Game-theoretic Foundations of Multi-agent Systems

Lecture 9: Learning in Games

Seyed Majid Zahedi

UNIVERSITY OF WATERLOO

Outline

Single-agent vs Muli-agent Learning

- In artificial intelligence (AI), learning is usually performed by single agent
- Learning agent learns to function successfully in unknown environment
- In multi-agent setting, environment contains other agents
- Agents' learning changes the environment
- These changes depend in part on actions of learning agents
- Learning of each agent is impacted by learning performed by others
- Different learning rules lead to different dynamical system
- Simple learning rules can lead to complex global behaviors of system

Learning and Teaching

- In multi-agent systems, learning and teaching are inseparable
- Agents must consider what they have learned from others' past behavior
- They also must consider how they wish to influence others' future behavior
- In such setting, learning as accumulating knowledge is not always beneficial
- Accumulating knowledge should never hurt, one can always ignore what is learned
- But when one pre-commits to particular strategy for
  acting on accumulated knowledge, sometimes less is more
- E.g., in game of Chicken, if your opponent is learning your strategy to play best response, then optimal strategy is to always dare

Is Agent Learning in Optimal Way?

- In (repeated or stochastic) zero-sum games, this question is meaningful to ask
- In general, answer depends not only on learning procedure but also on others' behavior
- When all agents adopt same strategy, the setting is called self-play
    - E.g., all agent adopt TfT, or all adopt reinforcement learning (RL)
- One way to evaluate learning procedures is based on their performance in self-play
- But learning agents can also be judged by how they do in context of other agent types
    - TfT agent may perform well against TfT agents, but less well against RL agents
- Note that in GT, optimal strategy is replaced by best response (and equilibrium)

Properties of Learning Rules

- Safety: Guarantee agents at least their maxmin value
- Rationality: Settle on best response to opponent's strategy whenever opponent settles on stationary strategy
    - Opponent adopts same mixed strategy each time, regardless of the past
- No regret: Yield payoff that is no less than payoff agent could have obtained by playing any pure strategy against any set of opponents (details later!)

Outline

Fictitious Play: Introduction

- What are agents learning about?
- Arguably, most plausible answer is strategies of others
- Fictitious play (FP), one of earliest learning rules, takes this approach
- FP was first introduced by G. W. Brown in 1951[1]
- Brown imagined that agents would "simulate" the game in their mind and update their future play based on this simulation; hence name fictitious play
- In its current use, FP is misnomer, since each play of the game actually occurs

[1] Brown, G. W. "Iterative solution of games by fictitious play." 1951

Fictitious Play

- Two agents repeatedly play stage game G
- $\eta_i^t(a_{-i})$ denotes number of times agent i has observed a_{-i} before time t
- η_i^1 represents fictitious past and cannot be zero for all a_{-i}
- Agents assume that their opponent is using stationary mixed strategy
- Agents update their beliefs about this strategy at each step according to:

$$\mu_i^t(a_{-i}) = \frac{\eta_i^t(a_{-i})}{\sum_{a'_{-i}} \eta_i^t(a'_{-i})}$$

- μ_i^t is empirical distribution of past actions and is treated as mixed strategy
- Agents best-respond to their beliefs about opponent' strategy

$$a_i^{t+1} = \operatorname*{argmax}_{a_i} u_i(a_i, \mu_i^t)$$

Fictitious Play: Example

- Consider the following coordination game

$$\begin{array}{c|c|c|} & \text{L} & \text{R} \\ \hline \text{U} & 3,3 & 0,0 \\ \hline \text{D} & 4,0 & 1,1 \\ \hline \end{array}$$

- Note that this game is dominant solvable with unique NE of (D, R)
- Suppose that $\eta_1^1 = (3, 0)$ and $\eta_2^1 = (1, 2.5)$
- FP proceeds as follows:

Round	1's η	2's η	1's action	2's action
1	(3, 0)	(1, 2.5)	D	L
2	(4, 0)	(1, 3.5)	D	R
3	(4, 1)	(1, 4.5)	D	R
4	(4, 2)	(1, 5.5)	D	R

Fictitious Play: Discussion

- In FP, agents do not need to know anything about their opponent's utilities
- FP is somewhat paradoxical as agents assume stationary strategy for their opponent, yet no agent plays stationary strategy except when FP converges
- Even though FP is belief based it is also myopic
- I.e., agents maximize current utility without considering their future ones
- Agents do not learn true model that generates empirical frequencies
- In other words, they do not learn how their opponent is actually playing the game

Convergence of Fictitious Play to Pure Strategies

- Let $\{a^t\}$ be sequence of action profiles generated by FP for G
- Sequence converges to a^* if there exists T s.t. $a^t = a^*$ for all $t \geq T$
- a^* is called steady state or absorbing state of FP
- (I) If sequence converges to a^*, then a^* is pure-strategy NE of G
- (II) If for some t, $a^t = a^*$, where a^* is strict NE of G, then $a^\tau = a^*$ for all $\tau > t$

Proof

- (I) is straightforward, for (II), let $a^t = a^*$, we want to show that $a^{t+1} = a^*$
- First, note that we can write μ as:

$$\mu_i^{t+1} = (1-\alpha)\mu_i^t + \alpha a_{-i}^t = (1-\alpha)\mu_i^t + \alpha a_{-i}^*$$

  here, abusing notation, a_{-i}^t denotes degenerate probability distribution and:

$$\alpha = \frac{1}{\sum_{a'_{-i}} \eta_i^t(a'_{-i}) + 1}$$

- By linearity of expected utility, we have for all a_i:

$$u_i(a_i, \mu_i^{t+1}) = (1-\alpha)u_i(a_i, \mu_i^t) + \alpha u_i(a_i, a_{-i}^*)$$

