HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Machine Learning

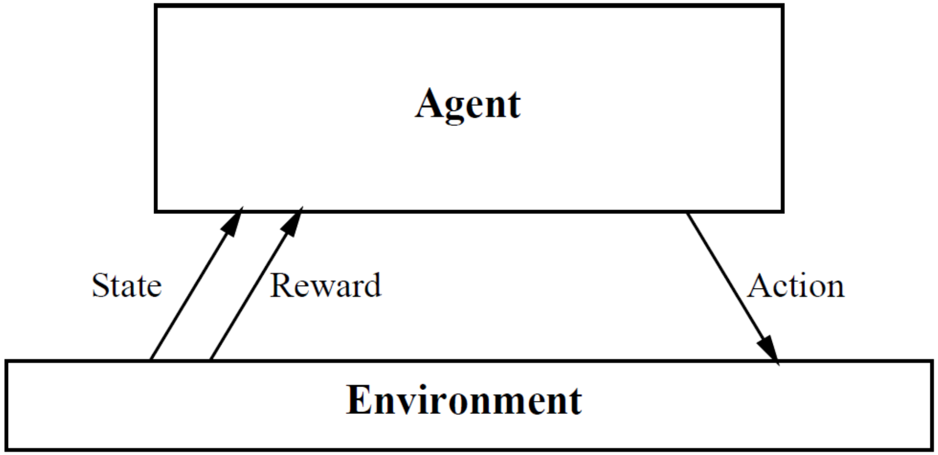
IT3190E

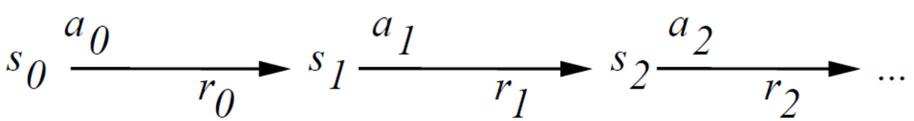
Lecture: Reinforcement Learning

Contents

- Lecture 1: Introduction to Machine Learning
- Lecture 2: Linear regression
- Lecture 3+4: Clustering
- Lecture 5: Decision tree and Random forest
- Lecture 6: Neural networks
- Lecture 7: Support vector machines
- Lecture 8: Performance evaluation
- Lecture 9: Probabilistic models
- Lecture 10: Ensemble learning
- **Lecture 11: Reinforcement learning**
- Lecture 12: Regularization
- Lecture 13: Discussion on some advanced topics

Reinforcement Learning problem





- **Goal:** Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots , \qquad where\ 0 \le \gamma < 1$$

(γ is the discount factor for future rewards)

(Mitchell, 1997)

Characteristics of Reinforcement learning

- What makes Reinforcement Learning (RL) different from other machine learning paradigms?
  - There is no explicit supervisor, only a **reward** signal
  - Training examples are of form ((S, A), R)
  - Feedback is often delayed
  - Time really matters (**sequential**, not independent data)
  - Agent's actions affect the subsequent data it receives
- Examples of RL
  - Play games better than humans
  - Manage an investment portfolio
  - Make a humanoid robot walk
  - …

Reward

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward
- Reinforcement learning is based on the **reward hypothesis**:
  - All goals can be described by the maximization of expected cumulative reward

Examples of reward

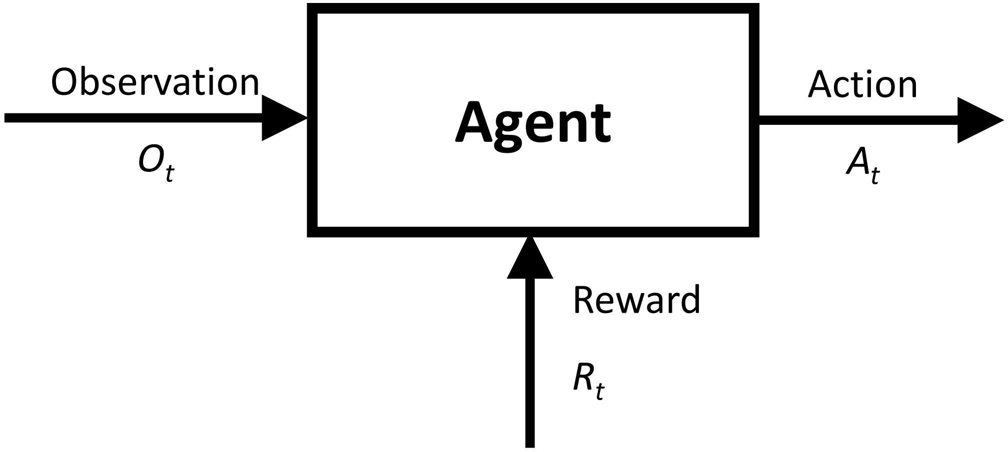
- Play games better than humans
  - + reward for increasing score
  - - reward for decreasing score
- Manage an investment portfolio
  - + reward for each $ in bank
- Make a humanoid robot walk
  - + reward for forward motion
  - - reward for falling over

