IF3270 Pembelajaran Mesin

Reinforcement Learning (RL)

Tim Pengajar IF3270

Review

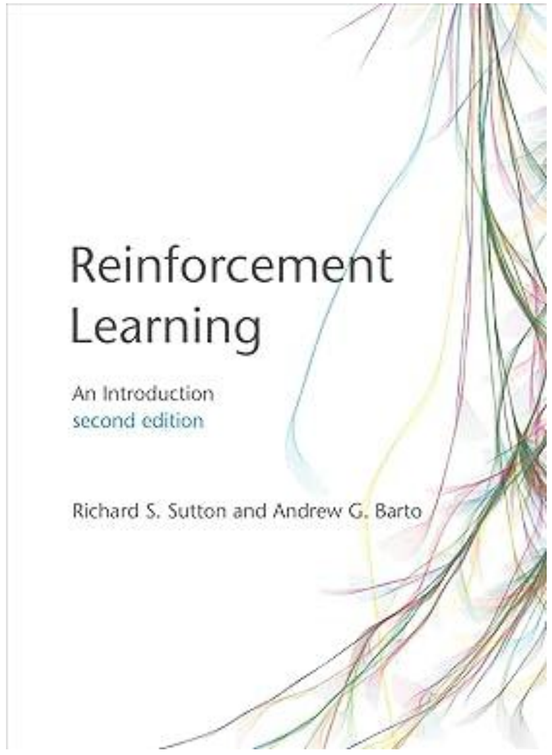
- Machine learning (ML) overview

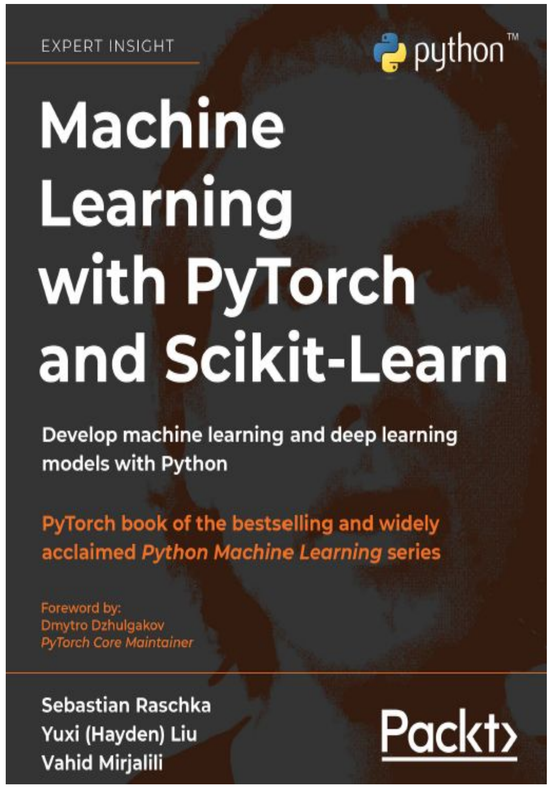
- Ensemble Methods → Supervised Learning

- Supervised Learning: Perceptron
- Supervised Learning: ANN: Feed Forward Neural Network
- Supervised Learning: ANN: Convolutional Neural Network
- Supervised Learning: ANN: Recurrent Neural Network
  - Simple RNN
  - LSTM
  - Encoder-Decoder
  - Attention

- Reinforcement Learning Today

References



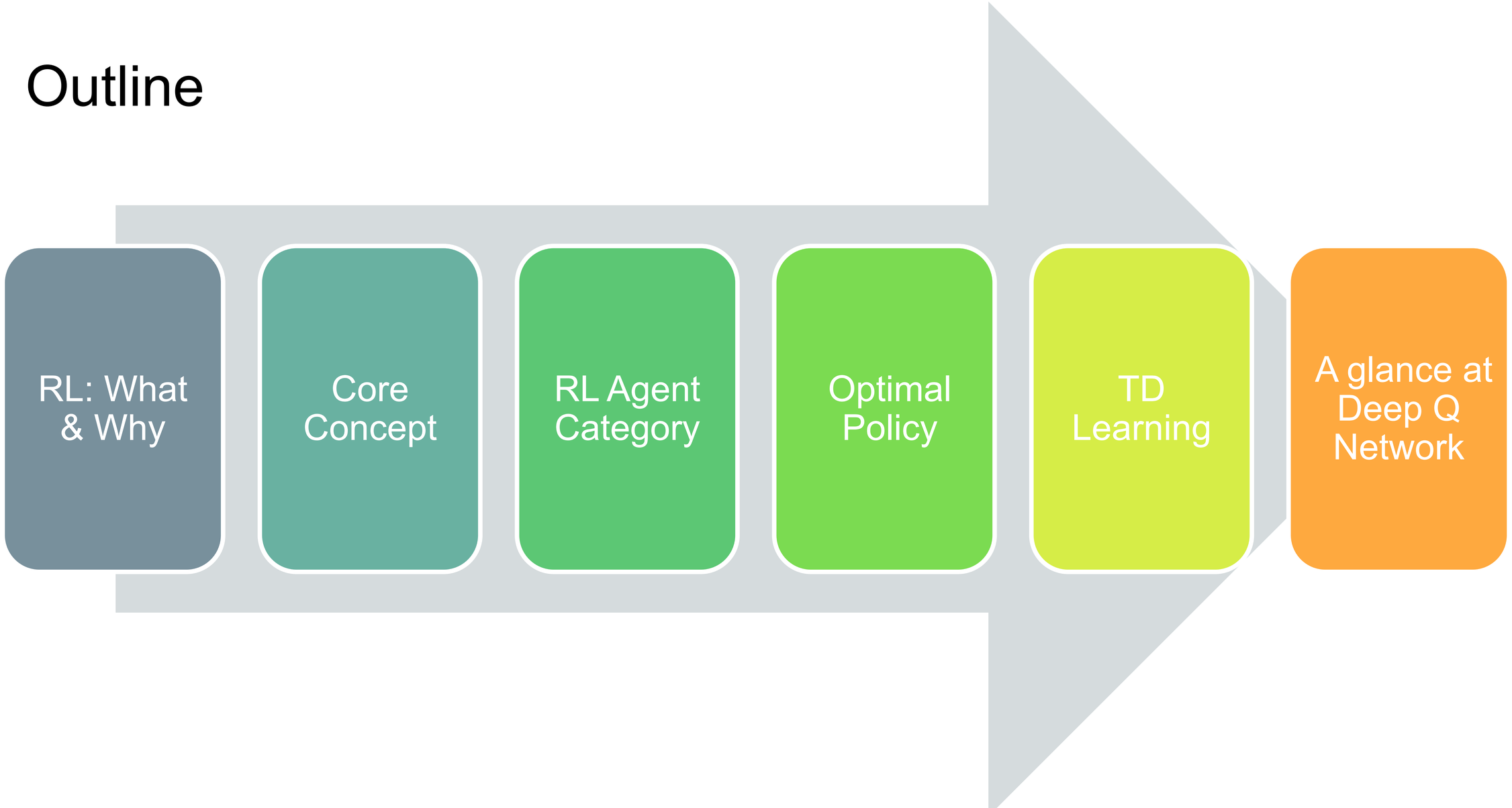


Reinforcement Learning, 2nd Edition. Richard Sutton, Andrew Barto. The MIT Press, 2018

Raschka, et.al., Machine Learning with Pytorch and Scikit-Learn, Packt Publishing Ltd., 2022 (Chapter 15)