- Since a_i^* maximizes both terms, it follows that it is played at $t+1$

Convergence of Fictitious Play to Mixed Strategies

- Of course, one cannot guarantee that fictitious play always converges to NE
- In FP, agents only play pure strategies and pure-strategy NE may not exist
- While FP sequence may not converge, its empirical distribution may
- Sequence $\{a^t\}$ converges to s^* in time-average sense if for all i and a_i:

$$\lim_{T\to\infty} \frac{\sum_{t=1}^{T} \mathbb{1}(a_i^t = a_i)}{T} = s_i^*(a_i)$$

  $\mathbb{1}(\cdot)$ denotes the indicator function

- If FP sequence converges to s^* in the time-average sense, then s^* is NE

Proof

- Suppose $\{a^t\}$ converges to s^* in time-average sense, but s^* is not NE
- There is some i, a'_i, and a_i with $s^*_i(a_i) > 0$ s.t. $u_i(a'_i, s^*_{-i}) > u_i(a_i, s^*_{-i})$
- Choose ϵ s.t. $\epsilon < \left(u_i(a'_i, s^*_{-i}) - u_i(a_i, s^*_{-i})\right)/2$
- Choose T s.t. for all $t \geq T$, $|\mu^t_i(a_{-i}) - s^*_{-i}(a_{-i})| < \epsilon / \max_{a'} u_i(a')$ for all a_{-i}
- This is possible because $\mu^t_i(a_{-i}) \to s^*_{-i}(a_{-i})$ by assumption

Proof (cont.)

- Then, for any $t \geq T$, we have:

$$\begin{aligned} u_i(a_i, \mu_i^t) &= \sum_{a_{-i}} u_i(a_i, a_{-i}) \mu_i^t(a_{-i}) \\ &\leq \sum_{a_{-i}} u_i(a_i, a_{-i}) s_{-i}^*(a_{-i}) + \epsilon \\ &\leq \sum_{a_{-i}} u_i(a_i', a_{-i}) s_{-i}^*(a_{-i}) - \epsilon \\ &\leq \sum_{a_{-i}} u_i(a_i', a_{-i}) \mu_i^t(a_{-i}) = u_i(a_i', \mu_i^t) \end{aligned}$$

- So after sufficiently large t, a_i is never played
- This implies that as $t \to \infty$, $\mu_i^t(a_i) \to 0$, which contradicts with $s_i^*(a_i) > 0$

Example: Matching Pennies

- Consider the matching-pennies game

$$\begin{array}{c|c|c|} & \text{H} & \text{T} \\ \hline \text{H} & 1,-1 & -1,1 \\ \hline \text{T} & -1,1 & 1,-1 \\ \hline \end{array}$$

Round	1's η	2's η	1's action	2's action
1	(1.5, 2)	(2, 1.5)	T	T
2	(1.5, 3)	(2, 2.5)	T	H
3	(2.5, 3)	(2, 3.5)	T	H
4	(3.5, 3)	(2, 4.5)	H	H
5	(4.5, 3)	(3, 4.5)	H	H
6	(5.5, 3)	(4, 4.5)	H	H
7	(6.5, 3)	(5, 4.5)	H	T

- FP continues as deterministic cycle, time average converges to unique NE

Example: (Anti-)Coordination Game

- Note that if empirical distribution of actions converges to NE, there is no guarantee on distribution of played outcomes
- Consider the following coordination game

$$\begin{array}{c|c|c|} & \text{A} & \text{B} \\ \hline \text{A} & 1,1 & 0,0 \\ \hline \text{B} & 0,0 & 1,1 \\ \hline \end{array}$$

- Note that this game is unique NE of ((0.5, 0.5), (0.5, 0.5))

Round	1's η	2's η	1's action	2's action
1	(0.5, 0)	(0, 0.5)	A	B
2	(0.5, 1)	(1, 0.5)	B	A
3	(1.5, 1)	(1, 1.5)	A	B
4	(1.5, 2)	(2, 1.5)	B	A

General Fictitious Play Convergence

- Fictitious play converges in time-average sense for game G if:
  - G is zero-sum game
  - G is two-player game where each agent has at most two actions (2x2 games)
  - G is solvable by iterated strict dominance
  - G is identical-interest game, i.e., all agents have same payoff function
  - G is potential game (more on this later!)

Non-convergence of Fictitious Play

- Convergence of fictitious play can not be guaranteed in general
- Shapley showed that in modified rock-scissors-paper game, FP does not converge

	Rock	Paper	Scissors
Rock	$0, 0$	$0, 1$	$1, 0$
Paper	$1, 0$	$0, 0$	$0, 1$
Scissors	$0, 1$	$1, 0$	$0, 0$

- This game has unique NE: each agent mixes uniformly
- Suppose $\eta_1^1 = (1, 0, 0)$ and $\eta_2^1 = (0, 1, 0)$
- Shapley showed that play cycles among 6 (off-diagonal) profiles with periods of ever-increasing length, thus non-convergence

Smooth Fictitious Play (SFP)

- Instead of best-responding to beliefs, agents respond randomly, but somewhat proportional to their expected utility

$$s_i^t(a_i \mid \mu_i^t) = \frac{\exp(u_i(a_i, \mu_i^t)/\gamma)}{\sum_{a_i'} \exp(u_i(a_i', \mu_i^t)/\gamma)}$$

- γ is called the smoothing parameter
- This is called soft-max policy
- Soft-max policy respects best replies, but leaves room for exploration
- If all agents use SFP with sufficiently small γ_i, empirical play converges to ϵ-CCE

Outline

Best-response Dynamics (BRD): Introduction

- Agents start playing arbitrary actions
- In arbitrary order, agents take turns updating their action
- Agent update their action only if doing so can improve their utility
- This is repeated until no agents wants to update their action

```
Initialize $a = (a_1, \ldots, a_n)$ to be arbitrary action profile;
while there exists i such that $a_i \notin \operatorname{argmax}_{a \in A_i} u_i(a, a_{-i})$ do
    Let $a'_i$ be such that $u_i(a'_i, a_{-i}) > u(a)$;
    Set $a_i \leftarrow a'_i$;
return $a$
```

