MIT 6.S890 — Topics in Multiagent Learning (F23) Thu, September 21st 2023

Lecture 5

Learning in Games: Algorithms

Instructor: Gabriele Farina*

In Lecture 4 we have mentioned how no-external-regret dynamics recover several solution concepts of interest, including Nash equilibria in two-player zero-sum games, normal-form coarse-correlated equilibria in multiplayer general-sum games, and more generally convex-concave saddle point problems. Furthermore, we have seen that no-external-regret dynamics are a fundamental building block for constructing no-Φ-regret dynamics for general sets Φ of deviation functions.

In this lecture, we start exploring how no-external-regret dynamics for players in normal-form games can be constructed. As a reminder, a player with n action has a strategy space that corresponds to the probability simplex of all distributions over those actions,

$$\Delta^m = \big\{(x_1, \dots, x_m) \in \mathbb{R}^m_{\ge 0} : x_1 + \cdots + x_m = 1\big\}.$$

Furthermore, remember that constructing a regret minimizer for Δ^m means that we need to build a mathematical object that supports two operations:

- NextStrategy() has the effect that the regret minimizer will output an element $\boldsymbol{x}^{(t)} \in \Delta^m$;
- ObserveUtility($u^{(t)}$) provides the environment's feedback to the regret minimizer, in the form of a utility function $u^{(t)}$. For today, we will focus on a case of particular relevance for normal-form games: the case of *linear* utilities (as such are the utilities in a normal-form game). In particular, to fix notation, we will let

  $$u^{(t)} : \boldsymbol{x} \mapsto (\boldsymbol{g}^{(t)})^\top \boldsymbol{x},$$

  where $\boldsymbol{g}^{(t)} \in \mathbb{R}^n$ is the vector that defines the linear function (called with the letter $\boldsymbol{g}$ to suggest the idea of it being the "*gradient vector*"). Since the utility in the game cannot be unbounded, we assume that the $\boldsymbol{g}^{(t)}$ can be arbitrary but *bounded in norm*.

In this notation, the (external) *regret* is defined as the quantity

$$\mathrm{Reg}^{(T)} \coloneqq \max_{\hat{\boldsymbol{x}} \in \Delta^m} \left\{ \sum_{t=1}^T \left(\langle \boldsymbol{g}^{(t)}, \hat{\boldsymbol{x}} \rangle - \langle \boldsymbol{g}^{(t)}, \boldsymbol{x}^{(t)} \rangle \right) \right\}.$$

Our goal is to make sure that the regret grows sublinearly in T no matter the utility vectors $\boldsymbol{g}^{(t)}$ chosen by the environment.

Most algorithms known today for the task operate by *prioritizing actions based on how much regret they have incurred*. In particular, a key quantity to define modern algorithms is the vector of cumulated actions regrets

$$\boldsymbol{r}^{(t)} \coloneqq \sum_{\tau=1}^t \boldsymbol{g}^{(\tau)} - \langle \boldsymbol{g}^{(\tau)}, \boldsymbol{x}^{(t)} \rangle \mathbf{1}.$$

(Note that $\mathrm{Reg}^{(t)} = \max_{i \in \{1,\dots,m\}} \boldsymbol{r}^{(t)}[i]$.) The natural question is: how to prioritize?

*MIT EECS. ✉ gfarina@mit.edu.

1 An algorithm that does NOT work: Follow-the-Leader

Perhaps the most natural idea would be to always play the action with the highest cumulated regret (breaking ties, say, lexicographically). After all, this is the action we wish the most we had played in the past. Unfortunately, this idea is known not to work. To see that, consider the following sequence of gradient vectors:

$$\boldsymbol{g}^{(1)} = \begin{pmatrix} 0 \\ 1/2 \end{pmatrix}, \quad \boldsymbol{g}^{(2)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{g}^{(3)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{g}^{(4)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{g}^{(5)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \text{etc.}$$

In this case, at time 1, we pick the first action and score a utility of 0. We then compute

$$\boldsymbol{r}^{(1)} = \begin{pmatrix} 0 \\ 1/2 \end{pmatrix}.$$

and at time $t = 2$ pick the second action since it has the highest regret. This results in a score of 0 and an updated regret vector

$$\boldsymbol{r}^{(2)} = \begin{pmatrix} 1 \\ 1/2 \end{pmatrix}$$

So, at time $t = 3$, we will pick the first action, resulting again in a score of 0 and an updated regret

$$\boldsymbol{r}^{(3)} = \begin{pmatrix} 1 \\ 3/2 \end{pmatrix},$$

and so on... Overall, it's easy to see that in all this jumping around, the regrets grow linearly.