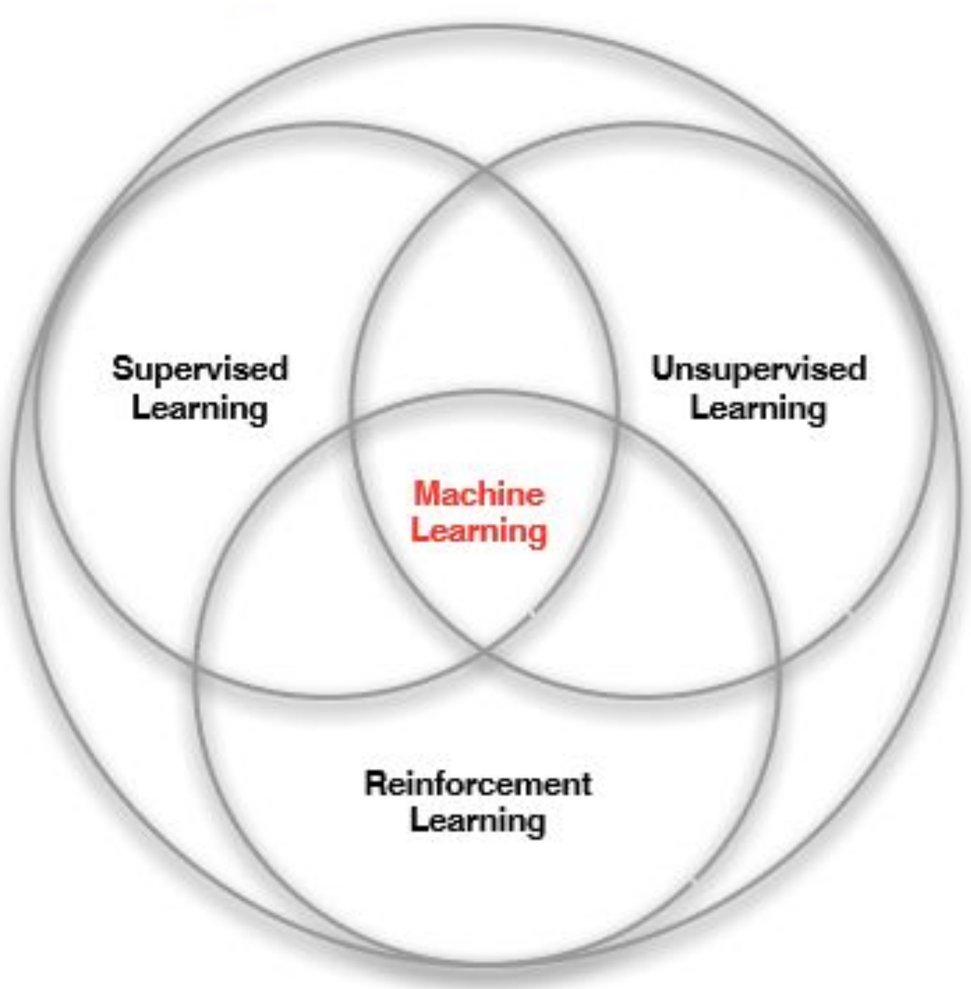
Slides: https://github.com/enggen/DeepMind-Advanced-Deep-Learning-and-Reinforcement-Learning/tree/master/lecture%20slides

Outline


RL: What & Why
Core Concept
RL Agent Category
Optimal Policy
TD Learning
A glance at Deep Q Network

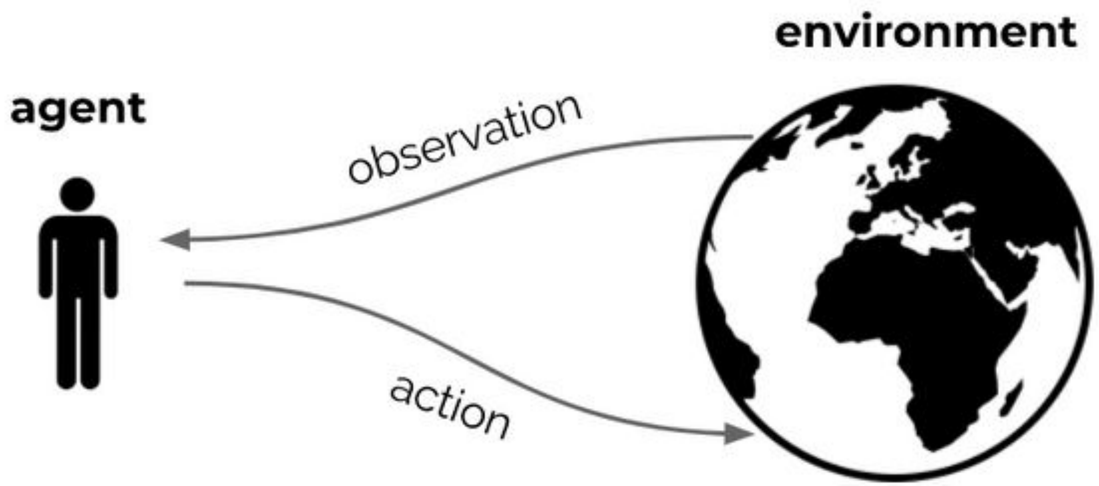
RL: What & Why

Branches of Machine Learning



What is Reinforcement Learning → belajar dgn berinteraksi

- We, and other intelligent beings, learn by interacting with our environment
- This differs from certain other types of learning
- We are goal-directed
- We can learn without examples of optimal behaviour
- Science of learning to make decisions from interaction



https://github.com/enggen/DeepMind-Advanced-Deep-Learning-and-Reinforcement-Learning/tree/master/lecture%20slides

Characteristics of Reinforcement Learning

How does reinforcement learning differ from other machine learning paradigms?

- No supervision, only a reward signal
- Feedback can be delayed, not instantaneous
- Time matters □ Earlier decisions affect later interactions

Supervised learning

Given labeled data: $\{(x_i, y_i)\}$, learn $f(x) \approx y$

- directly told what to output
- inputs x are independently, identically distributed (i.i.d.)

Reinforcement learning

Learn *behavior* $\pi(a \mid s)$.

- from experience, indirect feedback
- data **not** i.i.d.: actions a affect the future observations.

Behavior can include:

motor control

chat bots

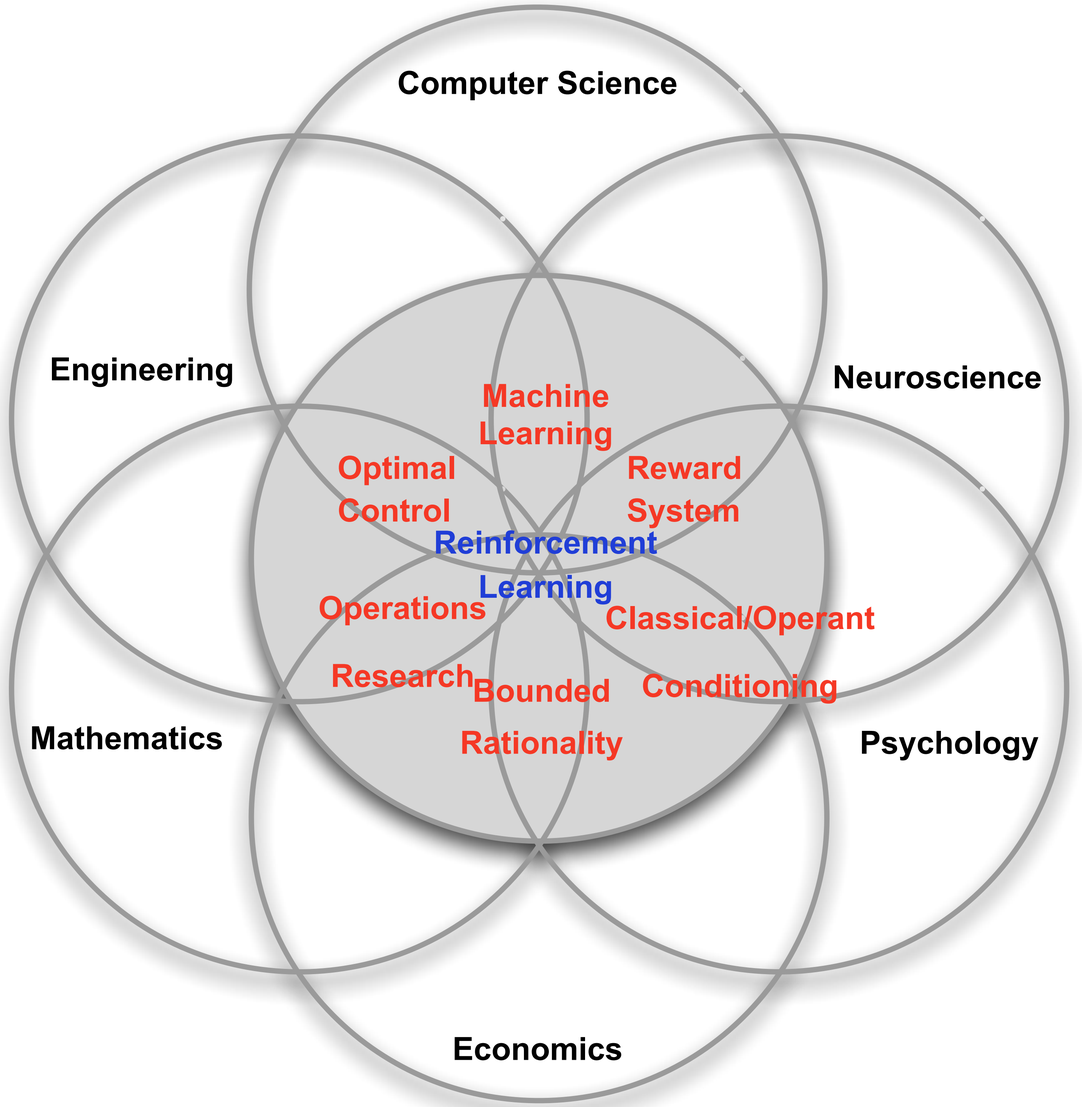
game playing

driving

web agents

https://cs224r.stanford.edu/slides/01_cs224r_intro_2025.pdf

Related Disciplines



Why RL → tdk semua case bs solved using direct supervising

https://cs224r.stanford.edu/slides/01_cs224r_intro_2025.pdf

1. Going beyond supervised (x, y) examples When direct supervision isn't available, decision making problems are everywhere
2. Widely used and deployed for performant AI systems example: learning complex physical tasks (legged robot)

3. Learning from experience seems fundamental to intelligence RL can discover new solutions

Ability to **discover** new solutions:

"Move 37" in Lee Sedol AlphaGo match surprises everyone

4. Plenty of exciting open research problems can we use RL to learn cooking a meal?; can robot practice fully autonomously?...

