15-850: Advanced Algorithms **CMU, Spring 2017**
Lecture #12: MW Applications: Zero-Sum Games and LP Solvers February 21, 2017
Lecturer: Anupam Gupta Scribe: Dongho Choi, Yang Jiao, Edgar Chen

1 Last Time

Last time, we covered a multiplicative weights algorithm for learning with small regret. Let $\ell^t \in [-1,1]^N$ be a "loss vector." Define $\Delta_N = \{x \in [0,1]^N \mid \sum_{i=1}^N x_i = 1\}$ to be the N-dimensional probability simplex. We showed in the last lecture that:

Theorem 12.1. *For every $0 < \varepsilon \leq 1$, there exists an algorithm Hedge(ε) such that for all times $T > 0$, for every sequence of loss vectors $(\ell^1, \dots, \ell^T)$, and for every $i \in \{1, \dots, n\}$, at every time $t \leq T$, Hedge(ε) produces $p^t \in \Delta_N$ such that*

$$\sum_{t=1}^T \langle \ell^t, p^t \rangle \leq \sum_{t=1}^T \langle \ell^t, e_i \rangle + \varepsilon T + \frac{\ln N}{\varepsilon},$$

where e_i is the ith vector in the standard basis of $\mathbb{R}^N$. Note that the first term on the right hand side represents the loss of the ith expert, and the last two terms represents the regret of not having always chosen the ith expert.

Note that if we choose $\varepsilon = \sqrt{\frac{\ln N}{T}}$, then $\varepsilon T + \frac{\ln N}{\varepsilon} = 2\sqrt{T \ln N}$, so that the regret term is *sublinear* in time T. This indicates that the average regret of Hedge(ε) converges towards the best expert, so that Hedge(ε) is in some sense "learning".

For future reference, we state the analogous result for gains g^t instead of losses ℓ^t, i.e., $g^t = -\ell^t$.

Theorem 12.2. *For every $0 < \varepsilon \leq 1$, there exists an algorithm Hedge$_g$(ε) such that for all times $T > 0$, for every sequence of gain vectors $(g^1, \dots, g^T)$, and for every $i \in \{1, \dots, n\}$, at every time $t \leq T$, Hedge$_g$(ε) produces $p^t \in \Delta_N$ such that*

$$\sum_{t=1}^T \langle g^t, p^t \rangle \geq \sum_{t=1}^T \langle g^t, e_i \rangle - \varepsilon T - \frac{\ln N}{\varepsilon},$$

where e_i is the ith vector in the standard basis of $\mathbb{R}^N$. Note that the first term on the right hand side represents the gain of the ith expert, and the last two terms represents the regret of not having always chosen the ith expert.

We also state a corollary of 12.2 that we will use.

Corollary 12.3. *Let $\rho \geq 1$. For every $0 < \varepsilon \leq \frac{1}{2}$, for all times $T \geq \frac{4\rho^2 \ln N}{\varepsilon^2}$, for all sequences of gain vectors $(g^1, \dots, g^T)$ with each $g^t \in [-\rho, \rho]^N$, and for all $i \in \{1, \dots, N\}$, at every time $t \leq T$, Hedge$_g$(ε) produces $p^t \in \Delta_N$ such that*

$$\frac{1}{T}\sum_{t=1}^T \langle g^t, p^t \rangle \geq \frac{1}{T}\sum_{t=1}^T \langle g^t, e_i \rangle - \varepsilon.$$

2 Zero-Sum Games

A zero sum game can be described by a pay-off matrix $M \in \mathbb{R}^{m \times n}$, where the two players are the "row player" and the "column player". Simultaneously, the row player chooses a row i and the column player chooses a column j, and the row player receives a pay-off of $M_{i,j}$. Alternatively, the column player loses $M_{i,j}$; hence, the name "zero-sum".

Given a strategy $x \in \Delta_m$ for the row player and a strategy $y \in \Delta_n$ for the column player, the expected pay-off to the row player is

$$\mathbf{E}[\text{pay-off to row}] = x^T M y.$$

The row player wants to maximize this value, while the column player wants to minimize this value. Note that the choices of x and y are made simultaneously. Later, we will see that these choices really do not have to be made at the same time.

