MIT 6.S890 — Topics in Multiagent Learning Thu, Sep 18th 2024

Lecture 5
Learning in games: Algorithms (Part I)

Instructor: Prof. Gabriele Farina (✉ gfarina@mit.edu)[*]

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. In this lecture, we begin exploring how no-external-regret dynamics can be constructed, starting from normal-form games.

1 No-regret algorithms for normal-form games

For a player in normal-form games the strategy space corresponds to the probability simplex $\Delta(A)$ of all distributions over the actions available to the player. As a reminder, constructing a regret minimizer for $\Delta(A)$ 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 $x^{(t)} \in \Delta(A)$;
- `ObserveUtility`$\big(u^{(t)}\big)$ 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

  $$u^{(t)} : x \mapsto \langle g^{(t)}, x \rangle,$$

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

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

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

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

General principle. 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

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

(Note that $\mathrm{Reg}^{(t)} = \max_{a \in A} r_a^{(t)}$.) The natural question is: *how to prioritize?*

[*] These notes are class material that has not undergone formal peer review. The TAs and I are grateful for any reports of typos.

1.1 Follow-the-leader

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

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

In this case, at time 1, we pick the first action and score a utility of 0. We then compute $r^{(1)} = (0, 1/2)$ 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 $r^{(2)} = (1, 1/2)$ So, at time $t = 3$, we will pick the first action, resulting again in a score of 0 and an updated regret $r^{(3)} = (1, 3/2)$ and so on... Overall, it's easy to see that in all this jumping around, the regrets grow linearly.

Where to go from here? A few ideas seem natural:

- We can replace picking the action with the highest regret with picking actions *proportionally* to their regret; this leads to the algorithm called *regret matching*, which we will discuss in Section 1.2.
- We can *smooth out* the maximum operator by using the *softmax* function. This leads to the *multiplicative weights update* algorithm, which we will discuss in Section 1.4.
- We can *regularize* the maximum operator by adding a term that penalizes large jumps in the strategy space. This leads to a very flexible algorithm called *follow-the-regularized-leader* algorithm, which we will discuss in Section 2 as well as in the next lecture.

All these ideas work. Before we move on, though, it is worth knowing that—while flawed—the follow-the-leader algorithm is not completely hopeless.

Remark 1.1 (Fictitious play). In the canonical learning setup (see Lecture 4), the use of follow-the-leader by all players goes under the name of *fictitious play* [Bro49; Bro51]. In certain classes of games, including two-player zero-sum games, fictitious play is able to recover a Nash equilibrium, albeit with a potentially exponentially slow convergence rate [Rob51].

1.2 The Regret Matching (RM) algorithm

The regret matching algorithm [HM00] picks probabilities proportional to the "ReLU" of the cumulated regrets, that is,

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

whenever $[r^{(t)}]^+ \neq 0$, and an arbitrary point otherwise. The algorithm is presented in pseudocode in Algorithm 1.

Algorithm 1: Regret matching (RM)

```
$r^{(0)} \leftarrow 0 \in \mathbb{R}^A, \quad x^{(0)} \leftarrow \mathbf{1}/|A| \in \Delta(A)$
function NextStrategy()
    if $\left[r^{(t-1)}\right]^+ \neq 0$
        return $x^{(t)} \leftarrow \dfrac{[r^{(t-1)}]^+}{\left\|[r^{(t-1)}]^+\right\|_1}$
    else
        return $x^{(t)} \leftarrow$ any strategy in $\Delta(A)$
function ObserveUtility($g^{(t)}$)
    $r^{(t)} \leftarrow r^{(t-1)} + g^{(t)} - \langle g^{(t)}, x^{(t)} \rangle \mathbf{1}$
```

Algorithm 2: Regret matching⁺ (RM⁺)

```
$r^{(0)} \leftarrow 0 \in \mathbb{R}^A, \quad x^{(0)} \leftarrow \mathbf{1}/|A| \in \Delta(A)$
function NextStrategy()
    if $\left[r^{(t-1)}\right]^+ \neq 0$
        return $x^{(t)} \leftarrow \dfrac{[r^{(t-1)}]^+}{\left\|[r^{(t-1)}]^+\right\|_1}$
    else
        return $x^{(t)} \leftarrow$ any strategy in $\Delta(A)$
function ObserveUtility($g^{(t)}$)
    $r^{(t)} \leftarrow [r^{(t-1)} + g^{(t)} - \langle g^{(t)}, x^{(t)} \rangle \mathbf{1}]^+$
```

Theorem 1.1 (Regret bound for RM). The regret matching algorithm (Algorithm 1) is an external regret minimizer, and satisfies the regret bound $\mathrm{Reg}^{(T)} \le \Omega\big(\sqrt{T}\big)$, where Ω is the maximum norm of $\big\|g^{(t)} - \langle g^{(t)}, x^{(t)}\rangle \mathbf{1}\big\|_2$ up to time T.