Best-response Dynamics: Discussion

- If BRD halts, it returns pure strategy Nash equilibrium
    - Every agent must be playing best response
- Does BRD always halt?
    - No: Consider matching pennies/Rock Paper Scissors

Example: Congestion Games

- N is set of n agents
- M is set of m resources
- A_i is set of actions available to agent i
  - a_i represents subset of resources that agent i chooses (i.e., $a_i \subseteq M$)
- ℓ_j is congestion cost function for resources $j \in M$
  - $\ell_j(k)$ represents cost of congestion on resource j when k agents choose j
- $n_j(a)$ is number of agents who choose resource j (i.e., $n_j(a) = |\{i \mid j \in a_i\}|$)
- $c_i(a) = \sum_{j \in a_i} \ell_j(n_j(a))$ is total cost of agent
- Agents minimize their total cost (instead of maximizing their total utility)

BRD in Congestion Games

- Consider potential function $\phi : A \to \mathbb{R}$:

$$\phi(a) = \sum_{j=1}^{m} \sum_{k=1}^{n_j(a)} \ell_j(k)$$

(Note: not social welfare)

- How does ϕ change in one round of BRD? Say i switches from a_i to $b_i \in A_i$
- Well... We know it must have decreased agent i's cost:

$$\begin{aligned} \Delta c_i &\equiv c_i(b_i, a_{-i}) - c_i(a_i, a_{-i}) \\ &= \sum_{j \in b_i \setminus a_i} \ell_j(n_j(a)+1) - \sum_{j \in a_i \setminus b_i} \ell_j(n_j(a)) < 0 \end{aligned}$$

BRD in Congestion Games (cont.)

$$\phi(a) = \sum_{j=1}^{m} \sum_{k=1}^{n_j(a)} \ell_j(k)$$

- Change in potential is:

$$\begin{aligned}\Delta\phi &\equiv \phi(b_i, a_{-i}) - \phi(a_i, a_{-i}) \\ &= \sum_{j \in b_i \setminus a_i} \ell_j(n_j(a)+1) - \sum_{j \in a_i \setminus b_i} \ell_j(n_j(a)) \\ &= \Delta c_i\end{aligned}$$

- Since ϕ can take on only finitely many values, this cannot go on forever
- And hence BRD halts in congestion games ...
- Which proves the existence of pure strategy Nash equilibria!

Example: Load Balancing Games on Identical Servers

- n clients $i \in N$ schedule jobs of size $w_i > 0$ on m identical servers M
- Action space $A_i = M$ for each client
- For each server $j \in M$, load $\ell_j(a) = \sum_{i:a_i=j} w_i$
- Cost of client i is load of server that i chooses : $c_i(a) = \ell_{a_i}(a)$

Load Balancing Games on Identical Servers: Discussion

- *Almost* congestion game — but server costs depend on which clients choose them
- BRD converges in load balancing games on identical servers
- Load balancing games on identical servers have pure strategy NE

BRD in Load Balancing Games on Identical Servers

- Consider potential function ϕ as:

$$\phi(a) = \frac{1}{2}\sum_{j=1}^{m} \ell_j(a)^2$$

- Suppose i switches from server j to server j':

$$\begin{aligned} \Delta c_i(a) &\equiv c_i(j', a_{-i}) - c_i(j, a_{-i}) \\ &= \ell_{j'}(a) + w_i - \ell_j(a) \\ &< 0 \end{aligned}$$

BRD in Load Balancing Games on Identical Servers (cont.)

$$\begin{aligned}
\Delta\phi(a) &\equiv \phi(j', a_{-i}) - \phi(j, a_{-i}) \\
&= \frac{1}{2}\left((\ell_{j'}(a) + w_i)^2 + (\ell_j(a) - w_i)^2 - \ell_{j'}(a)^2 - \ell_j(a)^2\right) \\
&= \frac{1}{2}\left(2w_i\ell_{j'}(a) + w_i^2 - 2w_i\ell_j(a) + w_i^2\right) \\
&= w_i\left(\ell_{j'}(a) + w_i - \ell_j(a)\right) \\
&= w_i \cdot \Delta c_i(a) \\
&< 0
\end{aligned}$$

Note: $\Delta c_i \neq \Delta\phi$

Potential Games

- $\phi : A \to \mathbb{R}_{\geq 0}$ is exact potential function for game G if for all a, i,a_i, and b_i:

$$\phi(b_i, a_{-i}) - \phi(a_i, a_{-i}) = c_i(b_i, a_{-i}) - c_i(a_i, a_{-i})$$

- $\phi : A \to \mathbb{R}_{\geq 0}$ is ordinal potential function for game G if for all a, i,a_i, and b_i:

$$(c_i(b_i, a_{-i}) - c_i(a_i, a_{-i}) < 0) \Rightarrow (\phi(b_i, a_{-i}) - \phi(a_i, a_{-i}) < 0)$$

(i.e. the change in utility is always equal in sign to the change in potential)

- BRD is guaranteed to converge in game G iff G has ordinal potential function

BRD and Potential Games

- We've already seen ordinal potential function $\Rightarrow$ BRD converges
- Lets prove other direction
- Consider graph $G = (V, E)$
- Let each $a \in A$ be a vertex in G (i.e., $V = A$)
- Add directed edge (a, b) if it is possible to go from b to a by best-response move
    - I.e., if there is i such that $b = (b_i, a_{-i})$, and $c_i(b_i, a_{-i}) < c_i(a)$
- BRD can be viewed as traversing this graph
    - Start at arbitrary vertex a, and then traverse arbitrary outgoing edges

BRD and Potential Games (cont.)