RL: Core Concept

Core Concepts

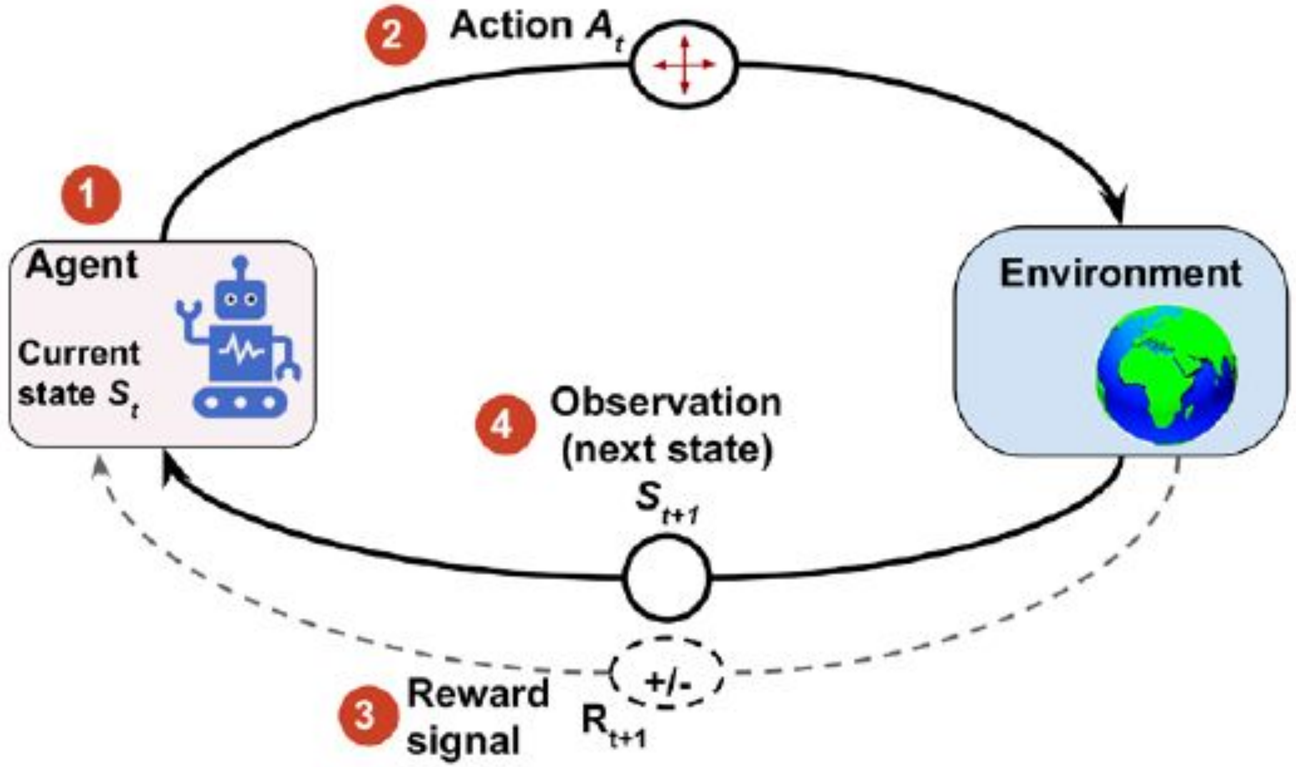
Environment

Reward Signal

Agent

- Agent state
- Policy
- Value function (probably)
- Model (optionally)

Agent and Environment



At each step t the agent:

- Receives observation O_t (and reward R_t)
- Executes action A_t

The environment:

- Receives action A_t
- Emits observation O_{t+1} (and reward R_{t+1})

Rewards

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t → defines the goal
- The agent's job is to maximize cumulative reward

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ ...$$

- We call this the return

Reinforcement learning is based on the reward hypothesis

Denition (Reward Hypothesis):

Any goal can be formalized as the outcome of maximizing a cumulative reward

Values

- We call the expected cumulative reward, from a state s, the value

$$v(s) = \mathbb{E}\left[G_t \mid S_t = s\right]$$
$$= \mathbb{E}\left[R_{t+1} + R_{t+2} + R_{t+3} + \ldots \mid S_t = s\right]$$

- Goal is then to maximize value, by picking suitable actions
- Rewards and values defines desirability of a state or action (no supervised feedback)
- Note that returns and values can be defined recursively

$$G_t = R_{t+1} + G_{t+1}$$

Actions in Sequential Problems

- Goal: select actions to maximize value
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
  a) A financial investment (may take months to mature)
  b) Refueling a helicopter (might prevent a crash in several hours)
  c) Blocking opponent moves (might help winning chances many moves from now)
- **A mapping from states to actions is called a policy**

Action Values

- It is possible to condition the value on actions:

$$q(s,a) = \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right]$$
$$= \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \ldots \mid S_t = s, A_t = a\,]$$

Agent Components

- Agent state
  - Action depends on the state
  - Both agent and environment my have an internal state
  - The state of agent generally differs from the state of environment
  - The agent may not know the full state of environment
  - History is a sequence of observations, action, rewards:

$$\mathcal{H}_t = O_0, A_0, R_1, O_1, \ldots, O_{t-1}, Q_{t-1}, R_t, O_t$$

  - This history can be used to construct an agent state S_t
  - Actions depend on this state
  - Full observability: observation = environment state: $S_t = O_t$

↳ bisa menangkap semua informasi

- Policy
- Value function
- Model

Agent Components

- Agent state
- Policy
  - Policy defines agent's behavior
  - A map from agent state to actions
  - Deterministic policy: $A = \pi(S)$
  - Stochastic policy: $\pi(A|S) = p(A|S)$
- Value function
- Model

Agent Components

- Agent state
- Policy
- Value function
  - The actual value function is the expected return

$$v(s) = \mathbb{E}[G_t \mid S_t = s, \pi]$$
$$= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s, \pi\right]$$

  - Discount factor $\gamma \in [0,1]$: trades off importance of immediate vs long-term rewards
  - The value depends on policy
  - Can be used to evaluate the desirability of states; can be used to select action
  - So: $v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma\, G_{t+1} \mid S_t = s, A_t \sim \pi(s)]$
    $= \mathbb{E}[R_{t+1} + \gamma\, v_t(S_{t+1}) \mid S_t = s, A_t \sim \pi(s)]$ → Bellman Equation
  - Optimal value (highest possible value)

$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma\, v_t(S_{t+1}) \mid S_t = s, A_t = a]$$

  - Agents usually approximate value functions → can behave well
- Model

Agent Components

- Agent state
- Policy
- Value function
- Model
  - A model predicts what the environment will do next
  - E.g., $\mathcal{P}$ predicts the next state:

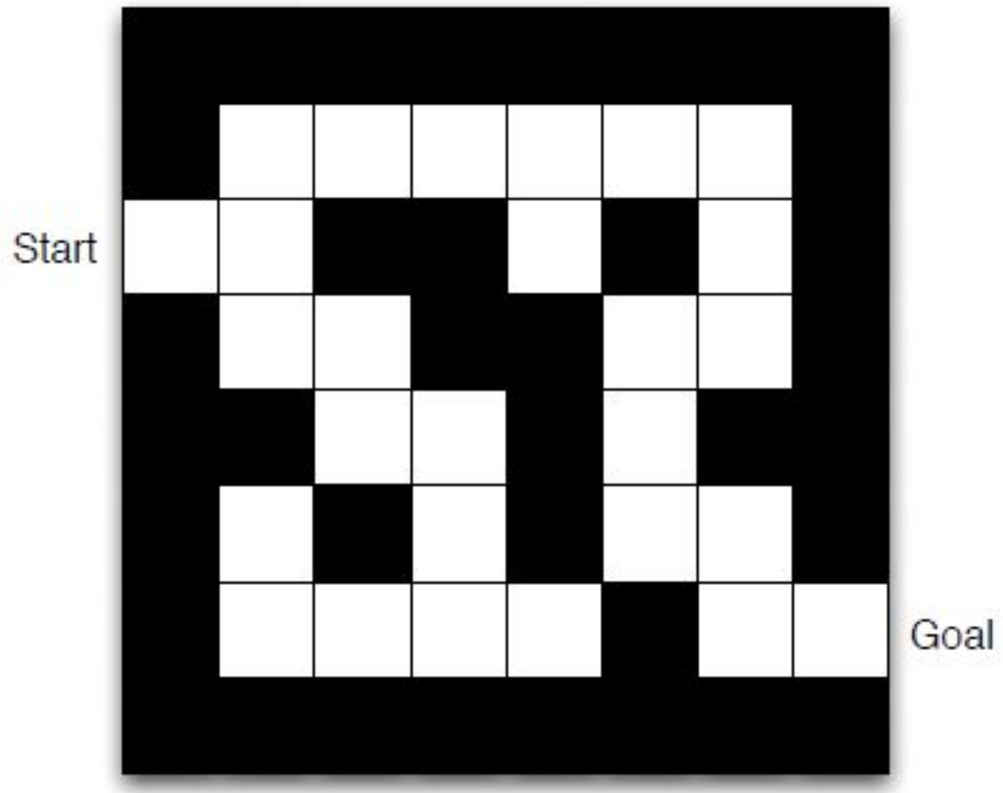
$$\mathcal{P}(s, a, s') \approx p(S_{t+1} = s' \mid S_t = s, A_t = a)$$

