Week 8 Precept Notes: MaxEnt RL and IRL

ECE 433/COS 435

April 6, 2024

1 Introduction

This week, we dive deeper into reinforcement learning, focusing on Maximum Entropy Reinforcement Learning (MaxEnt RL), exploration strategies, and Inverse Reinforcement Learning (IRL). We aim to understand how these advanced topics extend the foundational concepts of RL to improve learning efficiency, policy generalization, and applicability to real-world problems.

2 Maximum Entropy Reinforcement Learning (MaxEnt RL)

Motivation MaxEnt RL extends the standard RL objective by adding an entropy term to the reward function. This encourages exploration by rewarding policies that maintain a higher entropy, i.e., more randomness in the action selection process.

Algorithm 1 Soft Actor-Critic Algorithm

Require: Initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$

1: Initialize target Q-function parameters $\phi_{\text{targ},1} \leftarrow \phi_1, \phi_{\text{targ},2} \leftarrow \phi_2$
2: **for** each iteration **do**
3: Collect a set of transitions $\{(s_t, a_t, r_t, s_{t+1})\}$ using the current policy π_θ
4: Store transitions in the replay buffer $\mathcal{D}$
5: **for** each gradient step **do**
6: Sample a minibatch $\{(s, a, r, s')\}$ from $\mathcal{D}$
7: Update the Q-functions ϕ_i by minimizing the MSE loss:

$$\mathcal{L}(\phi_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_{\phi_i}(s,a) - Q_{\text{target}}(s,a)\right)^2\right]$$

8: Update the policy by minimizing the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_\theta}\left[\alpha \log(\pi_\theta(a|s)) - Q_\phi(s,a)\right]$$

9: Update target Q-function parameters:

$$\phi_{\text{targ},i} \leftarrow \tau\phi_i + (1-\tau)\phi_{\text{targ},i}$$

The objective of SAC is to maximize the expected sum of rewards along with the policy's entropy. This ensures a balance between exploration (high entropy) and exploitation (maximizing expected reward). The

objective function is given by:

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right)\right], \tag{1}$$

where α is a temperature parameter that determines the relative importance of the entropy term against the reward, encouraging the policy to explore more or less.

Architecture SAC incorporates two Q-functions (to mitigate positive bias in the policy improvement step resulting from the max operator in the standard Q-learning algorithm) and uses a target value network to stabilize training. The temperature parameter α is either fixed or adjusted.

Policy Evaluation In SAC, policy evaluation involves learning a Q-function that estimates the expected return of taking an action in a given state and following the current policy thereafter. The Q-function is updated using the Bellman backup operation:

$$Q_{\text{target}}(s_t,a_t) = r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\mathcal{P},a_{t+1}\sim\pi}\left[Q(s_{t+1},a_{t+1}) - \alpha\log\pi(a_{t+1}|s_{t+1})\right], \tag{2}$$

where $\mathcal{P}$ is the state transition probability.

Policy Improvement The policy is improved by minimizing the KL-divergence between the policy's action distribution and a softmax distribution derived from the Q-function, effectively performing a policy iteration step. The updated policy aims to maximize the expected return and the entropy of the policy.

3 Exploration in RL

Motivation Effective exploration strategies are crucial for RL agents to discover and learn optimal policies. We discuss the challenges of exploration and various strategies to address them.

This section covers several prominent exploration techniques.

3.1 ε-greedy Exploration

The ε-greedy strategy is one of the simplest exploration techniques, where with probability ε an agent takes a random action, and with probability $1-\varepsilon$, it chooses the action with the highest estimated value. This method ensures that the agent explores the action space indefinitely but with diminishing frequency over time, as ε is often decreased.

Application: Widely used in Q-learning and other value-based methods due to its simplicity and effectiveness in a wide range of environments.

3.2 Upper Confidence Bounds (UCB)

Upper Confidence Bounds (UCB) is an exploration technique that selects actions based on the principle of optimism in the face of uncertainty. The UCB for each action is calculated considering both the estimated value of that action and the uncertainty or variance associated with that estimate. Actions with higher UCB are preferred, promoting exploration of actions with less certainty.