- Nash Equilibria are the sinks in this graph
- Suppose BRD converges $\Rightarrow$ there are no cycles in this graph
- So, from every vertex a there is some sink s that is reachable (why?)
- We construct potential function $\phi(a)$ for each vertex a
- $\phi(a)$ is length of longest finite path from a to any sink s
- We need: for any edge $a \rightarrow b$, $\phi(b) < \phi(a)$.
- Its true! $\phi(a) \geq \phi(b) + 1$. (why?)

Outline

Sequential Prediction: Stock-prediction Example

- Every day GME goes up or down
- Goal is to predict direction each day before market opens (to buy or short)
- Market can behave arbitrarily/adversarially
- So there is no way we can promise to do well
- However, we get advice

Expert Advice

- There are N experts who make predictions in T rounds
- At each round t, each expert i makes prediction $p_i^t \in \{U, D\}$
- Expertise is self proclaimed — no promise experts know what they're talking about
- We (algorithm) want to aggregate predictions, to make our own prediction p_A^t
- We learn true outcome o^t at the end of each round
- If we predicted incorrectly (i.e. $p_A^t \neq o^t$), then we made a mistake

Expert Advice (cont.)

- Goal is to after a while do (almost) as well as best expert in hindsight
- To make things easy, we assume for now that there is one perfect expert
- Perfect expert never makes mistakes (but we don't know who the expert is)
- Can we find strategy that is guaranteed to make at most $\log(N)$ mistakes?

The Halving Algorithm

```
Let $S^1 \leftarrow \{1, \ldots, N\}$ be set of all experts;
for $t = 1$ to $T$ do
    Predict with majority vote;
    Observe the true outcome $o^t$;
    Eliminate all experts that made a mistake: $S^{t+1} = \{i \in S^t \mid p_i^t = o^t\}$;
```

The Halving Algorithm: Analysis

- Algorithm predicts with majority vote
- Every time it makes a mistake, at least half of remaining experts are eliminated
- Hence $|S^{t+1}| \leq |S^t|/2$
- On the other hand, perfect expert is never eliminated
- Hence $|S^t| \geq 1$ for all t
- Since $|S^1| = N$, this means there can be at most $\log N$ mistakes
- But what if no expert is perfect? Say the best expert makes OPT mistakes
- Can we find a way to make not too many more than OPT mistakes?

The Iterated Halving Algorithm

```
Let $S^1 \leftarrow \{1, \ldots, N\}$ be the set of all experts;
for $t = 1$ to $T$ do
    if $|S^t| = 0$ then
        Reset: Set $S^t \leftarrow \{1, \ldots, N\}$
    Predict with majority vote;
    Eliminate all experts that made a mistake: $S^{t+1} = \{i \in S^t \mid p_i^t = o^t\}$;
```

The Iterated Halving Algorithm: Analysis

- Whenever algorithm makes mistake, we eliminate half of experts
- So algorithm can make at most $\log N$ mistakes between any two resets
- But if we reset, it is because since last reset, every expert has made mistake
- In particular, between any two resets, best expert has made at least 1 mistake
- Algorithm makes at most $\log(N)(\mathrm{OPT}+1)$ mistakes
- Algorithm is wasteful in that every time we reset, we forget what we have learned!
- How about just downweight experts who make mistakes?

The Weighted Majority Algorithm

```
Set weights $w_i^1 \leftarrow 1$ for all experts $i$;
for $t = 1$ to $T$ do
    Predict with weighted majority vote;
    Down-weight experts who made mistakes: (i.e., if $p_i^t \neq o^t$, set $w_i^{t+1} \leftarrow w_i^t/2$)
```

The Weighted Majority Algorithm: Analysis

- Let M be total number of mistakes that algorithm makes
- Let $W^t = \sum_i w_i^t$ be total weight at step t
- When algorithm makes mistake, at least half of total weight is cut in half
- So: $W^{t+1} \leq (3/4)W^t$
- If algorithm makes M mistakes, $W^T \leq N \cdot (3/4)^M$
- Let i^* be the best expert, $W^T > w_i^T = (1/2)^{\mathrm{OPT}}$, which gives:

$$(1/2)^{\mathrm{OPT}} \leq W \leq N(3/4)^M \Rightarrow (4/3)^M \leq N \cdot 2^{\mathrm{OPT}} \Rightarrow M \leq 2.4(\mathrm{OPT} + \log(N))$$

- Algorithm makes at most $2.4\,(\mathrm{OPT} + \log(N))$ mistakes
- $\log(N)$ is constant, so ratio of mistakes to OPT is 2.4 in limit – not great, but not bad

What Do We Want in an Algorithm?

- Make only $1\times$ as many mistakes as OPT in limit, rather than $2.4\times$
- Handle N distinct actions (separate action for each expert), not just up and down
- Handle arbitrary costs in $[0, 1]$ per expert per round, not just right and wrong

New Model/Algorithm

- In rounds $1, \ldots, T$, algorithm chooses some expert i^t
- Each expert i experiences loss: $\ell_i^t \in [0, 1]$
- Algorithm experiences the loss of the expert it chooses: $\ell_A^t = \ell_{i^t}^t$
- Total loss of expert i is $L_i^T = \sum_{t=1}^T \ell_i^t$
- Total loss of algorithm is $L_A^T = \sum_{t=1}^T \ell_A^t$
- Goal is to obtain loss "not much worse" than that of the best expert: $\min_i L_i^T$

Multiplicative Weights (MW) Algorithm (a.w.a. Hedge Algorithm)

```
Set weights $w_i^1 \leftarrow 1$ for all experts $i$;
for $t = 1$ to $T$ do
    Let $W^t = \sum_{i=1}^{N} w_i^t$;
    Choose expert $i$ with probability $w_i^t / W^t$;
    For each $i$, set $w_i^{t+1} \leftarrow w_i^t \cdot \exp(-\epsilon \ell_i^t)$;
```

- Can be viewed as "smoothed" version of weighted majority algorithm
- Has parameter ϵ which controls how quickly it down-weights experts
- Is *randomized* — chooses experts w.p. proportional to their weights
- Can be used with alternative update: $w_i^{t+1} \leftarrow w_i^t \cdot (1 - \epsilon \ell_i^t)$

