Minimizing Swap Regret

Aaron Roth

University of Pennsylvania

February 16 2023
Recall last lecture we defined correlated equilibria and coarse correlated equilibria.
Overview

- Recall last lecture we defined correlated equilibria and coarse correlated equilibria.
- We observed that if players use the polynomial weights algorithm (or other similar methods) the empirical history of play will converge quickly to a CCE.

In this lecture, we give a learning algorithm to achieve this.
Recall last lecture we defined correlated equilibria and coarse correlated equilibria.

We observed that if players use the polynomial weights algorithm (or other similar methods) the empirical history of play will converge quickly to a CCE.

And we showed that if a player could minimize regret to arbitrary strategy modification rules, play would converge to CE.
Overview

- Recall last lecture we defined correlated equilibria and coarse correlated equilibria.
- We observed that if players use the polynomial weights algorithm (or other similar methods) the empirical history of play will converge quickly to a CCE.
- And we showed that if a player could minimize regret to arbitrary strategy modification rules, play would converge to CE.
- In this lecture, we give a learning algorithm to achieve this.
Recall

Definition
A distribution $\mathcal{D}$ over action profiles is an $\epsilon$-approximate correlated equilibrium if for every player $i$, and for every strategy modification rule $F_i : A_i \rightarrow A_i$:

$$E_{a \sim \mathcal{D}}[\text{Regret}_i(a, F_i)] \leq \epsilon.$$  

Recall that $\text{Regret}_i(a, F_i) = u_i(F_i(a_i), a_{-i}) - u_i(a)$. 

We’ll define a new notion of regret for sequences of action profiles. To disambiguate, we’ll start calling our old notion of regret “external regret”. 
Recall

Definition
A distribution $\mathcal{D}$ over action profiles is an $\epsilon$-approximate correlated equilibrium if for every player $i$, and for every strategy modification rule $F_i : A_i \rightarrow A_i$:

$$E_{a \sim \mathcal{D}}[\text{Regret}_i(a, F_i)] \leq \epsilon.$$  

Recall that $\text{Regret}_i(a, F_i) = u_i(F_i(a_i), a_{-i}) - u_i(a)$.

We’ll define a new notion of regret for sequences of action profiles. To disambiguate, we’ll start calling our old notion of regret “external regret”.  


A New Notion

Definition
A sequence of action profiles $a^1, \ldots, a^T$ has swap-regret $\Delta(T)$ if for every player $i$, and every strategy modification rule $F_i : A_i \rightarrow A_i$ we have:

$$\frac{1}{T} \sum_{t=1}^{T} u_i(a^t) \geq \frac{1}{T} \sum_{t=1}^{T} u_i(F_i(a_i), a_{-i}) - \Delta(T)$$

If $\Delta(T) = o_T(1)$, we say that the sequence of action profiles has no swap regret.
A New Notion

Definition
A sequence of action profiles \( a^1, \ldots, a^T \) has swap-regret \( \Delta(T) \) if for every player \( i \), and every strategy modification rule \( F_i : A_i \rightarrow A_i \) we have:

\[
\frac{1}{T} \sum_{t=1}^{T} u_i(a^t) \geq \frac{1}{T} \sum_{t=1}^{T} u_i(F_i(a_i), a_{-i}) - \Delta(T)
\]

If \( \Delta(T) = o_T(1) \), we say that the sequence of action profiles has no swap regret.

1. External regret measured regret to the best fixed action in hindsight.

   2. Swap regret measures regret to the counterfactual in which you can swap every action of a particular type with a different action in hindsight, separately for each action.
A New Notion

Definition
A sequence of action profiles $a^1, \ldots, a^T$ has swap-regret $\Delta(T)$ if for every player $i$, and every strategy modification rule $F_i : A_i \rightarrow A_i$ we have:

$$\frac{1}{T} \sum_{t=1}^{T} u_i(a^t) \geq \frac{1}{T} \sum_{t=1}^{T} u_i(F_i(a_i), a_{-i}) - \Delta(T)$$

If $\Delta(T) = o_T(1)$, we say that the sequence of action profiles has no swap regret.

1. External regret measured regret to the best fixed action in hindsight.

2. Swap regret measures regret to the counterfactual in which you can swap every action of a particular type with a different action in hindsight, separately for each action.
Why Sequences?