Sequential decision making

- Goal: **Select actions to maximize total future reward**
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice an immediate reward to gain more long-term reward
- Examples:
  - A financial investment (may take months to mature)
  - Blocking opponent moves (might help winning chances, after many moves from now)

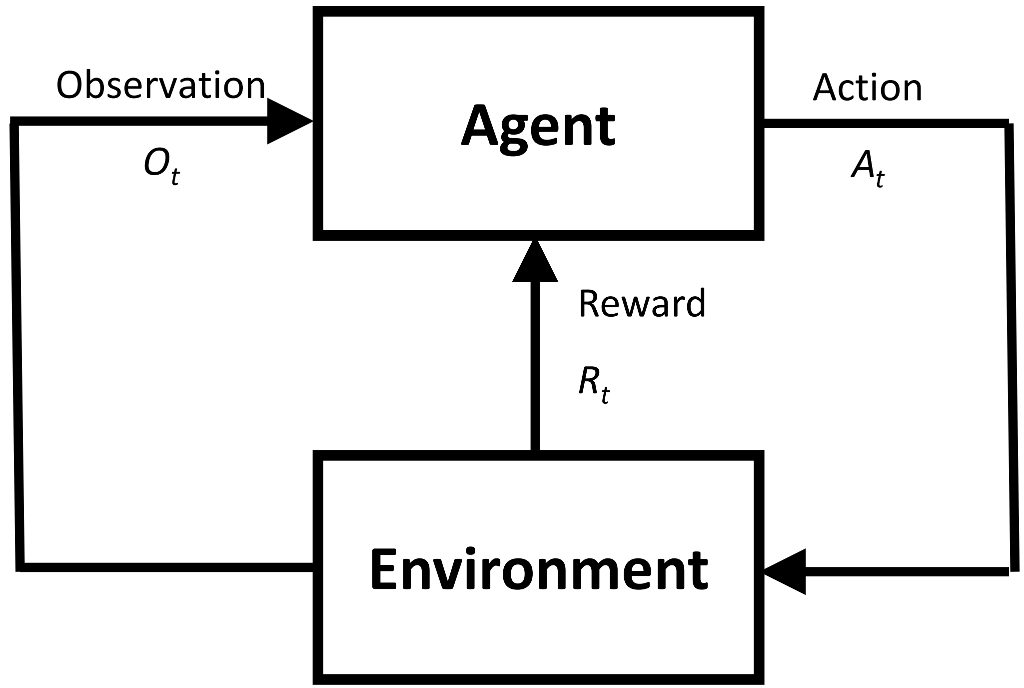
Agent and Environment (1)

- At each step *t*, the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t



Agent and Environment (2)

- At each step t, the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- At each step t, the environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at environment step