In particular, if all the gradient vectors satisfy $\big\|g^{(t)}\big\|_\infty \le 1$ at all times t then the regret satisfies

$$\mathrm{Reg}^{(T)} \le \sqrt{T \cdot |A|}.$$

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

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

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

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

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

Now, we use the following inequality:

$$\big\|[a+b]^+\big\|_2^2 \le \big\|[a]^+ + b\big\|_2^2,$$

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

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

Hence, by induction we have

$$\big\|[r^{(T)}]^+\big\|_2^2 \le T\Omega^2 \qquad \Longrightarrow \qquad \mathrm{Reg}^{(T)} = \max_{a \in A} r_a^{(t)} \le \max_{a \in A}\ [r_a^{(T)}]^+ \le \big\|[r^{(T)}]^+\big\|_2 \le \Omega\sqrt{T}.$$

The proof of the first part is then complete. The second part then just follows from using the inequality

$$\big\|g^{(t)} - \langle g^{(t)}, x^{(t)}\rangle \mathbf{1}\big\|_2 \le \big\|g^{(t)}\big\|_2 \le \sqrt{|A|} \cdot \big\|g^{(t)}\big\|_\infty.$$

□

Our interest for the regret bound under the specific condition that $\big\|g^{(t)}\big\|_\infty \le 1$ is as follows.

Remark 1.2. Within the canonical learning setup (see Lecture 4), the entries of $g^{(t)}$ are the expected payoffs of all actions of the players. Thus, the condition $\big\|g^{(t)}\big\|_\infty \le 1$ corresponds to the condition that the game's payoffs lie in the interval $[-1, 1]$.

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

Remark 1.3. One very appealing property of the regret matching algorithm is its *lack of hyperparameters*. It just works "out of the box".

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

The regret matching$^+$ algorithm [Tam14; Tam+15] 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 1.2 (Regret bound for RM$^+$). The RM$^+$ algorithm (Algorithm 2) is an external regret minimizer, and satisfies the regret bound $\mathrm{Reg}^{(T)} \le \Omega\sqrt{T}$, where Ω is the maximum norm $\left\|g^{(t)} - \langle g^{(t)}, x^{(t)}\rangle \mathbf{1}\right\|_2$ up to time T.

So again, if all the gradient vectors satisfy $\left\|g^{(t)}\right\|_\infty \le 1$ at all times t then the regret satisfies

$$\mathrm{Reg}^{(T)} \le \sqrt{T \cdot |A|}.$$

1.4 Multiplicative weights update (MWU)

If we replace the "hard" maximum of follow-the-leader with the "soft" maximum given by

$$x_a^{(t)} = \mathrm{softmax}_a\left(\eta r^{(t)}\right) := \frac{\exp\left(\eta r_a^{(t)}\right)}{\sum_{j=1}^m \exp(\eta r^{(t)}[j])},$$

where $\eta > 0$ is an inverse temperature parameter, then we obtain the *multiplicative weights update* algorithm [FS97]. This algorithm is presented in Algorithm 3.

Algorithm 3: Multiplicative weights update (MWU)

```
$r^{(0)} \leftarrow 0 \in \mathbb{R}^A, \quad x^{(0)} \leftarrow \mathbf{1}/|A| \in \Delta(A)$
function NextStrategy()
    return $x^{(t)} \leftarrow \mathrm{softmax}\big(\eta r^{(t-1)}\big)$
function ObserveUtility($g^{(t)}$)
    $r^{(t)} \leftarrow r^{(t-1)} + g^{(t)} - \langle g^{(t)}, x^{(t)}\rangle \mathbf{1}$
```

Compared to RM, MWU has a different flavor: it uses *softmax* instead of ReLU. This change in prioritization function has a pretty significant impact on the regret bound that multiplicative weights guarantees. In particular, the following can be shown:

Theorem 1.3 (Regret bound for MWU). The regret cumulated by the MWU algorithm can be upper bounded as

$$\mathrm{Reg}^{(T)} \le \frac{\log|A|}{\eta} + \eta \sum_{t=1}^{T} \left\|g^{(t)}\right\|_\infty^2 - \frac{1}{8\eta} \sum_{t=2}^{T} \left\|x^{(t)} - x^{(t-1)}\right\|_1^2.$$

no matter the sequence of gradient vectors $g^{(t)}$ chosen by the environment. In particular, if all the gradient vectors satisfy $\left\|g^{(t)}\right\|_\infty \le 1$ at all times t and $\eta = \sqrt{\log|A|/T}$, the regret satisfies

$$\mathrm{Reg}^{(T)} \le \sqrt{T \log|A|}.$$

We will see the proof of Theorem 1.3 in the next lecture, as a reflection of a substantially more general framework. For now, we remark a crucial aspect of MWU. Compared with the regret bound of RM and

RM$^+$, the regret bound of MWU has only a *logarithmic* dependence on the number of actions $|A|$. Despite in practice RM/RM$^+$ tend to outperform MWU (all while getting rid of any hyperparameter tuning), this property has profound *theoretical* implications. We will discuss this in more depth when discussing combinatorial games in Lecture 19. The gist of it is that several important classes of games can be converted into *exponentially large* normal-form games (this is the case of sequential games, for example). Since MWU only has logarithmic dependence on the number of actions of the resulting normal-form games, this shows that —at least ignoring computation—external regret minimization is possible with polynomial dependence on the game size even in these classes of complex, structured games.

2 More general approaches: FTRL and OMD

Finally, we turn our attention to the third way of obtaining no-regret algorithms, that 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.

2.1 FTRL in the normal-form case

Definition 2.1 (FTRL, simplex case). Let $\psi : \Delta(A) \to \mathbb{R}$ be a 1-strongly convex[1] function (with respect to some norm $\|\cdot\|$, for example, the Euclidean norm $\|\cdot\|_2$). The follow-the-*regularized*-leader algorithm (FTRL; sometimes also called "regularized follow-the-leader") defines the choice of strategy[2]

$$x^{(t)} := \arg\max_{\hat{x}\in\Delta(A)} \left\{ \langle r^{(t-1)}, \hat{x} \rangle - \frac{1}{\eta}\psi(\hat{x}) \right\}.$$

The regularization term $-\psi(\hat{x})$ *limits the amount of variation between consecutive strategies.* This makes intuitive sense: if $\eta \to 0$, for example, $x^{(t)}$ is constant (and equal to $\arg\min_{\hat{x}\in\Delta(A)} \psi(\hat{x})$). 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:

$$\|x^{(t)} - x^{(t)}\| \le \eta \|g^{(t)}\|_*,$$

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). We will see the regret bound for FTRL in the general case in Section 2.3.

2.2 OMD in the normal-form case

The last general method that we mention today is the *online mirror descent (OMD)* algorithm. This is the online generalization of the mirror descent optimization method, one of the workhorses of modern optimization theory. A good way to think about OMD is as a generalization of gradient ascent.

Definition 2.2 (OMD, simplex case). Let $\psi : \Delta(A) \to \mathbb{R}$ be a 1-strongly convex function. The online mirror descent algorithm (OMD) defines the choice of strategy

$$x^{(t)} := \arg\max_{\hat{x}\in\Delta(A)} \left\{ \langle g^{(t-1)}, \hat{x} \rangle - \frac{1}{\eta}\mathrm{D}_\psi\left(\hat{x} \,\|\, x^{(t-1)}\right) \right\}, \quad \text{where} \quad \mathrm{D}_\psi(\hat{x} \,\|\, x) := \psi(\hat{x}) - \psi(x) - \langle \nabla\psi(x), \hat{x} - x \rangle.$$

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

[2] In particular, $x^{(1)} := \arg\min_{\hat{x}\in\Delta(A)} \psi(\hat{x})$ since the regrets of all actions are 0 at the beginning.

