CSE6243: Advanced Machine Learning **Fall 2023**

Lecture 12: Learning with MDPs

Lecturer: Bo Dai *Scribes: Zihao Xiao; Sharma, Dhruv*

Note: *LaTeX template courtesy of UC Berkeley EECS Department.*

12.1 Recap

12.1.1 Bellman Equation

The Bellman equations can be expressed in the following linear algebra form:

$$V^\pi = R^\pi + \gamma P^\pi V^\pi$$

$$Q^\pi = R + \gamma P Q^\pi$$

Bellman Optimal Equation:

$$V^*(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V^*(s') \right] \right\}$$

The Bellman Optimal Equation for Q^* is given by:

$$Q^*(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in A} Q^*(s',a') \right]$$

12.1.2 Policy Evaluation via Solving Linear Equations in the Planning Setting

We can get the matrix form of the Bellman Equation for V^π:

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = \left(I + \gamma P^\pi + (\gamma P^\pi)^2 + \dots \right) R^\pi.$$

12.1.3 Policy Optimization

The analytical prediction algorithm is not feasible for large state spaces since it runs exponentially.

12.2 New Content

12.2.1 Policy Iteration

Policy iteration is an approach to finding the optimal policy by repeatedly improving an initial policy. The process alternates between policy evaluation (calculating the value function $V(s)$ for a given policy) and

policy improvement (updating the policy based on the current value function).These steps are repeated until the policy converges to the optimal policy, meaning further improvements do not yield a better outcome.

Algotithmn Steps-Policy Iteration

1. Initialize the policy π_0.
2. For $t = 0, 1, 2, \ldots$
   - **Policy Evaluation:** Solve $V^{\pi_t} = R^{\pi_t} + \gamma P^{\pi} V^{\pi_t}$
   - **Policy Improvement:** Update the policy by setting

$$\pi_{t+1} \leftarrow \arg\max_{\pi} \{R^{\pi} + \gamma P^{\pi} V^{\pi_t}\}$$

12.2.1.1 Theorem 12.1

Let u_0 initialize V_I, with $V^{\pi_0} = V_0$ as the initial policy π_0. If $u_0 = V_0$, then for all n, $u_n \leq V_n$.

12.2.1.2 Lemma 12.1

This statement is commonly used in policy iteration to show that the sequence of value functions $\{V_n\}$, generated by evaluating an improving sequence of policies $\{\pi_n\}$, is monotonic and converges to the optimal value function V^*.

$$V_n \leq V_{n+1} \leq V^*$$

Proof:

$$R_{n+1} = R^{\pi_{n+1}} = \sum_a R(s,a)\pi_{n+1}(a|s)$$

$$P_{n+1} = \sum_a P(s'|s,a)\pi_{n+1}(a|s)$$

$$R_{n+1} + \gamma P_{n+1} V_n \geq R_n + \gamma P_n V_n$$

$$R_n + \gamma P_n V_n = V_n$$

$$R_{n+1} \geq (I - \gamma P_{n+1}) V_n$$

$$V_{n+1} \geq (I - \gamma P_{n+1})^{-1} R_{n+1} = I + \gamma P_{n+1} + (\gamma P_{n+1})^2 + \ldots \geq V_n$$

Since each $V_n \leq V_{n+1}$ and the optimal value function V^* is the fixed point of the Bellam optimality operator, we conclude that:

$$V_n \leq V_{n+1} \leq V^*$$

This completes the proof.

12.2.1.3 Lemma 12.2

If ϕ is monotonic, assuming $u \leq v$, we have:

$$\phi(u) \leq \phi(v)$$

Here, $\phi(u)$ is defined as the maximum expected reward and discounted future value under a policy π. $R^{\pi u}$ and $P^{\pi u}$ represent the reward and transition probabilities for an optimal policy π_u associated with u.

Proof:

$$\phi(u) = \max_{\pi} (R^{\pi} + \gamma P^{\pi} u) = R^{\pi_u} + \gamma P^{\pi_u} u$$

$$R^{\pi_u} + \gamma P^{\pi_u} u \leq R^{\pi_u} + \gamma P^{\pi_u} v$$

$$R^{\pi_u} + \gamma P^{\pi_u} v \leq \max_{\pi} (R^{\pi} + \gamma P^{\pi} v) = \phi(v)$$

This completes the proof.

Proof of $\phi(V_n) \leq V_{n+1}$:

To prove that $\phi(V_n) \leq V_{n+1}$, we need to leverage the properties of the Bellman operator ϕ and the definition of V_{n+1} in the context of policy iteration.

