Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 4: Value-Based Methods

Summarized By: Sara Karimi

Monte-Carlo Prediction & Control

Monte-Carlo Prediction

Monte-Carlo (MC) methods allow learning without requiring a model of the environment. Unlike dynamic programming, which assumes knowledge of transition probabilities $P(s'|s,a)$, MC methods estimate value functions directly from sample episodes.

Key Concepts:

- **Model-Free Learning:** MC does not require knowledge of transition dynamics.
- **Learning from Episodes:** MC methods estimate value functions by averaging observed returns.
- **First-Visit vs. Every-Visit MC:**
  - **First-Visit MC:** Estimates $V^\pi(s)$ as the average return from the first time s is visited in each episode.
  - **Every-Visit MC:** Averages returns from all occurrences of s in an episode.

Monte-Carlo Policy Evaluation

To estimate the state-value function $V^\pi(s)$:

$$V^\pi(s) = \mathbb{E}[G_t|S_t = s]$$

where G_t is the return from time t, computed as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

Monte-Carlo updates $V^\pi(s)$ using:

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha(G_t - V^\pi(s))$$

where α is the learning rate.

Key Disadvantages of Monte-Carlo

- **Slow Convergence:** Requires many episodes to converge.
- **High Variance:** Returns depend on long trajectories.
- **Delayed Updates:** Can only update values at the end of an episode.

Monte-Carlo Control

Policy Improvement

MC methods can be extended to control problems, where the goal is to find the optimal policy. The policy improvement theorem ensures that improving the action-value function $Q^\pi(s,a)$ leads to a better policy.

Monte-Carlo Policy Iteration

- **Policy Evaluation:** Estimate $Q^\pi(s,a)$ using sample episodes.
- **Policy Improvement:** Update policy using:

$$\pi(s) = \arg\max_a Q(s,a)$$

  This greedy approach improves the policy iteratively.

Epsilon-Greedy Exploration

To ensure sufficient exploration, we use an ϵ-greedy policy:

- With probability $1 - \epsilon$, choose the action maximizing $Q(s, a)$.
- With probability ϵ, select a random action.

Monte-Carlo Control Algorithm

Repeat for each episode:

- Generate an episode following policy π.
- Compute returns G_t for each state-action pair.
- Update action-value estimates:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(G_t - Q(s, a))$$

- Improve policy using ϵ-greedy action selection.

Monte-Carlo vs. Temporal Difference Learning

Feature	Monte-Carlo (MC)	Temporal-Difference (TD)
Updates	At the end of an episode	After each step
Variance	High (full episode return)	Lower (bootstraps from estimates)
Bias	Unbiased	Can be biased (depends on initialization)
Environments	Episodic only	Works for both episodic & continuing tasks

Bootstrapping vs. Sampling

- **Bootstrapping:** Using an estimate to update another estimate (TD, DP).
- **Sampling:** Learning from actual experience (MC, TD).

Temporal-Difference Learning

TD(0) Update Rule

Instead of waiting for the full return, TD learning bootstraps from existing estimates:

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

where $R_{t+1} + \gamma V(S_{t+1})$ is the TD target.

SARSA (On-Policy TD Control)

SARSA is an on-policy TD control method that updates:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right)$$

It follows the policy being improved.

Q-Learning (Off-Policy TD Control)

Q-learning learns the optimal policy while following an exploratory policy:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right)$$

It learns from greedy actions, but follows a different policy.