CS540 Introduction to Artificial Intelligence

Lecture 15

Young Wu

Based on lecture slides by Jerry Zhu and Yingyu Liang

July 21, 2023

Reinforcement Learning

Motivation

- Reinforcement learning is about learning from the outcome of actions.

1. Sense world.
2. Reason.
3. Choose an action to perform.
4. Get feedback.
5. Learn.

Applications

Motivation

- Actions can be performed in the physical world or artificial ones.
- Board games.
- Robotic control.
- Autonomous helicopter performance.
- Economics models.

Q Learning

Description

- Select an action.
- Receive reward.
- Observe new state.
- Update (learn) the value of the state-action pair.

State and Actions

Definition

- The set of possible states is $s_t \in S$.
- The set of possible actions is $a_t \in A$.
- The set of possible rewards is $r_t \in R$.
- At each time t:

1. Observe state s_t.
2. Chooses action a_t.
3. Receives reward r_t.
4. Changes to state s_{t+1}.

Markov Decision Process

Definition

- Markov property on states and actions is assumed.

$$\mathbb{P}\left\{s_{t+1} \middle| s_t, a_t, s_{t-1}, a_{t-1}, \ldots\right\} = \mathbb{P}\left\{s_{t+1} \middle| s_t, a_t\right\}$$
$$\mathbb{P}\left\{r_{t+1} \middle| s_t, a_t, s_{t-1}, a_{t-1}, \ldots\right\} = \mathbb{P}\left\{r_{t+1} \middle| s_t, a_t\right\}$$

- The goal is to learn a policy function $\pi : S \to A$ for choosing actions that maximize the total expected discounted reward.

$$\mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots\right], \gamma \in [0, 1]$$

Expected Reward

Definition

- The expected reward at a given time t is the average reward weighted by probabilities.

$$\mathbb{E}\left[r_t\right] = \sum_{r_t \in R} r_t \mathbb{P}\left\{r_t \middle| s_{t-1}, a_{t-1}\right\}$$

Discounted Reward

Definition

- The discounted reward at time 0 is the sum of reward weighted given the time preference, usually described by a constant discount factor.

$$\text{PV}\ (r_t) = \gamma^t r_t, \gamma \in [0, 1]$$

$$\text{PV}\ (r_1, r_2, \ldots) = \sum_{t=0}^{\infty} \gamma^t r_t$$

- γ is the value of 1 unit of reward at time 1 perceived at time 0. If $\gamma = 1$, the sum over an infinite time period is usually infinity, therefore $\gamma < 1$ is usually used.

Value Function

Definition

- The value function is the expected discounted reward given a policy function π, assuming the action sequence is chosen according to π stating with state s.

$$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^{t} \mathbb{E}[r_t]$$

- The optimal policy $\pi^{\star}$ is the one that maximizes the value function.

$$\pi^{\star} = \operatorname*{argmax}_{\pi} V^{\pi}(s) \text{ for all } s \in S$$

$$V^{\star}(s) = V^{\pi^{\star}}(s)$$

Optimal Policy Given Value Function

Definition

- Given $V^\star(s), r(s,a), \mathbb{P}(s'|s,a), \pi^\star$ can be computed directly.

$$\pi^\star(s) = \operatorname*{argmax}_{a \in A} \left(\mathbb{E}[r|s,a] + \gamma \mathbb{E}\left[V^\star(s')\,|s,a\right]\right)$$
$$= \operatorname*{argmax}_{a \in A} \left(\sum_{r \in R} r\mathbb{P}\{r|s,a\} + \gamma \sum_{s' \in S} \mathbb{P}\{s'|s,a\} V^\star(s')\right)$$

- Define the function inside the argmax as the Q function.

