Online Optimization, Learning, and Games (O2LG)
Lesson 5: Introduction to Online Learning

Vinh Thanh Ho*, Panayotis Mertikopoulos

*Faculté des Sciences et Techniques
Université de Limoges
vinh-thanh.ho@unilim.fr

ACSYON
Master of Applied Mathematics

Table of Contents

Recall: Game-theoretic learning

Recall: Game-theoretic learning

Require: a finite game $\Gamma \equiv \Gamma(\mathcal{N}, \mathcal{A}, u)$.
  repeat for each epoch $t = 0, 1, 2, \ldots$, for all players $i \in \mathcal{N}$, **do**
    Choose mixed strategy $x_i(t) \in \mathcal{X}_i$. ▷ mixing
    Observe mixed payoff vector $u_i(x(t)) = \langle v_i(x(t)), x_i(t) \rangle$. ▷ feedback phase
  until end

Online learning scheme

Require: set of actions $\mathcal{A} = \{1, \ldots, A\}$, stream of payoff vectors $v_t \in [0, 1]^{\mathcal{A}}$, $t \geq 0$.
  repeat for each epoch $t \geq 0$ **do**
    Choose mixed strategy $x_t \in \mathcal{X}$. ▷ mixing
    Encounter payoff vector v_t and get mixed payoff $u_t(x_t) = \langle v_t, x_t \rangle$. ▷ feedback phase
  until end

Features: *continuous* time, *single* player, and *exogenous* payoffs.

Online learning versus Multi-agent learning

How are payoffs generated?

- Multi-agent viewpoint
  - Multiple agents.
  - Game-theoretic: underlying mechanism is a (finite) game.
  - Endogenous rewards: individual payoffs depend on other agents.
- Online viewpoint
  - Single agent.
  - Agnostic: no assumptions on mechanism generating $v(t)$.
  - Exogenous rewards: different payoff vector at each stage.

Online learning versus Multi-agent learning

What is the **interplay** between online learning and multi-agent learning?

- Online learning can be used to learn in multi-agent environments.
  - For example, an agent can learn to play a game against other agents by using online learning to update its strategy based on the actions of the other agents.
- Multi-agent learning can be used to improve the performance of online learning algorithms.
  - For example, multiple agents can cooperate to explore the environment and learn more quickly.
- Online learning and multi-agent learning can be used together to design new learning algorithms.

Table of Contents

The agent's regret

Require: set of actions $\mathcal{A} = \{1, \ldots, A\}$, stream of payoff vectors $v_t \in [0,1]^{\mathcal{A}}$, $t \geq 0$

repeat for each epoch $t \geq 0$ **do**

Choose mixed strategy $x_t \in \mathcal{X}$. $\triangleright$ mixing

Encounter payoff vector v_t and get mixed payoff $u_t(x_t) = \langle v_t, x_t \rangle$. $\triangleright$ feedback phase

until end

How to measure the performance of a policy x_t?

We define the agent's regret.

The agent's regret

Performance of a policy x_t is measured by the agent's (external) regret:

$$\mathrm{Reg}(T) = \max_{p \in \mathcal{X}} \int_0^T [u_t(p) - u_t(x_t)]\, dt = \max_{p \in \mathcal{X}} \int_0^T \langle v_t, p - x_t \rangle\, dt.$$

No regret: $\mathrm{Reg}(T) = o(T)$ # the smaller the better

"The chosen policy is as good as the best fixed strategy in hindsight."

Extensive body of work:

- Economics (e.g. Hannan 1958).
- Mathematics (e.g. Blackwell 1956, Bubeck et al. 2012).
- Computer science (e.g. Shalev-Shwartz 2012, Cesa-Bianchi et al. 2006).

Exponential weights for online learning

Exponential Weight Dynamics (EWD)

$$\dot{y}_t = v_t, \qquad x_t = \Lambda(y_t) \tag{EWD}$$

where $\Lambda\colon \mathbb{R}^{\mathcal{A}} \to \mathcal{X}$ is a logit map defined by

$$\Lambda_a(y) := \frac{\exp(y_a)}{\sum_{a' \in \mathcal{A}} \exp(y_{a'})}.$$

Does (EWD) lead to no regret?

Bounding the regret

- Fix a comparator $p \in \mathcal{X}$.
- The associated regret is computed by:

$$\mathrm{Reg}_p(T) = -\int_0^T \langle \dot{y}_t, \Lambda(y_t) - p \rangle \, dt.$$

- Suppose that there exists a potential function Φ such that

$$\nabla\Phi(y) = \Lambda(y) - p \implies \frac{d\Phi}{dt} = \langle \dot{y}_t, \Lambda(y_t) - p \rangle.$$

Then

$$\mathrm{Reg}_p(T) = -\int_0^T \frac{d\Phi}{dt} \, dt = \Phi(y_0) - \Phi(y_T).$$

If suitable potential exists $\implies$ $\mathrm{Reg}_p(T) \leq \Phi(y_0) - \min_y \Phi(y)$.

Potential function

Task

1) How to find such a potential function as described above?
2) What is the minimum value of the potential?

Table of Contents

Energy functions

We can encode the above with the help of the following energy functions:

- The Fenchel coupling:

$$F(p, y) = \sum_{a \in \mathcal{A}} p_a \log p_a + \log \sum_{a \in \mathcal{A}} \exp(y_a) - \sum_{a \in \mathcal{A}} p_a y_a.$$

- Defining $x := \Lambda(y)$ yields the Kullback–Leibler (KL) divergence:

$$D_{\mathrm{KL}}(p, x) = \sum_{a \in \mathcal{A}} p_a \log \frac{p_a}{x_a}.$$

Key property: $$\frac{d}{dt} F(p, y_t) = \langle v_t, x_t - p \rangle.$$

Regret of (EWD)

Theorem 1 (Sorin 2009)

Under (EWD), *the learner enjoys the regret bound*

$$\mathrm{Reg}_p(T) \leq F(p, y_0) = \sum_{a \in \mathcal{A}} p_a \log p_a + \log \sum_{a \in \mathcal{A}} \exp(y_{a,0}) - \sum_{a \in \mathcal{A}} p_a y_{a,0}.$$

In particular, if (EWD) *is initialized with* $y_0 = 0$*, we have*

$$\mathrm{Reg}(T) \leq \log A.$$

Summary

This lesson

- Introduction to online optimization
- Regret and regret minimization
- Fenchel coupling / Kullback–Leibler divergence
- $O(1)$ regret in continuous time

Next lesson

- Online learning in discrete time
- Multi-armed bandits

References

[1] David Blackwell. An analog of the minimax theorem for vector payoffs. In: *Pacific Journal of Mathematics* 6.1 (1956), pp. 1–8 (cited at slide -7).

[2] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. In: *Foundations and Trends®in Machine Learning* 5.1 (2012), pp. 1–122 (cited at slide -7).

[3] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006 (cited at slide -7).

[4] James Hannan. Approximation to Bayes risk in repeated play. In: *Contributions to the Theory of Games (AM-39), Volume III*. Princeton: Princeton University Press, 1958, pp. 97–140 (cited at slide -7).

[5] Shai Shalev-Shwartz. Online Learning and Online Convex Optimization. In: *Foundations and Trends® in Machine Learning* 4.2 (2012), pp. 107–194 (cited at slide -7).

[6] Sylvain Sorin. Exponential weight algorithm in continuous time. In: *Mathematical Programming* 116.1-2 (Apr. 2009), pp. 513–528 (cited at slide -1).