Advanced topics in ML and AGT **Fall Semester, 2018/9**

Lecture 9: December 10

Lecturer: Yishay Mansour *Scribe: Tal Lancewicki, Itsik Mor, Or Malkai*

9.1 Lecture Overview

In this lecture we will show applications of regret minimization to 2-player zero sum games and algorithm design. We will start with a few definitions.

Definition A game for N players is defined such that for each player $i \in [N]$ there is a group of possible actions A_i and a loss function ℓ of player $i : \ell_i(a_1...a_N), A_1 \times \cdots \times A_n \to \mathbb{R}$ and each players wants to minimize her loss.[1]

Definition A joint action $(a_1, ..., a_N)$ is a pure Nash equilibrium if for every player $i \in [N]$ and any action $a'_i \in A_i$ the loss of player i will not decrease by unilateraly deviating, i.e.,

$$\forall i \in [N]\ \forall a'_i \in A_i \quad \ell_i(a_1, ..., a_N) \leq \ell_i(a_1, ..., a_{i-1}, a'_i, a_{i+1}, ..., a_N)$$

Definition In a mixed Nash equilibrium every player $i \in [N]$ has a distribution D_i, and distributions $D_1...D_N$ are a mixed equilibrium if for every player i and any distribution D'_i, player i will not benefit by unilateraly deviating, i.e.

$$\forall i \in [N]\ \forall D'_i \in \Delta(A_i) \underset{a_j \sim D_j}{\mathbb{E}} \ell_i(a_1, ..., a_N) \leq \underset{\substack{j \neq i: a_j \sim D_j \\ a'_i \sim D'_i}}{\mathbb{E}} \ell_i(a_1, ..., a_{i-1}, a'_i, a_{i+1}, ..., a_N)$$

where $\Delta(A_i)$ is the set of ditributions over A_i.

9.2 2-Player zero-sum game

In a 2-player zero-sum game the actions and loss functions are defined such that:

$$\forall a_1 \in A_1, a_2 \in A_2 : \ell_1(a_1, a_2) + \ell_2(a_1, a_2) = 0$$

Based on the scribe of Yiftach Ginger, Eyal Shulman and Nitai Itzhacky, 2017

[1]Usually in game theory, games are defined with utility functions rather than loss functions, but here we use loss functions in order to be consistent with the Regret minimization model.

This clearly implies that for every two distribution D_1, D_2 the sum of expected losses is zero. That is,

$$\mathop{\mathbb{E}}_{a_i \sim D_i} \left[\ell_1 (a_1, a_2) + \ell_2 (a_1, a_2)\right] = 0$$

Usually the game is modeled as a matrix where the rows denote the first player action, the columns denote the second player action and the matrix entries are the corresponding to the loss of the row player (Recall that the loss of the column player is negative the loss of the row player). For example:

$$M = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & b_2 & b_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 2 \\ 3 & 1 & -1 \\ -2 & 4 & 1 \end{bmatrix} \end{array}$$

The rows player will try to minimize the games value and the columns player will try to maximize it.

9.2.1 Pure Nash Equilibrium

When the first player plays first, in the above example, we get:

$$min_i max_j M[i,j] = 2. \text{ This holds for } i = 1, j = 3$$

When the second player plays first:

$$max_j min_i M[i,j] = 0. \text{ This holds for } i = 1, j = 2$$

For this example we can see that there is no pure equilibrium.

In general it holds that

$$\min_i \max_j M[i,j] \geq \max_j \min_i M[i,j], \tag{9.1}$$

since when the minimizer get to choose i given j (right hand side), she can still choose the same action as if she chooses i before j is known (left hand side).

9.2.2 Mixed Strategies

Each player chooses a distribution. The other player knows the distribution but not the realized action.

The rows player will choose a distribution $p \in \Delta(A_1)$
The columns player will choose a distribution $q \in \Delta(A_2)$

The value of the game (expected loss of row player) is,

$$M[p,q] = p^T M q = \sum_{i,j} M[i,j]p[i]q[j]$$

Theorem 9.1 *MinMax Theorem (Von Neuman):*

$$min_p max_q M[p,q] = max_q min_p M[p,q]$$

Proof: Note that the player that moves last can be deterministic. That is,

$$min_p max_q M[p,q] = min_p max_j M[p,j] \tag{9.2}$$

and

$$max_q min_p M[p,q] = max_q min_i M[i,q] \tag{9.3}$$

Both of which can be rewritten as a linear program:

$$\begin{aligned} 1)\, min\, v\, s.t. \\ p^T M \leq v \cdot \bar{1} \\ p^T \cdot \bar{1} = 1 \\ p \geq 0 \end{aligned} \tag{9.4}$$

$$\begin{aligned} 2)\, max\, x\, s.t. \\ Mq \geq x \cdot \bar{1} \\ q^T \cdot \bar{1} = 1 \\ q \geq 0 \end{aligned} \tag{9.5}$$

