CS 309: Autonomous Intelligent Robotics
Instructor: Jivko Sinapov
Reinforcement Learning

- Internal state
- Reward
- Environment
- Action
- Observation
- Learning rate $\alpha$
- Inverse temperature $\beta$
- Discount rate $\gamma$
A little bit about next semester...

• New robots: robot arm, HSR-1 robot
• Virtually all of the grade will be based on a project
• There will still be some lectures and tutorials but much of the class time will be used to give updates on your projects and for discussions
Reinforcement Learning

internal state

environment

observation

learning rate $\alpha$
inverse temperature $\beta$
discount rate $\gamma$

reward
Activity: You are the Learner

At each time step, you receive an observation (a color)

You have three actions: “clap”, “wave”, and “stand”

After performing an action, you may receive a reward
Next time...

How can we formalize the strategy for solving this RL problem into an algorithm?
Project Breakout Session

Meet with your group

Summarize what you've done so far, identify next steps

Come up with questions for me, the TAs, and the mentors
Main Reference

Sutton and Barto, (2012). Reinforcement Learning: An Introduction, Chapter 1-3
What is Reinforcement Learning (RL)?
How Dog Training Works

1. Before Conditioning
   - Unconditioned Stimulus: Food
   - Unconditioned Response: Salivation
   - Response: Food → Salivation

2. Before Conditioning
   - Neutral Stimulus: Bell
   - Neutral Response: No Salivation
   - Response: Bell → No Salivation

3. During Conditioning
   - Unconditioned Stimulus: Bell + Food
   - Unconditioned Response: Salivation
   - Response: Bell + Food → Salivation

4. After Conditioning
   - Conditioned Stimulus: Bell
   - Conditioned Response: Salivation
   - Response: Bell → Salivation

©2006 HowStuffWorks
Ivan Pavlov (1849-1936)
WATCH WHAT I CAN MAKE PAVLOV DO. AS SOON AS I DROOL, HE'LL SMILE AND WRITE IN HIS LITTLE BOOK.
From Pavlov to Markov
Andrey Andreyevich Markov (1856 – 1922)
Markov Chain

- Bull Market
- Bear Market
- Recession

Transition Probabilities:
- From Bull Market to Bull Market: 0.9
- From Bull Market to Bear Market: 0.075
- From Bull Market to Recession: 0.15
- From Bear Market to Bull Market: 0.8
- From Bear Market to Bear Market: 0.25
- From Bear Market to Recession: 0.05
- From Recession to Bull Market: 0.25
- From Recession to Bear Market: 0.025
- From Recession to Recession: 0.5
Markov Decision Process
The Multi-Armed Bandit Problem

a.k.a. how to pick between Slot Machines (one-armed bandits) so that you walk out with the most $$$ from the Casino

Arm 1

Arm 2

......

Arm k
How should we decide which slot machine to pull next?
How should we decide which slot machine to pull next?
How should we decide which slot machine to pull next?

1 with prob = 0.6 and 0 otherwise

50 with prob = 0.01 and 0 otherwise
A value function encodes the “value” of performing a particular action (i.e., bandit)

\[ Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a} \]

Value function \( Q \)  

Rewards observed when performing action \( a \)  

\# of times the agent has picked action \( a \)
How do we choose next action?

- Greedy: pick the action that maximizes the value function, i.e.,

\[ Q_t(A_t^*) = \max_a Q_t(a) \]

- \( \varepsilon \)-Greedy: with probability \( \varepsilon \) pick a random action, otherwise, be greedy
10-armed Bandit Example

Average reward vs. Steps for different values of $\varepsilon$: $\varepsilon = 0.1$, $\varepsilon = 0.01$, $\varepsilon = 0$ (greedy)
Soft-Max Action Selection

Exponent of natural logarithm (~ 2.718)

As temperature goes up, all actions become nearly equally likely to be selected; as it goes down, those with higher value function outputs become more likely
What happens after choosing an action?

**Batch:**

\[ Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a} \]

**Incremental:**

\[ Q_{k+1} = \frac{1}{k} \sum_{i=1}^{k} R_i \]

\[ = \frac{1}{k} \left( R_k + \sum_{i=1}^{k-1} R_i \right) \]

\[ = \frac{1}{k} \left( R_k + (k - 1)Q_k + Q_k - Q_k \right) \]

\[ = \frac{1}{k} \left( R_k + kQ_k - Q_k \right) \]

\[ = Q_k + \frac{1}{k} \left[ R_k - Q_k \right], \]
Updating the Value Function

\[ \text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize} \left[ \text{Target} - \text{OldEstimate} \right] \]
What happens when the payout of a bandit is changing over time?

\[ Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a} \]
What happens when the payout of a bandit is changing over time?