2 The Regret Matching (RM) algorithm

The regret matching algorithm picks probabilities proportional to the "ReLU" of the cumulated regrets, that is,

$$\boldsymbol{x}^{(t+1)} \propto [\boldsymbol{r}^{(t)}]^+ \tag{1}$$

whenever $[\boldsymbol{r}^{(t)}] \neq \mathbf{0}$, and an arbitrary point otherwise.

The algorithm is presented in pseudocode in Algorithm 1.

Theorem 2.1. The regret matching algorithm (Algorithm 1) is an external regret minimizer, and satisfies the regret bound $R^T \leq \Omega\sqrt{T}$, where Ω is the maximum norm $\|\boldsymbol{g}^{(t)} - \langle \boldsymbol{g}^{(t)}, \boldsymbol{x}^{(t)} \rangle \mathbf{1}\|_2$ up to time T.

Proof. We start by observing that, at all times t,

$$\left(\boldsymbol{g}^{(t+1)} - \langle \boldsymbol{g}^{(t+1)}, \boldsymbol{x}^{(t+1)} \rangle \mathbf{1}\right)^\top \boldsymbol{x}^{(t+1)} = 0$$

(this is always true, not just for regret matching). Plugging in the definition (1) of how $\boldsymbol{x}^{(t+1)}$ is constructed, we therefore conclude that

$$\left(\boldsymbol{g}^{(t+1)} - \langle \boldsymbol{g}^{(t+1)}, \boldsymbol{x}^{(t+1)} \rangle \mathbf{1}\right)^\top [\boldsymbol{r}^{(t)}]^+ = 0. \tag{2}$$

(Note that the above equation holds trivially when $[\boldsymbol{r}^{(t)}]^+$ and therefore $\boldsymbol{x}^{(t+1)}$ is picked arbitrarily.)

Now, we use the following inequality:

$$\|[\boldsymbol{a}+\boldsymbol{b}]^+\|_2^2 \le \|[\boldsymbol{a}]^+ + \boldsymbol{b}\|_2^2,$$

applied to $\boldsymbol{a} = \boldsymbol{r}^{(t)}$ and $\boldsymbol{b} = \boldsymbol{g}^{(t+1)} - \langle \boldsymbol{g}^{(t+1)}, \boldsymbol{x}^{(t+1)} \rangle \mathbf{1}$. In particular, since by definition $\boldsymbol{a}+\boldsymbol{b} = \boldsymbol{r}^{(t+1)}$, we have

$$\begin{aligned}
\|[\boldsymbol{r}^{(t+1)}]^+\|_2^2 &\le \|[\boldsymbol{r}^{(t)}]^+ + (\boldsymbol{g}^{(t+1)} - \langle \boldsymbol{g}^{(t+1)}, \boldsymbol{x}^{(t+1)} \rangle \mathbf{1})\|_2^2 \\
&= \|[\boldsymbol{r}^{(t)}]^+\|_2^2 + \|\boldsymbol{g}^{(t+1)} - \langle \boldsymbol{g}^{(t+1)}, \boldsymbol{x}^{(t+1)} \rangle \mathbf{1}\|_2^2 + 2\Big(\boldsymbol{g}^{(t+1)} - \langle \boldsymbol{g}^{(t+1)}, \boldsymbol{x}^{(t+1)} \rangle \mathbf{1}\Big)^\top [\boldsymbol{r}^{(t)}]^+ \\
&= \|[\boldsymbol{r}^{(t)}]^+\|_2^2 + \|\boldsymbol{g}^{(t+1)} - \langle \boldsymbol{g}^{(t+1)}, \boldsymbol{x}^{(t+1)} \rangle \mathbf{1}\|_2^2 \qquad \text{(from (2))} \\
&\le \|[\boldsymbol{r}^{(t)}]^+\|_2^2 + \Omega^2.
\end{aligned}$$

