Announcements

- Project instruction is out. Let us know if you have any question or need any help!

CS6501: Topics in Learning and Game Theory
(Spring 2021)

Swap Regret and Convergence to CE

Instructor: Haifeng Xu

Outline

- (External) Regret vs Swap Regret
- Convergence to Correlated Equilibrium
- Converting Regret Bounds to Swap Regret Bounds

Recap: Online Learning

At each time step $t = 1, \cdots, T$, the following occurs in order:

1. Learner picks a distribution p_t over actions $[n]$
2. Adversary picks cost vector $c_t \in [0,1]^n$
3. Action $i_t \sim p_t$ is chosen and learner incurs cost $c_t(i_t)$
4. Learner observes c_t (for use in future time steps)

Recap: (External) Regret

- External regret

$$R_T = \mathbb{E}_{i_t \sim p_t} \sum_{t \in [T]} c_t(i_t) - \boxed{\min_{j \in [n]} \sum_{t \in [T]} c_t(j)}$$

- Benchmark $\min_{j \in [n]} \sum_t c_t(j)$ is the learner utility had he known $c_1, \cdots, c_T$ and is allowed to take the best single action across all rounds
- Describes how much the learner regrets, had he known the cost vector $c_1, \cdots, c_T$ in hindsight

Recap: (External) Regret

➢ A closer look at external regret

$$R_T = \mathbb{E}_{i_t \sim p_t} \sum_{t\in[T]} c_t(i_t) - \min_{j\in[n]} \sum_{t\in[T]} c_t(j)$$

$$= \sum_{t\in[T]} \sum_{i\in[n]} c_t(i) p_t(i) - \min_{j\in[n]} \sum_{t\in[T]} c_t(j)$$

$$= \max_{j\in[n]} \left[\sum_{t\in[T]} \sum_{i\in[n]} c_t(i) p_t(i) - \sum_{t\in[T]} c_t(j)\right]$$

$$= \max_{j\in[n]} \sum_{t\in[T]} \sum_{i\in[n]} \boxed{[c_t(i) - c_t(j)]} p_t(i)$$

Many-to-one action swap

Recap: (External) Regret

- A closer look at external regret

$$\begin{aligned} R_T &= \mathbb{E}_{i_t \sim p_t} \sum_{t\in[T]} c_t(i_t) - \min_{j\in[n]} \sum_{t\in[T]} c_t(j) \\ &= \sum_{t\in[T]} \sum_{i\in[n]} c_t(i) p_t(i) - \min_{j\in[n]} \sum_{t\in[T]} c_t(j) \\ &= \max_{j\in[n]} \left[\sum_{t\in[T]} \sum_{i\in[n]} c_t(i) p_t(i) - \sum_{t\in[T]} c_t(j) \right] \\ &= \max_{j\in[n]} \sum_{t\in[T]} \sum_{i\in[n]} \boxed{[c_t(i) - c_t(j)]} p_t(i) \end{aligned}$$

- In external regret, learner is allowed to swap to a single action *j* and can choose the best *j* in hindsight

Swap Regret

- A closer look at external regret

R_T

- Swap regret allows many-to-many action swap

$c_t(s(i))$

  - E.g., $s(1) = 2, s(2) = 1, s(3) = 4, s(4) = 4$

- Formally,

$$= \max_{j\in[n]} \sum_{t\in[T]} \sum_{i\in[n]} [c_t(i) - c_t(j)] p_t(i)$$

$$swR_T = \max_{s} \sum_{t\in[T]} \sum_{i\in[n]} [c_t(i) - c_t(s(i))] p_t(i)$$

where $\max$ is over all possible swap functions

  - Each action i has n choices to swap to, so n^n many swap functions
  - Quiz: how many many-to-one swaps?