Q Function

Definition

$$V^\star(s) = \mathbb{E}\left[r|s,\pi^\star(s)\right] + \gamma\mathbb{E}\left[V^\star\left(s'\right)|s,\pi^\star(s)\right]$$
$$Q(s,a) = \mathbb{E}\left[r|s,a\right] + \gamma\mathbb{E}\left[V^\star\left(s'\right)|s,a\right]$$

- If the agent knows Q, then the optimal action can be learned without $\mathbb{P}\left\{s'|s,a\right\}$.

$$\pi^\star(s) = \operatorname*{argmax}_a Q(s,a),\ V^\star(s) = \max_a Q(s,a)$$

Deterministic Q Learning

Definition

- In the deterministic case, $\mathbb{P}\left\{s' \middle| s, a\right\}$ is either 0 or 1, the update formula for the Q function is the following.

$$\hat{Q}\left(s, a\right) = r + \gamma \max_{a'} \hat{Q}\left(s', a'\right)$$

Non-Deterministic Q Learning

Definition

- In the nondeterministic case, the update formula for the Q function is the following.

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

- The learning rate α is sometimes set to $\dfrac{1}{1+\ \text{visits}\ (s,a)}$.
- Q learning will converge to the correct Q function in both deterministic and non-deterministic cases. In practice, it takes a very large number of iterations.

Q Learning, Part I

Algorithm

- Input: an MDP with states S, actions A, reward distribution R, transition probabilities P.
- Output: $\hat{Q}$ approximate Q function of the optimal policy.
- Initialize the Q table.

$$\hat{Q}(s, a) = 0, \text{ for each } s \in S, a \in A$$

Q Learning, Part II

Algorithm

- Observe current state s.
- Select an action a and execute it.
- Receive immediate reward r.
- Observe the new state s'.
- Update the table entry.

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

- Update the state and repeat forever.

$$s = s'$$

SARSA, On Policy Learning

Definition

- Q Learning uses the optimal action in state s', which is not necessarily the action a_{t+1} specified by the current (original) policy.
- Q Learning is an off-policy learning algorithm.
- To make the Q values learned consistent with the current policy, a_{t+1} can be in place of the $a^{\star}$ that maximizes $\hat{Q}\left(s', a'\right)$, this algorithm is called SARSA, which stands for $\left(S_t, A_t, R_t, S_{t+1}, A_{t+1}\right)$.
- SARSA is an on-policy learning algorithm.

SARSA, Part I

Algorithm

- Input: an MDP with states S, actions A, reward distribution R, transition probabilities P.
- Output: $\hat{Q}$ approximate Q function of a policy π.
- Initialize the Q table.

$$\hat{Q}(s, a) = 0, \text{ for each } s \in S, a \in A$$

SARSA, Part II

Algorithm

- Observe current state s.
- Select an action a and execute it.
- Receive immediate reward r.
- Observe the new state s'.
- Select an action a' in the next period.
- Update the table entry.

$$\hat{Q}(s,a) = (1-\alpha)\,\hat{Q}(s,a) + \alpha\left(r + \gamma \hat{Q}(s',a')\right)$$

- Update the state and repeat forever.

$$s = s'$$

Exploration vs Exploitation

Discussion

- There is a trade-off between learning about possibly better alternatives and following the current policy. Sometimes, random actions should be selected.

$$\mathbb{P}\left\{a \middle| s\right\} = \frac{c^{\hat{Q}(s,a)}}{\sum_{a' \in A} c^{\hat{Q}(s,a')}}$$

- $c > 0$ is a constant that determines how strongly selection favors actions with higher Q values.

Q Table vs Q Net

Discussion

- In practice, Q table is too large to store since the number of possible states is very large.
- If there are m binary features that represent the state, the Q table contains $2^m |A|$.
- However, it can be stored in a neural network called Q net.
- If there is a single hidden layer with m units, there are only $m^2 + m|A|$ weights to store.

Q Net Training

Discussion

- Observe the features x given a state s.
- Apply action a and observe new state s' with features x' and reward r.
- Train the network with new instance (x, y)

$$y = (1 - \alpha)\,\hat{y}(x, a) + \alpha\left(r + \gamma \max_{a'} \hat{y}(x', a')\right)$$

- $\hat{y}(x, a)$ is the activation of output unit a given the input x in the current neural network.
- $\hat{y}(x', a')$ is the activation output unit a' given the input x' in the current neural network.

Multi-Agent Reinforcement Learning

Discussion

- Value function and policy function iteration methods can be applied to solve dynamic games with multiple agents.
- It will be discussed in the game theory lectures.