History and State

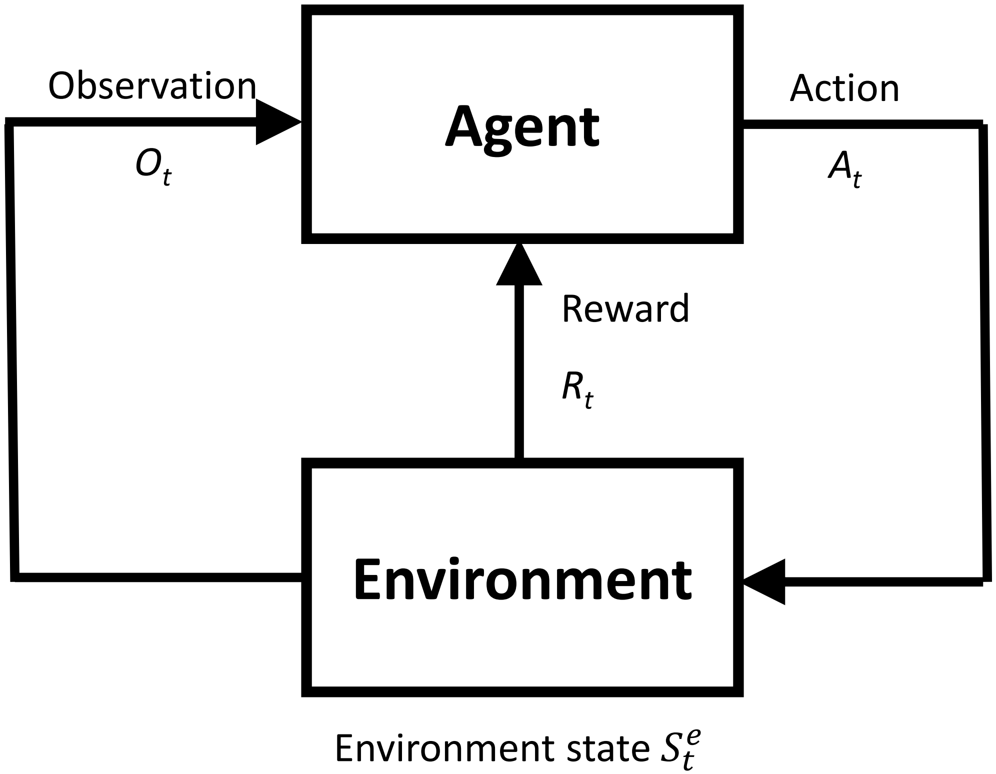
- The **history** is the sequence of observations, actions, rewards:

$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t$$

  - All observable variables up to time *t*
  - The sensorimotor stream of the agent
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- **State** is the information used to determine what happens next
- Formally, state is a function of the history:

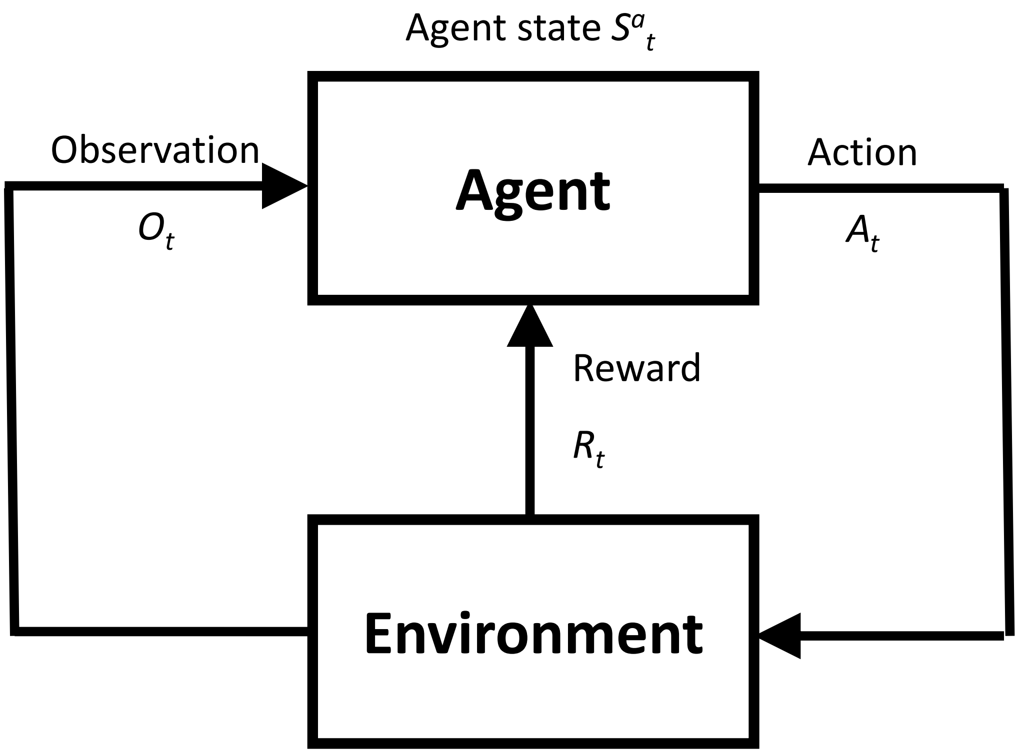
$$S_t = f(H_t)$$

Environment state



- The **environment state** S_t^e is the environment's private representation
  - The information the environment uses to pick the next observation or reward
- The environment state is not usually visible to the agent

