

Algorithmic game theory

Martin Balko

6th lecture

November 11th 2025



Regret minimization

Regret minimization

- We introduce a completely new model of interactions based on so-called **regret minimization**. We apply **online learning**.



Sources: <https://blogger.googleusercontent.com/>

- Later, we apply these new methods to design new fast algorithms to **approximate correlated equilibria**.
- Today, we introduce the model and some basic algorithms on how to minimize regret.

The setting

- Since we are introducing a new model, we will need some notation.
- We have an agent A in an adversary environment.
- There are N available actions for A in the set $X = \{1, \dots, N\}$.
- At each step $t = 1, \dots, T$:
 - Our agent A selects a probability distribution $p^t = (p_1^t, \dots, p_N^t)$ over X , where p_i^t is the probability that A selects i in step t .
 - Then, the adversary chooses a loss vector $\ell^t = (\ell_1^t, \dots, \ell_N^t)$, where $\ell_i^t \in [-1, 1]$ is the loss of action i in step t .
 - The agent A then experiences loss $\ell_A^t = \sum_{i=1}^N p_i^t \ell_i^t$. This is the expected loss of A in step t .
- After T steps, the cumulative loss of action i is $L_i^T = \sum_{t=1}^T \ell_i^t$.
- The cumulative loss of A is $L_A^T = \sum_{t=1}^T \ell_A^t$.

External regret

- We need to be able to tell how well is our agent doing. We choose an “external approach” and compare his loss to the loss of the best agent from some **comparison class** $\mathcal{A}$.
- We will mostly consider the class $\mathcal{A}_X = \{A_i : i \in X\}$, where an agent A_i always chooses action i .
- Let $R_A^T = L_A^T - \min\{L_B^T : B \in \mathcal{A}_X\}$ be the **external regret** of A (with respect to $\mathcal{A}_X$). That is, $R_A^T = L_A^T - \min\{L_i^T : i \in X\}$
- Until specified otherwise, we consider only loss vectors from $\{0, 1\}^N$. This is only to simplify the notation, all presented results can be extended to the general case.

Example

No Regret Learning (review)

No single action significantly outperforms the dynamic.



	
0	1
1	0

Weather					Loss
Algorithm					1
Umbrella					1
Sunscreen					3

Is the setting too restrictive?

- It might seem that the class $\mathcal{A}_X$ contains too simple agents. However, we show that **large comparison classes lead to a very large regret**.
- Let $\mathcal{A}_{all}$ be the set of agents that assign probability 1 to an arbitrary action from X in every step.
 - In $\mathcal{A}_X$ each agent has to select the same action in all steps.

Observation 2.45

For any agent A and every $T \in \mathbb{N}$, there is a sequence of T loss vectors and an agent $B \in \mathcal{A}_{all}$ such that $L_A^T - L_B^T \geq T(1 - 1/N)$.

- That is almost as bad as it can get.
- **Proof:** For every t , let i_t be the action with the lowest probability $p_{i_t}^t$. We set $\ell_{i_t}^t = 0$ and $\ell_i^t = 1$ for every $i \neq i_t$.
- Since $p_{i_t}^t \leq 1/N$, we have $\ell_A^t \geq 1 - 1/N$ and thus the cumulative loss L_A^T of A after T steps is at least $T(1 - 1/N)$.
- The algorithm $B \in \mathcal{A}_{all}$ that selects the action i_t in step t with probability 1 has the cumulative loss $L_B^T = 0$. □

Greedy algorithm

- So we are good with the comparison class $\mathcal{A}_X$. How to **design an agent A** that performs well against agents from $\mathcal{A}_X$?
- We first try a natural **greedy approach**: select an action $i \in X$ for which the cumulative loss L_i^{t-1} at step $t - 1$ is the smallest.

Algorithm 0.12: GREEDY ALGORITHM(X, T)

Input : A set of actions $X = \{1, \dots, N\}$ and number of steps $T \in \mathbb{N}$.