Multiplicative Weights Algorithm: Discussion

- For any sequence of losses, and any expert k:

$$\frac{1}{T}\ \mathbb{E}[L_{MW}^T] \leq \frac{1}{T}\ L_k^T + \epsilon + \frac{\ln(N)}{\epsilon \cdot T}$$

- In particular, setting $\epsilon = \sqrt{\ln(N)/T}$:

$$\frac{1}{T}\ \mathbb{E}[L_{MW}^T] \leq \frac{1}{T}\ \min_k L_k^T + 2\sqrt{\frac{\ln(N)}{T}}$$

- Average loss quickly approaches that of best expert exactly, at rate of $1/\sqrt{T}$
- This works for arbitrary sequence of losses (e.g., chosen adaptively by adversary)
- So we could us it to play games (experts $\leftrightarrow$ actions and losses $\leftrightarrow$ costs)

Recall: Minimax Theorem (John von Neumann, 1928)

In any finite, two-player, zero-sum game, in any NE, each agent receives a payoff that is equal to both their maxmin value and their minmax value

$$\max_{s_i} \min_{s_{-i}} u_i(s_i, s_{-i}) = \min_{s_{-i}} \max_{s_i} u_i(s_i, s_{-i})$$

Simple Proof for Minimax Theorem

- Scale utilities such that u_1 is in $[0, 1]$
- Write $v_1 = \min_{s_2} \max_{s_1} u_1(s_1, s_2)$ and $v_2 = \max_{s_1} \min_{s_2} u_1(s_1, s_2)$
- Suppose theorem were false: $v_1 = v_2 + \epsilon$ for some constant $\epsilon > 0$
- Suppose A1 and A2 repeatedly play against each other as follows
  - A2 uses MW algorithm: at round t, $s_2^t(a_2) = w_{a_2}^t / W^t$
  - A1 plays best response to A2's strategy: $s_1^t = \text{argmax}_{s_1} u_1(s_1, s_2^t)$

Simple Proof for Minimax Theorem (cont.)

- For A2's MW algorithm, we have:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[u_1(a_1^t, a_2^t)] \leq \frac{1}{T}\min_{a_2}\sum_{t=1}^{T}u_1(a_1^t, a_2) + 2\sqrt{\frac{\log n}{T}}$$

- Let $\bar{s}_1$ be mixed strategy that puts weight $1/T$ on each action a_1^t, we have:

$$\frac{1}{T}\min_{a_2}\sum_{t=1}^{T}u_1(a_1^t, a_2) = \min_{a_2}\sum_{t=1}^{T}\frac{1}{T}u_1(a_1^t, a_2) = \min_{a_2}u_1(\bar{s}_1, a_2)$$

- By definition, we have: $\min_{a_2} u_1(\bar{s}_1, a_2) \leq \max_{s_1}\min_{a_2} u_1(s_1, a_2) = v_2$, and so:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[u_1(a_1^t, a_2^t)] \leq v_2 + 2\sqrt{\frac{\log n}{T}}$$

Simple Proof for Minimax Theorem (cont.)

- On the other hand, A1 best responds to A2's mixed strategy:

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[u_1(a_1^t, a_2^t)] &= \frac{1}{T}\sum_{t=1}^{T} \max_{a_1} u_1(a_1, s_2^t) \\ &\geq \frac{1}{T}\sum_{t=1}^{T} \min_{s_2} \max_{a_1} u_1(a_1, s_2) = \frac{1}{T}\sum_{t=1}^{T} v_1 = v1 \end{aligned}$$

- Combining these inequalities, we get: $v_1 \leq v_2 + 2\sqrt{\log n / T}$
- Since $v_1 = v_2 + \epsilon$, we have: $\epsilon \leq 2\sqrt{\log n / T}$
- Taking T large enough leads to contradiction

External Regret

- Sequence $a^1, \ldots, a^T$ has external regret of $\Delta(T)$ if for every agent i and action a'_i:

$$\frac{1}{T}\sum_{t=1}^{T} u_i(a^t) \geq \frac{1}{T}\sum_{t=1}^{T} u_i(a'_i, a_{-i}) - \Delta(T)$$

- If $\Delta(T) = o_T(1)$, we say that sequence of action profiles has *no* external regret
- External regret measures regret to the best fixed action in hindsight
- If $a^1, \ldots, a^T$ has ϵ external regret, then distribution π that puts weight $1/T$ on each a^t (i.e., empirical distribution of actions) forms ϵ-approximate CCE

$$\mathbb{E}_{a\sim\pi}[u_i(a)] = \frac{1}{T}\sum_{t=1}^{T} u_i(a^t) \geq \frac{1}{T}\sum_{t=1}^{T} u_i(a'_i, a_{-i}) - \epsilon = \mathbb{E}_{a\sim\pi}[u_i(a'_i, a_{-i})] - \epsilon$$

No-(external-)regret Dynamics

- Suppose that all agents use MW algorithm to choose between k actions
- After T steps, sequence of outcomes has external regret of $\Delta(T) = 2\sqrt{\log k/T}$
- Empirical distribution of outcomes forms $\Delta(T)$-approximate CCE
- For $T = 4\log(k)/\epsilon^2$, distribution of outcomes converges to ϵ-approximate CCE

Swap Regret

- Sequence $a^1, \dots, a^T$ has swap regret of $\Delta(T)$ if for every agent i and every switching function $F_i : A_i \to A_i$:

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

- If $\Delta(T) = o_T(1)$, we say that sequence of action profiles has no swap regret
- This measures regret to counterfactual case where every action of particular type is swapped with different action in hindsight, separately for each action
- E.g., "Every time i bought Microsoft, i should have bought Apple, and every time i bought Google, i should have bought Comcast."
- If $a^1, \dots, a^T$ has ϵ swap regret, then distribution π that picks among $a^1, \dots, a^T$ uniformly at random is ϵ-approximate correlated equilibrium