Agent state



- The **agent state** S_t^a is the agent's internal representation
  - The information the agent uses to pick the next action
  - It is the information used by reinforcement learning algorithms
- It can be a function of history:

$$S_t^a = f(H_t)$$

Information state

- An **information state** (a.k.a. Markov state) contains all useful information from the history
- A state S_t is **Markov** if and only if:

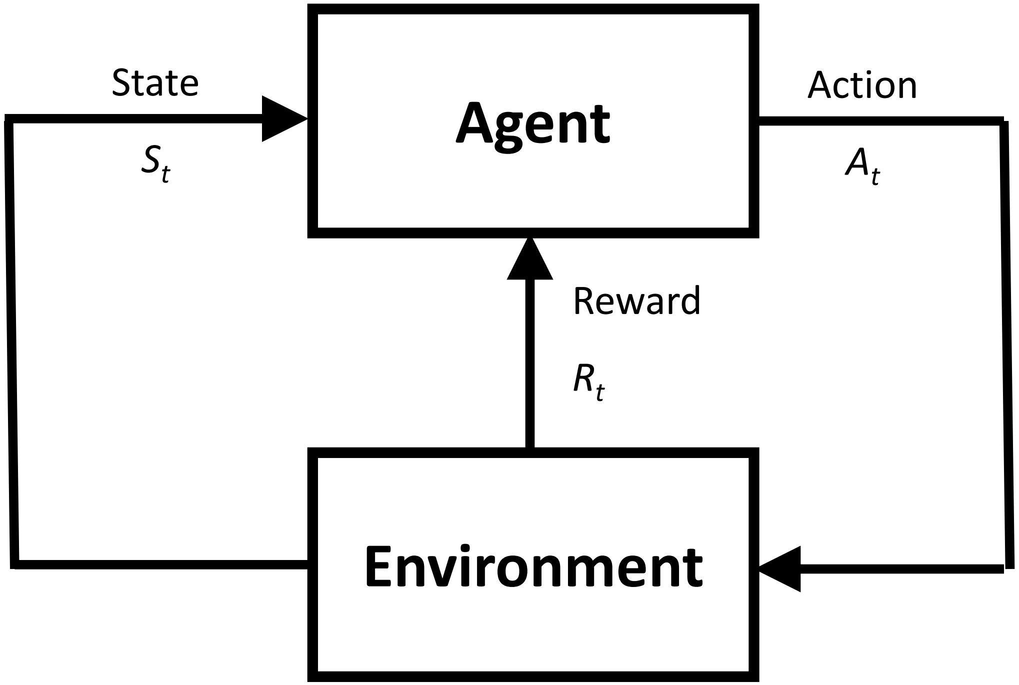
$$P(S_{t+1}|S_t) = P(S_{t+1}|\, S_1, \ldots, S_t)$$

  - The future is independent of the past given the present

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

  - Once the state is known, the history may be thrown away
  - The state is a sufficient statistic of the future
  - The environment state S_t^e is Markov
  - The history H_t is Markov

Fully observable environments



- **Full observability**: Agent **directly** observes environment state

$$O_t = S_t^a = S_t^e$$

- Agent state = Environment state = Information state
- Formally, this is a Markov decision process (MDP)

Partially observable environments

- **Partial observability**: Agent **indirectly** observes environment:
  - E.g., a robot with camera vision isn't told its absolute location
  - E.g., a trading agent only observes current prices
  - E.g., a poker playing agent only observes public cards
- Now, Agent state ≠ Environment state
- Formally this is a **partially observable Markov decision process (POMDP)**
- Agent must construct its own state representation S_t^a:
  - E.g., by using complete history: $S_t^a = H_t$
  - E.g., by using a recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Major components of a RL agent

A RL agent may include one or more of these components:

- **Policy**: Agent's behavior function
- **Value function**: How good is each state and/or action
- **Model**: Agent's representation of the environment

Policy

- A **policy** is the agent's behavior
- It is a map from state to action
- *Deterministic* policy: $a = \pi(s)$
- *Stochastic* policy: $\pi(a|s) = P(A_t = a \,|S_t = s)$

Value function

- **Value function** is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore, to select between actions

$$v_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s)$$

where $R_{t+1}, R_{t+2}, \ldots$ are generated by following policy π starting at state *s*