  - E.g., $\mathcal{R}$ predicts the next (immediate) reward:

$$\mathcal{R}(s, a) \approx \mathbb{E}[R_{t+1} S_t = s, A_t = a]$$

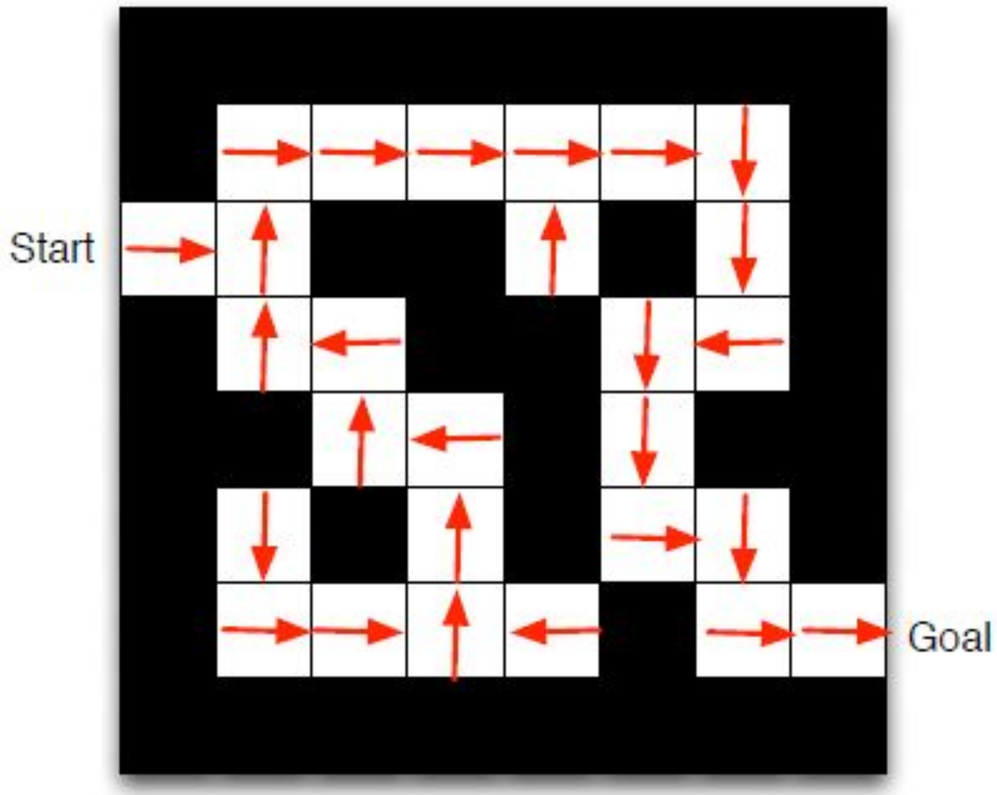
  - A model does not immediately gives us a good policy – we would still need good plan

MAZE Example



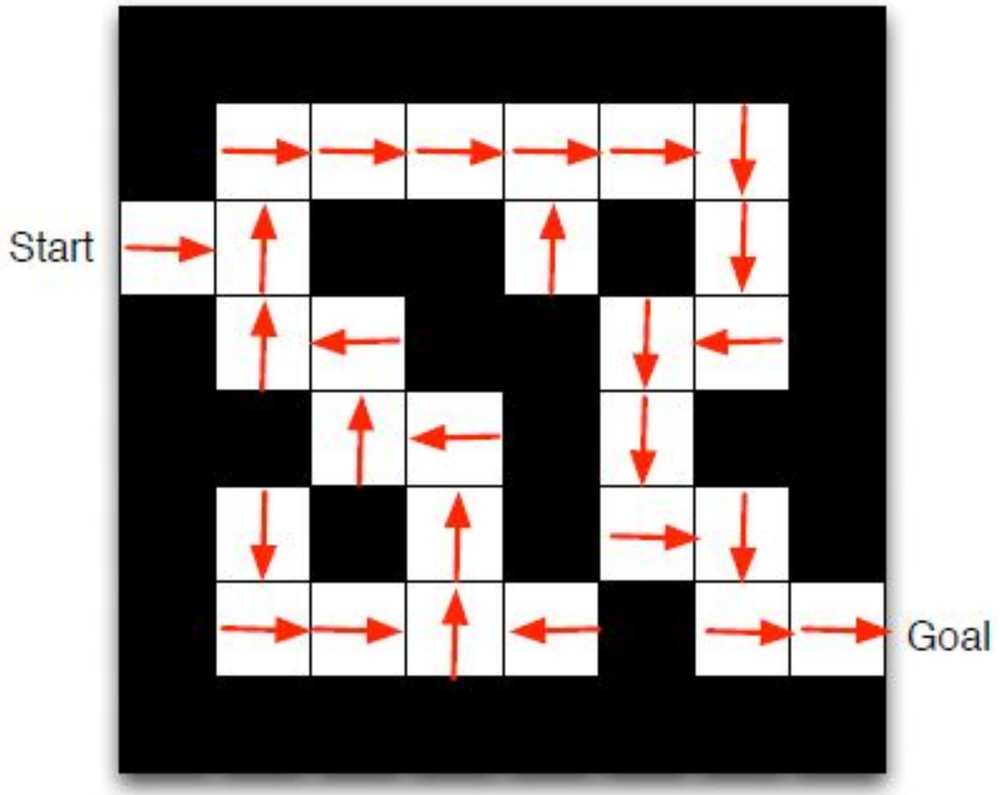
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