Suppose row player fixes a strategy $x \in \Delta_m$. Knowing the row player's strategy, the column player can choose a response to minimize the row player's expected winnings:

$$C(x) = \min_{y \in \Delta_n} x^T M y = \min_{j \in [n]} x^t M e_j.$$

The equality holds because if the column player already knows the row player's strategy (x), then the column player's best strategy is to choose the column that minimizes the row player's expected winnings.

Analogously, suppose the column player fixes a strategy $y \in \Delta_n$. Knowing the column player's strategy, the row player can choose a response to maximize his own expected winnings:

$$R(y) = \max_{x \in \Delta_m} x^T M y = \max_{i \in [m]} e_i^T M y.$$

Overall, the row player wants to achieve $\max_{x \in \Delta_m} C(x)$, and the column player wants to achieve $\min_{y \in \Delta_n} R(y)$. It is easy to see that for any $x \in \Delta_m, y \in \Delta_n$, we have

$$C(x) \leq R(y) \tag{12.1}$$

Intuitively, in the latter the column player commits to a strategy y, and hence gives more power to the row player. Formally, the row player could always play strategy x in response to y, and hence could always get value $C(x)$. But $R(y)$ is the best response, which could be even higher.

Theorem 12.4. *(Von Neumann's Minimax) For any finite zero-sum game $M \in \mathbb{R}^{m \times n}$,*

$$\max_{x \in \Delta_m} C(x) = \min_{y \in \Delta_n} R(y).$$

The common value V is called the value of the game M.

Proof. By scaling, we assume for the sake of contradiction that $\exists M \in [-1,1]^{m \times n}$ such that $\max_{x \in \Delta_m} C(x) < \min_{y \in \Delta_n} R(y) - \delta$ for $\delta > 0$.

We treat each row of M as an expert. At each time step t, the row player produces $p^t \in \Delta_m$. Initially, $p^1 = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$, which represents that the row will choose any row with equal probability when he has no information to work with.

At each time t, the column player plays the best response to p^t, i.e.,

$$j_t := \arg\max_{j\in[n]} (p^1)^T M e_j.$$

Now define the gain vector for the row player as

$$g_t := M e_{j_t}.$$

This the row player uses to update the weights as get p_{t+1}, etc. Define

$$\hat{x} := \frac{1}{T}\sum_{t=1}^{T} p^t \quad \text{and} \quad \hat{y} := \frac{1}{T}\sum_{t=1}^{T} e_{j_t}.$$

These are the average long-term plays of the row player, and the best responses of the column player to those plays. We know that $C(\hat{x}) \le R(\hat{y})$ by (12.1). Now by 12.3, after $T \ge \frac{4\ln m}{\varepsilon^2}$ steps,

$$\begin{aligned}
\frac{1}{T}\sum_t \langle p^t, g^t\rangle &\ge \max_i \frac{1}{T}\sum_t \langle e_i, g^t\rangle - \varepsilon && \text{(by Hedge)}\\
&= \max_i \langle e_i, \frac{1}{T}\sum_t g^t\rangle - \varepsilon\\
&= \max_i \langle e_i, M(\tfrac{1}{T}\textstyle\sum_t e_{j_t})\rangle - \varepsilon && \text{(by definition of } g_t\text{)}\\
&= \max_i \langle e_i, M\hat{y}\rangle - \varepsilon = R(\hat{y}) - \varepsilon
\end{aligned}$$

We also know that since p^t is the row player's strategy, and C is concave (i.e., the payoff on the average strategy $\hat{x}$ is no more than the average of the payoffs),

$$\frac{1}{T}\sum \langle p^t, g^t\rangle = \frac{1}{T}\sum C(p^t) \le C(\frac{1}{T}\sum p^t) = C(\hat{x})$$

Putting it all together:

$$R(\hat{y}) - \varepsilon \le C(\hat{x}) \le R(\hat{y})$$

Since we control ε, for any δ we can make ε smaller than δ to get the contradiction. □