Generalization

- For any agent i, F_i, and $a \in A$, define regret as:

$$\mathrm{Regret}_i(a, F_i) = u_i(F_i(a_i), a_{-i}) - u_i(a)$$

- F_i is constant switching function if $F_i(a_i) = F_i(a_i')$ for all $a_i, a_i' \in A_i$
- π is CCE if for every agent i and every constant switching function F_i:

$$\mathbb{E}_{a\sim\pi}[\mathrm{Regret}_i(a, F_i)] \leq 0$$

- π is CE if for every agent i and every switching function F_i:

$$\mathbb{E}_{a\sim\pi}[\mathrm{Regret}_i(a, F_i)] \leq 0$$

How to Achieve No Swap Regret

- Define set of time steps that expert j is selected:

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

- Observation: To achieve no swap regret it would be sufficient that for every j:

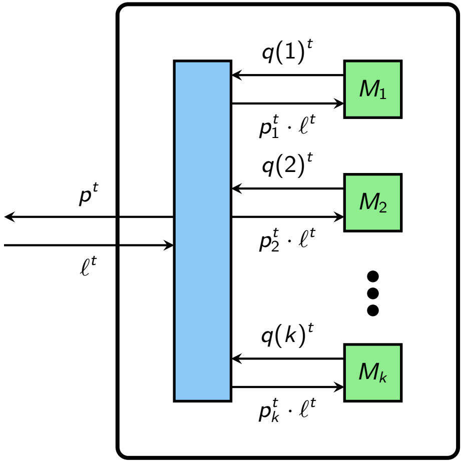
$$\frac{1}{|S_j|} \sum_{t \in S_j} \ell_{a_t}^t \leq \frac{1}{|S_j|} \min_i \sum_{t \in S_j} \ell_i^t + \Delta(T)$$

- No swap regret = no external regret separately on each sequence of actions S_j
- Best switching function in hindsight = swapping each action j for best fixed action in hindsight over S_j
- Idea: Run k copies of PW, one responsible for each S_j

Algorithm Sketch for No Swap Regret

- Initialize k copies of MW algorithm one for each of k actions
- Let $q(i)_1^t, \ldots, q(i)_k^t$ be distribution over experts for copy i at time t
- Combine these into single distribution over experts: $p_1^t, \ldots, p_k^t$ (details later!)
- Let $\ell_1^t, \ldots, \ell_k^t$ be losses for experts at time t
- For copy i of MW algorithm, we report losses $p_i^t \ell_1^t, \ldots, p_i^t \ell_k^t$
- I.e., to copy i, we report the true losses scaled by p_i^t

No-swap-regret Algorithm



No-swap-regret Algorithm: Analysis

- Expected cost of the master algorithm:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{k} p_i^t \cdot \ell_i^t \tag{1}$$

- Expected cost under switching function F

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{k} p_i^t \cdot \ell_{F(i)}^t \tag{2}$$

- Goal: prove that (1) is at most (2) plus $\Delta(T) = o_T(1)$

No-swap-regret Algorithm: Analysis (cont.)

- Expected cost of M_j:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{k} q(j)_i^t \left(p_j^t \cdot \ell_i^t\right) \tag{3}$$

- M_j is no-regret algorithm, so its cost is at most:

$$\frac{1}{T}\sum_{t=1}^{T} p_j^t \cdot \ell_{F(j)}^t + \Delta(T) \tag{4}$$

for any any arbitrary F

No-swap-regret Algorithm: Analysis (cont.)

- Summing inequality between (3) and (4) over all copies:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{k}\sum_{j=1}^{k} q(j)_i^t \left(p_j^t \cdot \ell_i^t\right) \leq \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} p_j^t \cdot \ell_{F(j)}^t + k \cdot \Delta(T) \tag{5}$$

- Right-hand side is equal to (2)
- For left-hand side to be equal to (1), we need:

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

Combining Distributions

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

- These might be familiar as those defining stationary distribution of Markov chain
  - There are k states, probability of going to state i from j is $q(j)_i^t$
  - Stationary distribution over states is $(p_1^t \dots p_k^t)$
- These equations always have solution as probability distribution
- Crucial property: two ways of viewing the distribution over experts:
  - Each expert i is chosen with probability p_i^t or
  - W.p. p_j^t we select copy j and then select expert i w.p. $q(j)_i^t$

Regret Matching

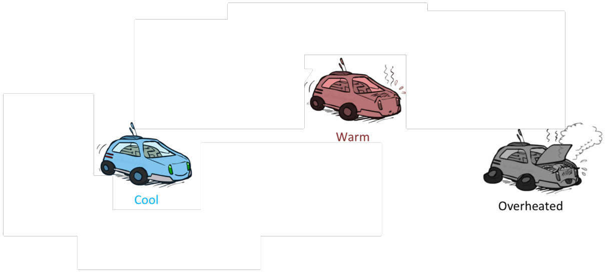
- α^t: Average per-step reward received by agent up until time t
- $\alpha^t(a)$: Average per-period reward that would have been received up until time t had pure strategy a was played by agent, assuming others played the same
- Regret at time t for not having played a: $R^t(a) = \alpha^t(a) - \alpha^t$
- Regret matching: At time t, choose action a w.p. proportional to its regret:

$$s^t(a) = \frac{R^t(a)^+}{\sum_{a'} R^t(a')^+}$$

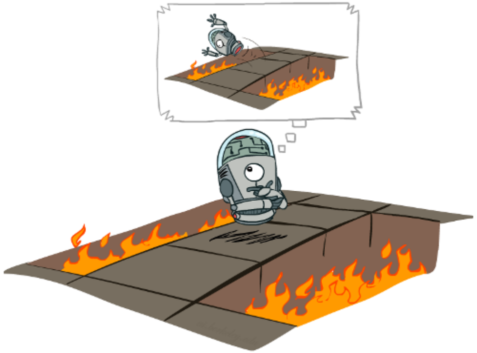
Outline

Reinforcement Learning

- Still assume MDP
  - Set of states $s \in S$
  - Set of actions $a \in A$
  - Model $p(s, a, s')$
  - Reward $r(s, a, s')$
- Still looking for policy $\pi(s)$
- New twist: we do not know p or r
- I.e. we do not know which states are good or what actions do
- Must actually try actions and states out to learn



Offline (MDPs) vs. Online (RL)


Offline solution

Online solution

Why Not Use Policy Evaluation?

