Reinforcement Learning algorithms

Alexander Shleyfman

December 30, 2015
Model

MDP

- MDP type:
  - Goal-directed, Indefinite Horizon, Cost Minimization;
  - Infinite Horizon, Discounted, Reward Maximization;
  - ...
- $S$ – state space,
- $A$ – action space,
- $R(s, a, s')$ – reward model.

Auxiliary notations

- $\alpha, \beta$ – learning rates.
- $\gamma$ – discount factor.
- ...
MDP

- MDP type:
  - Goal-directed, Indefinite Horizon, Cost Minimization;
  - Infinite Horizon, Discounted, Reward Maximization;
  - ...

- $S$ – state space,
- $A$ – action space,
- $R(s, a, s')$ – reward model.

Auxiliary notations

- $\alpha, \beta$ – learning rates.
- $\gamma$ – discount factor.
- ...

Definition I

A randomized policy $\pi$ can be described as a mapping of states to probability distributions over actions,

$$\pi : S \times A \rightarrow [0, 1],$$

where $\pi(s, a)$ defines the probability that the agent will execute action $a$ in the state $s$.

Definition II

- An off-policy learner learns the value of the optimal policy independently of the agent’s actions.
- An on-policy learner learns the value of the policy being carried out by the agent, including the exploration steps.
Boltzmann Exploration

\[ \pi(s, a) = \frac{e^{Q(s, a)/T}}{\sum_{a' \in A} e^{Q(s, a')/T}} \]

- \( T \) is the “temperature”. Large \( T \) means that each action has about the same probability. Small \( T \) leads to more “greedy” behavior.
- Typically: start with large \( T \) and decrease it with time (\( T \to 0 \)).
**Q-Learning**

Initialize $Q(s, a)$

**for each** episode **do:**

  Initialize $s$

  **while** $s$ is not terminal **do:**

    Choose $a$ from $s$ using policy $\pi$ derived from $Q$

    Take action $a$, observe $r, s'$

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

    $s \leftarrow s'$
Q-Learning

Characteristics

- It is off-policy (Q-values approximate \(Q^*-\)values, regardless of exploration).
- It learns state-action values (Q-values).

Advantages

- Can reach optimal solutions, even under continued exploration.
- Being the oldest of the considered algorithms, it is well-researched and successfully applied.

Disadvantages

- Sometimes diverge when function approximators are used.
- Cannot handle continuous action spaces.
- Has no “natural” extension to eligibility traces (Watkins, Peng, etc.).
Algorithm

Initialize $Q(s, a)$

**for each** episode **do**:
  - Initialize $s$
  - Choose $a$ from $s$ using policy $\pi$ derived from $Q$

**while** $s$ is not terminal **do**:
  - Take action $a$, observe $r, s'$
  - Choose $a'$ from $s'$ using policy $\pi$ derived from $Q$
  - $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha (r + \gamma Q(s', a'))$
  - $s \leftarrow s'$
  - $a \leftarrow a'$
SARSA

Characteristics

- It is on-policy ($Q$-values approximate the value including the effects of exploration).
- It learns state-action values ($Q$-values).

Advantages

- Can reach optimal solutions, when exploration decreases properly.
- Similar to Q-Learning, it is well-researched and successfully applied.
- SARSA has a more natural extension to eligibility traces than Q-Learning.

Disadvantages

- Cannot handle continuous action spaces.
Expected SARSA

Algorithm

Initialize \( Q(s, a) \)

for each episode do:

Initialize \( s \)

while \( s \) is not terminal do:

Choose \( a \) from \( s \) using policy \( \pi \) derived from \( Q \)

Take action \( a \), observe \( r, s' \)

Derived \( \pi \) for \( s' \) according to \( Q \)

\[
Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left( r + \gamma \sum_{a' \in A} \pi(s', a')Q(s', a') \right)
\]

\( s \leftarrow s' \)
Advantages over SARSA

▶ Does not have to know the next action to update its value.
▶ Has lower variance with the same bias.

Note:

▶ The algorithm is simpler than SARSA and more similar in structure to Q-Learning.
▶ Still two policy calculation has to be made.
Algorithm

Initialize $Q(s, a)$
Initialize $V(s)$

for each episode do:
  Initialize $s$
  while $s$ is not terminal do:
    Choose $a$ from $s$ using policy $\pi$ derived from $Q$
    Take action $a$, observe $r, s'$
    $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha (r + \gamma V(s'))$
    $V(s) \leftarrow (1 - \beta)V(s) + \beta (r + \gamma V(s'))$
    $s \leftarrow s'$
**QV-Learning**

**Characteristics**
- It is on-policy.
- It learns state-action values.
- It learns state values.

**Advantages**
- Using state values decreases the variance compared to Q-Learning and SARSA.
- Using state values often speeds up learning.
- State values are easily extendable to eligibility traces.

**Disadvantages**
- Cannot handle continuous action spaces.
Algorithm

Initialize \( P(s, a) \)
Initialize \( V(s) \)
for each episode do:
  Initialize \( s \)
  while \( s \) is not terminal do:
    Choose \( a \) from \( s \) using policy \( \pi \) derived from \( P \)
    Take action \( a \), observe \( r, s' \)
    \( P(s, a) \leftarrow P(s, a) + \alpha (r + \gamma V(s') - V(s)) \)
    \( V(s) \leftarrow (1 - \beta)V(s) + \beta (r + \gamma V(s')) \)
    \( s \leftarrow s' \)
Actor-Critic Learning

Characteristics
- It is on-policy.
- Learns preference values that do not hold explicit information on the expected discounted rewards.

Advantages
- Using state values often speeds up learning.
- State values are easily extendable to eligibility traces.

Disadvantages
- Cannot handle continuous action spaces.
Intuition

An eligibility trace is a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action (backward view). The trace marks the memory parameters associated with the event as eligible for undergoing learning changes.

Algorithms

Algorithms with eligibility trace are denoted by $\lambda$:

- $\text{TD}(\lambda)$,
- $\text{SARSA}(\lambda)$,
- $\text{Q}(\lambda)$-Learning,
- etc.
Eligibility trace equation

New variable $e(s) \in \mathbb{R}$ called eligibility trace:

- On each step, decay all traces by $\gamma \lambda$ and increment the trace for the current state by 1.
- Accumulating trace for each $s \in S$:

$$e(s) = \begin{cases} 
\gamma \lambda e(s) + 1, & \text{if } s \text{ is the current state} \\
\gamma \lambda e(s), & \text{else}
\end{cases}$$