In the special case in which $\psi(x) := \frac{1}{2}\|x\|_2^2$, then the OMD algorithm reduces to online projected gradient descent:

$$x^{(t)} = \arg\max_{\hat{x}\in\Delta(A)} \left\{ \langle g^{(t-1)}, \hat{x}\rangle - \frac{1}{\eta}\left\|\hat{x} - x^{(t-1)}\right\|_2^2 \right\} = \Pi_{\mathcal{X}}\left(g^{(t-1)} + \eta g^{(t-1)}\right).$$

2.3 The general case

FTRL and OMD apply well beyond the case of probability simplices. In fact, they can be applied to any convex and compact domain $\mathcal{X}$. In this case, the vector of regrets $r^{(t)}$ must be replaced with the cumulative gradient vector $\sum_{\tau=1}^{t} g^{(\tau)}$, as follows:

Definition 2.3 (FTRL, general version). For a generic convex and compact domain $\mathcal{X}$, the FTRL algorithm produces strategies $x^{(t)}$ by solving the optimization problem

$$x^{(t)} := \arg\max_{\hat{x}\in\mathcal{X}} \left\{ \left\langle \sum_{\tau=1}^{t-1} g^{(\tau)}, \hat{x} \right\rangle - \frac{1}{\eta}\psi(\hat{x}) \right\}.$$

Definition 2.4 (OMD, general version). The OMD algorithm produces strategies $x^{(t)}$ by solving the optimization problem

$$x^{(t)} := \arg\max_{\hat{x}\in\mathcal{X}} \left\{ \langle g^{(t-1)}, \hat{x}\rangle - \frac{1}{\eta}\mathrm{D}_{\psi}\left(\hat{x} \,\|\, x^{(t)}\right) \right\}.$$

We mention the following regret bound for the general case. We will mention an even more powerful result next time.

Theorem 2.1 (Regret bound for FTRL and OMD). The regret cumulated by the FTRL and OMD algorithms is upper bounded by

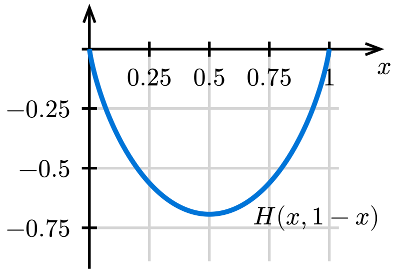
$$\mathrm{Reg}^{(T)} \le \max_{\hat{x}\in\mathcal{X}} \frac{\psi(\hat{x}) - \psi(x^{(1)})}{\eta} + \eta\sum_{t=1}^{T} \left\|g^{(t)}\right\|_*^2 - \frac{1}{8\eta}\sum_{t=2}^{T}\left\|x^{(t)} - x^{(t-1)}\right\|^2,$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. In particular, if the norm of the gradient vectors is bounded, then by picking learning rate $\eta = \frac{1}{\sqrt{T}}$, we obtain that $\mathrm{Reg}^{(T)}$ is bounded as roughly $\sqrt{T}$ (a sublinear function!) at all times T.

2.4 Multiplicative Weights Update algorithm as a special case

Multiplicative weights update is the special case of *both FTRL and OMD* in which the regularizer ψ is set to the *negative entropy* function

$$H(x) := \sum_{a\in A} x_a \log x_a.$$



Example 2.1. The plot on the right displays the negative entropy function in the case of $|A| = 2$ 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

  $$\overline{x} = (1/m, ..., 1/m),$$

  at which $H(\overline{x}) = m \cdot \frac{1}{m} \log \frac{1}{m} = -\log m$.

Bibliography

[Bro49] G. W. Brown, *Some notes on computation of games solutions*. Rand Corporation, 1949.

[Bro51] G. W. Brown, “Iterative solution of games by fictitious play,” *Act. Anal. Prod Allocation*, vol. 13, no. 1, p. 374–375, 1951.

[Rob51] J. Robinson, “An iterative method of solving a game,” *Annals of Mathematics*, vol. 54, no. 2, pp. 296–301, 1951, [Online]. Available: http://www.jstor.org/stable/1969530

[HM00] S. Hart and A. Mas-Colell, “A Simple Adaptive Procedure Leading to Correlated Equilibrium,” *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2000, [Online]. Available: http://www.jstor.org/stable/2999445

[Tam14] O. Tammelin, “Solving large imperfect information games using CFR+,” *arXiv*, 2014, [Online]. Available: https://arxiv.org/abs/1407.5042

[Tam+15] O. Tammelin, N. Burch, M. Johanson, and M. Bowling, “Solving Heads-up Limit Texas Hold'em,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[FS97] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.