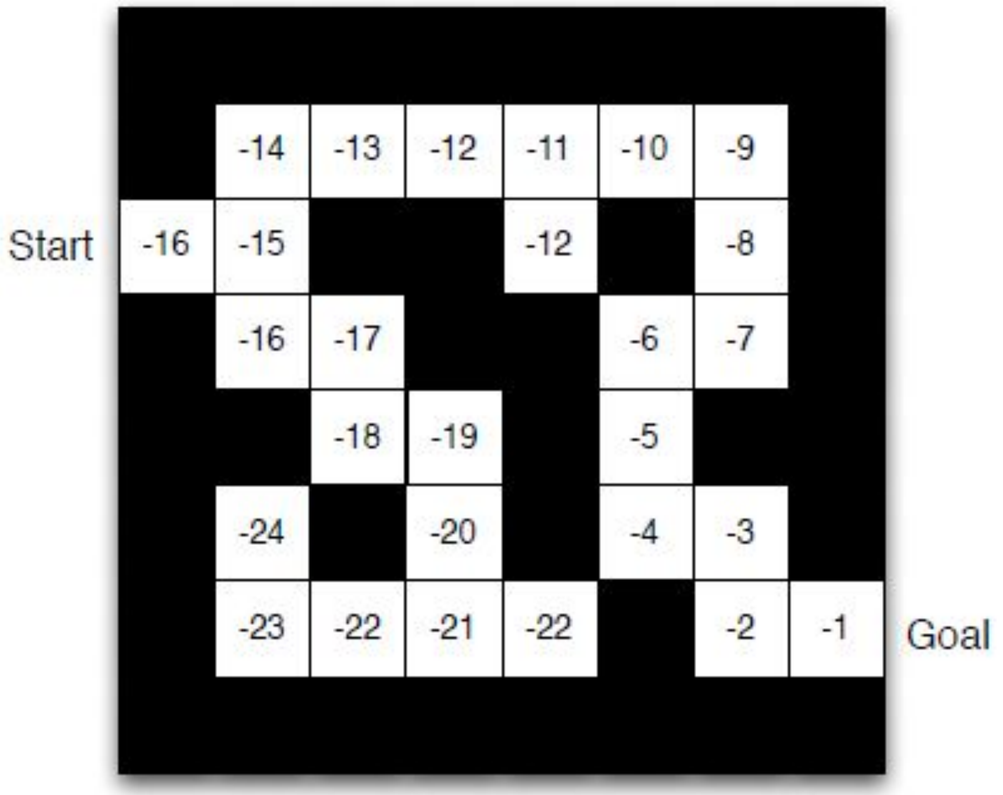
MAZE Example: Policy



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

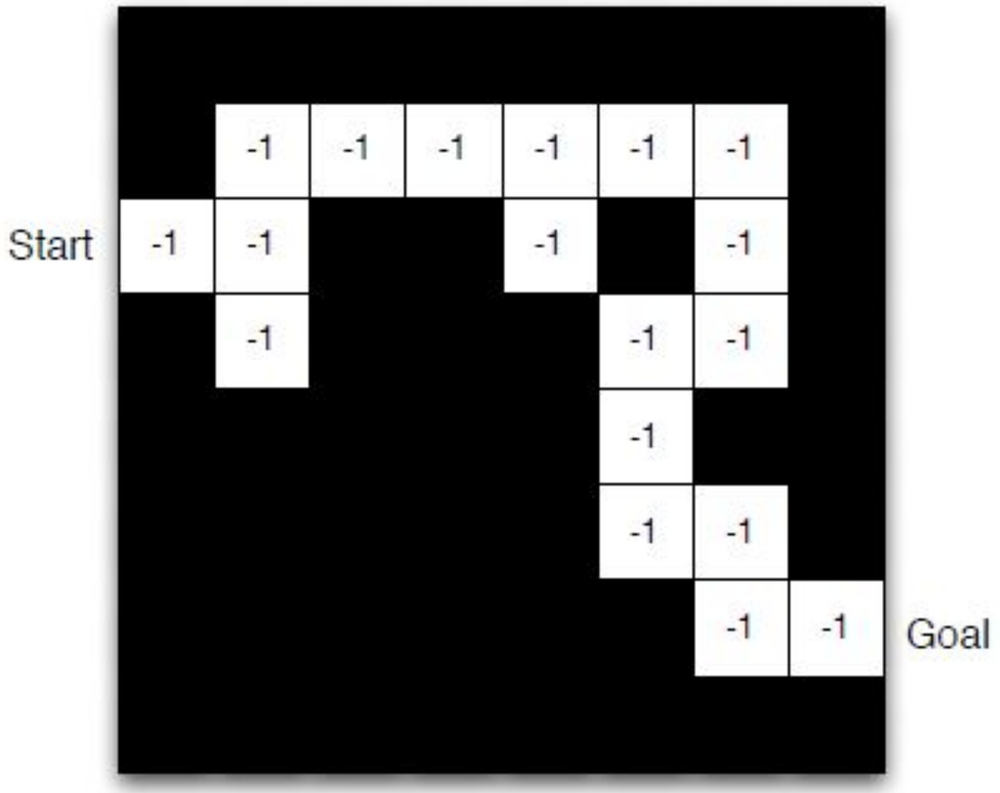
MAZE Example: Value Function





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

MAZE Example: Model



- Grid layout represents partial transition model $\mathcal{P}^a_{ss'}$
- Numbers represent immediate reward $\mathcal{R}^a_{ss'}$ from each state s (same for all a and s' in this case)

Exploration and Exploitation in RL

- We learn by trial and error
- The agent should discover a good policy
  - from new experiences
  - without sacrificing too much reward along the way
- Exploration finds more information
- Exploitation exploits known information to maximize reward
- It is important to explore as well as exploit
- This is a fundamental problem that does not occur in supervised learning
- Examples:
  - Restaurant Selection
    - Exploitation Go to your favourite restaurant
    - Exploration Try a new restaurant
  - Oil Drilling
    - Exploitation Drill at the best known location
    - Exploration Drill at a new location
  - Game Playing
    - Exploitation Play the move you currently believe is best
    - Exploration Try a new strategy

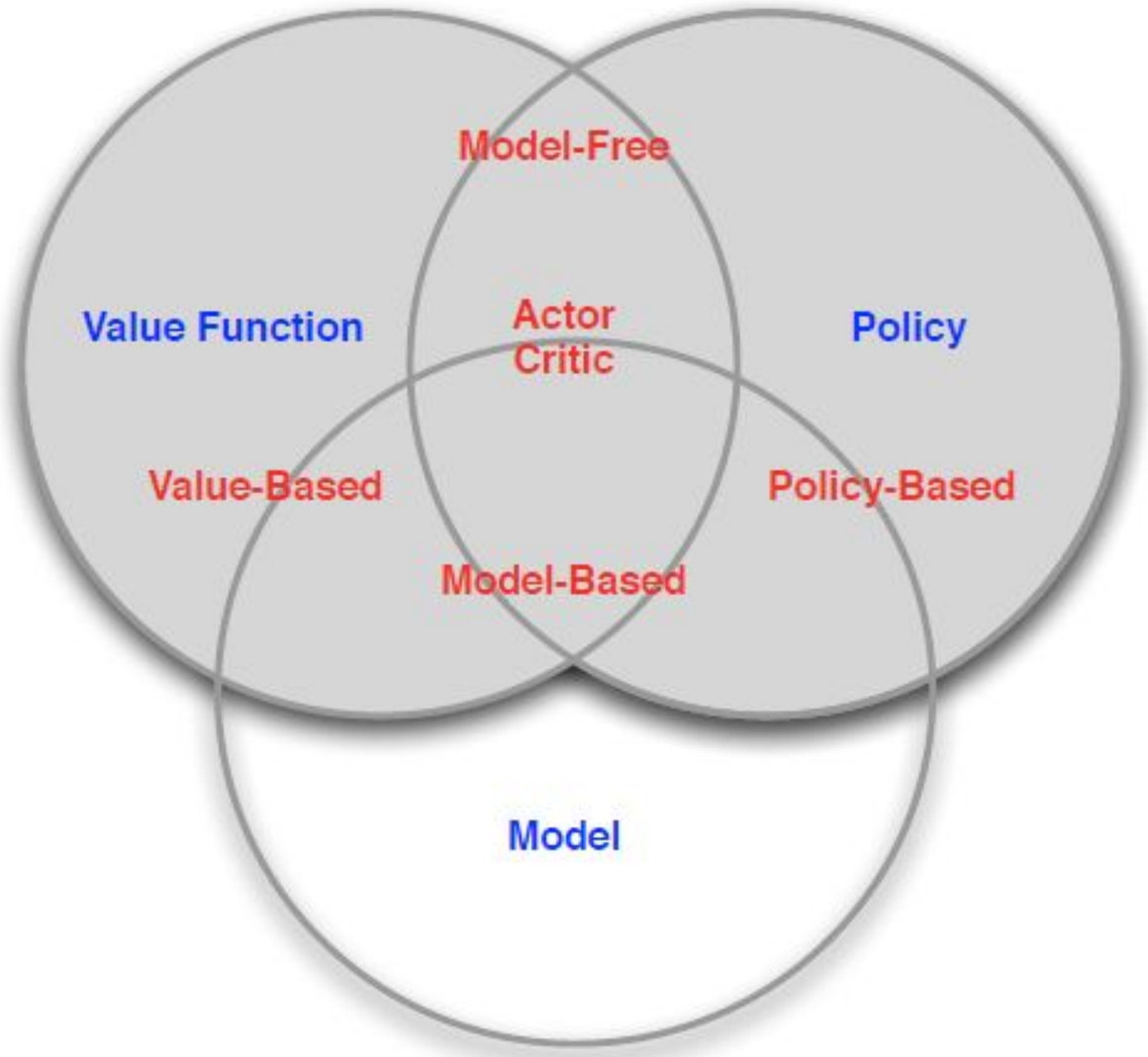
RL: Categorizing Agent

Categorizing Agents

- Value Based
  - No Policy (implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function

- **Model Free our course**
  - Policy and/ or Value Function
  - No Model
- Model Based
  - Optional Policy and/ or Value Function
  - Model

Agent Taxonomy



RL: Optimal Policy

Formalizing RL Interface

- We discuss a mathematical formulation of the agent-environment interaction
- This is called a Markov Decision Process (MDP)
- We can then talk clearly about the objective and how to reach it
- Assume the environment is fully observable → the current observation contains all relevant information
- Markov Property: The future is independent of the past given the present
- Markov Decision Process is a tuple $(S, \mathcal{A}, p, \gamma)$, where

  S : set of all possible states

  $\mathcal{A}$: set of all possible actions

  $p(r, s'|s, a)$: joint probability of reward r and state s', given a state s and action a

  $\gamma \in [0,1]$: discount factor

Return

- Acting in a MDP results in returns G_t : total discounted reward from time-step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The value function v(s) gives the long-term value of state s:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t \mid S_t = s, \pi] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, \pi] \\ &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t \sim \pi(s)] \\ &= \sum_a \pi(a|s) \sum_r \sum_{s'} p(r, s'|s, a)(r + \gamma v_\pi(s')) \end{aligned}$$