Define V_{n+1} :

$$V_{n+1} = \max_{\pi} (R^{\pi} + \gamma P^{\pi} V_n) = R^{\pi_{n+1}} + \gamma P^{\pi_{n+1}} V_{n+1}$$

Apply the Bellman Operator ϕ to V_n:

$$\phi(V_n) = \max_{\pi} (R^{\pi} + \gamma P^{\pi} V_n) = R^{\pi_{n+1}} + \gamma P^{\pi_{n+1}} V_n$$

3. According to Lemma 12.1 $V_n \leq V_{n+1}$:

Therefore, it follows that:

$$\phi(V_n) \leq V_{n+1}$$

This completes the proof.

12.2.2 Value Iteration

In value iteration, we calculate the optimal state-value function by repeatedly updating the estimate $V(s)$. Each new value of $V(s)$ is derived using the Bellman equations based on the current estimates. This process continues until a convergence criterion is satisfied.

Algorithm Steps- Value Iteration with Exploration Bonus

1. **Initialization**:
   - Initialize the value function $V_0(s)$ arbitrarily (e.g., $V_0(s) = 0$ for all s).
   - Initialize counts $n(s, a) = 0$ for all state-action pairs (s, a).
2. **For** $t = 1$ **to** T **do**:
   (a) **Policy Execution and Data Collection**:

- Derive the policy π_t from the current value estimates:

$$\pi_t(s) = \arg\max_{a \in A} \left(\hat{R}(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a) V_t(s') + \hat{b}(s,a) \right)$$

- Execute policy π_t and interact with the environment.
- Collect new data (state transitions, rewards) and update counts:

$$n(s,a) \leftarrow n(s,a) + \text{number of times action } a \text{ is taken in state } s \text{ during iteration } t$$

(b) **Estimate Model Parameters**:

- Estimated reward function $\hat{R}(s,a)$:

$$\hat{R}(s,a) = \frac{\text{Total reward received when taking action } a \text{ in state } s}{n(s,a)}$$

- Estimated transition probabilities $\hat{P}(s'|s,a)$:

$$\hat{P}(s'|s,a) = \frac{\text{Number of times transitioned to } s' \text{ from } s \text{ using } a}{n(s,a)}$$

(c) **Compute Exploration Bonus**:

- Define the exploration bonus term $\hat{b}(s,a)$:

$$\hat{b}(s,a) = \sqrt{\frac{c}{n(s,a)}}$$

where $c > 0$ is a constant (tuning parameter).

(d) **Value Function Update**:

- Update the value function for all states $s \in S$:

$$V_{t+1}(s) = \max_{a \in A} \left(\hat{R}(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a) V_t(s') + \hat{b}(s,a) \right)$$

3. **Convergence Check**:

- If $\|V_{t+1} - V_t\|_\infty < \epsilon$, where $\epsilon > 0$ is a small threshold, then stop; else, continue to the next iteration.

Output:

After the value iteration process converges, we obtain an approximation of the optimal value function V^* given by V_T, where T is the final iteration. The corresponding optimal policy π^* can be derived from V_T as follows:

$$\pi^*(s) = \arg\max_{a \in A} \left(\hat{R}(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a) V_T(s') \right)$$

Optimal Value Updates:

For the optimal value updates during the iteration process, we use the following equations:

1. Optimal State-Value Update:

The state-value function is updated using the Bellman optimality equation:

$$V_{t+1}^*(s) = \max_{a \in A} \left(\hat{R}(s,a) + \gamma \sum_{s'} \hat{P}(s'|s,a) V_t^*(s') \right)$$

Alternatively, using the expectation over next states:

$$V_{t+1}^*(s) = \max_{a \in A} \left(R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V_t^*(s')] \right)$$

To incorporate a learning rate α (where $0 < \alpha \leq 1$), we can use a weighted update:

$$V_{t+1}^*(s) \leftarrow \alpha V_t^*(s) + (1-\alpha) \left(\max_{a \in A} [R(s,a) + \gamma V_t^*(s')] \right)$$

2. Optimal Action-Value (Q-Value) Updates:

The action-value function $Q^*(s,a)$ can be updated using:

$$Q_{t+1}^*(s,a) \leftarrow \alpha Q_t^*(s,a) + (1-\alpha) \left(R(s,a) + \gamma \max_{a' \in A} Q_t^*(s',a') \right)$$

Alternatively, defining an intermediate value $\Delta Q_t^*(s,a)$:

$$\Delta Q_t^*(s,a) = R(s,a) + \gamma \max_{a' \in A} Q_t^*(s',a')$$

Then the update becomes:

$$Q_{t+1}^*(s,a) \leftarrow \alpha Q_t^*(s,a) + (1-\alpha) \Delta Q_t^*(s,a)$$

3.Q-Value Update with Probability Transition Matrix:

Incorporating the transition probability matrix P_Q, the Q-value update can be expressed as:

$$Q_{t+1}^*(s,a) \leftarrow (1-\alpha) Q_t^*(s,a) + \alpha \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a' \in A} Q_t^*(s',a') \right)$$