CS 474/574 Machine Learning
9. Reinforcement Learning

Prof. Dr. Forrest Sheng Bao
Dept. of Computer Science
Iowa State University
Ames, IA, USA

April 30, 2021

Images that were not created by the instructor but from the web are cited in the Markdown source code in the syntax:

```
![a web image](URL)
```

Reinforcement learning (RL)

- Three kinds of learning: Supervised, unsupervised, and reinfocement.

Reinforcement learning (RL)

- Three kinds of learning: Supervised, unsupervised, and reinfocement.
- Like supervised learning, reinforcement learning is also about fitting a function to match the expected outcome.

Reinforcement learning (RL)

- Three kinds of learning: Supervised, unsupervised, and reinfocement.
- Like supervised learning, reinforcement learning is also about fitting a function to match the expected outcome.
- Unlike supervised learning, the expected outcome is not given by human but through letting the AI agent gain rewards via interaction with the environment.

Reinforcement learning (RL)

- Three kinds of learning: Supervised, unsupervised, and reinfocement.
- Like supervised learning, reinforcement learning is also about fitting a function to match the expected outcome.
- Unlike supervised learning, the expected outcome is not given by human but through letting the AI agent gain rewards via interaction with the environment.
- RL is just like how we learn: try-and-error.

Reinforcement learning (RL)

- Three kinds of learning: Supervised, unsupervised, and reinfocement.
- Like supervised learning, reinforcement learning is also about fitting a function to match the expected outcome.
- Unlike supervised learning, the expected outcome is not given by human but through letting the AI agent gain rewards via interaction with the environment.
- RL is just like how we learn: try-and-error.
- What can RL do?

MDP (Markov Decision Process): the framework of RL

- The agent interacts with the environment at timesteps: it takes one action to transit from the current state to the next.

MDP (Markov Decision Process): the framework of RL

- The agent interacts with the environment at timesteps: it takes one action to transit from the current state to the next.
- Unlike supervised or unsupervised learning where the learning outcome is called a model, in RL, it's called a **policy**.

MDP (Markov Decision Process): the framework of RL

- The agent interacts with the environment at timesteps: it takes one action to transit from the current state to the next.
- Unlike supervised or unsupervised learning where the learning outcome is called a model, in RL, it's called a **policy**.
- The agent picks an action to take in a state based on a policy.

MDP (Markov Decision Process): the framework of RL

- The agent interacts with the environment at timesteps: it takes one action to transit from the current state to the next.
- Unlike supervised or unsupervised learning where the learning outcome is called a model, in RL, it's called a **policy**.
- The agent picks an action to take in a state based on a policy.
- Of course there are good policies and bad policies. The **optimal** policy π^* is the one that will eventually maximize the reward of the agent.

MDP II

- An MDP problem has four components:

MDP II

- An MDP problem has four components:
  - a set of states, S

MDP II

- An MDP problem has four components:
  - a set of states, S
  - a set of actions, A

MDP II

- An MDP problem has four components:
  - a set of states, S
  - a set of actions, A
  - A probablistic transition function $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ where $s, s' \in S$ are states, t is a timestep, and $a \in A$ an action

MDP II

- An MDP problem has four components:
  - a set of states, S
  - a set of actions, A
  - A probablistic transition function $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ where $s, s' \in S$ are states, t is a timestep, and $a \in A$ an action
  - a reward function $r(s, a)$ of the immediate reward after taking action a at state s.

MDP II

- An MDP problem has four components:
  - a set of states, S
  - a set of actions, A
  - A probablistic transition function $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ where $s, s' \in S$ are states, t is a timestep, and $a \in A$ an action
  - a reward function $r(s, a)$ of the immediate reward after taking action a at state s.
- a policy is a function $\pi : A \times S \mapsto [0, 1]$ that maps a pair of action and state to a probability. A policy can be deterministic or stochastic.

MDP II

- An MDP problem has four components:
  - a set of states, S
  - a set of actions, A
  - A probablistic transition function $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ where $s, s' \in S$ are states, t is a timestep, and $a \in A$ an action
  - a reward function $r(s, a)$ of the immediate reward after taking action a at state s.
- a policy is a function $\pi : A \times S \mapsto [0, 1]$ that maps a pair of action and state to a probability. A policy can be deterministic or stochastic.
- The agent takes an action based on probabilities suggested by the policy.