- We can define state-action value

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}[G_t \mid S_t = s, A_t = a, \pi] \\ &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a)] \\ &= \sum_r \sum_{s'} p(r, s'|s, a)(r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')) \end{aligned}$$

- An MDP is solved when we know the optimal value function

  - optimal state value function v*(s) ; optimal action-value function q*(s,a)

Optimal Policy

- There are equivalences between state and action values

$$v_\pi(s) = \sum_a \pi(a|s)\, q_\pi(s,a)$$

$$v_*(s) = max_a\ q_*(s,a)$$

- An optimal policy can be found by maximizing over q*(s,a)
- There can be multiple optimal policies → if multiple actions maximize q*(s, ·), we can just pick any of these
- Many iterative solution methods:

  - Using models (dynamic programming) → when $p(r, s'|s, a)$ is known

  - Using samples (Monte Carlo, **Temporal Difference Learning (Q-Learning, SARSA)**)

RL: Temporal Difference Learning

Temporal Difference (TD) Learning

- Does not need to wait until we get Return (G_t) □ Monte Carlo does

  - Can learn from incomplete sequences of observations (samples)

- Natural Idea:

  - Apply TD to q(s,a)

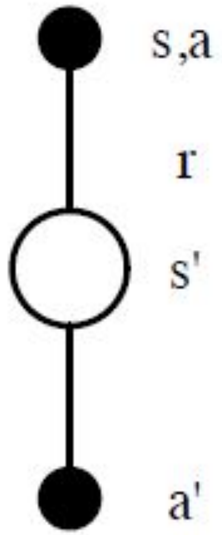
  - Update every time-step

- Two algorithm for TD Control:
  - on-policy control (SARSA)

  - off-policy control (Q-Learning)

SARSA

- “Learn on the job”
- Learn about policy π from experience sampled from π



$$q(s,a) \leftarrow q(s,a) + \alpha\left(r + \gamma q(s',a') - q(s,a)\right)$$

Tabular SARSA

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-Learning

- Look over someone's shoulder"
- Learn about policy π from experience sampled from b
- E.g. the target policy π is greedy w.r.t q(s,a)
- The behavior policy b is $\epsilon - greedy$ w.r.t q(s,a)

Q-Learning

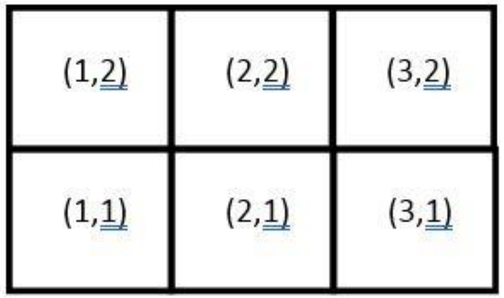
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

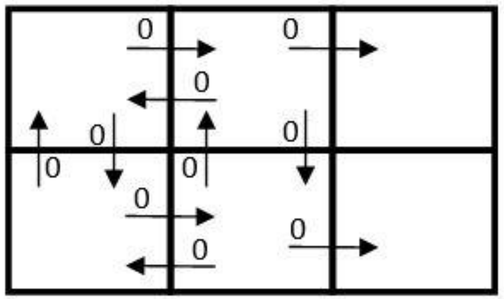
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Contoh

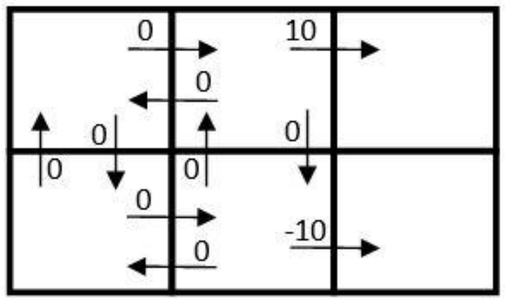
- Diberikan *GridWorld* dengan (3,2) dan (3,1) sebagai states terminal, nilai-nilai awal $Q(s,a)$ serta nilai-nilai reward $R(s,a)$ sebagai berikut:



Grid World



$Q(s,a)$initial values



$R(s,a)$– immediate reward

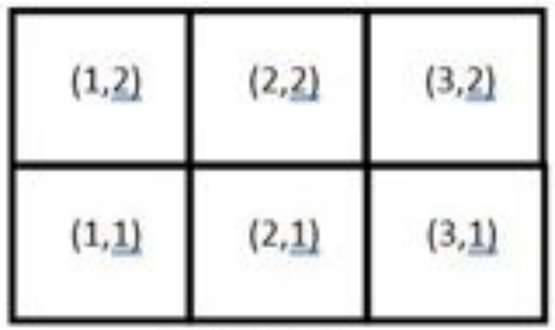
Jika *learning rate* α=0.5 dan *discounted factor* γ=0.9, untuk setiap episode berikut hitung nilai Q yang bersesuaian setiap kali agen berpindah dari satu state ke state lain. Gunakan policy pemilihan aksi berdasarkan urutan sekuen state yang diberikan pada episode tersebut. Selanjutnya rangkum nilai-nilai Q setelah satu episode selesai dalam *GridWorld.*

- Episode-1: (1,1) ⇒(1,2) ⇒(2,2) ⇒ (3, 2)
- Episode-2:(1,1) ⇒(2,1) ⇒(3,1)
- Episode-3: (1,1) ⇒(2,1) ⇒(2,2) ⇒ (3, 2)

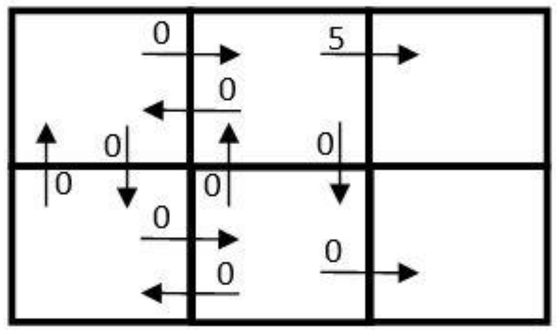
SARSA

- Episode-1: (1,1) ⇒(1,2) ⇒(2,2) ⇒ (3, 2)

Step	
(1,1) ↑	
(1,2) →	
(2,2) →	



Grid World

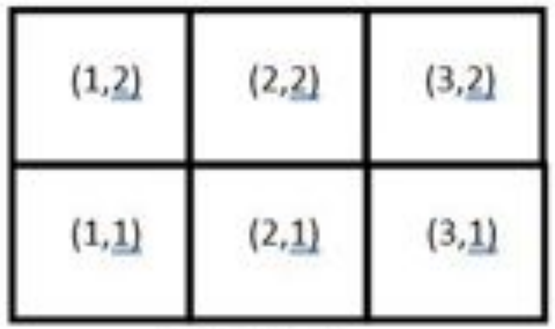


$Q(s, a)$ values after episode-1

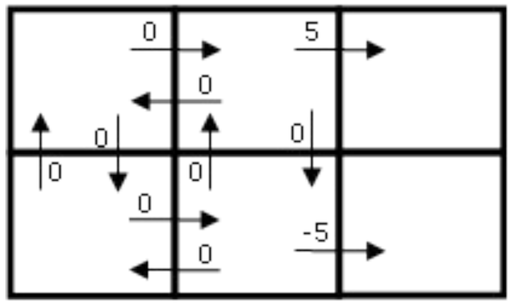
SARSA

- Episode-2: (1,1) ⇒(2,1) ⇒(3,1)

Step	
(1,1) →	
(2,1) →	



Grid World

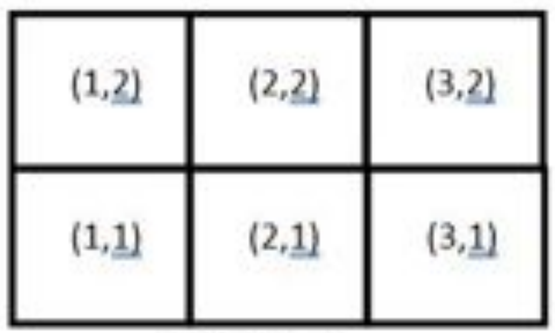


$Q(s,a)$ values after episode-2

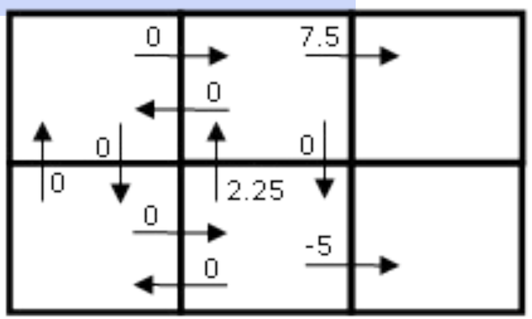
SARSA

- Episode-3: (1,1) ⇒(2,1) ⇒(2,2) ⇒ (3, 2)