Theorem
If a sequence of action profiles $a^1, \ldots, a^T$ has $\Delta(T)$ swap-regret, then the distribution $D = \frac{1}{T} \sum_{t=1}^{T} a^t$ (i.e. the distribution that picks among the action profiles $a^1, \ldots, a^T$ uniformly at random) is a $\Delta(T)$-approximate correlated equilibrium.

Proof.
This follows immediately from the definitions.
Why Sequences?

Theorem
If a sequence of action profiles $a^1, \ldots, a^T$ has $\Delta(T)$ swap-regret, then the distribution $D = \frac{1}{T} \sum_{t=1}^{T} a^t$ (i.e. the distribution that picks among the action profiles $a^1, \ldots, a^T$ uniformly at random) is a $\Delta(T)$-approximate correlated equilibrium.

Proof.
This follows immediately from the definitions.
Why Sequences?

Theorem
If a sequence of action profiles $a^1, \ldots, a^T$ has $\Delta(T)$ swap-regret, then the distribution $D = \frac{1}{T} \sum_{t=1}^{T} a^t$ (i.e. the distribution that picks among the action profiles $a^1, \ldots, a^T$ uniformly at random) is a $\Delta(T)$-approximate correlated equilibrium.

Proof.
This follows immediately from the definitions.

For any player $i$:

$$E_{a^t \sim D}[\text{Regret}_i(a^t, F_i)] = \frac{1}{T} \sum_{t=1}^{T} (u_i(F_i(a^t_i), a^t_{-i}) - u_i(a^t)) \leq \Delta(T)$$
Back to Experts: The Setting

In rounds $t = 1, \ldots, T$:

1. The algorithm picks an expert $a_t \in \{1, \ldots, k\}$ from among the set of $k$ experts.

2. Each expert $i$ experiences loss $\ell_t^i$, and the algorithm experiences loss $\ell_{a_t}^t$.

Write $L_T^\text{Alg} = \sum_{t=1}^{T} \ell_{a_t}^t$ for the cumulative loss of the algorithm after $T$ rounds. We want to find an algorithm that can guarantee, for arbitrary sequences of losses:

$$1_T L_T^\text{Alg} \leq 1_T \sum_{t=1}^{T} \ell_t^i(a_t) + \Delta(T)$$

for all $F_i : [k] \to [k]$ and for $\Delta(T) = o(1)$. 
In rounds $t = 1, \ldots, T$:

1. The algorithm picks an expert $a_t \in \{1, \ldots, k\}$ from among the set of $k$ experts.

2. Each expert $i$ experiences loss $\ell_{i}^t$, and the algorithm experiences loss $\ell_{a_t}^t$. Write $L_{Alg}^T = \sum_{t=1}^{T} \ell_{a_t}^t$ for the cumulative loss of the algorithm after $T$ rounds.
Back to Experts: The Setting

In rounds \( t = 1, \ldots, T \):

1. The algorithm picks an expert \( a_t \in \{1, \ldots, k\} \) from among the set of \( k \) experts.

2. Each expert \( i \) experiences loss \( \ell_t^i \), and the algorithm experiences loss \( \ell_t^{a_t} \).

Write \( L_T^{\text{Alg}} = \sum_{t=1}^{T} \ell_t^{a_t} \) for the cumulative loss of the algorithm after \( T \) rounds.

We want to find an algorithm that can guarantee, for arbitrary sequences of losses:

\[
\frac{1}{T} L_T^{\text{Alg}} \leq \frac{1}{T} \sum_{t=1}^{T} \ell_t^{F_i(a_t)} + \Delta(T)
\]

for all \( F_i : [k] \to [k] \) and for \( \Delta(T) = o(1) \).
What Should We Do?

1. For a fixed sequence of decisions by our algorithm, define:

\[ S_j = \{ t : a_t = j \} \]

to be the set of time steps that the algorithm chose expert \( j \).
What Should We Do?

1. For a fixed sequence of decisions by our algorithm, define:

   \[ S_j = \{ t : a_t = j \} \]

   to be the set of time steps that the algorithm chose expert \( j \).