Hence, by induction we have

$$\|[\boldsymbol{r}^{(T)}]^+\|_2^2 \le T\Omega^2 \quad \Longrightarrow \quad \mathrm{Reg}^{(T)} = \max_{i \in \{1,\dots,m\}} \boldsymbol{r}^{(t)}[i] \le \max_{i \in \{1,\dots,m\}} \big[\boldsymbol{r}^{(t)}[i]\big]^+ \le \|[\boldsymbol{r}^{(T)}]^+\|_2 \le \Omega\sqrt{T}.$$

The proof is then complete. □

Algorithm 1: Regret matching

$\boldsymbol{r}^0 \leftarrow \mathbf{0} \in \mathbb{R}^n, \quad \boldsymbol{x}^0 \leftarrow \mathbf{1}/m \in \Delta^m$

function NextStrategy()

  if $[\boldsymbol{r}^{(t-1)}]^+ \neq \mathbf{0}$ **return** $\boldsymbol{x}^{(t)} \leftarrow \dfrac{[\boldsymbol{r}^{(t-1)}]^+}{\|[\boldsymbol{r}^{(t-1)}]^+\|_1}$

  else return $\boldsymbol{x}^{(t)} \leftarrow$ any point in Δ^m

function ObserveUtility($\boldsymbol{g}^t$)

  $\boldsymbol{r}^{(t)} \leftarrow \boldsymbol{r}^{(t-1)} + \boldsymbol{g}^{(t)} - \langle \boldsymbol{g}^{(t)}, \boldsymbol{x}^{(t)} \rangle \mathbf{1}$

Algorithm 2: Regret matching$^+$

$\boldsymbol{z}^0 \leftarrow \mathbf{0} \in \mathbb{R}^n, \quad \boldsymbol{x}^0 \leftarrow \mathbf{1}/m \in \Delta^m$

function NextStrategy()

  if $[\boldsymbol{r}^{(t-1)}]^+ \neq \mathbf{0}$ **return** $\boldsymbol{x}^{(t)} \leftarrow \dfrac{[\boldsymbol{r}^{(t-1)}]^+}{\|[\boldsymbol{r}^{(t-1)}]^+\|_1}$

  else return $\boldsymbol{x}^{(t)} \leftarrow$ any point in Δ^m

function ObserveUtility($\boldsymbol{g}^t$)

  $\boldsymbol{z}^{(t)} \leftarrow [\boldsymbol{z}^{(t-1)} + \boldsymbol{g}^{(t)} - \langle \boldsymbol{g}^{(t)}, \boldsymbol{x}^{(t)} \rangle \mathbf{1}]^+$

As of today, regret matching and its variants are still often some of the most practical algorithms for learning in games.

Remark 2.1. One very appealing property of the regret matching algorithm is its *lack of hyperparameters*. It just works "out of the box".

2.1 The regret matching$^+$ (RM$^+$) algorithm

The regret matching$^+$ algorithm was introduced by Tammelin [2014], Tammelin et al. [2015], and is given in Algorithm 2. It differs from RM only on the last line, where a further thresholding is added. That small change has the effect that actions with negative cumulated regret (that is, "bad" actions) are treated as actions with 0 regret. Hence, intuitively, if a bad action were to become good over time, it would take less time for RM$^+$ to notice and act on that change. Because of that, regret matching$^+$ has stronger practical performance and is often preferred over regret matching in the game solving literature.

With a simple modification to the analysis of RM, the same bound as RM can be proven.

Theorem 2.2. The RM$^+$ algorithm (Algorithm 2) is an external regret minimizer, and satisfies the regret bound $R^T \leq \Omega\sqrt{T}$, where Ω is the maximum norm $\|\boldsymbol{g}^{(t)} - \langle \boldsymbol{g}^{(t)}, \boldsymbol{x}^{(t)} \rangle \mathbf{1}\|_2$ up to time T.

3 Follow-the-Regularized-Leader (FTRL)

Another way of obtaining no-regret algorithms is by considering a regularized (i.e., smoothed) version of the follow-the-leader algorithm discussed above. The idea is that, instead of playing by always putting 100% of the probability mass on the action with highest cumulated regret, we look for the distribution that maximizes the expected cumulated regret, *minus* some regularization term that prevents us from putting all the mass on a single action.

