ROB 537: Learning-Based Control

Week 4, Lecture 1
Reinforcement Learning

Announcements:
HW 2 due: 10/16
Project Background due: 10/23

Reading:
Kaelbling et al.
Stone et al.

So far

• Neural Networks
  – Learn an unknown function
    ▪ Regression
    ▪ Classification
  – Direct mapping of input to output
    ▪ Features to class (classification)
    ▪ Features to real value (regression)
    ▪ State to action (control)
Reinforcement Learning

- Don’t directly map state to action

- Learn expected rewards
  - For state
  - For state, action pairs

Reinforcement Learning Concept

- Learn from interactions with the environment
  - Take action
  - Receive feedback from the environment
  - Modify your behavior
  - Achieve some goal

- Examples:
  - Baby playing
    - Connection to the environment guides sensory input/output
  - Driving a car
    - Learn from interaction
Reinforcement Learning

- The Reinforcement Learning Concept
- N-armed Bandit Problem
- Value Functions
- Markov Decision Processes
- Monte Carlo Methods
- Sarsa Learning
- Q-Learning

Agent Environment Interaction in RL

\[ s_t \rightarrow a_t \rightarrow s_{t+1} \rightarrow r_t \rightarrow s_{t+1} \rightarrow a_t \rightarrow s_{t+2} \rightarrow r_{t+1} \rightarrow s_{t+2} \rightarrow \ldots \]
Policy: Map States to Actions

System dynamics

\[ s_t \times a_t \rightarrow s_{t+1}, r_t \]

Policy (\(\pi_t\)): map states to probabilities of taking those actions

\[ \pi_t(s,a) = P(a_t = a \mid s_t = s) \]

N-Armed Bandit

- Think multiple armed slot machine
  - N actions
  - Each slot has a reward (randomly chosen from a distribution)
  - Aim: maximize long term reward
  - Which arm to pull?
Value Function

- Reward is what you receive for taking an action

- Value:
  - Actual “worth” of that action
  - Long term reward

- N-armed bandit:
  - Action: which arm to pull
  - Reward: immediate gain on this time step
  - Value: long term payoff of a particular action

N-Armed Bandit

- Let’s say the actual value of an action is \( V(a) \)
- If action \( a \) has been sampled \( k_a \) times
  - The estimate at time step \( t \) is \( V_t(a) \):

\[
V_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}
\]

- By the law of large numbers

\[
\lim_{t \to \infty} V_t(a) \to V^*(a)
\]
Computing $V$

- For large values of $t$:
  - Need to keep all values in memory.
  - How about we do this incrementally?
  - $V_{k+1}$: average at $k+1$
  - $V_k$: average after $k$

$$V_{k+1} = \frac{r_1 + r_2 + r_3 + \cdots + r_{k+1}}{k+1}$$

$$= \frac{1}{k+1} \left( r_{k+1} + \sum_{i=1}^{k} r_i \right)$$

$$= \frac{1}{k+1} \left( r_{k+1} + kV_k \right)$$

$$= \frac{1}{k+1} \left( r_{k+1} + kV_k + V_k - V_k \right)$$

$$= \frac{1}{k+1} \left( r_{k+1} + (k+1)V_k - V_k \right)$$

$$= V_k + \frac{1}{k+1} \left( r_{k+1} - V_k \right)$$
Summary

- Result:

\[ V_{k+1} = V_k + \frac{1}{k+1}(r_{k+1} - V_k) \]

- Key Reinforcement Learning Concept:

\[ \text{NewEstimate} \leftarrow \text{OldEstimate} + \text{Stepsize} \cdot (\text{Target} - \text{OldEstimate}) \]

What if process is non-stationary?

- How about a fixed “stepsize”

\[ V_{k+1} = V_k + \alpha (r_{k+1} - V_k) \]

- Impact:
  - New observations count more
  - Slowly forget old info
  - Track non-stationary process
Example: N-Armed Bandit

- Given values at time $t$:

  | 0.1 | 0.4 | 0 | 0.05 | 0.1 | 0 | 0.05 | 0.1 | 0 | 0.2 |

  - Greedy selection:
    - Action 2 (highest value)

  - $\epsilon$-greedy selection:
    - Action 2 with probability $(1-\epsilon)$
    - Another action with probability $\epsilon$

  - Softmax:
    - Each action with probability based on its value

Example: N-Armed Bandit, Softmax

- Given values at time $t$:

  | 0.1 | 0.4 | 0 | 0.05 | 0.1 | 0 | 0.05 | 0.1 | 0 | 0.2 |

  - Softmax:
    - Select each action with probability $p_t(a)$ based on:
      - Temperature $t$
      - The Q-value of that action at that time step
      \[ p_t(a) = \frac{\exp\left(\frac{V_t(a)}{\tau}\right)}{\sum_b \exp\left(\frac{V_t(b)}{\tau}\right)} \]

      - If $\tau \rightarrow 0$, then greedy selection (highest value has probability 1)
      - If $\tau \rightarrow \infty$, then $p_t(a) = 1/n$ for all actions
Initial Values?

- How do we start?
- What are initial Q values?
  - Zero
  - Random
  - Optimistic
    - Encourage exploration of actions not yet taken

Interesting Questions

- What if reward is not immediate?
  - Reward/value for sequences of actions
  - How do I assign/blame to sequences of actions

- Do I choose best reward?
  - What about potential for better reward of unexplored action
  - Exploration/Exploitation
    - Explore: try a new action
    - Exploit: repeat (so far) best action
Rewards and Time Horizons

- Maximize reward
  - What does this really mean?
  - Maximize expected reward
    - Simplest case: sum of rewards:
      \[ R_s = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \]
    - Makes sense when there is a natural final step \( T \)
    - Episodic task:
      - Break learning into episodes of \( T \) steps
      - Example: Play a game

What if there is no final time \( T \)?

- Maximize discounted sum of rewards
  \[ R_i = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T \]
  \[ = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \]
  - A reward \( k \) step in the future is worth only \( \gamma^k \) times what it would be worth now
  - \( \gamma < 1 \) means the infinite sum has a finite value as long each reward is bounded
  - \( \gamma = 0 \) means agent is myopic: care only about immediate rewards
Markov Property

- Markov property:
  - A state that retains all the information relevant to future actions and rewards is said to be Markov, or to have the Markov property
  - Key: How you got to the state doesn’t matter
  - Example: Chess board: all relevant info is there
  - Important in Reinforcement Learning because decisions and values are based on state

Markov Property

- Markov property depends on representation of problem, not the problem

- Example: Pole balancing
  - State: location of pole tip
  - Not Markov
  - State: location and velocity of pole tip
  - Not Markov
  - State: location/velocity/acceleration of pole tip, location/velocity of cart
  - Markov (approximation)
Markov Decision Processes

- Markov Decision Process (MDP)

- Finite MDP (finite number of states) defined by:
  - Set of states \( S \)
  - Set of actions \( A \)
  - Transition probabilities
    \[
    P^a_{ss'}(s, s') = \Pr \{ s_{t+1} = s' \mid s_t = s, a_t = a \}
    \]
  - Rewards
    \[
    R^a_{ss'}(s, s') = \mathbb{E} \{ r_t \mid s_t = s, a_t = a, s_{t+1} = s' \}
    \]
  - Transition probabilities and Rewards fully specify an MDP

\[
(S, A, P^a(s, s'), R^a(s, s'))
\]