Application: Particularly effective in multi-armed bandit problems and has been adapted for use in Monte Carlo Tree Search (MCTS) and other RL scenarios.

3.3 Thompson Sampling

Thompson Sampling is a probabilistic method that models the problem of action selection as a Bayesian inference problem. It samples from the posterior distributions of the action-value estimates to decide on the next action, inherently balancing exploration and exploitation based on the uncertainty of action outcomes.

Application: It has shown success in both multi-armed bandit problems and as a principle for action selection in more complex RL tasks.

4 Inverse Reinforcement Learning (IRL)

4.1 Introduction to IRL

Inverse Reinforcement Learning (IRL) infers the reward function an expert follows based on their observed behavior. This is crucial in scenarios like autonomous driving, where defining a comprehensive reward function is challenging, but examples of desired behavior (e.g. human driving) are plentiful. By working backward from these behaviors, IRL helps derive implicit objectives in complex tasks, offering a practical approach when explicit reward specification is elusive.

4.2 IRL Formulation

The goal of IRL is to find a reward function $r(s, a)$ under which the observed behavior appears optimal. Formally, the IRL problem can be expressed as follows:

$$\max_r \left\{ \mathbb{E}_{\tau \sim \pi_{\text{expert}}} \left[\sum_{t=0}^{T} r(s_t, a_t) \right] - \log \mathbb{E}_{\tau \sim \pi} \left[e^{\sum_{t=0}^{T} r(s_t, a_t)} \right] \right\}, \tag{3}$$

where τ represents a trajectory of states and actions, π_{expert} is the policy under which the expert's behavior is generated, and π is a policy derived from the learned reward function. This formulation seeks a reward function that makes the expert's policy uniquely optimal, highlighting the inherent ambiguity in IRL—multiple reward functions can often explain the same observed behavior.

4.3 Maximum Entropy IRL

Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) is a prominent approach that addresses the ambiguity inherent in IRL by incorporating a principle of maximum entropy. This principle asserts that among all reward functions that could explain the observed expert behavior, preference is given to the one that results in the most probabilistically diverse set of optimal policies, effectively making the least prior assumptions about unobserved preferences.

Methodology The core idea behind MaxEnt IRL is to treat the problem as a probabilistic inference task, where the probability of observing a trajectory under a given reward function is proportional to the exponential sum of rewards encountered along the trajectory. Formally, the probability of a trajectory τ is given by:

$$P(\tau|r) \propto \exp\left(\sum_{(s_t,a_t)\in\tau} r(s_t, a_t)\right). \tag{4}$$

This formulation naturally leads to a preference for reward functions under which the observed expert trajectories have higher likelihood, while also promoting entropy or diversity among plausible trajectories.

Algorithm The MaxEnt IRL algorithm involves iteratively adjusting the estimated reward function to increase the likelihood of the observed expert trajectories.

Algorithm 2 Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL)

1: Initialize the reward function $R(s, a)$ with zeros or random values
2: **repeat**
3: Solve the RL problem using the current $R(s, a)$ to obtain policy π that maximizes the expected sum of rewards plus entropy
4: Compute the expected feature counts under π, $E_\pi[\phi(s, a)]$
5: Compute observed feature counts from expert trajectories, $E_{\text{expert}}[\phi(s, a)]$
6: Adjust $R(s, a)$ to minimize the difference between $E_\pi[\phi(s, a)]$ and $E_{\text{expert}}[\phi(s, a)]$ using gradient-based optimization
7: **until** convergence

Why is it good? MaxEnt IRL has several advantages, making it a popular choice for IRL applications:

- It provides a principled way to handle the inherent ambiguity in IRL by preferring solutions that maximize entropy, thus avoiding overfitting to the observed data.
- The probabilistic framework allows for straightforward incorporation of prior knowledge and uncertainty into the IRL process.
- MaxEnt IRL has been shown to be effective in complex environments, including those with continuous states and actions, through the use of function approximation techniques.