Step	
(1,1) →	
(2,1) ↑	
(2,2) →	



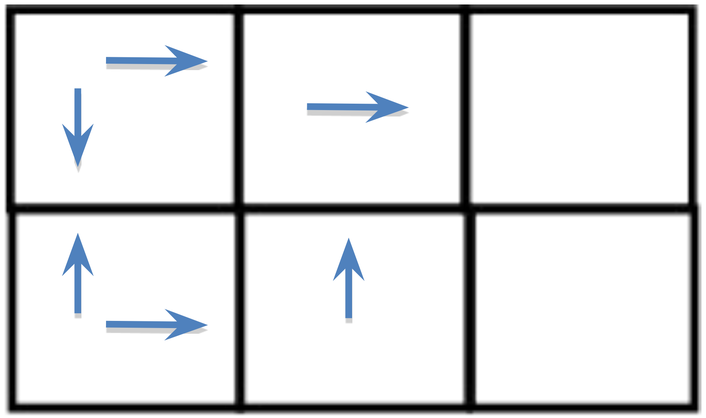
Grid World



$Q(s,a)$ values after eposide-3

SARSA

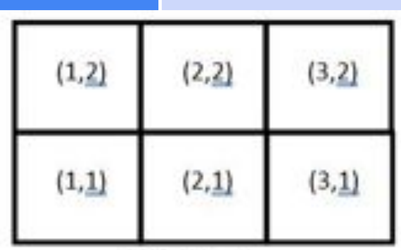
- Optimal Policy SARSA:



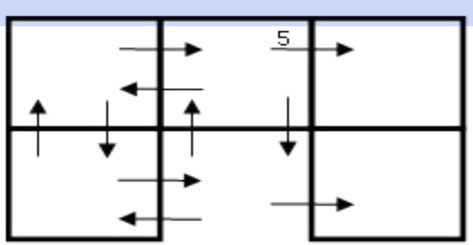
TD Q-Learning (Off-Policy)

- Episode-1: (1,1) ⇒(1,2) ⇒(2,2) ⇒ (3, 2)

Step	
(1,1) ↑	
(1,2) →	
(2,2) →	



Grid World

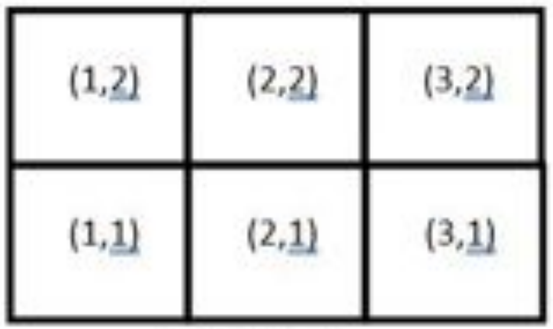


$Q(s,a)$ values after episode-1

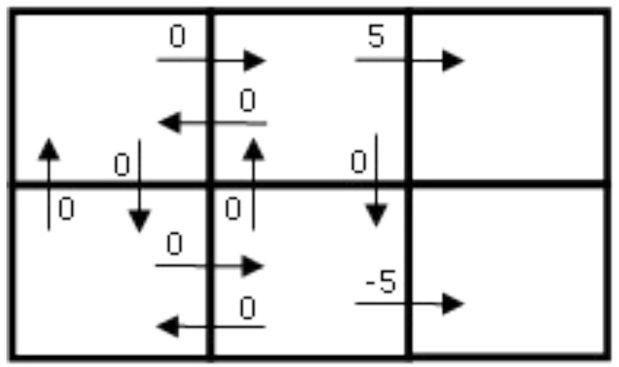
TD Q-Learning (Off-Policy)

- Episode-2: (1,1) ⇒(2,1) ⇒(3,1)

Step	
(1,1) →	
(2,1) →	



Grid World

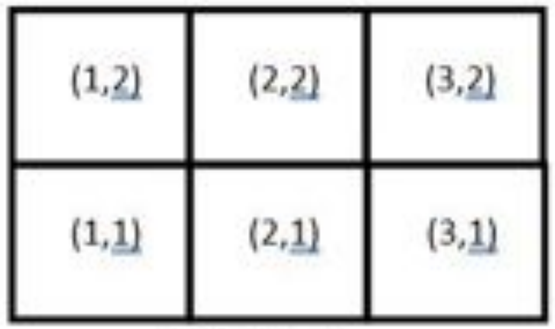


$Q(s,a)$ values after episode-2

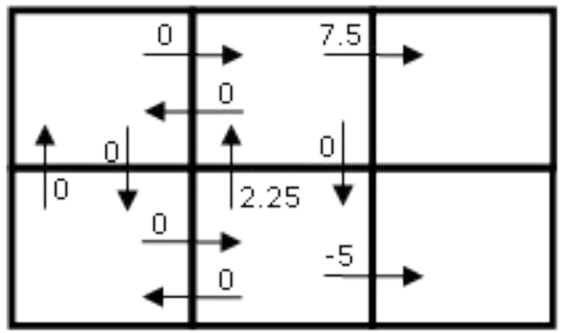
TD Q-Learning (Off-Policy)

- Episode-3: (1,1) ⇒(2,1) ⇒(2,2) ⇒ (3, 2)

Step	
(1,1) →	
(2,1) ↑	
(2,2) →	



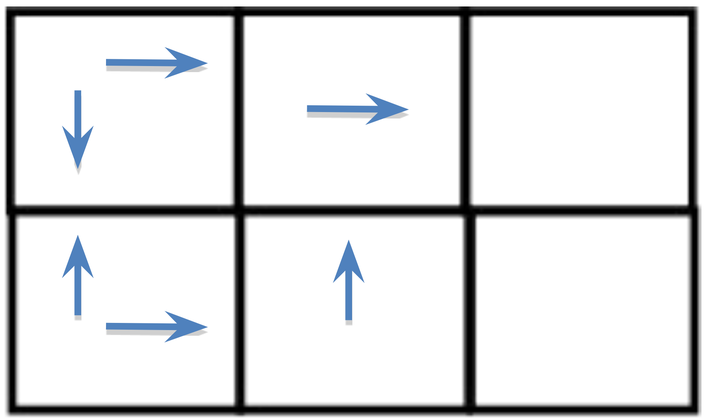
Grid World



$Q(s,a)$ values after eposide-3

TD Q-Learning (Off-Policy)

- Optimal Policy :



A glance at Deep Q-Learning

Source: Rascha, Machine Learning with Pytorch and Scikit Learn, Chapter 19

Deep Q-Learning

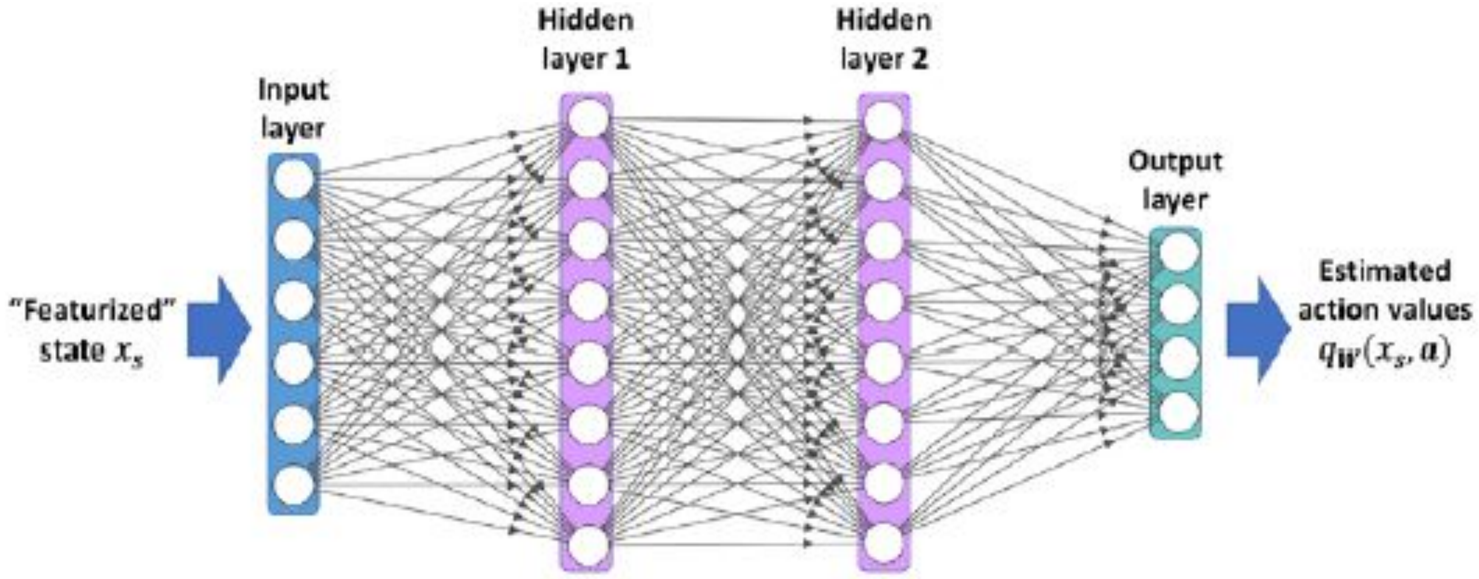
- sometimes the number of states can get very large, possibly almost infinitely large
- we may be dealing with a continuous state space instead of working with discrete states
- some states may not be visited at all during training, which can be problematic when generalizing the agent to deal with such unseen states later