MDP II

- An MDP problem has four components:
  - a set of states, S
  - a set of actions, A
  - A probablistic transition function $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ where $s, s' \in S$ are states, t is a timestep, and $a \in A$ an action
  - a reward function $r(s, a)$ of the immediate reward after taking action a at state s.
- a policy is a function $\pi : A \times S \mapsto [0, 1]$ that maps a pair of action and state to a probability. A policy can be deterministic or stochastic.
- The agent takes an action based on probabilities suggested by the policy.
- A **(state-)value function** defines the expected return starting with the state s: $V_\pi(s) = \mathrm{E}[R|s, \pi] = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$ where $r_t = r(s_t, a_t)$ is the reward at a timestep t, and $\gamma \in [0, 1]$ is the discount rate.

MDP II

- An MDP problem has four components:
  - a set of states, S
  - a set of actions, A
  - A probablistic transition function $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ where $s, s' \in S$ are states, t is a timestep, and $a \in A$ an action
  - a reward function $r(s, a)$ of the immediate reward after taking action a at state s.
- a policy is a function $\pi : A \times S \mapsto [0, 1]$ that maps a pair of action and state to a probability. A policy can be deterministic or stochastic.
- The agent takes an action based on probabilities suggested by the policy.
- A **(state-)value function** defines the expected return starting with the state s: $V_\pi(s) = \mathrm{E}[R|s, \pi] = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$ where $r_t = r(s_t, a_t)$ is the reward at a timestep t, and $\gamma \in [0, 1]$ is the discount rate.
- Another kind of value functions is **action-value function**: $Q : S \times A \mapsto \mathbb{R}$.

MDP II

- An MDP problem has four components:
  - a set of states, S
  - a set of actions, A
  - A probablistic transition function $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ where $s, s' \in S$ are states, t is a timestep, and $a \in A$ an action
  - a reward function $r(s, a)$ of the immediate reward after taking action a at state s.
- a policy is a function $\pi : A \times S \mapsto [0, 1]$ that maps a pair of action and state to a probability. A policy can be deterministic or stochastic.
- The agent takes an action based on probabilities suggested by the policy.
- A **(state-)value function** defines the expected return starting with the state s: $V_\pi(s) = \mathrm{E}[R|s, \pi] = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$ where $r_t = r(s_t, a_t)$ is the reward at a timestep t, and $\gamma \in [0, 1]$ is the discount rate.
- Another kind of value functions is **action-value function**: $Q : S \times A \mapsto \mathbb{R}$.
- An **optimal policy** is the one that maximizes the value function, not the immediate reward.

MDP II

- An MDP problem has four components:
    - a set of states, S
    - a set of actions, A
    - A probablistic transition function $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ where $s, s' \in S$ are states, t is a timestep, and $a \in A$ an action
    - a reward function $r(s, a)$ of the immediate reward after taking action a at state s.
- a policy is a function $\pi : A \times S \mapsto [0, 1]$ that maps a pair of action and state to a probability. A policy can be deterministic or stochastic.
- The agent takes an action based on probabilities suggested by the policy.
- A **(state-)value function** defines the expected return starting with the state s: $V_\pi(s) = \mathrm{E}[R|s, \pi] = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$ where $r_t = r(s_t, a_t)$ is the reward at a timestep t, and $\gamma \in [0, 1]$ is the discount rate.
- Another kind of value functions is **action-value function**: $Q : S \times A \mapsto \mathbb{R}$.
- An **optimal policy** is the one that maximizes the value function, not the immediate reward.
- How to find the optimal policy: gradient descent. Because the value function is

Exploration vs. exploitation

- The agent needs to take actions in order to know the environment.

Exploration vs. exploitation

- The agent needs to take actions in order to know the environment.
- So how to pick actions?

Exploration vs. exploitation

- The agent needs to take actions in order to know the environment.
- So how to pick actions?
- Exploitation: use the policy to decide. but purely relying on the policy, especially at early stage, will result in overfitting.