Output : A probability distribution p^t for every $t \in \{1, \dots, T\}$.

$p^1 \leftarrow (1, 0, \dots, 0),$

for $t = 2, \dots, T$

do
$$\begin{cases} L_{\min}^{t-1} \leftarrow \min_{j \in X} \{L_j^{t-1}\}, \\ S^{t-1} \leftarrow \{i \in X : L_i^{t-1} = L_{\min}^{t-1}\}, \\ k \leftarrow \min S^{t-1}, \\ p_k^t \leftarrow 1, p_i^t \leftarrow 0 \text{ for } i \neq k, \end{cases}$$

Output $\{p^t : t \in \{1, \dots, T\}\}.$

Analysis of the Greedy algorithm

Proposition 2.46

For any sequence of $\{0, 1\}$ -valued loss vectors, the cumulative loss L_{Greedy}^T of the Greedy algorithm at time $T \in \mathbb{N}$ satisfies

$$L_{\text{Greedy}}^T \leq N \cdot L_{\min}^T + (N - 1).$$

- **Proof:** At step t , if the Greedy algorithm incurs a loss of 1 and $L_{\min}^t$ does not increase, then at least one action disappears from S^t in the next step. This occurs at most N times and then $L_{\min}^t$ increases by 1.
- Thus, the Greedy algorithm incurs a loss of at most N between successive increments of $L_{\min}^t$ by 1. It follows that

$$L_{\text{Greedy}}^T \leq N \cdot L_{\min}^T + N - |S^T| \leq N \cdot L_{\min}^T + (N - 1).$$

□

- This is **rather weak** since A can perform roughly N times worse than the best action.

Randomized Greedy algorithm

- There is a good reason for the poor behavior. **No deterministic algorithm can perform significantly better** (Exercise).
- So it makes sense to introduce some **randomness**. We break ties at random, splitting weights between the currently best actions.

Algorithm 0.25: RANDOMIZED GREEDY ALGORITHM(X, T)

Input : A set of actions $X = \{1, \dots, N\}$ and number of steps $T \in \mathbb{N}$.

Output : A probability distribution p^t for every $t \in \{1, \dots, T\}$.

$p^1 \leftarrow (1/N, \dots, 1/N),$

for $t = 2, \dots, T$

do
$$\begin{cases} L_{\min}^{t-1} \leftarrow \min_{j \in X} \{L_j^{t-1}\}, \\ S^{t-1} \leftarrow \{i \in X : L_i^{t-1} = L_{\min}^{t-1}\}, \\ p_i^t \leftarrow 1/|S^{t-1}| \text{ for every } i \in S^{t-1} \text{ and } p_i^t \leftarrow 0 \text{ otherwise.} \end{cases}$$

Output $\{p^t : t \in \{1, \dots, T\}\}$.

Analysis of the Randomized greedy algorithm

Proposition 2.48

For any sequence of $\{0, 1\}$ -valued loss vectors, the cumulative loss L_{RG}^T of the Randomized greedy algorithm at time $T \in \mathbb{N}$ satisfies

$$L_{\text{RG}}^T \leq (1 + \ln N) \cdot L_{\min}^T + \ln N.$$

- **Proof** (sketch): We proceed as in the previous proof. For $j \in \mathbb{N}$, let t_j be the time step t at which the loss $L_{\min}^t$ first reaches value j . We estimate the **loss of the algorithm between steps t_j and t_{j+1}** .
- Note that $1 \leq |S^t| \leq N$. If the size of S^t shrinks by k from n' to $n' - k$ at some time $t \in (t_j, t_{j+1}]$, then the loss of the algorithm at step t is k/n' , since the weight of each such action is $1/n'$.
- Clearly, $k/n' \leq 1/n' + 1/(n' - 1) + \dots + 1/(n' - k + 1)$, so we obtain that the loss for the entire time interval $(t_j, t_{j+1}]$ is at most $1/N + 1/(N - 1) + \dots + 1/1 \leq 1 + \ln N$. It follows that

$$L_{\text{RG}}^T \leq (1 + \ln N) \cdot L_{\min}^T + (1/N + 1/(N - 1) + \dots + 1/(|S^T| + 1)).$$



Polynomial weights algorithm

- This is better, but still not optimal. The losses are greatest when the sets S^t are small since the loss can be viewed as proportional to $1/|S^t|$.
- We overcome this issue by assigning larger **weights** to actions that are close to the best one.