- For each policy π, we have a value $v_\pi(s)$
- We want to find the *optimal policy* π^* such that

$$v^*(s) = \max_\pi v_\pi(s), \qquad \forall s$$

Model

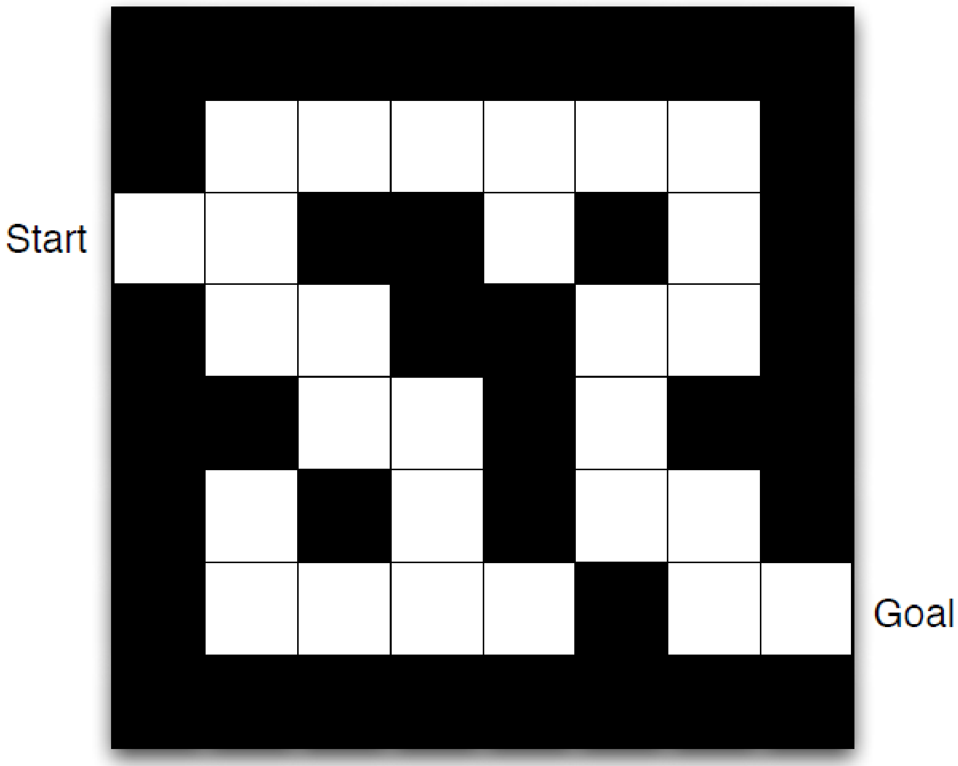
- A **model** predicts what the environment will do next
- *P* predicts the next state