The two programs have the same value due to the LP duality. Hence $v = x$ which is also the value of the game. □

Returning to the example matrix M. Consider the following distribution for the row player:

$$p = \begin{bmatrix} 6/11 \\ 3/11 \\ 2/11 \end{bmatrix}$$

Consider the columns player distribution:

$$q = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

Namely, the game looks like:

$$\begin{array}{c|ccc} & 1/3 & 1/3 & 1/3 \\ \hline 6/11 & 1 & 0 & 2 \\ 3/11 & 3 & 1 & -1 \\ 2/11 & -2 & 4 & 1 \end{array}$$

Now if the row's player goes second, the expected value for her is:

$$Mq = \begin{bmatrix}1\\1\\1\end{bmatrix}$$

And whatever distribution she will choose the outcome will be the same, so she has no reason to change her distribution from p.
Likewise, if the columns player goes second he will see the game as:

$$p^\top M = \begin{bmatrix}1 & 1 & 1\end{bmatrix}$$

And for him the there will be no preference for any distribution either, so he will play q.
Finally this means that the game has a mixed strategy equlibrium and the value of the game is 1.

Alternative proof of the minimax theorem using regret minimization

Proof:
Define:
$v^1_{min} = max_q min_i M[i,q]$
$v^1_{max} = -v^2_{min} = min_p max_j M[p,j]$
The MinMax theorem implies that $v^1_{min} = v^1_{max}$.
As we've seen in (9.1), $v^1_{min} \leq v^1_{max}$
Assume by contradiction that $\exists \gamma > 0 \, s.t. \, v^1_{max} = v^1_{min} + \gamma$

We will play a game where each player does regret minimization for T steps and has regret $R(T)$. We have that $lim_{T\to\infty} R(T)/T = 0$ and so for some T, $\gamma/2 > R(T)/T$.

Game definitions

- L_{ON} player 1's loss (and $-L_{ON}$ for player 2 loss)
- L^1_{min}, L^2_{min} - the losses for the best action for players 1 and 2
- Given a sequence of the players action we define p, q as the empirical distributions of the players' actions.

We now observe a few properties that follow from regret minimization:
Since L^1_{min} is selected is respect to a specific distribution q, we get that:

$$v^1_{min} \geq L^1_{min}/T \tag{9.6}$$

And similarly:

$$v^2_{min} = -v^1_{max} \geq L^2_{min}/T \tag{9.7}$$

From the regret minimization algorithm regret guarantee we get:

$$L_{ON} \leq L^1_{min} + R(T) \qquad (9.8) \qquad\qquad -L_{ON} \leq L^2_{min} + R(T) \qquad (9.9)$$

Which combines to give us:

$$v^1_{max} - \frac{R(T)}{T} \leq -\frac{L^2_{min}}{T} - \frac{R(T)}{T} \leq \frac{L_{ON}}{T} \leq \frac{L^1_{min}}{T} + \frac{R(T)}{T} \leq v^1_{min} + \frac{R(T)}{T} \qquad (9.10)$$

Therefore $v^1_{max} - v^1_{min} \leq \dfrac{2R(T)}{T} < \gamma$ in contradiction to our initial assumptions.

□

We can also derive the following corollary from the proof.

Corollary 9.2 *If a player in a 0-sum 2-player game plays a regret minimization strategy with regret* $\dfrac{R(T)}{T}$*, then it guarantees an average loss of at most* $v + \frac{R(T)}{T}$*, where* v *is the value of the game.*

9.3 Deterministic and Randomized Algorithms

In the following examples, let $\mathbb{A}$ denote the set of deterministic algorithms for solving the given problem, and $\mathbb{X}$ the set of the possible inputs. Let $Time(A_i, x_j)$ denote the time complexity of deterministic algorithm A_i with the input x_j. $Time()$ can be interpreted as a matrix, where the rows represent algorithms and columns represent time.

Definition A *Randomized Algorithm* defines distribution p over $\mathbb{A}$ (the algorithm space).