- Simplified Bellman updates calculate V and Q for a fixed policy

$$V_t^{\pi}(s) \leftarrow \sum_{s'} p(s, \pi(s), s') \left(r(s, \pi(s), s') + \delta V_{t-1}^{\pi}(s') \right)$$

- This approach fully exploited connections between the states
- Unfortunately, we need p and r to do it!

Temporal Difference (TD) Learning

- Main idea: learn from every experience!
  - Update $V(s)$ each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often
- Temporal difference learning of values
  - Policy still fixed, still doing evaluation!
  - Move values toward value of whatever successor occurs: running average

$$\begin{aligned}
\text{Sample of } V(s): &\quad r(s,a,s') + \delta V^{\pi}(s') \\
\text{Update of } V(s): &\quad V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left(r(s,a,s') + \delta V^{\pi}(s')\right) \\
\text{Same update}: &\quad V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha\left(r(s,a,s') + \delta V^{\pi}(s') - V^{\pi}(s)\right)
\end{aligned}$$

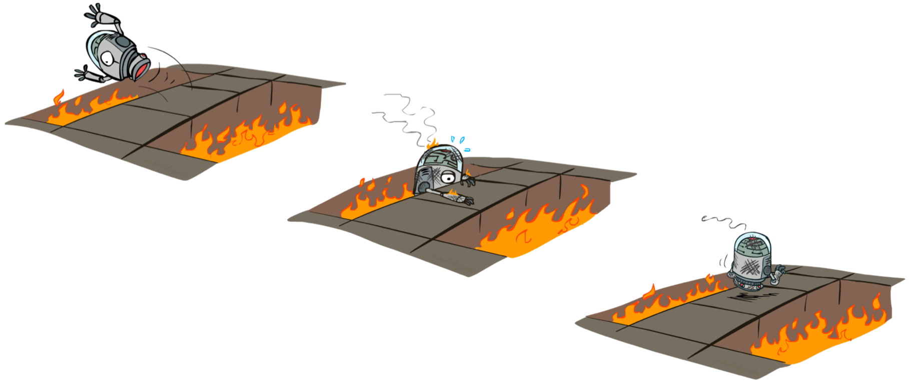
Problems with TD Value Learning

- TD value leaning is model-free way to do policy evaluation
- It mimics Bellman updates with running sample averages
- However, if we want to turn values into (new) policy, we need p and r!

$$\begin{aligned} \pi(s) &= \operatorname*{argmax}_a Q(s,a) \\ Q^{\pi}(s,a) &= \sum_{s'} p(s,a,s')\left(r(s,a,s') + \delta V(s')\right) \end{aligned}$$

- To solve this, we can learn Q-values instead of values
- This makes action selection model-free too!

Active Reinforcement Learning



Q-learning

- Q-Learning is sample-based Q-value iteration

$$Q_t(s,a) \leftarrow \sum_{s'} p(s,a,s') \left(r(s,a,s') + \delta \max_{a' \in A} Q_{t-1}(s',a') \right)$$

- We learn $Q(s,a)$ values as we go

$$\text{Sample}: \quad r(s,a,s') + \delta \max_{a' \in A} Q(s',a')$$

$$\text{Update}: \quad Q(s,a) \leftarrow (1-\alpha_t) Q(s,a) + \alpha_t \left(r(s,a,s') + \delta \max_{a' \in A} Q(s',a') \right)$$

Q-learning Algorithm

```
repeat until convergence
    observe current state $s$;
    select action $a$ and take it (e.g., via $\epsilon$-greedy policy);
    observe next state $s'$ and reward $r(s, a, s')$;
    $Q_{t+1}(s, a) \leftarrow (1 - \alpha_t)Q_t(s, a) + \alpha_t \left(r(s, a, s') + \delta V_t(s')\right)$;
    $V_{t+1}(s) \leftarrow \max_a Q_t(s, a)$;
```

- ϵ-greedy: W.p. ϵ, act randomly, w.p. $(1 - \epsilon)$ act according to Q_t

Q-learning Properties

- Q-learning converges to optimal policy – even if agent acts sub-optimally!
- This is called off-policy learning
- There are some caveats
    - We have to explore enough
    - We have to eventually make the learning rate small enough
    - But we should not decrease it too quickly
    - Q-learning converges if $\sum_0^\infty \alpha_t = \infty$ and $\sum_0^\infty \alpha_t^2 < \infty$
    - Basically, in the limit, it does not matter how you select actions (!)

Outline

Independent Single-agent RL

- Setting: Two-player zero-sum games
- Naive idea: Agents ignore the existence of their opponent
- $Q_i^\pi(s, a_i)$: Value for i if both agents follow π starting from s and i plays a_i
- Learning dynamics: Agents deploy independent Q-learning
- Good news: No-regret property if opponent plays stationary policy
- Bad news: No convergence guarantee if both agents are learning (e.g., self play)!

