CS 440/ECE448 Lecture 24:
Repeated Games

Mark Hasegawa-Johnson, 3/2024
These slides are in the public domain.
Outline

• Mathematical foundations
• Repeated games and the rational basis for cooperation
• Learning an episodic game
• Learning a sequential game
Review: Markov decision process

- \( s \in S \): state of the environment (could be int, real, tuple, whatever)
  - \( r(s) \in \mathbb{R} \): reward received in state \( s \)
  - \( u(s) \in \mathbb{R} \): utility of state \( s \) = expected discounted sum of all future rewards
- \( a \in A \): action (usually \( A \) is a discrete finite set)
  - \( \pi : S \rightarrow A \): policy = best action for each state
- The optimum action is given by Bellman’s equation:
  \[
  u(s) = r(s) + \gamma \max_a \sum_{s'} P(S_{t+1} = s'|S_t = s, a)u(s')
  \]
Review: Expectiminimax

• In a two-player, zero-sum game, each player wins exactly as much as the other player loses.
• The player trying to maximize $u(s)$ is called “Max,” the player trying to minimize $u(s)$ is called “Min.”
• Bellman’s equation:

$$u(s) = \begin{cases} 
  r(s) + \gamma \max_a \sum_{s'} P(S_{t+1} = s' | S_t = s, a) u(s') & \text{s is a max state} \\
  r(s) + \gamma \min_a \sum_{s'} P(S_{t+1} = s' | S_t = s, a) u(s') & \text{s is a min state}
\end{cases}$$
Simultaneous games

• If players play simultaneously, then the best strategy might be to choose a move at random (e.g., game of Chicken: make sure opponent can’t predict your action with certainty)

• Instead of a scalar $\pi(s) =$action, we can use

$$
\pi(s) = \begin{bmatrix} 
\pi_1(s) \\
\vdots \\
\pi_{|\mathcal{A}|}(s) 
\end{bmatrix}, \quad \varphi(s) = \begin{bmatrix} 
\varphi_1(s) \\
\vdots \\
\varphi_{|\mathcal{B}|}(s) 
\end{bmatrix}
$$

• $\pi_a(s) =$probability that player 1 chooses action $a$ in state $s$

• $\varphi_b(s) =$probability that player 2 chooses action $b$ in state $s$

$$
0 \leq \pi_a(s) \leq 1, \quad \sum_{a=1}^{|\mathcal{A}|} \pi_a(s) = 1, \quad 0 \leq \varphi_b(s) \leq 1, \quad \sum_{b=1}^{|\mathcal{B}|} \varphi_b(s) = 1,
$$
Rewards and utility for simultaneous games

• $r_1(s, a, b) = $ Reward that player 1 receives in state $s$ if player 1 chooses action $a$ and player 2 chooses action $b$

• $r_2(s, a, b) = $ Reward that player 2 receives in state $s$ if player 1 chooses action $a$ and player 2 chooses action $b$

• $u_1(s) = $ Utility of state $s$ for player 1

• $u_2(s) = $ Utility of state $s$ for player 2

• $P(s'|s, a, b) = $ Probability of a transition to $s'$ from $s$ if player 1 chooses action $a$ and player 2 chooses action $b$
Bellman’s equation for repeated simultaneous two-player games

The probability of action $a$ is $\pi_a(s)$, the probability of action $b$ is $\varphi_b(s)$, and the probability of a transition to state $s'$ is $P(s'|s,a,b)$, so the expected sum of all future rewards under policies $\pi$ and $\varphi$ is:

$$u_1(s) = \sum_{a,b} \pi_a(s)\varphi_b(s) \left( r_1(s,a,b) + \gamma \sum_{s'} P(s'|s,a,b)u_1(s') \right)$$

$$u_2(s) = \sum_{a,b} \pi_a(s)\varphi_b(s) \left( r_2(s,a,b) + \gamma \sum_{s'} P(s'|s,a,b)u_2(s') \right)$$

The best policies, for each player, are:

$$\pi(s) = \arg\max_{\pi} u_1(s)$$

$$\varphi(s) = \arg\max_{\varphi} u_2(s)$$
Outline

• Mathematical foundations
• Repeated games and the rational basis for cooperation
• Learning an episodic game
• Learning a sequential game
Repeated games and the rational basis for cooperation

The Iterated Prisoner's Dilemma and The Evolution of Cooperation

https://www.youtube.com/watch?v=BOvAbjfJ0x0&t=331s

Image © This Place blog
Outline

• Mathematical foundations
• Repeated games and the rational basis for cooperation
• Learning an episodic game
• Learning a sequential game
Learning an episodic game

Repeated games can be either episodic or sequential:

• Sequential: your actions can change the environment $s$, which changes the reward $r_1(s, a, b)$ and the other player’s strategy $\varphi(s)$

• Episodic: your actions don’t change the environment. Your reward is $r_1(a, b)$, your strategy is $\pi$, your opponent’s reward is $r_2(a, b)$, and their strategy is $\varphi$.

