}} f_I \sum_{e\in I} \ell(e) \leq 1,$$

where $\ell(e) \overset{\text{def}}{=} \frac{p_e(t)}{|p(t)|c_e}$ can be interpreted as the "length" of edge e. By Remark 2, this reduces to finding a shortest path I with these edge lengths, and checking if $\ell(I) \leq \frac{1}{z^*}$. Then the update step 5 will essentially mean pushing a flow of z^* along this path. By taking the average of these flows, over $t = O(\frac{\rho^2 \log n}{\varepsilon^2})$ iterations, we get an optimal flow (because each one was optimal) and by Lemma 5.2, we also have $\sum_{I\in\mathcal{I}:e\in I} f_I \leq c_e + \varepsilon \leq c_e(1+\varepsilon)$, assuming that c_e is integral. Thus by scaling we get a feasible flow which is within $(1-\varepsilon)$ of the optimal flow.

The problem with this approach (as the usual one with all these techniques in general), is that the running time of the procedure is quadratic in $\rho = \max_{f,e}\{|\sum_{I:e\in I} f_I - c_e|\}$ which could be as large as $\max_{e\in E} c_e$. Thus, this is only a pseudo-polynomial time procedure. Does the special structure of the problem make it possible to fix this problem by a (slight) change of the procedure? Is this, more generally, possible for the class of LP's in which the constraints and the objective function are non-negative. These questions were answered in the affirmative by Garg and Könemann in 1998 [GK98], and will be the subject of the next two lectures.

Bibliography

[AHK05] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semide.nite programming using the multiplicative weights update method. In *FOCS*, pages 339–348, 2005.

[AHK06] Sanjeev Arora, Elad Hazan, and Satyen Kale. Multiplicative weights method: a meta-algorithm and its applications. Technical report, Princeton University, USA, available at: http://www.cs.princeton.edu/ arora/pubs/MWsurvey.pdf, 2006.

[AK07] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC*, pages 227–236, 2007.

[FS99] Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[GK98] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 300–309, 1998.

[GK04] Naveen Garg and Rohit Khandekar. Fractional covering with upper bounds on the variables: Solving lps with negative entries. In *ESA*, pages 371–382, 2004.

[Kha04] Rohit Khandekar. *Lagrangian Relaxation based Algorithms for Convex Programming Problems.* PhD thesis, Indian Institute of Technology, Delhi, 2004.

[KY07] Christos Koufogiannakis and Neal E. Young. Beating simplex for fractional packing and covering linear programs. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 494–504, 2007.

[LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.

[PST91] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *FOCS*, pages 495–504, 1991.

[You01] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS*, pages 538–546, 2001.