Minimax-Q

- Littman[2] extended Q-learning algorithm to zero-sum stochastic games
- Main idea is to modify Q-function to consider actions of opponent

$$Q_{i,t+1}(s_t, a_t) = (1 - \alpha_t) Q_{i,t}(s_t, a_t) + \alpha_t \left(r_i(s_t, a_t) + \delta V_{i,t}(s_{t+1}) \right)$$

- Since game is zero sum, we can have

$$V_{i,t}(s) = \max_{\pi_i} \min_{a_{-i}} Q_{i,t}(s, \pi_i, a_{-i})$$

[2] Littman, M. L. "Markov games as a framework for multi-agent reinforcement learning." 1994

Minimax-Q Algorithm

```
**repeat** *until convergence*
    observe current state $s$;
    select action $a_i$ and take it (e.g., via $\epsilon$-greedy policy);
    observe action profile $a$;
    observe next state $s'$ and reward $r(s, a, s')$;
    $Q_{i,t+1}(s, a) \leftarrow (1 - \alpha_t)Q_{i,t}(s, a) + \alpha_t \left(r(s, a) + \delta V_{i,t}(s')\right)$;
    $\pi_i(s, \cdot) \leftarrow \mathrm{argmax}_{\pi'} \min_{a_{-i}} \sum_{a_i} \pi'(s, a_i)Q_{i,t}(s, a_i, a_{-i})$;
    $V_{t+1}(s) \leftarrow \min_{a_{-i}} \sum_{a_i} \pi(s, a_i)Q_{i,t}(s, a_i, a_{-i})$;
```

Minimax-Q Algorithm: Discussion

- It guarantees agents payoff at least equal to that of their maxmin strategy
- In zero-sum games, minimax-Q converges to the value of the game in self play
- It no longer satisfies no-regret property
- If opponent plays sub-optimally, minimax-Q does not exploit it in most games

Nash-Q

- Hu and Wellman[3] extended minimax-Q to general-sum games
- Algorithm is structurally identical to minimax-Q
- Extension requires that each agent maintains values for all other agents
- LP to find maxmin value is replaced with quadratic programming to find NE
- Nash-Q makes number of very limiting assumptions (e.g., uniqueness of NE)

[3] Hu, J, and Wellman, M. P. "Multiagent reinforcement learning: theoretical framework and an algorithm." 1998

Recall: Stochastic Games Model

- Focus on stationary Markov strategies (a mixed strategy per state)
- $\pi_i : S \mapsto \Delta(A_i)$ denotes (mixed) strategy of agent i at state s
- $\pi = (\pi_1, \ldots, \pi_n)$ denotes strategy profile of all agents
- Expected utility (value) function of agent i is

$$v_i(s, \pi) := \mathbb{E}_{a_k \sim \pi(s_k)} \left[\sum_{k=0}^{\infty} \delta^k r_i(s_k, a_k) \mid s_0 = s \right]$$

Equilibrium Characterization

- Equilibrium value function is defined using one-stage deviation principle (multi-agent extension of Bellman's equation) as

$$v_i(s,\pi^*) = \max_{\pi_i} \mathbb{E}_{a\sim(\pi_i,\pi^*_{-i}(s))}\left[r_i(s,a) + \delta \sum_{s'\in S} p(s,a,s')v_i(s',\pi^*)\right]$$

- Q-function is defined as

$$Q_i(s,a,\pi^*) = r_i(s,a) + \delta \sum_{s'\in S} p(s,a,s')v_i(s',\pi^*)$$

- Recursion is then defined as

$$v_i(s,\pi^*) = \max_{\pi_i} \mathbb{E}_{a\sim(\pi_i,\pi^*_{-i}(s))}\left[Q_i(s,a,\pi^*)\right]$$

FP for Model-based Learning

- Consider learning dynamic that combines FP with value-function (or Q-function) iteration
- Agents form beliefs on opponent strategies (using empirical frequencies and assuming opponent uses stationary strategy)
- Agents also form beliefs about equilibrium value function, or Q-function
- Agents then choose best response action in auxiliary game given their beliefs (where payoffs are given by Q-function estimates)
- Key challenge is that payoffs or value functions in these auxiliary games are non-stationary (unlike repeated play of stage games)

FP for Model-based Learning: Model

- At time t, i's belief on $-i$'s strategy is μ_i^t and on own Q-function is

$$Q_i^t := \mathbb{E}_{a_{-i} \sim \mu_i^t(s)}[Q_i^t(s, a_i, a_{-i})]$$

- Agent i selects best response $a_i^t(s) \in \operatorname{argmax}_{a_i} Q_i^t(s, a_i, \mu_i^t(s))$
- Agent i updates μ_i as

$$\mu_i^{t+1}(s) = (1 - \alpha_t)\mu_i^t(s) + \alpha_t a_{-i}^t(s)$$

- Agent i updates Q_i as

$$Q_i^{t+1}(s, a) = (1 - \beta_t)Q_i^t(s, a) + \beta_t \left(r_i(s, a) + \delta \sum_{s' \in S} p(s, a, s') v_i^t(s') \right)$$

where $v_i^t(s') = \max_{a_i} Q_i^t(s', a_i, \mu_i^t(s))$

Two-timescale Learning Framework

- Beliefs on Q-functions are updated at slower rate than beliefs on opponent strategies
- This postulate agents' choices to be more dynamic than changes in their preferences
- Q-functions in auxiliary games can be viewed as slowly evolving agent preferences
- This enables weakening the dependence between evolving strategies and Q-functions

Convergence of Two-timescale Learning Framework

- If each state is visited infinitely many times
- And, if $\lim_{k\to\infty} \alpha_k = \lim_{k\to\infty} \beta_k = 0$ and $\sum_k \alpha_k = \sum_k \beta_k = \infty$
- And, if $\lim_{k\to\infty} \beta_k/\alpha_k = 0$ (two-timescale learning: $\beta_k \to 0$ faster than $\alpha_k \to 0$)
- Then Q and μ converge to NE value and strategy in zero-sum stochastic games
- They also converge to NE value for single-controller stochastic games

Acknowledgment

- This lecture is a slightly modified version of ones prepared by
    - Asu Ozdaglar [MIT 6.254]
    - Vincent Conitzer [Duke CPS 590.4]
    - Aaron Roth [UPenn NETS 412]
    - Dan Klein and Pieter Abbeel [UC Berkeley CS 188]