$$P^a_{ss^*} = P(S_{t+1} = s^* \mid S_t = s, A_t = a)$$

- *R* predicts the next (*immediate*) reward

$$R^a_s = \mathbb{E}(R_{t+1} \mid S_t = s;\ A_t = a)$$

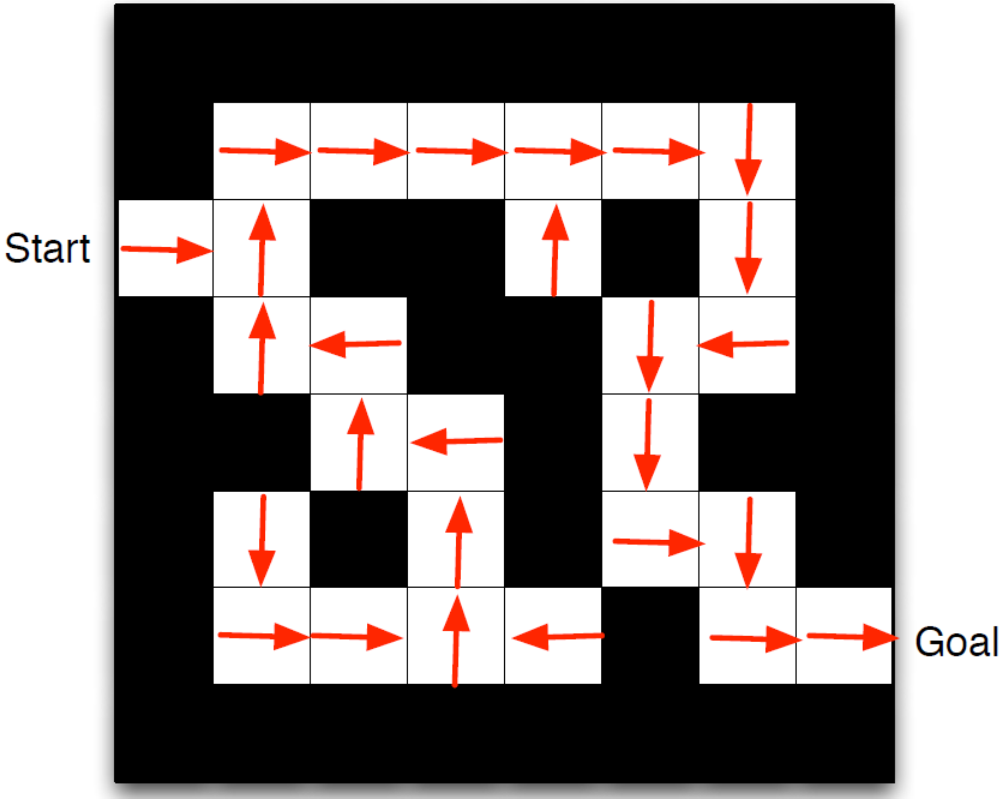
Maze example



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

(https://www.davidsilver.uk/wp-content/uploads/2020/03/intro_RL.pdf)

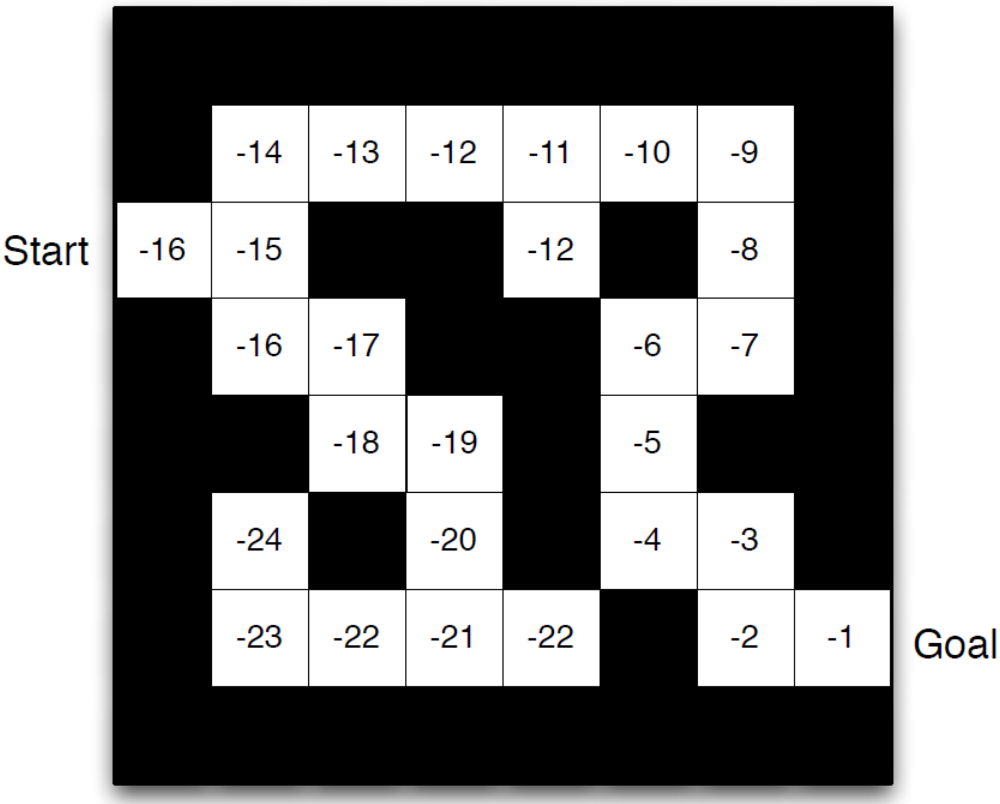
Maze example: Policy



(https://www.davidsilver.uk/wp-content/uploads/2020/03/intro_RL.pdf)