Algorithm 0.37: POLYNOMIAL WEIGHTS ALGORITHM(X, T, η)

Input : A set of actions $X = \{1, \dots, N\}$, $T \in \mathbb{N}$, and $\eta \in (0, 1/2]$.

Output : A probability distribution p^t for every $t \in \{1, \dots, T\}$.

$w_i^1 \leftarrow 1$ for every $i \in X$,

$p^1 \leftarrow (1/N, \dots, 1/N)$,

for $t = 2, \dots, T$

do
$$\begin{cases} w_i^t \leftarrow w_i^{t-1}(1 - \eta \ell_i^{t-1}), \\ W^t \leftarrow \sum_{i \in X} w_i^t, \\ p_i^t \leftarrow w_i^t / W^t \text{ for every } i \in X. \end{cases}$$

Output $\{p^t : t \in \{1, \dots, T\}\}$.

Analysis of the Polynomial weights algorithm I

Theorem 2.49

For $\eta \in (0, 1/2]$, every sequence of $[-1, 1]$ -valued loss vectors, and every $k \in X$, the cumulative loss L_{PW}^T of the Polynomial weights algorithm satisfies

$$L_{PW}^T \leq L_k^T + \eta Q_k^T + \ln N / \eta,$$

where $Q_k^T = \sum_{t=1}^T (\ell_k^t)^2$. In particular, if $T \geq 4 \ln N$, then by setting $\eta = \sqrt{\ln N / T}$ and noting that $Q_k^T \leq T$, we obtain

$$L_{PW}^T \leq L_{min}^T + 2\sqrt{T \ln N}.$$

- **Proof** (sketch): We show that if there is a significant loss, then the total weight W^t must drop substantially. For step t , we have $\ell_{PW}^t = \sum_{i=1}^N w_i^t \ell_i^t / W^t$, that is, ℓ_{PW}^t is the expected loss at step t .
- The weight w_i^t of every action i is multiplied by $(1 - \eta \ell_i^{t-1})$ at step t . Thus, $W^{t+1} = W^t - \sum_{i=1}^N \eta w_i^t \ell_i^t = W^t(1 - \eta \ell_{PW}^t)$.

Analysis of the Polynomial weights algorithm II

- Using $W^1 = N$ and $1 - z \leq e^{-z}$ for every $z \in \mathbb{R}$, we obtain

$$W^{T+1} = W^1 \prod_{t=1}^T (1 - \eta \ell_{PW}^t) \leq N \prod_{t=1}^T e^{-\eta \ell_{PW}^t} = N e^{-\eta \sum_{t=1}^T \ell_{PW}^t}.$$

- Taking the logarithms, we obtain

$$\ln W^{T+1} \leq \ln N - \eta \sum_{t=1}^T \ell_{PW}^t = \ln N - \eta L_{PW}^T.$$

- For the lower bound, we have $W^{T+1} \geq w_k^{T+1}$ and thus, by taking logarithms, using the recursive definition of weights and $\ln(1 - z) \geq -z - z^2$ for $z \leq 1/2$, we obtain

$$\ln W^{T+1} \geq \ln w_k^{T+1} = \sum_{t=1}^T \ln(1 - \eta \ell_k^t) \geq -\eta L_k^T - \eta^2 Q_k^T.$$

- Combining the lower and the upper bound, we have

$$-\eta L_k^T - \eta^2 Q_k^T \leq \ln N - \eta L_{PW}^T.$$



Polynomial weights algorithm: remarks

- This algorithm produces very good external regret. Time-averaged external regret goes to zero.
- The bound $L_{PW}^T \leq L_{min}^T + 2\sqrt{T \ln N}$ is essentially optimal.