Useful Facts about Swap Regret

Fact 1. For any algorithm: $swR_T \geq R_T$

Fact 2. For any algorithm execution $p_1, \cdots, p_T$, the optimal swap function s^* satisfies, for any i,

$$s^*(i) = \arg\max_{j \in [n]} \sum_{t \in [T]} [c_t(i) - c_t(j)] p_t(i)$$

Recall swap regret

$$swR_T = \max_s \sum_{t \in [T]} \sum_{i \in [n]} [c_t(i) - c_t(s(i))] p_t(i)$$

Proof:

- $s(i)$ only affects term $\sum_{t \in [T]} [c_t(i) - c_t(s(i))] p_t(i)$, so should be picked to maximize this term

Useful Facts about Swap Regret

Fact 1. For any algorithm: $swR_T \geq R_T$

Fact 2. For any algorithm execution $p_1, \cdots, p_T$, the optimal swap function s^* satisfies, for any i,

$$s^*(i) = \arg\max_{j \in [n]} \sum_{t \in [T]} [c_t(i) - c_t(j)] p_t(i)$$

Remarks:

- The optimal swap can be decided "independently" for each i

Useful Facts about Swap Regret

Fact 1. For any algorithm: $swR_T \geq R_T$

Fact 2. For any algorithm execution $p_1, \cdots, p_T$, the optimal swap function s^* satisfies, for any i,

$$s^*(i) = \arg\max_{j \in [n]} \sum_{t \in [T]} [c_t(i) - c_t(j)] p_t(i)$$

Remarks:

- Benchmark of swap regret depends on the algorithm execution $p_1, \cdots, p_T$, but benchmark of external regret does not.
- This raises a subtle issue: an algorithm minimize swap regret does not necessarily minimize the total loss
    - An algorithm may intentionally take less actions so the benchmark does not have many opportunities to swap

Useful Facts about Swap Regret

Fact 1. For any algorithm: $swR_T \geq R_T$

Fact 2. For any algorithm execution $p_1, \cdots, p_T$, the optimal swap function s^* satisfies, for any i,

$$s^*(i) = \arg\max_{j \in [n]} \sum_{t \in [T]} [c_t(i) - c_t(j)] p_t(i)$$

pick worst i

$$\max_{i \in [n]} \max_{j \in [n]} \sum_{t \in [T]} [c_t(i) - c_t(j)] p_t(i)$$

is also called the *internal regret*

Note: internal regret $\leq$ swap regret $\leq n\times$ internal regret

Outline

- (External) Regret vs Swap Regret
- Convergence to Correlated Equilibrium
- Converting Regret Bounds to Swap Regret Bounds

Recap: Normal-Form Games and CE

- n players, denoted by set $[n] = \{1, \cdots, n\}$
- Player i takes action $a_i \in A_i$
- Player utility depends on the outcome of the game, i.e., an action profile $a = (a_1, \cdots, a_n)$
  - Player i receives payoff $u_i(a)$ for any outcome $a \in \Pi_{i=1}^{n} A_i$
- Correlated equilibrium is an action recommendation policy

A recommendation policy π is a **correlated equilibrium** if

$$\sum_{a_{-i}} u_i(a_i, a_{-i}) \cdot \pi(a_i, a_{-i}) \geq \sum_{a_{-i}} u_i({a'}_i, a_{-i}) \cdot \pi(a_i, a_{-i}), \forall\, {a'}_i \in A_i, \forall i \in [n].$$

- That is, for any recommended action a_i, player i does not want to "swap" to another a_i'

Repeated Games with No-Swap-Regret Players

- The game is played repeatedly for T rounds
- Each player uses an online learning algorithm to select a mixed strategy at each round t
- For any player i's perspective, the following occurs in order at t
  - Picks a mixed strategy $x_i^t \in \Delta_{|A_i|}$ over actions in A_i
  - Any other player $j \neq i$ picks a mixed strategy $x_j^t \in \Delta_{|A_j|}$
  - Player i receives expected utility $u_i(x_i^t, x_{-i}^t) = \mathbb{E}_{a \sim (x_i^t, x_{-i}^t)}\, u_i(a)$
  - Player i learns x_{-i}^t (for future use)

From No Swap Regret to Correlated Equ

Theorem. If all players use no-swap-regret learning algorithms with strategy sequence $\{x_i^t\}_{t\in[T]}$ for i. The following recommendation policy π^T converges to a CE: $\pi^T(a) = \frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)$, $\forall\, a \in A$.