- Arrows represent policy $\pi(s)$ for each state s

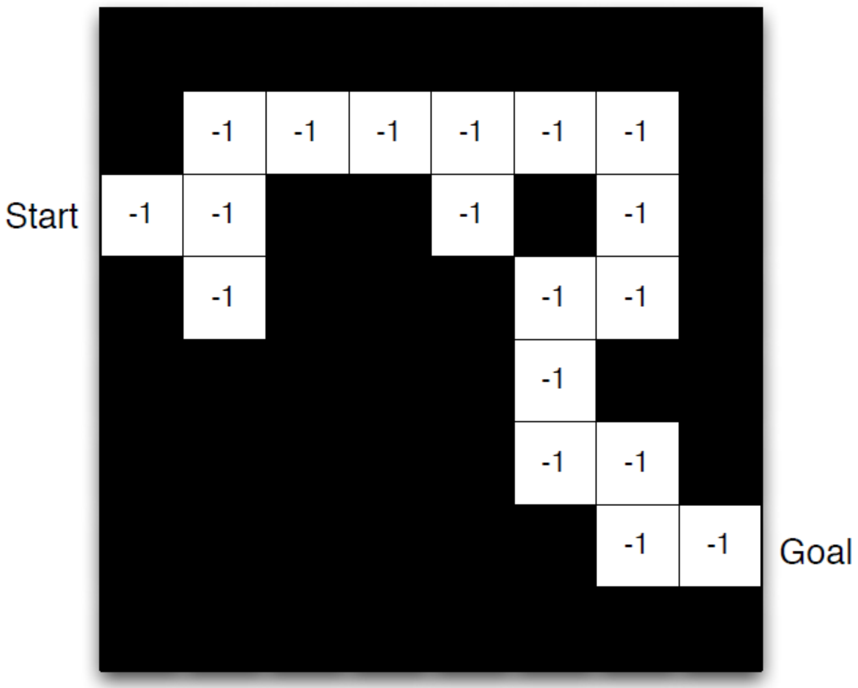
Maze example: Value function



(https://www.davidsilver.uk/wp-content/uploads/2020/03/intro_RL.pdf)

- Numbers represent value $v_\pi(s)$ of each state s

Maze example: Model



(https://www.davidsilver.uk/wp-content/uploads/2020/03/intro_RL.pdf)

- Agent may have an internal model of the environment
- Dynamics: How actions change the state
- Rewards: How much reward from each state
- Grid layout represents transition model $P_{ss'}^{a}$
- Numbers represent immediate reward R_s^a from each state *s* (same for all actions *a*)

Categorizing RL agents (1)

- Value-based
  - No policy
  - Value function
- Policy-based
  - Policy
  - No value function
- Actor critic
  - Policy
  - Value function

Categorizing RL agents (2)

- Model-free
  - Policy and/or Value function
  - No model
- Model-based
  - Policy and/or Value function
  - Model

Exploration and Exploitation (1)

- Reinforcement learning is like *trial-and-error learning*
- The agent should discover a good policy
- from its experiences of the environment
- without losing too much reward along the way

Exploration and Exploitation (2)

- **Exploration** finds more information about the environment
- **Exploitation** exploits known information to maximize reward
- It is usually important to **both** *explore* and *exploit*

Exploration and Exploitation: Examples

- Restaurant selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Online banner advertisements
  - Exploitation: Show the most successful advertisement
  - Exploration: Show a different advertisement
- Game playing
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move

Q-Learning: What to learn

- We might try to have agent learn the value function v_π
- It could then do a *lookahead* search to choose best action from any state s because

$$\pi(s) = \arg\max_a \left(r(s,a) + \gamma v_\pi\big(\delta(s,a)\big)\right)$$

  - δ: $S \times A \rightarrow S$ will map a given action a and state s to *the next state*
  - r: $S \times A \rightarrow R$ provides the reward of action a, from state s
- A problem:
  - This works well if agent knows functions δ and r
  - But when it doesn't, it can't choose actions by this way

Q-Function

- Define new function very similar to v:

$$Q(s,a) = r(s,a) + \gamma v_\pi(\delta(s,a))$$

  - $Q(s,a)$ shows how good it is to perform action a when in state s
  - whereas $v_\pi(s)$ shows how good it is for the agent to be in state s

- If agent learns Q, it can choose optimal action

$$\pi(s) = \arg\max_a \left(r(s,a) + \gamma v_\pi\left(\delta(s,a)\right)\right) = \arg\max_a Q(s,a)$$