Proposition 2.50

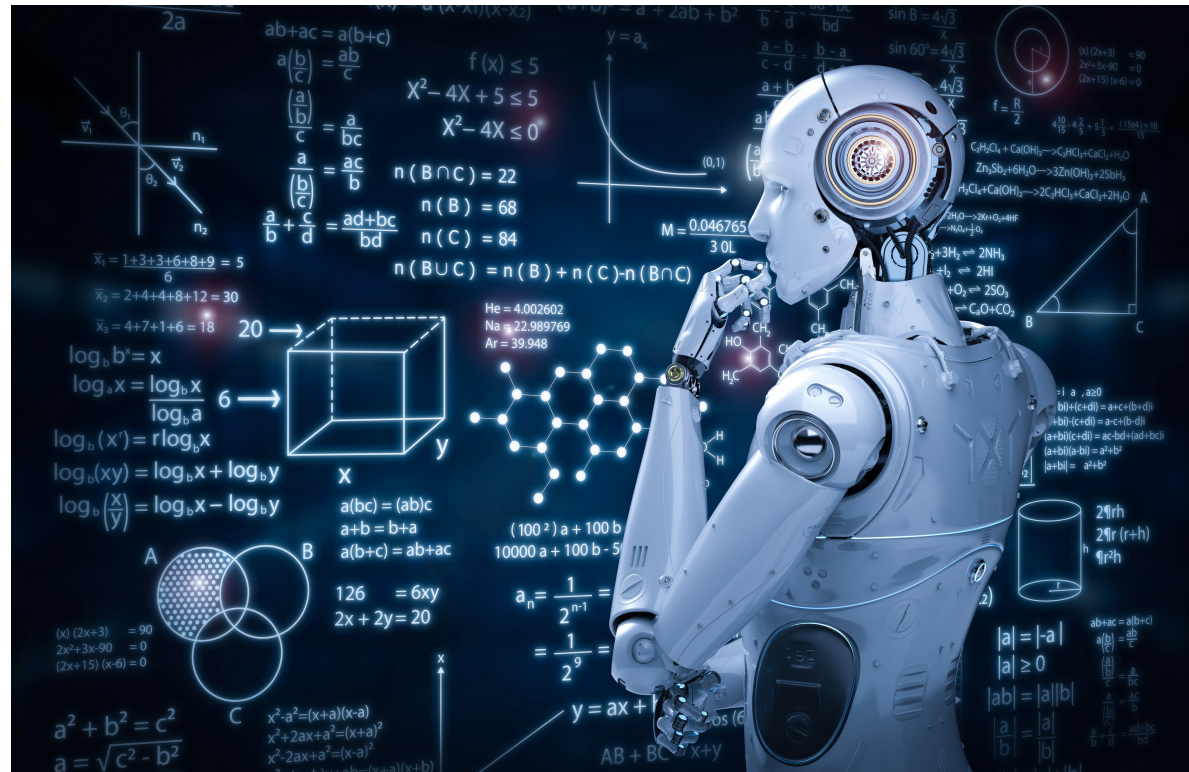
For integers N and T with $T < \lfloor \log_2 N \rfloor$, there exists a stochastic generation of losses such that, for every online algorithm A , we have $\mathbb{E}[L_A^T] \geq T/2$ and yet $L_{min}^T = 0$.

Proposition 2.51

In the case of $N = 2$ actions, there exists a stochastic generation of losses such that, for every online algorithm A , we have $\mathbb{E}[L_A^T - L_{min}^T] \geq \Omega(\sqrt{T})$.