→ Tabular format $v(S_t)$ or $q(S_t, A_t)$ is not sufficient to store the values

→ We can use function approximation approach

When approximation function, $q_w(x_s, a)$ [where x_s is a set of input feature (or "featured" states)] is deep neural network, the resulting model is called a deep Q-network (DQN)

Deep Q-Network (DQN)



- Weights are updated using Q-Learning algorithm
- Choose action from Q learning is modified to a function that performs a forward pass of the NN to compute the action values
- DQN:
  - reply memory
  - determining the target value for computing loss

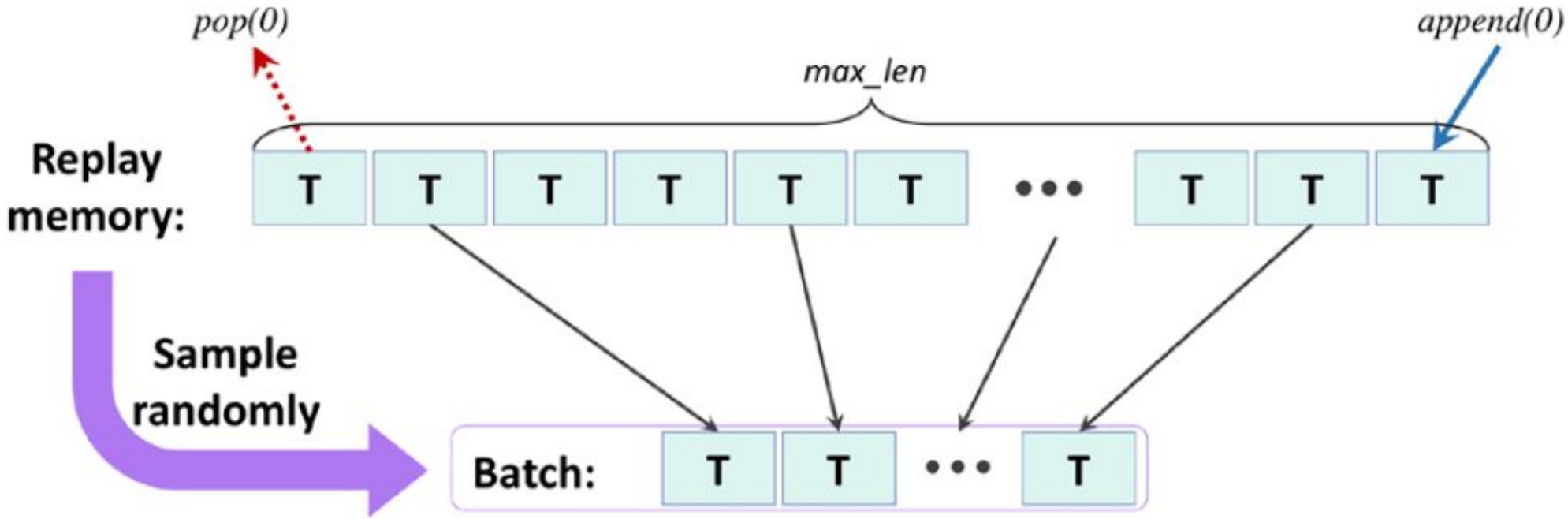
Replay Memory

- updating the weights for a state-action pair will likely affect the output of other states
- Multiple epoch is not feasible, since the episodes will change during the training and some states that were visited in the early stages of training will become less likely to be visited later
- the samples taken from an episode of the agent are not IID, as they form a sequence of transitions

 generates a transition quintuple T: ((x_s, a, r, $x_{s'}$,done)

Mini-batch examples is randomly selected from memory buffer (the memory size is bounded), which will be used for computing the loss and updating the network parameters

Replay Memory Illustration



Determining the target value for computing loss

- we perform two forward passes of the DQN model
- The first forward pass uses the features of the current state (x_s) → obtained $q_w(x_s, :)$
- The second forward pass uses the features of the next state ($x_{s'}$) → obtained $q_w(x_{s'}, :)$

Note: $q_w(x_s, :)$ → a vector of Q-values for all actions in *A*

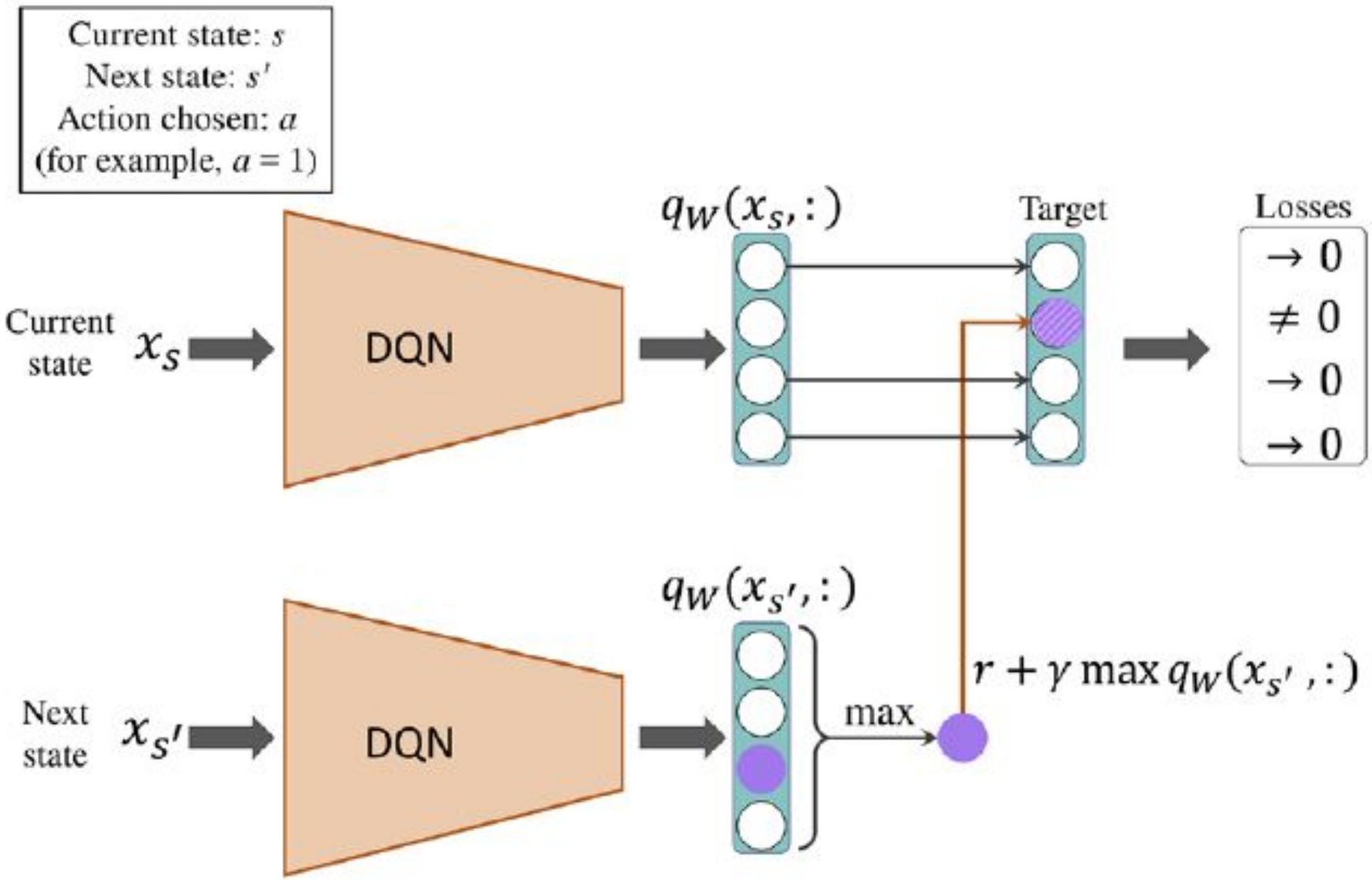
We treat this as a regression problem, using the following three quantities:

a) The currently predicted values, $q_w(x_s, :)$
b) The target value vector as described
c) The standard mean squared error (MSE) loss function

The losses will be zero for every action except for *a;* the computed loss will be

backpropagated to update the network parameters

Determining the target value for computing loss



Thank you