Earlier rewards may not be indicative of how the bandit performs now.

\[ Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{K_a}}{K_a} \]
What happens when the payout of a bandit is changing over time?

\[ Q_{k+1} = Q_k + \alpha [R_k - Q_k] \]

instead of

\[ Q_k + \frac{1}{k} [R_k - Q_k] \]
How do we construct a value function at the start (before any actions have been taken)
How do we construct a value function at the start (before any actions have been taken)?

<table>
<thead>
<tr>
<th></th>
<th>Arm 1</th>
<th>Arm 2</th>
<th>Arm k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zeros</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Random</td>
<td>-0.23</td>
<td>0.76</td>
<td>-0.9</td>
</tr>
<tr>
<td>Optimistic</td>
<td>+5</td>
<td>+5</td>
<td>+5</td>
</tr>
</tbody>
</table>
The Multi-Armed Bandit Problems

The casino always wins – so why is this problem important?
The Reinforcement Learning Problem

Diagram showing the interaction between the Agent and the Environment. The Agent observes the state $S_t$, takes an action $A_t$, receives a reward $R_t$, and the Environment transitions to the next state $S_{t+1}$. The diagram includes the following elements:

- Agent
- Environment
- State $S_t$
- Reward $R_t$
- Action $A_t$
- Next State $S_{t+1}$
RL in the context of MDPs
The Markov Assumption

The award and state-transition observed at time \( t \) after picking action \( a \) in state \( s \) is independent of anything that happened before time \( t \).
Maze Example

- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent’s location

[slide credit: David Silver]
Maze Example: Value Function

- Numbers represent value $v_\pi(s)$ of each state $s$
Maze Example: Policy

- Arrows represent policy $\pi(s)$ for each state $s$
Maze Example: Model

- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect

- Grid layout represents transition model $P_{ss'}$
- Numbers represent immediate reward $R_s^a$ from each state $s$ (same for all $a$)

[slide credit: David Silver]
Notation

Set of States: \( S \)
Set of Actions: \( A \)
Transition Function:
\[
P : S \times A \mapsto \Pi(S)
\]
Reward Function:
\[
R : S \times A \mapsto \mathbb{R}
\]
Action-Value Function

\[ Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s' | s, a) \max_{a'} Q^*(s', a') \]
Action-Value Function

\[ Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s' | s, a) \max_{a'} Q^*(s', a') \]

- Discount factor (between 0 and 1)
- Probability of going to state \( s' \) from \( s \) after \( a \)
- The value of taking action \( a \) in state \( s \)
- The reward received after taking action \( a \) in state \( s \)
- \( a' \) is the action with the highest action-value in state \( s' \)
Action-Value Function

\[ Q^*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s'} \mathcal{P}(s' | s, a) \max_{a'} Q^*(s', a') \]

Common algorithms to learn the action-value function include Q-Learning and SARSA

The policy consists of always taking the action that maximize the action-value function
Q-Learning Example

• Example Slides
Q-Learning Algorithm

Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in S$ and $a \in A(s)$

Do forever:

(a) $s \leftarrow$ current (nonterminal) state
(b) $a \leftarrow \varepsilon$-greedy($s, Q$)
(c) Execute action $a$; observe resultant state, $s'$, and reward, $r$
(d) $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
(e) $Model(s, a) \leftarrow s', r$ (assuming deterministic environment)
(f) Repeat $N$ times:
   s $\leftarrow$ random previously observed state
   a $\leftarrow$ random action previously taken in $s$
   $s', r \leftarrow Model(s, a)$
   $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
Pac-Man RL Demo
How does Pac-Man “see” the world?
How does Pac-Man “see” the world?
The state-space may be continuous...
How does Pac-Man “see” the world?
Q-Function Approximation

\[ Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a') \]

\[ a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \]
Curriculum Development for RL Agents
Curriculum Development for RL Agents

Goal

Most difficult region
Main Approach
Main Approach

Rewind back \( k \) game steps and branch out
Figure 4: Results of MISTAKELEARNING applied to the Ms. Pac-Man domain. See Section 5.1.2 for details. Dashed lines indicate standard error.

Resources

• BURLAP: Java RL Library: http://burlap.cs.brown.edu/
THE END