3 Solving LPs

In this lecture, we use the above theorem to solve LPs approximately. We are given an LP with n variables and m constraints (excluding non-negativity constraints):

$$\begin{aligned}
\max \quad & c\cdot x\\
Ax &\le b\\
x &\ge 0
\end{aligned}$$

Denote by OPT the optimal value of the LP. Let $K \subseteq \mathbb{R}^n$ be the polyhedron defined by the easy constraints, i.e. $K = \{x \in \mathbb{R}^n : x \ge 0, c\cdot x = OPT\}$, where OPT is found by binary search over possible objective values. We will find $x \in K$ such that $\langle a_i, x\rangle \le b_i + \varepsilon$ for all $i \in [m]$.

3.1 The Oracle

First, we make a simple observation.

Proposition 12.5. *Given a single constraint $\alpha \cdot x \geq \beta$, there is a fast oracle that finds $x \in K$ such that $\alpha \cdot x \geq \beta$ if such an x exists, or else outputs that no such x exists.*

Proof. We give the proof only for the case where $c_i \geq 0$ for all i; the general case is left as an exercise. Find $j^* = \operatorname{argmax}_j \frac{\alpha_j OPT}{c_j}$. Set $x^* = \frac{OPT}{c_{j^*}} e_{j^*}$. Then x^* is a vector that maximizes $\alpha \cdot x$ subject to $x \in K$. Now check whether $\alpha \cdot x^* \geq \beta$ and output x^* if the inequality is satisfied; otherwise output no such x exists.

Note that the runtime of this oracle procedure is polynomial in n, the dimension of the problem. (As we'll see later in the lecture, when we consider max-flows, that even if the dimension is exponential, we may be able to efficiently figure out the argmax in the calculation above.) □

To solve the LP, we will call the above oracle while using Hedge by repeatedly asking the oracle to produce a solution to a convex combination of the m constraints and updating the weight/importance of each constraint based on how badly the constraint was violated by the current solution. To this end, we will set our gain vectors to be the amount of violation of the corresponding constraint.

An upper bound on the maximum possible violation is the *width* ρ of the LP, defined by

$$\rho := \max_{x \in K, i \in [m]} \{|a_i \cdot x - b_i|\}.) \tag{12.2}$$

We will assume that $\rho \geq 1$.

3.2 The Algorithm

Below we give the formal algorithm to approximately solve the LP.

1. $p^1 \leftarrow (1/m, \dots, 1/m)$
2. Call oracle to give x^t such that $\sum_{i=1}^m \langle p_i^t a_i, x\rangle \leq \sum_{i=1}^m p_i^t b_i$. If oracle says no, it means that the original LP is infeasible, as any feasible solution to the original LP would satisfy a convex combination of the LP's constraints.
3. Define gain vector by $g_i^t := \langle a_i, x^t\rangle - b_i$.
4. Update p^{t+1} using Hedge(ε).
5. Run steps 1 to 4 for $T := \Theta(\rho^2 \ln m/\varepsilon^2)$ rounds.
6. return $\hat{x} \leftarrow (x_1 + \dots + x_T)/T$.

3.3 The Analysis

Theorem 12.6. *For every $0 \leq \varepsilon \leq 1/4$, the above algorithm returns $\hat{x} \in K$ such that $\langle a_i, x\rangle \leq b_i + \varepsilon$ for all $i \in [m]$ and calls the oracle $O(\rho^2 \ln m/\varepsilon^2)$ times.*

Proof. Let $i \in [m]$. Define $\alpha^t = \sum_{i=1}^m p_i^t a_i$ and $\beta^t = \sum_{i=1}^m p_i^t b_i$. Then

$$\begin{aligned}\langle p^t, g^t\rangle &= \langle p^t, Ax^t - b\rangle \\ &= \langle p^t, Ax^t\rangle - \langle p^t, b\rangle \\ &= \langle \alpha^t, x^t\rangle - \beta^t \\ &\leq 0.\end{aligned}$$