Definition The *Random Time Complexity* of a random algorithm A_i with probability p_i is denoted by $RTime(p, x_j)$ and is defined as a weighted average over algorithms:

$$RTime(p, x_j) = \sum_i p_i Time(A_i, x_j)$$

Definition The *Average Time Complexity* of a deterministic algorithm A_i given a distribution q over its inputs is denoted by $AvgTime(A_i, q)$ and is defined as a weighted average over inputs:

$$AvgTime(A_i, q) = \sum_j q_j Time(A_i, x_j)$$

Definition The **worst case time complexity** of a deterministic algorithm is

$$\min_i \max_j Time(A_i, x_j)$$

the best algorithm that will minimize the time on the worst input.

Definition The **non-deterministic time complexity** is

$$\max_j \min_i Time(A_i, x_j)$$

the best algorithm that will minimize the time given that we already know the input.

Definition The **random worst case time complexity** is

$$\min_p \max_j RTime(p, x_j)$$

the best distribution over the algorithm space that will minimize the complexity on the worst input.

Definition The **average worst case time complexity** is

$$\max_q \min_i AvgTime(A_i, q)$$

the worst distribution over the input space that will maximize complexity for the best algorithm that will minimize it.

Theorem 9.3 *(Yao's Lemma): For any distribution p over the algorithm space $\mathbb{A}$ and q over the input space $\mathbb{X}$*

$$\max_j RTime(p, x_j) \geq \min_i AvgTime(A_i, q)$$

Proof: Construct a 2-player zero sum game. Player 1 chooses $A_i \in \mathbb{A}$ (min player). Player 2 chooses $x_j \in \mathbb{X}$ (max player). The payoff is the time complexity $Time(A_i, x_j)$. According to the Minimax Theorem we get:

$$\min_p \max_q E_{i\sim p, j\sim q}[Time(A_i, x_j)] = \max_q \min_p E_{i\sim p, j\sim q}[Time(A_i, x_j)]$$

Since the last player can be deterministic

$$\min_p \max_j E_{i\sim p}[Time(A_i, x_j)] = \max_q \min_i E_{j\sim q}[Time(A_i, x_j)]$$

Notice that for any distribution $\bar{p}$ over $\mathbb{A}$ and distribution $\bar{q}$ over $\mathbb{X}$

$$\min_p \max_j E_{i\sim p}[Time(A_i, x_j)] \leq \max_j E_{i\sim \bar{p}}[Time(A_i, x_j)]$$

$$\max_q \min_i E_{j\sim q}[Time(A_i, x_j)] \geq \min_i E_{j\sim \bar{q}}[Time(A_i, x_j)]$$

So, according to the results from the minimax theorem we get:

$$\max_j E_{i\sim \bar{p}}[Time(A_i, x_j)] \geq \min_i E_{j\sim \bar{q}}[Time(A_i, x_j)]$$

Therefore,

$$\max_j RTime(\bar{p}, x_j) \geq \min_i AvgTime(A_i, \bar{q})$$

□

9.3.1 Lower Bound for Sorting Algorithm

We want to lower-bound the complexity of a random algorithm for sorting n numbers (comparison based sort). We can describe any deterministic comparison based sort algorithm as a decision tree, where each internal node corresponds to a comparison the algorithm performs, with 2 possible outcomes (we assume all elements are different). For a specific input, the execution of the algorithm corresponds to a path from the root to a leaf. It is impossible for 2 different permutations to result in the same path. The running time for the algorithm over an input is the length of the path.

Therefore, the decision tree must have at least $n!$ leaves. Thus the depth of the tree is at least $\log(n!) = \Theta(nlogn)$ nodes. The number of leaves whose depth l is at most 2^{l+1}.

This means that for every deterministic algorithm at least half of the permutations ($\frac{1}{2}n!$) are in depth of $\log(\frac{1}{2}n!) = \Omega(nlogn)$

Let us choose a uniform distribution q over all possible inputs (all permutations of n numbers). The running time of any algorithm over this distribution is simply the average of the depth of the leaves for all possible inputs. But at least $\frac{1}{2}n!$ inputs are at depth at least $log(n!) - 1$, so the average running time will be at least

$$\frac{\frac{n!}{2}(\log(n!) - 1)}{n!} = \Omega(n\log(n))$$

9.3.2 Best Arm Identification

Another example for using Yao's lemma. In the last lecture we saw that for every deterministic algorithm that runs in $T < \frac{ck}{\epsilon^2}$, there exists a set S of $\lceil \frac{k}{3} \rceil$ actions so that every action $j \in S$

$$Pr(y_T \neq j | I_j) \geq \frac{1}{4}$$

We want to derive from that a lower bound for randomized algorithm. We will construct a matrix M such that the rows are algorithms, the columns are profiles and each entry $M[i, j]$ is the cost:

$$M[i, j] := Pr[y_T \neq j | I_j, A_i]$$

Let us define a uniform distribution over I_j, then for any algorithm A_i,

$$E_j[Pr[y_T \neq j | I_j, A_i]] \geq \frac{1}{3} \cdot \frac{1}{4} = \frac{1}{12}$$