More specifically, let $\varphi : \Delta^m \to \mathbb{R}$ be a 1-strongly convex[1] function (with respect to some norm $\|\cdot\|$, for example, the Euclidean norm $\|\cdot\|_2$) with bounded range, and consider the choice of strategy

$$\boldsymbol{x}^{(t)} := \arg\max_{\hat{\boldsymbol{x}} \in \Delta^m} \left\{ (\boldsymbol{r}^{(t-1)})^\top \hat{\boldsymbol{x}} - \frac{1}{\eta} \varphi(\hat{\boldsymbol{x}}) \right\}.$$

(so, in particular $\boldsymbol{x}^{(1)} := \arg\min_{\hat{\boldsymbol{x}} \in \Delta^m} \varphi(\hat{\boldsymbol{x}})$ since the regrets of all actions are 0 at the beginning). This algorithm is called the *follow-the-regularized-leader (FTRL)* algorithm. The following two properties are known about FTRL:

- Regularization *limits the amount of variation between consecutive strategies*. This makes intuitive sense: if $\eta \to 0$, for example, $\boldsymbol{x}^{(t)}$ is constant (and equal to $\arg\min_{\hat{\boldsymbol{x}} \in \Delta^m} \varphi(\hat{\boldsymbol{x}})$). When $\eta = \infty$, we recover the follow-the-leader algorithm, where strategies can jump arbitrarily. For intermediate η, as you might expect, the amount of variation between strategies at consecutive times is bounded above by a quantity proportional to η:

$$\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t)}\| \leq \eta \|\boldsymbol{g}^{(t)}\|_*, \tag{3}$$

  where $\|\cdot\|_*$ is the *dual* norm of $\|\cdot\|$ (if $\|\cdot\|$ is the Euclidean norm, its dual is also the Euclidean norm; if $\|\cdot\|$ is the ℓ_1 norm, then its dual is the ℓ_∞ norm).

- A useful regret bound of FTRL is given by

$$\mathrm{Reg}^{(T)} \leq \max_{\hat{\boldsymbol{x}} \in \Delta^m} \frac{\varphi(\hat{\boldsymbol{x}}) - \varphi(\boldsymbol{x}^{(1)})}{\eta} + \sum_{t=1}^{T} (\boldsymbol{g}^{(t)})^\top \big(\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t)}\big). \tag{4}$$

Using the Cauchy-Schwarz inequality in (4) and plugging in (3), we obtain the following

Theorem 3.1. The regret cumulated by the FTRL algorithm is upper bounded by

$$\mathrm{Reg}^{(T)} \leq \max_{\hat{\boldsymbol{x}} \in \Delta^m} \frac{\varphi(\hat{\boldsymbol{x}}) - \varphi(\boldsymbol{x}^{(1)})}{\eta} + \eta \sum_{t=1}^{T} \|\boldsymbol{g}^{(t)}\|_*^2.$$

Since the norm of the gradient vectors is bounded, it follows immediately that by picking learning rate $\eta = 1/\sqrt{T}$, we obtain that $\mathrm{Reg}^{(T)}$ is bounded as roughly $\sqrt{T}$ (a sublinear function!) at all times T.

One of the appeals of FTRL is that it is a very general method, and in fact it generalized well beyond probability simplexes.

We will now see a particular instantiation of FTRL that is so important and well-studied that it deserves its own name: the *Multiplicative Weights Update* method (MWU).

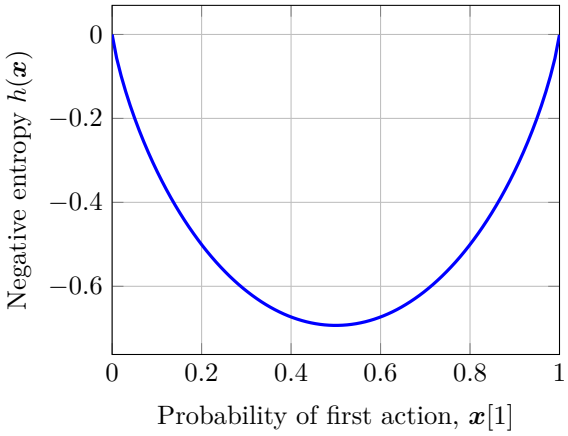
[1] A differentiable function f is 1-strongly convex if for all $\boldsymbol{x}, \boldsymbol{y}$ in its domain one has $(\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}))^\top (\boldsymbol{x} - \boldsymbol{y}) \geq \|\boldsymbol{x} - \boldsymbol{y}\|^2$.