Repeating an episodic game allows us to iteratively optimize our strategy vector $\pi$, using methods kind of like gradient descent.
Example: The lunch game

• Alice and Bob have agreed that they should always meet at the boat club for lunch.
• Each day, each of them must decide to either:
  • Cooperate: go to the boat club for lunch
  • Defect: go to the museum for lunch
• If they both cooperate, they eat lunch together
• If they both defect, they eat lunch together
• If one cooperates and the other defects, they each eat lunch alone
Example: The lunch game

- Alice prefers reading to talking
  - If they eat lunch together, she gets 1 happiness point
  - If they eat lunch alone, she gets 2 happiness points
- Bob prefers talking to reading
  - If they eat lunch together, he gets 2 happiness points
  - If they eat lunch alone, he gets 1 happiness point
Example: The lunch game

- Alice’s strategy is $\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix}$
  - $\pi_2 = \frac{1}{1+e^{-x}}$ is the probability she cooperates
  - $\pi_1 = 1 - \pi_2$

- Bob’s strategy is $\varphi = \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix}$
  - $\varphi_2 = \frac{1}{1+e^{-y}}$ is the probability he cooperates
  - $\varphi_1 = 1 - \varphi_2$

<table>
<thead>
<tr>
<th></th>
<th>Cooperate</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect</td>
<td>$2$</td>
<td>$1$</td>
</tr>
<tr>
<td></td>
<td>$2$</td>
<td>$1$</td>
</tr>
<tr>
<td>Alice</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$2$</td>
<td>$1$</td>
</tr>
<tr>
<td>Bob</td>
<td>$1$</td>
<td>$2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: The lunch game

- Alice’s strategy is $\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix}$

- Bob’s strategy is $\varphi = \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix}$

- Alice’s expected reward is
  $$u_1 = \pi^T R_1 \varphi = [\pi_1, \pi_2] \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix}$$

- Bob’s expected reward is
  $$u_2 = \pi^T R_2 \varphi = [\pi_1, \pi_2] \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix}$$
The Nash Equilibrium

- Alice’s expected reward is
  \[ u_1 = \pi^T \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \varphi \]
- Bob’s expected reward is
  \[ u_2 = \pi^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \varphi \]

- The Nash equilibrium is:
  \[ \pi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad \varphi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \]

...you can verify that this is a Nash equilibrium by noticing that

- If \( \pi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \), then Bob has no preference between cooperating and defecting, so he can choose at random.

- If \( \varphi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \), then Alice has no preference, and can choose at random.
Simultaneous gradient ascent

\[ u_1 = \pi^T R_1 \phi, \quad u_2 = \pi^T R_2 \phi, \quad \pi = \begin{bmatrix} e^{-x}/(1 + e^{-x}) \\ 1/(1 + e^{-x}) \end{bmatrix}, \quad \phi = \begin{bmatrix} e^{-y}/(1 + e^{-y}) \\ 1/(1 + e^{-y}) \end{bmatrix} \]

• Can we use some type of machine learning algorithm to find the ”optimum values” of \( \pi \) and \( \phi \), i.e., the Nash equilibrium?

• Alice chooses \( \pi \) to maximize \( u_1 \) for any given \( \phi \)

• Bob chooses \( \phi \) to maximize \( u_2 \) for any given \( \pi \)

• One thing we can try is “simultaneous gradient ascent:” adjust \( x \) and \( y \) in order to maximize both \( u_1 \) and \( u_1 \) simultaneously:

\[
\begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix} + \eta \begin{bmatrix} \nabla_x u_1 \\ \nabla_y u_2 \end{bmatrix}
\]

• (...where \( \nabla_x u_1 \) is general notation for the gradient of \( u_1 \) w.r.t. \( x \). In this case, since \( x \) is a scalar, \( \nabla_x u_1 = \partial u_1 / \partial x \))
Try the quiz!

Try the quiz:
https://us.prairielearn.com/pl/course_instance/147925/assessment/2408126
Simultaneous gradient ascent

- Surprisingly, simultaneous gradient ascent fails.
- The graph at right is the sequence of vectors
  \[
  \begin{bmatrix}
  \pi_2 \\ \varphi_2
  \end{bmatrix} = \begin{bmatrix}
  1/(1 + e^{-x}) \\
  1/(1 + e^{-y})
  \end{bmatrix}
  \]
  ...obtained using
  \[
  \begin{bmatrix}
  x' \\ y'
  \end{bmatrix} \leftarrow \begin{bmatrix}
  x \\ y
  \end{bmatrix} + \eta \begin{bmatrix}
  \partial u_1/\partial x \\
  \partial u_2/\partial y
  \end{bmatrix}
  \]
Simultaneous gradient ascent

