Lecture 10: Q-Learning

1 Introduction

Admin:

- Midterm next week. Will be in Friend 101. If you haven't received an email about accomodations, please email me or ODS. We've released last year's midterm and solution on the course website.
- HW4 has been released. The final question is about proposing quiz questions. These will be available to use for studying.
- Review Session next Monday, Mar 3: 6pm – 8pm in CS 105.
- What are you confused about – answer on sheet.

Review: Learning value functions These methods learned value functions directly from data, without requiring a model (hence these are *model-free* methods). SARSA and expected SARSA are more sample efficient than MC estimation. Expected SARSA allows us to do off-policy evaluation.

- Bellman equations.

$$Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)\pi(a'|s')}[Q^\pi(s',a')] \qquad \text{(Bellman optimality eq.)}$$
$$Q^\star(s,a) = r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)}[\max_{a'} Q^\star(s',a')]. \qquad \text{(Bellman expectation eq.)}$$

- MC Estimation.
- SARSA.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \tag{1}$$
$$\alpha\left(r(s,a) + \gamma Q(s',a')\right). \tag{2}$$

- Expected SARSA.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r(s,a) + \gamma Q(s',a')\right) \tag{3}$$
$$\text{where} \quad y = r(s,a) + \gamma Q(s', a' \sim \pi(a' \mid s')). \tag{4}$$

- MC Regression

$$\min_Q \frac{1}{|\mathcal{D}|} \sum_{(s,a,R)} (Q(s,a) - R)^2 \tag{5}$$
$$\tag{6}$$

Learning objectives for today At the end of this lecture, you will be able to

1. learn $Q^\star$ from data, without access to a simulator or the expert policy.
2. extend this approach to continuous states.

1.1 Assumptions for today.

1. **Tabular actions.** We'll start by assuming states are also tabular, but will lift this by the end of the lecture.
2. No model. This lifts the limitation of value iteration

2 Q-learning [1]

Inputs: $\{(s, a, r, s')\}$

Output: $Q^\star(s, a)$, the value function for the optimal policy at each state-action pair.

Q-learning is an algorithm that iteratively updates a table of Q-values given data from the MDP. Q-learning will produce $Q^\star$, even if your data is collected from a policy that is not optimal.

Recall Bellman optimality equation:

$$Q(s, a) = r(s, a) + \gamma \mathbb{E}_{p(s'|s,a)}[\max_{a'} Q(s', a')] \qquad \text{(Bellman optimality eq.)} \tag{7}$$

A first attempt:

$$Q(s, a) \leftarrow r(s, a) + \gamma \mathbb{E}_{p(s'|s,a)}[\max_{a'} Q(s', a')]. \tag{8}$$

Q: What could go wrong? A: In stochastic settings, the updates would oscillate. Instead, we need to average over the possible next states. This is the same problem as in SARSA.

Our actual update for Q-learning:

$$Q(s, a) \leftarrow \alpha \left(r(s, a) + \gamma \max_{a'} Q(s', a') \right) + (1 - \alpha) Q(s, a) \tag{9}$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha \underbrace{\left(r(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)}_{\text{Bellman/TD error}}. \tag{10}$$

Q: How would you modify Q-learning to obtain $V^\star$ instead of $Q^\star$?

Gradient descent interpretation. Define the TD error or Bellman error as

$$\mathcal{L} = \frac{1}{2} \left(Q(s, a) - (r(s, a) + \gamma \max_{a'} Q(s', a')) \right)^2. \tag{11}$$

2.1 Complete Algorithm

Algorithm 1 Q-learning

```
Initialize Q(s, a) = 0 for all states and actions.
while not converged do
    s ← ENV.RESET()
    for t = 1, ··· , T do
        a = arg max_a Q(s, a)                          ▷ or ε-greedy
        Take action a, observe r, s'.
        Q(s, a) ← Q(s, a) + α[r + γ max_a' Q(s', a') − Q(s, a)]
        s ← s'
Return Q(s, a).
```

Convergence? It's hard to directly prove that Q-learning converges because convergence depends on the data you collect. If your exploration never collects all the transitions in an optimal trajectory, then Q-learning won't learn the correct values.

3 Preview: Deep Q-Learning

To extend Q-learning to work in settings with continuous states, we'll represent the Q-values as a learned function $Q_\theta(s, a)$. This function will be trained using the TD error as the loss:

$$\min_\theta \frac{1}{2}\left(Q_\theta(s,a) - (r(s,a) + \gamma \max_{a'} \hat{Q}_\theta(s', a'))\right)^2. \tag{12}$$

One important thing to note is that we only update the Q function at the current time step (like in the tabular update for Q-learning), and we do not update the Q function at the next time step. Next time, we'll see some more tricks for making Q-learning with function approximation work.

Another way of writing this is to use θ_i to denote the weights at iteration i:

$$\theta_{i+1} \leftarrow \arg\min_{\theta_{i+1}} \frac{1}{2}\left(Q_{\theta_{i+1}}(s,a) - (r(s,a) + \gamma \max_{a'} Q_{\theta_i}(s', a'))\right)^2. \tag{13}$$

4 Topics to review

References

[1] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8:279–292.