Remarks:

- In mixed strategy profile $(x_1^t, x_2^t, \cdots, x_n^t)$, prob. of a is $\Pi_{i\in[n]} x_i^t(a_i)$
- $\pi^T(a)$ is simply the average of $\Pi_{i\in[n]} x_i^t(a_i)$ over T rounds

From No Swap Regret to Correlated Equ

Theorem. If all players use no-swap-regret learning algorithms with strategy sequence $\{x_i^t\}_{t\in[T]}$ for i. The following recommendation policy π^T converges to a CE: $\pi^T(a) = \frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)\,, \forall\, a \in A$.

Proof:

➢Derive player i's expected utility from π^T

$$\sum_{a\in A}\left[\frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)\right]\cdot u_i(a)$$

$$= \frac{1}{T}\sum_t \sum_{a\in A} \Pi_{i\in[n]} x_i^t(a_i)\cdot u_i(a)$$

From No Swap Regret to Correlated Equ

Theorem. If all players use no-swap-regret learning algorithms with strategy sequence $\{x_i^t\}_{t\in[T]}$ for i. The following recommendation policy π^T converges to a CE: $\pi^T(a) = \frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)$, $\forall\, a \in A$.

Proof:

➢ Derive player i's expected utility from π^T

$$\sum_{a\in A} \left[\frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)\right] \cdot u_i(a)$$

$$= \frac{1}{T}\sum_t \sum_{a\in A} \Pi_{i\in[n]} x_i^t(a_i) \cdot u_i(a)$$

$$= \frac{1}{T}\sum_t u_i(x_i^t, x_{-i}^t)$$

From No Swap Regret to Correlated Equ

Theorem. If all players use no-swap-regret learning algorithms with strategy sequence $\{x_i^t\}_{t\in[T]}$ for i. The following recommendation policy π^T converges to a CE: $\pi^T(a) = \frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)$, $\forall\, a \in A$.

Proof:

➢Derive player i's expected utility from π^T

$$\sum_{a\in A}\left[\frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)\right]\cdot u_i(a)$$

$$= \frac{1}{T}\sum_t \sum_{a\in A} \Pi_{i\in[n]} x_i^t(a_i)\cdot u_i(a)$$

$$= \frac{1}{T}\sum_t u_i(x_i^t, x_{-i}^t)$$

$$= \frac{1}{T}\sum_{a_i\in A_i}\sum_{t=1}^{T} u_i\left(a_i, x_{-i}^t\right)\cdot x_i^t(a_i)$$

➢Player i's expected utility conditioned on being recommended a_i is

$\frac{1}{T}\sum_{t=1}^{T} u_i\left(a_i, x_{-i}^t\right)\cdot x_i^t(a_i)$ (normalization factor omitted)

From No Swap Regret to Correlated Equ

Theorem. If all players use no-swap-regret learning algorithms with strategy sequence $\{x_i^t\}_{t\in[T]}$ for i. The following recommendation policy π^T converges to a CE: $\pi^T(a) = \frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)\,, \forall\, a \in A$.

Proof:

- The CE condition requires for all player i and all $a_i \in A_i$

$$\geq \frac{1}{T}\sum_{t=1}^{T} u_i\big(s(a_i), x_{-i}^t\big)\cdot x_i^t(a_i), \;\; \forall s(a_i) \in A_i$$

- Let s^* be the optimal swap function in the swap regret:

$$swR_T^i = \max_s \sum_{t=1}^{T}\sum_{a_i\in A_i}[u_i(s(a_i), x_{-i}) - u_i\big(a_i, x_{-i}^t\big)]\cdot x_i^t(a_i)$$

$$= \sum_{a_i}\Big(\;\sum_{t=1}^{T}[u_i(s^*(a_i), x_{-i}) - u_i\big(a_i, x_{-i}^t\big)]\cdot x_i^t(a_i) \;\Big)$$

$$\geq \sum_{t=1}^{T}\big[u_i(s^*(a_i), x_{-i}) - u_i\big(a_i, x_{-i}^t\big)\big]\cdot x_i^t(a_i)\,, \quad \forall a_i$$

$$\frac{1}{T}\sum_{t=1}^{T} u_i\big(a_i, x_{-i}^t\big)\cdot x_i^t(a_i)$$

From No Swap Regret to Correlated Equ

Theorem. If all players use no-swap-regret learning algorithms with strategy sequence $\{x_i^t\}_{t\in[T]}$ for i. The following recommendation policy π^T converges to a CE: $\pi^T(a) = \frac{1}{T}\sum_t \Pi_{i\in[n]} x_i^t(a_i)\,, \forall\, a \in A$.