2. One guiding observation: To achieve the desired bound, it would be sufficient that for every \( j \):

   \[
   \frac{1}{|S_j|} \sum_{t \in S_j} \ell^t_{a_t} \leq \frac{1}{|S_j|} \min_i \sum_{t \in S_j} \ell^t_i + \Delta(T)
   \]
What Should We Do?

1. For a fixed sequence of decisions by our algorithm, define:

\[ S_j = \{ t : a_t = j \} \]

to be the set of time steps that the algorithm chose expert \( j \).

2. One guiding observation: To achieve the desired bound, it would be sufficient that for every \( j \):

\[
\frac{1}{|S_j|} \sum_{t \in S_j} \ell^t_{a_t} \leq \frac{1}{|S_j|} \min_i \sum_{t \in S_j} \ell^t_i + \Delta(T)
\]

3. i.e. we can achieve no swap regret if we can achieve no external regret separately on each sequence of actions \( S_j \).
What Should We Do?

1. For a fixed sequence of decisions by our algorithm, define:

\[ S_j = \{ t : a_t = j \} \]

to be the set of time steps that the algorithm chose expert \( j \).

2. One guiding observation: To achieve the desired bound, it would be sufficient that for every \( j \):

\[
\frac{1}{|S_j|} \sum_{t \in S_j} \ell^t_{a_t} \leq \frac{1}{|S_j|} \min_i \sum_{t \in S_j} \ell^t_i + \Delta( T )
\]

3. i.e. we can achieve no swap regret if we can achieve no external regret separately on each sequence of actions \( S_j \).

4. The best strategy modification rule in hindsight simply swaps each action \( j \) for the best fixed action in hindsight over \( S_j \)...
What Should We Do?

1. For a fixed sequence of decisions by our algorithm, define:

$$S_j = \{ t : a_t = j \}$$

to be the set of time steps that the algorithm chose expert $j$.

2. One guiding observation: To achieve the desired bound, it would be sufficient that for every $j$:

$$\frac{1}{|S_j|} \sum_{t \in S_j} \ell_{a_t}^t \leq \frac{1}{|S_j|} \min_i \sum_{t \in S_j} \ell_i^t + \Delta(T)$$

3. i.e. we can achieve no swap regret if we can achieve no external regret separately on each sequence of actions $S_j$.

4. The best strategy modification rule in hindsight simply swaps each action $j$ for the best fixed action in hindsight over $S_j$...

5. Idea: Run $k$ copies of PW, one responsible for each $S_j$...
Algorithm Sketch

The algorithm will work as follows:

1. Initialize $k$ copies of the PW algorithm one for each action $j \in [k]$.

2. At each time $t$, denote by $q(1)^t, \ldots, q(k)^t$ the distribution maintained by each copy of the PW algorithm over the experts. We will combine these into a single distribution over experts $p^t \equiv (p^t_1, \ldots, p^t_k)$.

3. The losses $\ell^t_1, \ldots, \ell^t_k$ for the experts arrive. To each copy $i$ of the PW algorithm, we report losses $p^t_i \ell^t_1, \ldots, p^t_i \ell^t_k$ for each of the $k$ experts. (i.e. to copy $i$, we report the true losses scaled by $p^t_i$).
The algorithm will work as follows:

1. Initialize $k$ copies of the PW algorithm one for each action $j \in [k]$.

2. At each time $t$, denote by $q(1)^t, \ldots, q(k)^t$ the distribution maintained by each copy of the PW algorithm over the experts. We will combine these into a single distribution over experts $p^t \equiv (p^t_1, \ldots, p^t_k)$

3. The losses $\ell_1^t, \ldots, \ell_k^t$ for the experts arrive. To each copy $i$ of the PW algorithm, we report losses $p_i^t \ell_1^t, \ldots, p_i^t \ell_k^t$ for each of the $k$ experts. (i.e. to copy $i$, we report the true losses scaled by $p_i^t$).