Exploration vs. exploitation

- The agent needs to take actions in order to know the environment.
- So how to pick actions?
- Exploitation: use the policy to decide. but purely relying on the policy, especially at early stage, will result in overfitting.
- Exploration: not follow the recommendation from the policy sometimes, e.g., ϵ-greedy (execute the best action per the policy with a probability $1-\epsilon$, and a random action otherwise), UCB, etc.

Model-based and model-free RL

- Model-based: To solve model-based RL, the agent first use a random policy to interact with the environment to obtain state transition data. Then compute the best policy that maximizes the value function using methods such as gradient descent.

Model-based and model-free RL

- Model-based: To solve model-based RL, the agent first use a random policy to interact with the environment to obtain state transition data. Then compute the best policy that maximizes the value function using methods such as gradient descent.
- It works well for problems of confined and small environment, e.g., teaching a robot to assemble parts together in the right order.

Model-based and model-free RL

- Model-based: To solve model-based RL, the agent first use a random policy to interact with the environment to obtain state transition data. Then compute the best policy that maximizes the value function using methods such as gradient descent.
- It works well for problems of confined and small environment, e.g., teaching a robot to assemble parts together in the right order.
- Model-free: The algorithm doesn't need/know the transition function, which is sometimes difficult or expensive to obtain.

Q-learning, a model-free RL approach

- Optimization goal is the action-value function Q. We wanna maximize it.

Q-learning, a model-free RL approach

- Optimization goal is the action-value function Q. We wanna maximize it.
- Bellman equation (dynamic programming):
  $Q(s_t, a_t) = E\left[r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a)\right]$ "The value of your starting point is the reward you expect to get from being there, plus the value of wherever you land next."

Q-learning, a model-free RL approach

- Optimization goal is the action-value function Q. We wanna maximize it.
- Bellman equation (dynamic programming): $Q(s_t, a_t) = E\left[r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a)\right]$ "The value of your starting point is the reward you expect to get from being there, plus the value of wherever you land next."
- Q update based on Bellman equation:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha \cdot (r(s_t, a_t) + \gamma \cdot \max_a Q(s_{t+1}, a))$$

where α is the learning rate.

Q-learning, a model-free RL approach

- Optimization goal is the action-value function Q. We wanna maximize it.
- Bellman equation (dynamic programming):
  $Q(s_t, a_t) = E\left[r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a)\right]$ "The value of your starting point is the reward you expect to get from being there, plus the value of wherever you land next."
- Q update based on Bellman equation:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha \cdot (r(s_t, a_t) + \gamma \cdot \max_a Q(s_{t+1}, a))$$

  where α is the learning rate.
- Initially, $Q(s, a) = 0$ for all s and a.

Further reading

- Value functions link 1 link 2

Further reading

- Value functions link 1 link 2
- Policy Gradient Algorithms by Lilian Weng

Further reading

- Value functions link 1 link 2
- Policy Gradient Algorithms by Lilian Weng
- OpenAI Spinning up

Further reading

- Value functions link 1 link 2
- Policy Gradient Algorithms by Lilian Weng
- OpenAI Spinning up
- On-policy vs. off-policy in Reinforcement Learning by Lei Mao

Further reading

- Value functions link 1 link 2
- Policy Gradient Algorithms by Lilian Weng
- OpenAI Spinning up
- On-policy vs. off-policy in Reinforcement Learning by Lei Mao
- Paper: Mastering the game of Go with deep neural networks and tree search, by Silver et al. at DeepMind, Nature, volume 529, pages 484–489 (2016)

Further reading

- Value functions link 1 link 2
- Policy Gradient Algorithms by Lilian Weng
- OpenAI Spinning up
- On-policy vs. off-policy in Reinforcement Learning by Lei Mao
- Paper: Mastering the game of Go with deep neural networks and tree search, by Silver et al. at DeepMind, Nature, volume 529, pages 484–489 (2016)
- Paper: Comparing exploration strategies for Q-learning in random stochastic mazes by Tijsma et al.