So the left hand side in Corollary 12.3

$$\frac{1}{T}\sum_{t=1}^{T}\langle p^t, g^t\rangle \leq 0.$$

Second, the right hand side in Corollary 12.3

$$\frac{1}{T}\sum_{t=1}^{T}\langle a_i, x^t\rangle - b_i - \varepsilon = \langle a_i, \hat{x}\rangle - b_i - \varepsilon.$$

By Corollary 12.3, we have

$$0 \geq \frac{1}{T}\sum_{t=1}^{T}\langle p^t, g^t\rangle \geq \frac{1}{T}\sum_{t=1}^{T}\langle a_i, x^t\rangle - b_i - \varepsilon \geq \langle a_i, \hat{x}\rangle - b_i - \varepsilon.$$

Hence, for each constraint i, we have that $\langle a_i, \hat{x}\rangle \leq b_i + \varepsilon$. $\square$

3.4 A Small Extension: Approximate Oracles

Recall the definition of the problem width from (12.2). Two comments:

- In the above analysis, we do not care about the maximum value of $|a_i \cdot x - b_i|$ over all points $x \in K$, we only care about the largest this expression can get over all the points that can be potentially returned by the oracle. While this seems a pedantic point, it will be useful — if there are many solutions we can return for $\alpha \cdot x \leq \beta$, we can try to return the one with least width. But we can do more, as the next point outlines.
- We can also relax the oracle to satisfy $\alpha \cdot x \leq \beta + \delta$ for some small $\delta > 0$ instead. Define the *width* of the LP with respect the relaxed oracle to be

$$\rho_{\text{relaxed}} := \max_{i\in[m], x \text{ returned by relaxed oracle}} \{|a_i \cdot x - b_i|\}. \tag{12.3}$$

  The new range of g^t using the relaxed oracle will be $[-\rho_{\text{relaxed}}, \rho_{\text{relaxed}}]^N$. So running the same algorithm with the relaxed oracle will give us that $a_i \cdot \hat{x} \leq b_i + \varepsilon + \delta$ for all $i \in [m]$ and the number of calls to the relaxed oracle is $O(\rho_{\text{relaxed}}^2 \ln m/\varepsilon^2)$.

We will not use these today, but these will be crucial to us in the next lecture.

4 Application to Finding Max Flows

In the max flow problem, we are given an $s - t$ network G possibly having multi-arcs, each arc having capacity 1. Let $\mathcal{P}$ be the set of all s-t paths in G. We use a path-based LP formulation:

$$\begin{aligned} \max \quad & \sum_{P\in\mathcal{P}} f_P \\ & \sum_{P\ni e} f_P \leq 1 && \forall e \in E \\ & f_P \geq 0 && \forall P \in \mathcal{P} \end{aligned}$$

Let F be the optimal flow value over feasible s-t flows in G. The "easy" constraints are then given by

$$K := \{f : f \geq 0, \sum_{P \in \mathcal{P}} f_p = F\}.$$

Now given probabilities $p^t \in \Delta_m$, the convex combination of the LP constraints weighted by p^t is the single constraint

$$\sum_{e \in E} p_e^t \sum_{P \ni e} f_P \leq \sum_{e \in E} p_e^t \leq 1,$$

or equivalently

$$\sum_{P \in \mathcal{P}} f_P \left(\sum_{e \in P} p_e^t \right) \leq 1.$$

Let us define $\ell_t(P) := \sum_{e \in P} p_e^t$ as the length of path P.

In that case, one solution to minimize $\sum_{P \in \mathcal{P}} f_P \ell_t(P)$ subject to $f \in K$ is to place all the F flow on a shortest s-t path according to $\ell_t(P)$ (and output "no" if this results in value more than 1). This can be done by a single call to Dijkstra's algorithm, in $O(m + n \log n)$ time.

4.1 The Algorithm

Plugging this into the Hedge-based algorithmic framework, we get the following algorithm:

1. $p^1 \leftarrow (1/m, \ldots, 1/m)$.
2. Call shortest-path oracle to give $f^t \in K$ such that $\sum_{P \in \mathcal{P}} f_P \sum_{e \in P} p_e^t \leq 1$. (If oracle says no, it means that the original LP is infeasible, as any feasible solution to the original LP would satisfy a convex combination of the LP's constraints.)
3. Define gain vector by $g_e^t = f_e^t - 1$, where f^t is the flow found by the oracle at round t.
4. Update p^{t+1} using Hedge(ε).
5. Run steps 1 to 4 for $T := \Theta(\rho^2 \ln m/\varepsilon^2)$ rounds, where $\rho := F$.
6. Return $\hat{f} \leftarrow (f^1 + \cdots + f^T)/T$.

The analysis above ensures $\hat{f} \in K$, and also that

$$\sum_{P : e \in P} \hat{f}_P \leq 1 + \varepsilon.$$

To get a flow that respects the constraints, we can scale down $\hat{f}$ by $(1+\varepsilon)$ to get a feasible flow that has value $\frac{1}{1+\varepsilon} F \geq (1-\varepsilon)F$ in time $O(\rho^2 \log m/\varepsilon^2)$.

A few observations:

- Observe that every gain vector g^t here is within the range $[-1, F]^m$. So we can use an asymmetric version of the Hedge guarantee (which appears in hand-written notes on the webpage). This analysis shows that if the gains are in $[-\gamma, \rho]^N$ for N experts, then we need only $O(\gamma\rho \ln N/\varepsilon^2)$ rounds. Using this we immediately get the runtime down to $O(F \log m/\varepsilon^2)$ iterations.

  Now each iteration takes $O(m + n \log n)$ to find the shortest path according to the lengths induced by p^t. So the total runtime is $\tilde{O}(mF)$. Although, this runtime is no better than the Ford Fulkerson algorithm, we will see a way to reduce the runtime via electrical flows next lecture.

- The algorithm repeats the following "natural" process: it finds a shortest path in the graph, pushes flow on it, and exponentially increases the length of the edge. This makes congested edges (those with a lot of flow) become very undesirable. Note that unlike usual network flows, these algorithms are greedy and cannot "undo" past actions (which is what pushing flow in residual flow networks does, when we use an arc backwards). So MW-based algorithms must ensure that very little flow goes on edges that are "wasteful".

4.2 An Example

To see how the algorithm corrects the paths it chooses towards better paths, we demonstrate the first few steps of the algorithm on an example. For simplicity of calculations, we ignore the ε in updating the weights, i.e., assume that $w_e^{t+1} \leftarrow w_e^t(1 + g_e^t/\rho)$. The label on edge e in the tth round indicates w_e^t. The green path in the tth round indicates the shortest path the oracle gives according to weights w_e^t.

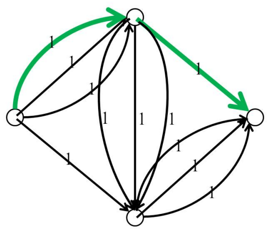

Figure 12.1: Round 1

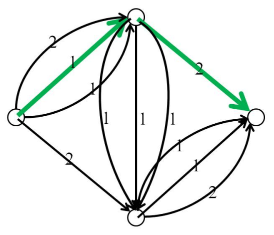

Figure 12.2: Round 2

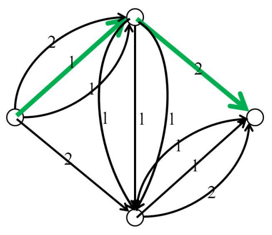

Figure 12.3: Round 3

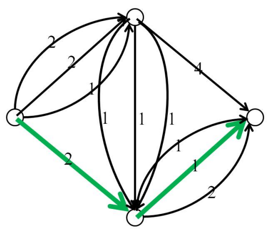

Figure 12.4: Round 4

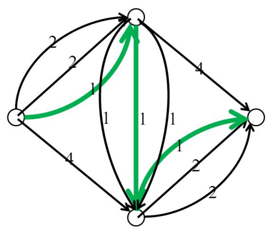

Figure 12.5: Round 5

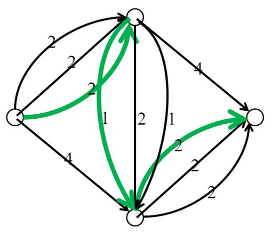

Figure 12.6: Round 6