Proof:

- The CE condition requires for all player i and all $a_i \in A_i$

$$\frac{1}{T}\sum_{t=1}^T u_i\left(a_i, x_{-i}^t\right) \cdot x_i^t(a_i) \geq \frac{1}{T}\sum_{t=1}^T u_i\left(s(a_i), x_{-i}^t\right) \cdot x_i^t(a_i), \quad \forall s(a_i) \in A_i$$

- Let s^* be the optimal swap function in the swap regret:

$$swR_T^i \geq \sum_{t=1}^T \left[u_i(s^*(a_i), x_{-i}) - u_i\left(a_i, x_{-i}^t\right)\right] \cdot x_i^t(a_i)\,, \quad \forall a_i$$

- From **Fact 2** before, optimal swap function s^* satisfies

$$s^*(a_i) = \arg\max_{s(a_i)\in A_i} \sum_{t=1}^T \left[u_i(s(a_i), x_{-i}) - u_i\left(a_i, x_{-i}^t\right)\right] \cdot x_i^t(a_i)$$

- This implies

Thm follows by diving both sides by $T(\rightarrow \infty)$

$$swR_T^i \geq \sum_{t=1}^T \left[u_i(s(a_i), x_{-i}) - u_i\left(a_i, x_{-i}^t\right)\right] \cdot x_i^t(a_i)\,, \quad \forall a_i \text{ and } s(a_i)$$

Outline

- (External) Regret vs Swap Regret
- Convergence to Correlated Equilibrium
- Converting Regret Bounds to Swap Regret Bounds

Good External Regret ≠ Good Swap Regret

- An algorithm with small swap regret also has small external regret
- The reverse is not true – an algorithm with small external regret does not necessarily have small swap regret
  - Examples are not difficult to construct

Do there exist online learning algorithms with sublinear regret?

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

$n =$ number of actions

- ➢ H utilizes A but is different and more complicated
- ➢ There exists no-swap-regret online learning algorithm
  - Since there exists online algorithm with $O(\sqrt{T \ln n})$ regret

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

Proof Overview:

- The idea starts from the following observations

Let s^* be the optimal swap function, then:

$$\begin{aligned} swR_T &= \max_s \sum_{t\in[T]} \sum_{i\in[n]} [c_t(i) - c_t(s(i))] p_t(i) \\ &= \sum_{i\in[n]} \left(\sum_{t\in[T]} [c_t(i) - c_t(s^*(i))] p_t(i) \right) \end{aligned}$$

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

Proof Overview:

- The idea starts from the following observations

Let s^* be the optimal swap function, then:

$$swR_T = \max_s \sum_{t\in[T]} \sum_{i\in[n]} [c_t(i) - c_t(s(i))] p_t(i)$$
$$= \sum_{i\in[n]} \Big(\underbrace{\sum_{t\in[T]} [c_t(i) - c_t(s^*(i))] p_t(i)}_{\text{regret from action } i\text{'s swap}} \Big)$$

Two observations:

1. The red terms "looks like" an external regret term
   - Swap to a single action, but $\sum_{t\in[T]} c_t(i) p_t(i)$ does not look quite right yet
2. If the red term is less than R for any i, then we are done

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

Proof Step 1: constructing H

- Make n copies of algorithm A as $A_1, \cdots, A_n$
  - Intuitively, A_i takes care of the regret from action i's swap
- Construction of H
  - At round t, H picks action i with probability $p_t(i)$ (to be designed)
  - Let $q_t^i \in \Delta_n$ be the randomized action of A_i generated at round t
  - Choose $p_t(i) \in [0,1]$ to satisfy the following:

$$\sum_i p_t(i) = 1 \xrightarrow{\qquad} p_t \text{ is a distribution}$$

$$\sum_i p_t(i) q_t^i(j) = p_t(j), \forall j \in [n] \xrightarrow{\qquad} p_t \text{ is stationary}$$

That is, following two ways for H to select actions are equivalent
1. Select i with probability $p_t(i)$
2. Select algorithm A_i with prob $p_t(i)$, then use A_i to pick an action

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

Proof Step 1: constructing H

- Make n copies of algorithm A as $A_1, \cdots, A_n$
  - Intuitively, A_i takes care of the regret from action i's swap
- Construction of H
  - At round t, H picks action i with probability $p_t(i)$ (to be designed)
  - Let $q_t^i \in \Delta_n$ be the randomized action of A_i generated at round t
  - Choose $p_t(i) \in [0,1]$ to satisfy the following:

$$\sum_i p_t(i) = 1 \longrightarrow p_t \text{ is a distribution}$$

$$\sum_i p_t(i) q_t^i(j) = p_t(j), \forall j \in [n] \longrightarrow p_t \text{ is stationary}$$

  - After observing cost vector c_t, allocate $p_t(i) \cdot c_t$ as the "simulated cost" to algorithm A_i for its future use

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

Proof Step 2: deriving regret bound

- A_i has external regret R, so

$$\sum_{t\in[T]} \sum_j q_t^i(j)\,[p_t(i)c_t(j) - p_t(i)c_t(j')] \le R \quad \forall j' \in [n] \qquad (1)$$

- Swap regret of H

$$swR_T = \max_s \sum_{t\in[T]} \sum_{j\in[n]} p_t(j)[c_t(j) - c_t(s(j))]$$

Need to somehow relate swR_T to q_t^i's, because Inequality (1) is the only bound we have

By our construction: $\sum_i p_t(i)q_t^i(j) = p_t(j)\,, \forall j \in [n]$

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

Proof Step 2: deriving regret bound

- A_i has external regret R, so

$$\sum_{t\in[T]}\sum_j q_t^i(j)\,[p_t(i)c_t(j) - p_t(i)c_t(j')] \le R \quad \forall j' \in [n] \qquad (1)$$

- Swap regret of H

$$\begin{aligned} swR_T &= \max_s \sum_{t\in[T]}\sum_{j\in[n]} p_t(j)[c_t(j) - c_t(s(j))] \\ &= \max_s \sum_{t\in[T]}\sum_{j\in[n]}\sum_i p_t(i)q_t^i(j)\,[c_t(j) - c_t(s(j))] \end{aligned}$$

By our construction: $\sum_i p_t(i)q_t^i(j) = p_t(j)\,, \forall j \in [n]$

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

Proof Step 2: deriving regret bound

- A_i has external regret R, so

$$\sum_{t\in[T]}\sum_j q_t^i(j)\,[p_t(i)c_t(j) - p_t(i)c_t(j')] \le R \quad \forall j' \in [n] \qquad (1)$$

- Swap regret of H

$$\begin{aligned} swR_T &= \max_s \sum_{t\in[T]}\sum_{j\in[n]} p_t(j)[c_t(j) - c_t(s(j))] \\ &= \max_s \sum_{t\in[T]}\sum_{j\in[n]} \sum_i p_t(i) q_t^i(j)\,[c_t(j) - c_t\big(s(j)\big)] \\ &= \max_s \ \sum_i \Big(\sum_{t\in[T]}\sum_{j\in[n]} p_t(i) q_t^i(j)[c_t(j) - c_t(s(j))]\Big) \end{aligned}$$

Theorem. Any online algorithm A with external regret R can be converted to another online algorithm H swap regret nR.

Proof Step 2: deriving regret bound

- A_i has external regret R, so

$$\sum_{t\in[T]} \sum_j q_t^i(j)\,[p_t(i)c_t(j) - p_t(i)c_t(j')] \le R \quad \forall j' \in [n] \qquad (1)$$

- Swap regret of H

$$\begin{aligned} swR_T &= \max_s \sum_{t\in[T]} \sum_{j\in[n]} p_t(j)[c_t(j) - c_t(s(j))] \\ &= \max_s \sum_{t\in[T]} \sum_{j\in[n]} \sum_i p_t(i) q_t^i(j)\,[c_t(j) - c_t(s(j))] \\ &= \max_s \sum_i \Big(\sum_{t\in[T]} \sum_{j\in[n]} p_t(i) q_t^i(j)[c_t(j) - c_t(s(j))] \Big) \\ &\le n \cdot R \end{aligned}$$

Thank You

Haifeng Xu

University of Virginia

hx4ad@virginia.edu