This implies that we have a distribution over the inputs, I_j, such that for any randomized algorithm defind by the distribution p, the error is at least $1/12$.

$$E_{A_i \sim p}[E_j[Pr[y_T \neq j | I_j, A_i]]]] \geq \frac{1}{12}$$

This implies a lower bound of $\Omega(k/\epsilon^2)$ for any randomized algorithm.

9.4 Boosting: Weak vs. Strong Learning

The model: For a target function f and a hypothesis class H

- $f : X \longrightarrow \{0, 1\}$
- $H = \{h : X \longrightarrow \{0, 1\}\}$

The ϵ-WL (weak learning) assumption:

$$\forall D \; \exists h \in H : Pr_{x \sim D}[h(x) = f(x)] \geq 1/2 + \epsilon$$

Question: How well can members of H approximate f?

We shall define a reduction from this problem to a 2-player zero sum game. Where the max player chooses an input $x \in X$ and the min player chooses a function $h \in H$. The value function is an error indicator:

$$M(h, x) = \begin{cases} 0 & \text{if } f(x) = h(x) \\ 1 & \text{if } f(x) \neq h(x) \end{cases}$$

The ϵ-WL assumption implies that for each mixed strategy D of the inputs (max) player

$$\forall D, \exists h \in H : E_{x \sim D}[M[h, x]] = Pr_{x \sim D}[h(x) \neq f(x)] \leq 1/2 - \epsilon$$

Thus
$$\max_D \min_h E_{x\sim D}[M(h,x)] \leq 1/2-\epsilon$$
Which then means that the value of the game V is,
$$V \leq 1/2-\epsilon$$

According to the minimax theorem, we can reverse the roles of the players.
Therefore
$$\exists q \in \Delta(H) : \max_x E_{h\sim q}[M(h,x)] \leq 1/2-\epsilon$$
Hence
$$\forall x \in X : E_{h\sim q}[M(h,x)] \leq 1/2-\epsilon$$
Define a hypothesis $G(x)=\sum_{h\in H} q(h)h(x)$
$$\forall x \in X : |f(x)-G(x)| = \left|\sum_{h\in H} q(h)(f(x)-h(x))\right| \leq \sum_{h\in H} q(h)\underbrace{|f(x)-h(x)|}_{M(h,x)} \leq 1/2-\epsilon < 1/2$$

So by using the hypothesis $\hat{f}(x)=sign(G(x)-1/2)=\mathbb{I}\{G(x)\geq \frac{1}{2}\}$ we obtain $\forall x \in X$: $\hat{f}(x)=f(x)$.
Actually, this is stronger corollary than the regular boosting since $\hat{f}$ is excatly f, though it is not constructive.

We would now like to show that we can have a small size $G(x)$ at the cost of not having a perfect predictor i.e., which is always correct. In the previous proof we show that there is a distribution q and we define $G(x)=\sum_{h\in H} q(h)h(x)$ such that $f(x)=sign(G(x)-1/2)$.

In the proof we also show that if $f(x)=1$ then $G(x)\geq 1/2+\epsilon$ and if $f(x)=0$ then $G(x)\leq 1/2-\epsilon$.

Now consider $G_n(x)=\sum_{i=1}^{n} h_i(x)$, where h_i is sampled from H using q. Fix an input x. Then $E[G_n(x)]=G(x)$. Also, with probability $1-\delta$ we have that $|G(x)-G_n(x)|\leq \epsilon/2$, assuming that $n\geq c\epsilon^{-2}\log\delta^{-1}$ for some $c>0$.

This implies that $G_n(x)$ has error on at most δ fraction of the inputs and is composed from at most n hypotheses $h\in H$.

9.5 Max Multi-Commodity Flow

Problem definition:

- $G(V,E)$ a directed graph

- $c : E \to \mathbb{R}^+$ the edge capacity
- $(s_1, t_1), (s_2, t_2), ..., (s_k, t_k)$ are k source and target pairs

The goal: find a flow f that maximize the total flow (sum of flows from all source and target pairs) while satisfying all constraints

The constraints:

- Capacity: the total flow on an edge should not exceed its capacity
- Flow conservation: $\forall i \in [k], \forall u \in V \setminus \{(s_i, t_i)\} : \sum_{w \in V} f_i(u, w) = \sum_{w \in V} f_i(w, u)$

We'd like to find a $1 - \epsilon$ approxiamtion to the optimal solution using the regret algorithm!