It remains to specify: how we combine the distributions $q(i)$ into a single distribution $p$?
Combining Distributions

1. For each expert \( j \), define:

\[
p_j^t = \sum_{i=1}^{k} p_i^t \cdot q(i)_j^t
\]
Combining Distributions

1. For each expert \( j \), define:

\[
p_j^t = \sum_{i=1}^{k} p_i^t \cdot q(i)_j^t
\]

2. The above equations always have a solution as a probability distribution. (Not obvious — but comes from the fact that stochastic matrices always have an eigenvector with eigenvalue 1)
Combining Distributions

1. For each expert \( j \), define:

\[
p_j^t = \sum_{i=1}^{k} p_i^t \cdot q(i)_j^t
\]

2. The above equations always have a solution as a probability distribution. (Not obvious — but comes from the fact that stochastic matrices always have an eigenvector with eigenvalue 1)

3. Crucial property: two ways of viewing the distribution over experts:

- Each expert \( i \) is chosen with probability \( p_i^t \)
- With probability \( p_i^t \) we select the \( i \)'th copy of the polynomial weights algorithm, and then select expert \( j \) according to the probability distribution \( q(i)_j^t \).
Combining Distributions

1. For each expert \( j \), define:

\[
p_j^t = \sum_{i=1}^{k} p_i^t \cdot q(i)_j^t
\]

2. The above equations always have a solution as a probability distribution. (Not obvious — but comes from the fact that stochastic matrices always have an eigenvector with eigenvalue 1)

3. Crucial property: two ways of viewing the distribution over experts:
   3.1 Each expert \( i \) is chosen with probability \( p_i^t \) or
Combining Distributions

1. For each expert $j$, define:

$$ p_j^t = \sum_{i=1}^{k} p_i^t \cdot q(i)_j^t $$

2. The above equations always have a solution as a probability distribution. (Not obvious — but comes from the fact that stochastic matrices always have an eigenvector with eigenvalue 1)

3. Crucial property: two ways of viewing the distribution over experts:

   3.1 Each expert $i$ is chosen with probability $p_i^t$ or
   3.2 With probability $p_i^t$ we select the $i$'th copy of the polynomial weights algorithm, and then select expert $j$ according to the probability distribution $q(i)^t$. 
Analysis

1. From the perspective of the \(i\)'th copy of polynomial weights, its expected loss at round \(t\) is:

\[
\sum_{j=1}^{k} q(i)_j^t \cdot (p_i^t \ell_j^t) = p_i^t \sum_{j=1}^{k} q(i)_j^t \ell_j^t
\]
Analysis

1. From the perspective of the $i$'th copy of polynomial weights, its expected loss at round $t$ is:

$$
\sum_{j=1}^{k} q(i)_j^t \cdot (p_i^t \ell_j^t) = p_i^t \sum_{j=1}^{k} q(i)_j^t \ell_j^t
$$

2. So the PW guarantee tells us that for all experts $j^*$:

$$
\frac{1}{T} \sum_{t=1}^{T} p_i^t \sum_{j=1}^{k} q(i)_j^t \ell_j^t \leq \frac{1}{T} \sum_{t=1}^{T} p_i^t \ell_j^t + 2\sqrt{\frac{\log k}{T}}
$$

$LHS$ $RHS$
Analysis

1. From the perspective of the $i'$th copy of polynomial weights, its expected loss at round $t$ is:

$$\sum_{j=1}^{k} q(i)_{j}^t \cdot (p_{i}^t \ell_{j}^t) = p_{i}^t \sum_{j=1}^{k} q(i)_{j}^t \ell_{j}^t$$

2. So the PW guarantee tells us that for all experts $j^*$:

$$\frac{1}{T} \sum_{t=1}^{T} p_{i}^t \sum_{j=1}^{k} q(i)_{j}^t \ell_{j}^t \leq \frac{1}{T} \sum_{t=1}^{T} p_{i}^t \ell_{j^*}^t + 2\sqrt{\frac{\log k}{T}}$$

3. Summing the LHS:

$$LHS = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{k} \sum_{j=1}^{k} p_{i}^t q(i)_{j}^t \ell_{j}^t = \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{k} p_{i}^t \ell_{j}^t = \frac{1}{T} L_{ALG}$$
Analysis

\[ \frac{1}{T} \sum_{t=1}^{T} p_i^t \sum_{j=1}^{k} q(i)_j^t \ell_j^t \leq \frac{1}{T} \sum_{t=1}^{T} p_i^t \ell_j^{*t} + 2 \sqrt{\frac{\log k}{T}} \]