- Q is the value function the agent will learn

Training rule to learn Q

- Note that Q and v_π are closely related

$$v_\pi(\mathrm{s}) = \max_{a'} Q(s, a')$$

- Which allows us to write Q recursively as

$$\begin{aligned} Q(s_{t,} a_t) &= r(s_t, a_t) + \gamma v_\pi(\delta(s_t, a_t)) \\ &= r(s_t, a_t) + \gamma v_\pi(s_{t+1}) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

- Let Q* denote learner (agent)'s current approximation to Q, consider the training rule

$$Q^*(s, a) \leftarrow r(s, a) + \gamma \max_{a'} Q^*(s', a')$$

  - where s' is the state resulting from applying action a in state s

Q-Learning for deterministic worlds

For each s, initialize table entry $Q^*(s,a) \leftarrow 0$

Observe current state s

Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $Q^*(s,a)$ as follows:

$$Q^*(s,a) \leftarrow r + \gamma \max_{a'} Q^*(s',a')$$

- $s \leftarrow s'$

Note:
- *Finite* action space
- *Finite* state space

Updating Q*

- $Q^*(s_1, a_{right}) \leftarrow r + \gamma . \left(\max_{a'} Q^*(s_2, a') \right)$

$$\leftarrow 0 + 0.9 \, . \, max(63, 81, 100)$$

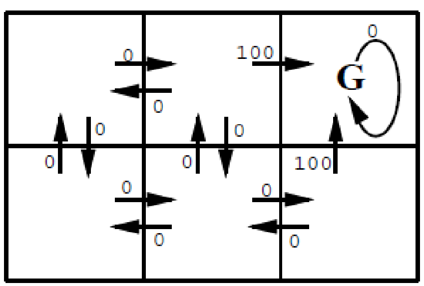
$$\leftarrow 90$$

- Note that if rewards are non-negative, then

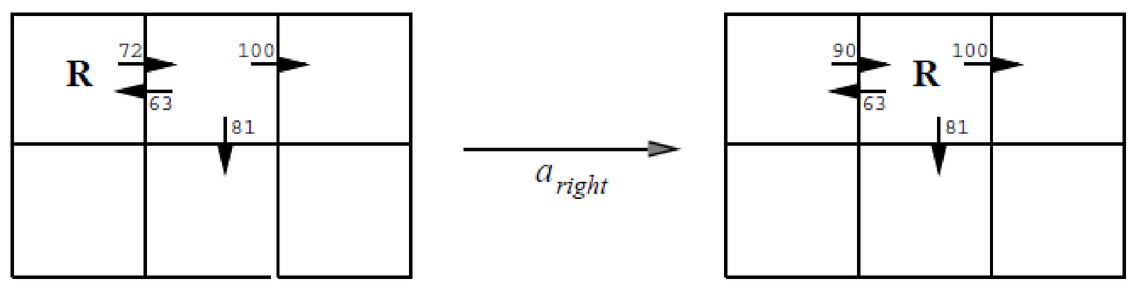
$$\forall\, s, a, n\text{:}\;\; Q^*_{n+1}(s,a) \geq Q^*_n(s,a)$$

$$\forall\, s, a, n\text{:}\;\; 0 \leq Q^*_n(s,a) \leq Q(s,a)$$

  - Where Q^*_n is the value at iteration n



$r(s, a)$ (immediate reward) values



initial state: s_1

next state: s_2

(Mitchell, 1997)

Q-Learning for non-deterministic worlds

- What if reward and next state are non-deterministic?
- We redefine v_π and Q by taking expected values

$$v_\pi(s) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots]$$

$$Q(s,a) = \mathbb{E}\big[r(s,a) + \gamma v_\pi\big(\delta(s,a)\big)\big]$$

$$= \sum_{s',r} P(s',r|\, s,a)\,[r + \gamma v_\pi(s')]$$

- Q-learning generalizes to non-deterministic worlds
  - Alter the training rule at iteration n to:

$$Q_n^*(s,a) \leftarrow (1-\alpha_n).\, Q_{n-1}^*(s,a) + \alpha_n \left[r + \max_{a'} Q_{n-1}^*(s',a')\right]$$

  - where α_n is sometimes known as learning rate

References

- D. Silver. *Lecture 1: Introduction to Reinforcement Learning* (https://www.davidsilver.uk/wp-content/uploads/2020/03/intro_RL.pdf).
- T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !