Deep Reinforcement Learning

Course Outline

Part 1: Reinforcement Learning

1) Introduction
2) Markov Decision Process(MDP)
3) Dynamic Programming
4) Model-free Prediction
5) Model-free Control

Part 2: Deep Reinforcement Learning

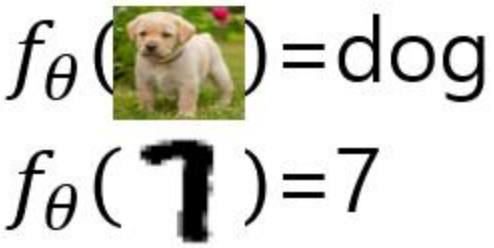
6) Value Function Approximation
7) Deep Neural Network and Gradient Method
8) Deep Q Network
9) Policy Gradient
10) Advanced Policy Gradient
11) Imitation Learning
12) Recent Trends in Reinforcement Learning

Contents of Chapter 1

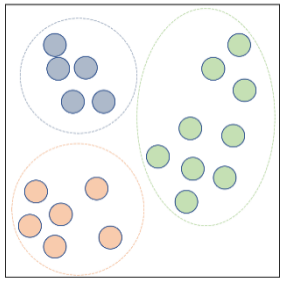
- Definition of reinforcement learning
- Vocabularies of RL
  - State
  - Action
  - Reward and return
  - Policy
  - Value function
  - Model
  - Model-based vs Model-free
  - Exploration vs Exploitation
  - Prediction vs Control
  - On policy vs off policy
  - Observability
- Supervised Learning vs RL
- Examples of RL

Learning Methods

- Supervised learning
  - classification, regression
  - $(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n)$
  - $t = f_\theta(x)$



- Unsupervised learning
  - clustering



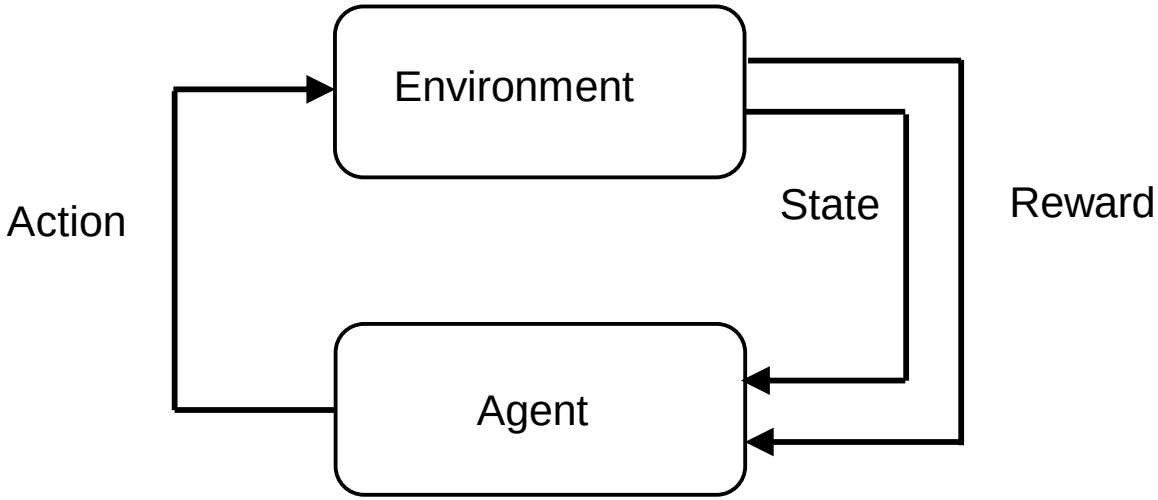
- Reinforcement learning
  - learn from interaction w/ environment to achieve a goal
  - learn the sequence of optimal actions to achieve a goal

Reinforcement Learning

- Reinforcement learning(RL) : An area of machine learning concerned with how intelligent agents find *optimal actions* in an environment in order to achieve its goals.
  - e.g.: mobile robot, optimize operations in factories, learning to play board games

- Each time the agent performs an action, its environment may provide a reward/penalty to indicate the desirability of the resulting state

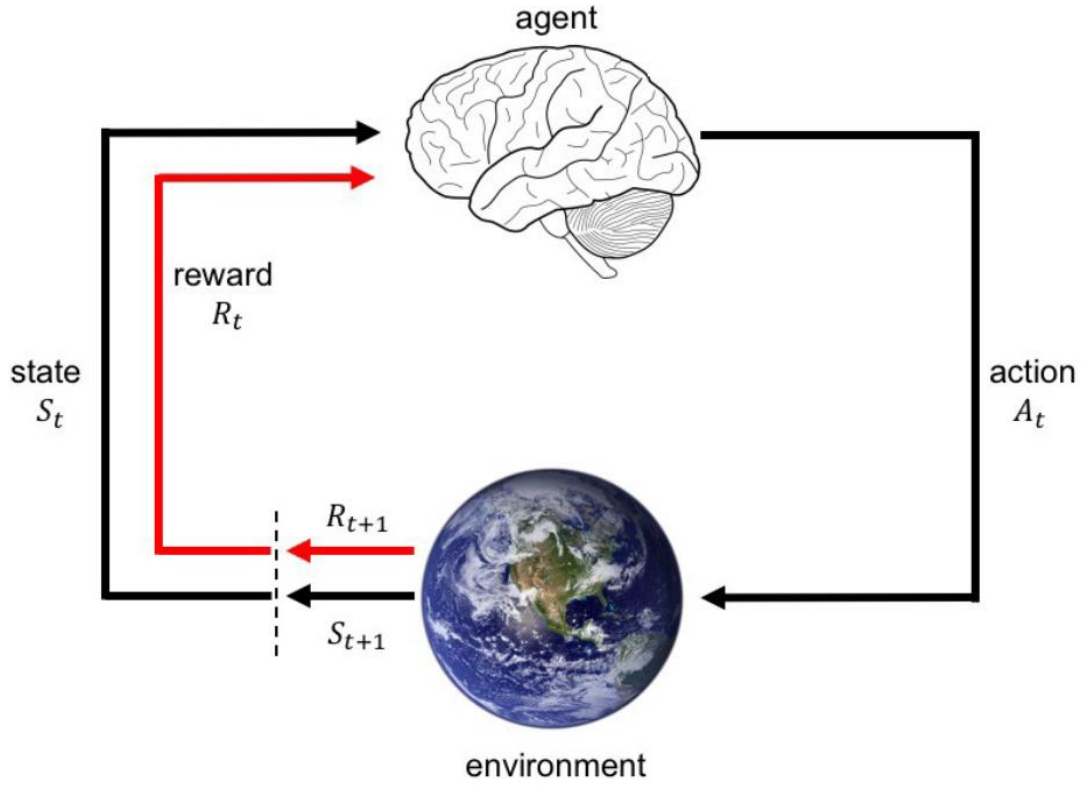
- Learn successful action policies by experimenting in their environment
  - Learn from trial and error

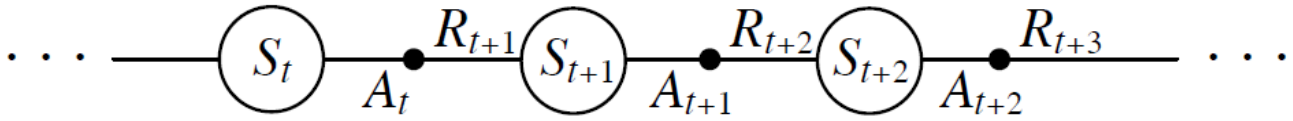


Reinforcement Learning(RL)

- Agent and environment interact at discrete time steps: t = 0, 1, 2, …, K

  Agent
  - observes state at step t : S_t
  - chooses action at step t : A_t
  - gets resulting reward : R_{t+1}
  - and resulting next state : S_{t+1}





Goal: Learn to choose actions that maximize future rewards

$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$, where $0 \leq \gamma \leq 1$

Characteristics of Reinforcement Learning

- What makes reinforcement learning different from other machine learning paradigms?
  - There is no supervisor, only a reward signal
  - Feedback is delayed, not instantaneous
  - Time really matters (sequential, non i.i.d data)
  - Agent's actions affect the subsequent data it receives

Vocabularies of RL

- Define some important RL terms:
  - State
  - Action
  - Reward (and return)

  } input

  - Policy (policy-based RL)
  - Value function (value-based RL)
  - Model (model-based RL)

  } output

  - Exploration and exploitation
  - Model-based and model-free
  - Prediction and control
  - On-policy and off-policy

Rewards

- A reward R is a scalar feedback that indicates how well agent is doing
  - e.g., 1 if goal is achieved, 0 otherwise
  - Credit assignment problem
  - $R(s,a)$ or $R(s)$
- Reward Hypothesis: All goals can be described by the maximization of expected cumulative reward.
- Rewards specify *what* the agent needs to achieve and RL specifies *how* the agent achieves the goals
- The simplest and cheapest form of supervision, and surprisingly general:
- Common assumption: *stationary* reward function (rules in env don't change)
  - $R(s,a)$ (or $R(s)$) is the same with time

Reward Examples

- Flipping a pancake:
  - Pos. reward: catching the 180 rotated pancake
  - Neg. reward: droping the pancake on the floor
- Stock trading:
  - Trading portfolio monetary value
- Playing Atari games:
  - Highscore value at the end of a game episode
- Driving an autonomous car:
  - Pos. reward: getting save from A to B without crashing
  - Neg. reward: hitting another car, pedestrian, bicycle,...
- Classical control task (e.g inverted pendulum, electric drive):
  - Pos. reward: following a given reference trajectory precisely
  - Neg. reward: violating system constraints and/or large control error

Returns

- Return is a function of reward sequence, which can be:
  - simple sum of rewards (also called cumulative reward)
  - sum of discounted rewards (also called discounted cumulative reward)

- Goal-seeking behavior of an agent can be formalized as the behavior that seeks maximization of the expected value of the cumulative sum of (potentially time discounted) rewards, we call it return.

- We want to maximize returns.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \text{ where } 0 \leq \gamma \leq 1$$

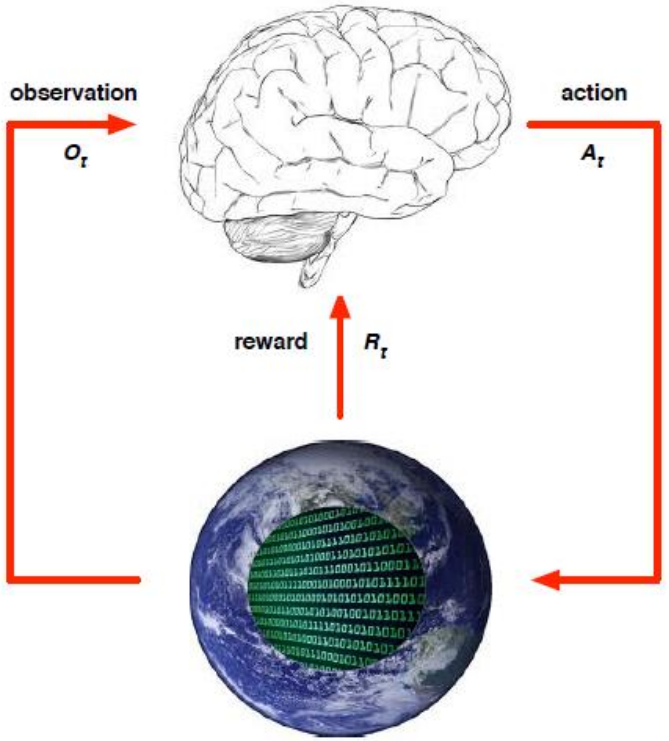
History and State

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

$$h_t = s_0, a_0, r_1, \dots, s_{t-1}, a_{t-1}, r_t, s_t, a_t, r_{t+1}$$

  - i.e. all observable variables up to time t

- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards



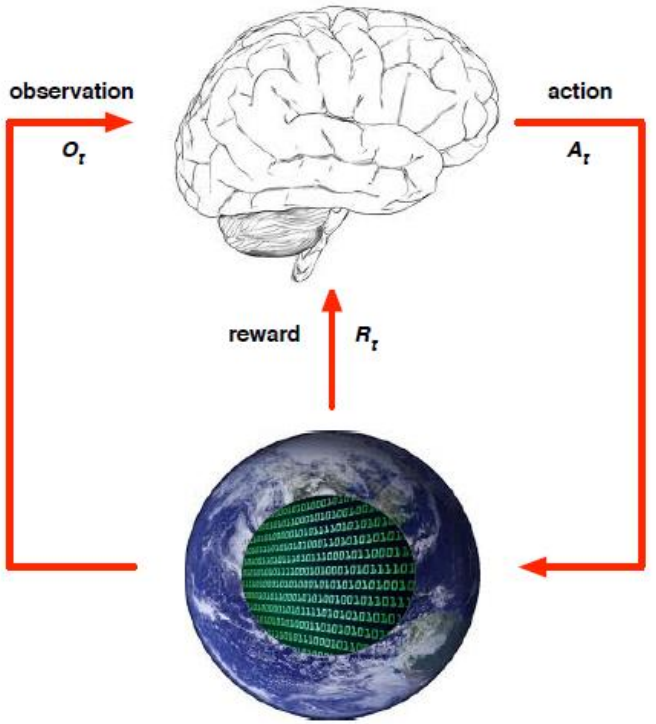
- *State* is the information used to determine what happens next
- Formally, state is a function of the history:

$$s_t = f(h_t)$$

State

1) Environment state

- The environment state is the environment's private representation
  - Whatever data the environment uses to pick the next observation/reward
- Internal status representation of the environment, e.g
  - Physical states e.g. car velocity or motor current
  - Game states e.g. current chess board situation
  - Financial states e.g. stock market status
- Often hidden or unknown to agent
- Even if known, may contain information not needed by agent



2) Agent state

- Internal status representation of the agent
  - Whatever information the agent/algorithm uses to pick the next action

Information/Markov State

- If the current state contains all useful information from the history it is called an information state (a.k.a. Markov state)

- Definition: A state S_t is called information (Markov) state if and only if

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots, S_{t-1}, S_t]$$

  - History is fully condensed in S_t, i.e. S_{t+1} is only depending on S_t
  - A given system can be fully described by S_t
  - i.e. The state is a sufficient statistic of the future

- Why is Markov assumption popular?
  - In practice, often assume most recent observation is sufficient statistic of history

Actions

- Actions are the agent's methods which allow it to interact change its environment, and thus transfer between states.
- Every action performed by the agent yields a reward from the environment.
- The decision of which action to choose is made by the policy.

- Major distinction:
  - Finite number of actions
  - Infinite number of actions

- Examples:
  - Drive an autonomous car
  - Buy a stock for your trading portfolio

Policy

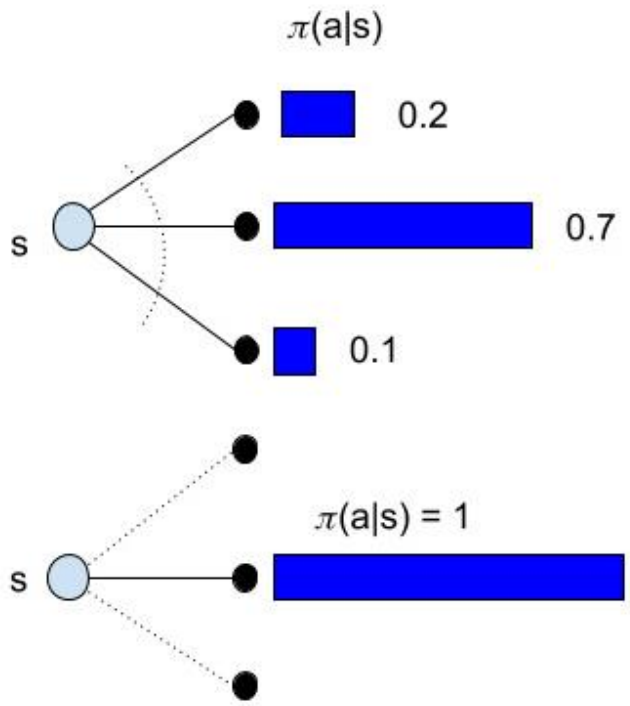
- Policy π determines how the agent chooses actions
- If the agent is following policy π at time t, then π(a|s) is the probability that $A_t = a$ if $S_t = s$.