- See lecture notes for the proofs.
- We do not need to know T in advance (Exercise).

- Besides game theory, the “multiplicative weight update method” has **many applications** in various fields of science, for example in optimization, theoretical computer science, and machine learning.



Sources: <https://clubitc.ro>

- See https://en.wikipedia.org/wiki/Multiplicative_weight_update_method#Applications
- There are other algorithms producing small external regret, for example, the **Regret matching algorithm**.

Regret matching

- Algorithm with external regret at most $O(\sqrt{T})$ for fixed N .
- Introduced by Hart and Mas-Colell in 2000.
- For a vector $v \in \mathbb{R}^N$, we use $[v]^+$ to denote the vector

$$(\max\{0, v_i\} : i \in [N]).$$

- The algorithm works as follows. We choose $R_1 = \mathbf{0} \in \mathbb{R}^N$ and then let for $t \geq 1$

$$p^t = \frac{[R^t]^+}{\|[R^t]^+\|_1} \quad \text{and} \quad R^{t+1} = R^t + \langle p^t, \ell^t \rangle \cdot \mathbf{1} - \ell^t,$$

where $\mathbf{0}/0$ is defined as the uniform distribution.

- Strong empirical performance and admits many variants.

The No-regret dynamics

- “Players in a normal-form game play against each other by selecting actions according to the Polynomial-weights algorithm.”

Algorithm 0.40: NO-REGRET DYNAMICS(G, T, ε)

Input : A normal-form game $G = (P, A, C)$ of n players, $T \in \mathbb{N}$, and $\varepsilon > 0$.

Output : A prob. distribution p_i^t on A_i for each $i \in P$ and $t \in \{1, \dots, T\}$.

for every step $t = 1, \dots, T$

do $\left\{ \begin{array}{l} \text{Each player } i \in P \text{ independently chooses a mixed strategy } p_i^t \\ \text{using an algorithm with average regret at most } \varepsilon, \text{ with actions} \\ \text{corresponding to pure strategies.} \\ \text{Each player } i \in P \text{ receives a loss vector } \ell_i^t = (\ell_i^t(a_i))_{a_i \in A_i}, \text{ where} \\ \ell_i^t(a_i) \leftarrow \mathbb{E}_{a_{-i}^t \sim p_{-i}^t} [C_i(a_i; a_{-i}^t)] \text{ for the product distribution} \\ p_{-i}^t = \prod_{j \neq i} p_j^t. \end{array} \right.$

Output $\{p^t : t \in \{1, \dots, T\}\}$.

"ENROLL IN AGT" THEY SAID

Algorithm 2.6.4: NO-REGRET DYNAMICS(G, T, ε)

Input : A normal-form game $G = (P, A, C)$ of n players, $T \in \mathbb{N}$ and $\varepsilon > 0$.

Output : A probability distribution p_i^t on A_i for each $i \in P$ and $t \in \{1, \dots, T\}$.

for every step $t = 1, \dots, T$

do $\left\{ \begin{array}{l} \text{Each player } i \in P \text{ independently chooses a mixed strategy } p_i^t \text{ using} \\ \text{an algorithm with average regret at most } \varepsilon, \text{ with actions} \\ \text{corresponding to pure strategies.} \\ \text{Each player } i \in P \text{ receives a loss vector } \ell_i^t = (\ell_i^t(a_i))_{a_i \in A_i}, \text{ where} \\ \ell_i^t(a_i) \leftarrow \mathbb{E}_{a_{-i}^t \sim \sigma_{-i}^t} [C_i(a_i; a_{-i}^t)] \text{ for the product distribution} \\ \sigma_{-i}^t = \prod_{j \neq i} p_j^t. \end{array} \right.$

Output $\{p^t : t \in \{1, \dots, T\}\}$.

"THERE'LL BE NO REGRET" THEY SAID

Sources: Students of MFF UK

Thank you for your attention.