3.1 The Multiplicative Weights Update Algorithm

A natural choice of convex regularizer for probability simplexes is the *negative entropy* function

$$h(\boldsymbol{x}) \coloneqq \sum_{i \in \{1,\dots,m\}} \boldsymbol{x}[i] \log \boldsymbol{x}[i].$$

Example 3.1. For example, this is what negative entropy looks like in the case of $m = 2$ actions:



The negative entropy function has the following properties:

- It is 1-strongly convex with respect to the ℓ_1 norm $\|\cdot\|_1$.
- The maximum of the function is 0, which is attained at any deterministic strategy.
- Its minimum (that is, maximum entropy) is attained at the uniformly random strategy $\bar{\boldsymbol{x}} = (1/m, \dots, 1/m)$, at which the function has value $h(\bar{\boldsymbol{x}}) = m \cdot 1/m \log 1/m = -\log m$.

In this case, then, the MWU algorithm reduces to the following:

$$\boldsymbol{x}^{(1)} \coloneqq (1/m, \dots, 1/m),$$

$$\boldsymbol{x}^{(t)} \coloneqq \arg\max_{\hat{\boldsymbol{x}} \in \Delta^m} \left\{ (\boldsymbol{r}^{(t-1)})^\top \hat{\boldsymbol{x}} - \frac{1}{\eta} h(\hat{\boldsymbol{x}}) \right\} = \arg\max_{\hat{\boldsymbol{x}} \in \Delta^m} \left\{ (\boldsymbol{r}^{(t-1)})^\top \hat{\boldsymbol{x}} - \frac{1}{\eta} \sum_{i=1}^{m} \boldsymbol{x}[i] \log \boldsymbol{x}[i] \right\}. \tag{5}$$

As it turns out via a simple application of the Lagrange multiplier theorem, the concave optimization problem in (5) has a closed-form solution:

Proposition 3.1. The concave optimization problem (5) has the closed-form solution

$$\boldsymbol{x}^{(t)} = \mathrm{softmax}\{\eta \boldsymbol{r}^{(t)}\},$$

that is, for every action $i \in \{1, \dots, m\}$,

$$\boldsymbol{x}^{(t)}[i] = \frac{e^{\eta \boldsymbol{r}^{(t)}[i]}}{\sum_{j=1}^{m} e^{\eta \boldsymbol{r}^{(t)}[j]}}.$$

Algorithm 3: Multiplicative Weights Update

$\boldsymbol{r}^0 \leftarrow \mathbf{0} \in \mathbb{R}^n, \quad \boldsymbol{x}^0 \leftarrow \mathbf{1}/m \in \Delta^m$

function NextStrategy()
  return $\boldsymbol{x}^{(t)} = \text{softmax}(\eta \boldsymbol{r}^{(t-1)})$

function ObserveUtility($\boldsymbol{g}^t$)
  $\boldsymbol{r}^{(t)} \leftarrow \boldsymbol{r}^{(t-1)} + \boldsymbol{g}^{(t)} - \langle \boldsymbol{g}^{(t)}, \boldsymbol{x}^{(t)} \rangle \mathbf{1}$

This gives rise to Algorithm 3, which has a similar flavor as RM, but uses a different prioritization of actions (in particular, it uses *softmax instead of ReLU*). From the general analysis of FTRL (Theorem 3.1), remembering that:

- the maximum of h is 0,
- the minimum (attained in $\boldsymbol{x}^{(1)}$) is $-\log m$,
- negative entropy is 1-strongly convex with respect to $\|\cdot\|_1$ and that the dual norm to $\|\cdot\|_1$ is $\|\cdot\|_\infty$,

we obtain the following.

Corollary 3.1. The regret cumulated by the MWU algorithm is upper bounded by

$$\text{Reg}^{(T)} \leq \frac{\log m}{\eta} + \eta \sum_{t=1}^{T} \|\boldsymbol{g}^{(t)}\|_\infty^2.$$

References

Oskari Tammelin. Solving large imperfect information games using CFR+, 2014.

Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit Texas hold'em. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.