- Why does it never converge?
- If Bob is at the boat club w/prob $\varphi_2 < 0.5$, then Alice increases $x$ (so she can eat alone more often)
- If Alice is at the boat club w/prob $\pi_2 > 0.5$, then Bob increases $y$ (so he can eat with her more often)
- If Bob is at the boat club w/prob $\varphi_2 > 0.5$, then Alice decreases $x$ (so she can eat alone more often)
- If Alice is at the boat club w/prob $\pi_2 < 0.5$, then Bob decreases $y$ (so he can eat with her more often)
- ... and so on, forever.
Digression: orbital mechanics

• This is exactly like the orbit of a spaceship around a planet (the equations are the same)
• We can make it converge the same way we would make a spaceship’s orbit decay: apply friction
The symplectic correction

- The solution is to apply friction. The friction term we apply is something that (Balduzzi et al., 2018) called “the symplectic correction” (named after orbital mechanics a.k.a. symplectic mechanics). It looks like this:

\[
\begin{bmatrix}
x \\
y
\end{bmatrix} \leftarrow \begin{bmatrix}
x \\
y
\end{bmatrix} + \eta (I + C) \begin{bmatrix}
\frac{\partial u_1}{\partial x} \\
\frac{\partial u_2}{\partial y}
\end{bmatrix}
\]

- The matrix \( C \) is called the symplectic correction. It is \( C = \lambda (H^T - H) \), where \( \lambda \) is a scalar, and \( H \) is something called the Hessian:

\[
H = \begin{bmatrix}
\frac{\partial^2 u_1}{\partial x^2} & \frac{\partial^2 u_1}{\partial x \partial y} \\
\frac{\partial^2 u_2}{\partial x \partial y} & \frac{\partial^2 u_2}{\partial y^2}
\end{bmatrix}
\]
Corrected gradient ascent

• The graph at right is the sequence of vectors
  \[
  \begin{bmatrix} \pi_2 \\ \varphi_2 \end{bmatrix} = \begin{bmatrix} 1/(1 + e^{-x}) \\ 1/(1 + e^{-y}) \end{bmatrix}
  \]
  ...obtained using
  \[
  \begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix} + \eta(I + C) \begin{bmatrix} \partial u_1/\partial x \\ \partial u_2/\partial y \end{bmatrix}
  \]
  • As you can see, the correction causes it to “fall” toward the Nash equilibrium at \( \pi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \ \varphi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}. \)
Outline

• Mathematical foundations
• Repeated games and the rational basis for cooperation
• Learning an episodic game
• Learning a sequential game
What about tit-for-tat?

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix} + \eta (I + C) \begin{bmatrix} \partial u_1 / \partial x \\ \partial u_2 / \partial y \end{bmatrix}
\]

...works if we only want each player to maximize their reward one game at a time, without thinking about future games.

• Strategies like tit-for-tat are a little more complicated: Tit-for-tat remembers how its opponent played last time, and retaliates if its opponent defected.
What about tit-for-tat?

We can model tit-for-tat by supposing that:

1. the reward matrix depends only on the actions, not the state
   \[ r_1(s_t, a_t, b_t) = r_1(a_t, b_t) \]
   \[ r_2(s_t, a_t, b_t) = r_2(a_t, b_t) \]

2. each player remembers a “state” variable consisting of the other player’s recent move:
   \[ s_t = (a_{t-1}, b_{t-1}) \]
   \[ P(S_{t+1} = s' | S_t = s, a, b) = \begin{cases} 
1 & s' = (a, b) \\
0 & \text{otherwise}
\end{cases} \]

Under these simplifications, the learning algorithm is four times harder than that of the episodic game: we have to learn \( \pi(s_t) \) and \( \varphi(s_t) \) separately for each state.
What about tit-for-tat?

The MP08 extra credit will ask you to create a strategy for a sequential game. You can learn it if you want to, or you can just specify it. For example, the tit-for-tat strategy is

\[
\pi(s) = \begin{cases} 
[1] & b_{t-1} = 1 \\
[0] & b_{t-1} = 2 \text{ or } t = 1 
\end{cases}
\]
Conclusions

• Policy probabilities:
  \[
  \pi_a(s) = \Pr(A_t = a | S_t = s) \\
  \varphi_b(s) = \Pr(B_t = b | S_t = s)
  \]

  \[
  u_1(s) = \sum_{a,b} \pi_a(s) \varphi_b(s) \left( r_1(s, a, b) + \gamma \sum_{s'} P(s' | s, a, b) u_1(s') \right)
  \]
  \[
  \pi(s) = \arg\max_{\pi} u_1(s)
  \]

• Learning episodic games using corrected gradient ascent:
  \[
  \begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \end{bmatrix} + \eta(I + C) \begin{bmatrix} \partial u_1 / \partial x \\ \partial u_2 / \partial y \end{bmatrix}
  \]