1. Now the RHS: We can instantiate each term with any \( j^* \).
Analysis

\[
\frac{1}{T} \sum_{t=1}^{T} p_i^t \sum_{j=1}^{k} q(i)_j^t \ell_j^t \leq \frac{1}{T} \sum_{t=1}^{T} p_i^t \ell_{j^*}^t + 2\sqrt{\frac{\log k}{T}}
\]

1. Now the RHS: We can instantiate each term with any \( j^* \).
2. Fixing an arbitrary strategy modification rule \( F : [k] \rightarrow [k] \), for each \( i \) choose \( j^* = F(i) \).
Analysis

\[
\frac{1}{T} \sum_{t=1}^{T} p_i^t \sum_{j=1}^{k} q(i)_j^t \ell_j^t \leq \frac{1}{T} \sum_{t=1}^{T} p_i^t \ell_{j^*}^t + 2 \sqrt{\frac{\log k}{T}}
\]

1. Now the RHS: We can instantiate each term with *any* \( j^* \).
2. Fixing an arbitrary strategy modification rule \( F : [k] \rightarrow [k] \), for each \( i \) choose \( j^* = F(i) \).
3. Summing:

\[
\text{RHS} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{k} p_i^t \ell_{F(i)}^t + 2k \sqrt{\frac{\log k}{T}}
\]
Analysis

\[
\frac{1}{T} \sum_{t=1}^{T} p_i^t \sum_{j=1}^{k} q(i)_j^t \ell_j^t \leq \frac{1}{T} \sum_{t=1}^{T} p_i^t \ell_{j^*}^t + 2 \sqrt{\frac{\log k}{T}}
\]

LHS \hspace{10cm} RHS

1. Now the RHS: We can instantiate each term with any \( j^* \).
2. Fixing an arbitrary strategy modification rule \( F : [k] \rightarrow [k] \), for each \( i \) choose \( j^* = F(i) \).
3. Summing:

\[
RHS = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{k} p_i^t \ell_{F(i)}^t + 2k \sqrt{\frac{\log k}{T}}
\]

4. Combining, we get:

\[
\frac{1}{T} L_{ALG} \leq \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{k} p_i^t \ell_{F(i)}^t + 2k \sqrt{\frac{\log k}{T}}
\]
So, we have proven:

**Theorem**

*There is an experts algorithm that, against an arbitrary sequence of losses, after $T$ rounds achieves $\Delta(T)$-swap regret for:*

$$\Delta(T) = 2k\sqrt{\frac{\log k}{T}}$$
Things of Note

1. $\Delta(T) = o(1)$, and so this is a no-swap-regret algorithm. And if every player plays according to it in an arbitrary game, play converges to CE.
Things of Note

1. $\Delta(T) = o(1)$, and so this is a no-swap-regret algorithm. If every player plays according to it in an arbitrary game, play converges to CE.

2. Players need not know anything about the game to play it - they only need to be able to compute their utilities for the action profiles \textit{actually played}. 

$T = 4 k^2 \ln(k) \epsilon^2$
Things of Note

1. \( \Delta(T) = o(1) \), and so this is a no-swap-regret algorithm. and If every player plays according to it in an arbitrary game, play converges to CE.

2. Players need not know anything about the game to play it - they only need to be able to compute their utilities for the action profiles actually played.

3. Convergence is fast. Setting \( \Delta(T) \leq \epsilon \), we see that we reach \( \epsilon \)-swap regret after \( T \) steps for:

\[
T = \frac{4k^2 \ln(k)}{\epsilon^2}
\]
Things of Note

1. $\Delta(T) = o(1)$, and so this is a no-swap-regret algorithm. If every player plays according to it in an arbitrary game, play converges to CE.

2. Players need not know anything about the game to play it - they only need to be able to compute their utilities for the action profiles actually played.

3. Convergence is fast. Setting $\Delta(T) \leq \epsilon$, we see that we reach $\epsilon$-swap regret after $T$ steps for:

   $$T = \frac{4k^2 \ln(k)}{\epsilon^2}$$

4. So not only do CE exist in all games, they are easy to find.
Thanks!

See you next class — stay healthy!