Algorithm definitions:

- Let P be the set of simple directed paths from s_1 to t_1, s_2 to t_2, etc.
- For $\rho \in P$ let $f_\rho \geq 0$ be the flow on ρ

We want to maximize $\sum_{\rho \in P} f_\rho$ s.t.

- $\forall e \in E : f_e = \sum_{\rho \ni e} f_\rho \leq c_e$
- $\forall \rho \in P : f_\rho \geq 0$

The algorithm: At each iteration:

- Given the distribution $p_t(e)$ over the edges, find the shortest path $path_t \in P$ with respect to weights $\frac{p_t(e)}{c(e)}$
- Route c_t units of flow on $path_t$, where $c_t = min_{e \in path_t} c(e)$ the path bottleneck
- Set gains: $r_t(e) = \begin{cases} \frac{c_t}{c(e)} & \text{if } e \in path_t \\ 0 & \text{else} \end{cases}$
  Note that $0 \leq r_t(e) \leq 1$ and $\exists e \in path_t : r_t(e) = 1$
- Let $f_t(e)$ be the total flow on e at time t, stop when $\exists e \in E : \frac{f_t(e)}{c(e)} \geq \frac{R(t)}{\epsilon} = \frac{\sqrt{Tln(|E|)}}{\epsilon}$.
- At termination, let $C = max_{e \in E} \frac{F_{ON}(e)}{c(e)}$ and set the flow to be $\frac{F_{ON}(e)}{C}$ (This guarantees that the flow on each edge is feasible).

Finally the total flow is $F_{ON} = \sum_{t=1}^{T} c_t$ and we down-scale F_{ON} to maintain all capacity constraints.

Analysis:
Denote $f_{OPT}(\rho)$ the optimal flow on path ρ and $F_{OPT} = \sum_{\rho \in P} f_{OPT}(\rho)$ the optimal flow in G

$$\sum_{t=1}^{T} p_t^T r_t = \sum_{t=1}^{T} \sum_{e \in path_t} p_t(e) \frac{c_t}{c(e)} = \sum_{t=1}^{T} c_t \sum_{e \in path_t} \frac{p_t(e)}{c(e)} \overset{\star}{\leq} \frac{1}{F_{OPT}} \sum_{t=1}^{T} c_t = \frac{F_{ON}}{F_{OPT}}$$

We later show ★. The above implies,

$$\frac{F_{ON}}{F_{OPT}} \geq \sum_{t=1}^{T} p_t^T r_t \geq \underbrace{\max_e \frac{f_{ON}(e)}{c(e)}}_{C} - R(T)$$

We scale down the flow by C which is the normalization factor for fitting the flow to edges capacities.

$$\frac{F_{ON}}{C} \geq F_{OPT} - \frac{R(T)}{C} F_{OPT}$$

For $C \geq \frac{R(T)}{\epsilon}$ we will get $F_{ON} \geq (1-\epsilon) F_{OPT}$ as desired (we need to show that this holds at termination)

Proof of ★:

$$1 = \sum_{e \in E} p_t(e) \geq \sum_{e \in E} p_t(e) \overbrace{\sum_{\rho : e \in \rho} \frac{f_{OPT}(\rho)}{c(e)}}^{\substack{\leq 1 \text{ as total flow on } e \\ \text{cannot exceed } c(e)}} = \sum_{\rho \in P} f_{OPT}(\rho) \sum_{e \in \rho} \frac{p_t(e)}{c(e)} \geq \sum_{\rho \in P} f_{OPT}(\rho) \sum_{e \in path_t} \frac{p_t(e)}{c(e)} =$$
$$= F_{OPT} \left(\sum_{e \in path_t} \frac{p_t(e)}{c(e)} \right)$$

Where the last inequality follows since $path_t$ is the shortest path at t. □

Runtime:
Each iteration adds 1 to the congestion of at least one edge (the bottleneck edge), so after T iterations there is an edge with congestion $\frac{T}{|E|}$.

Given ϵ we are looking for T such that $C \geq \frac{T}{|E|} \geq \frac{R(T)}{\epsilon} = \frac{\sqrt{T \log(|E|)}}{\epsilon} \Rightarrow T \geq \frac{|E|^2 \log(|E|)}{\epsilon^2}$.