  - Stochastic policy: a probability distribution over actions

$$\pi(a|s) = P[A_t = a \,|S_t = s]$$

  - Deterministic policy: probability of only one action is 1 (special case of stochastic policy)

$$a = \pi(s)$$



Policy

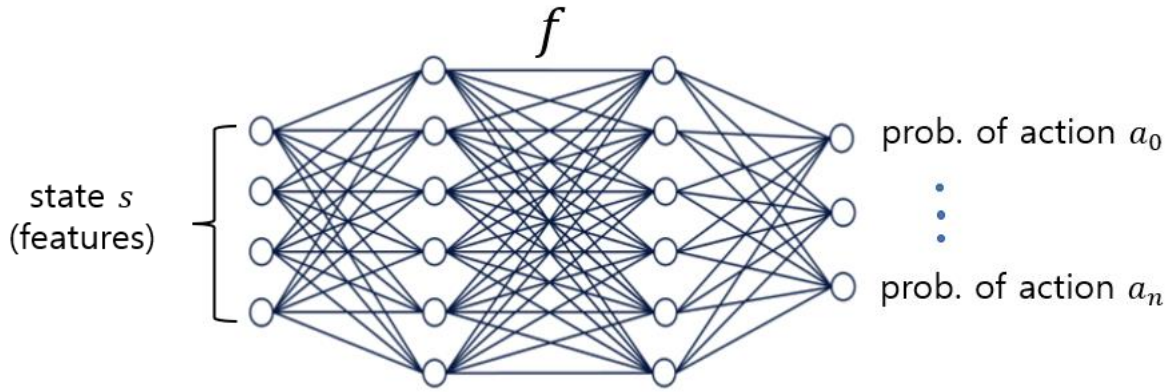
- Representation of policy

Tabular

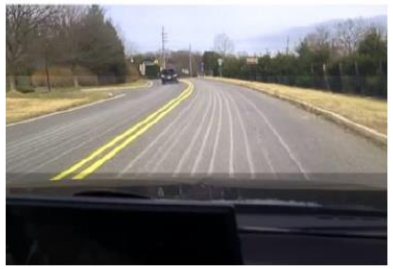
action(a)

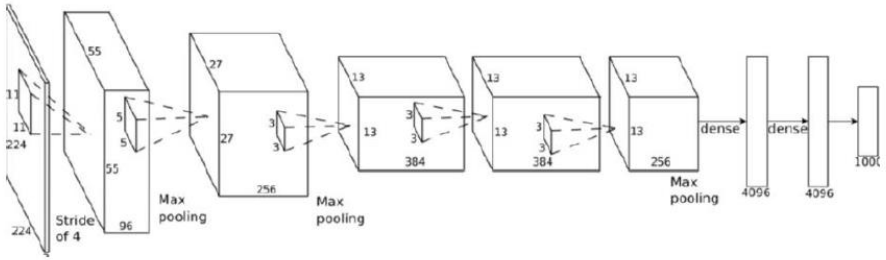
state(s) / $\pi(a\|s)$	a_0	a_1	a_2	a_3
s_0	0.2	0.3	0.1	0.4
s_1	0	1.0	0	0
...				
s_m	0.1	0.5	0	0.4

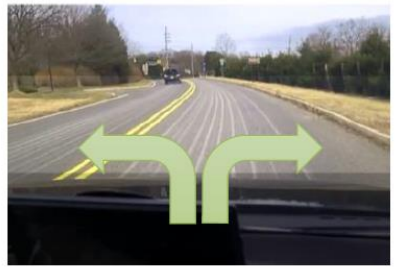
Function(neural network)



- Policy can be a shallow or a deep function mapping







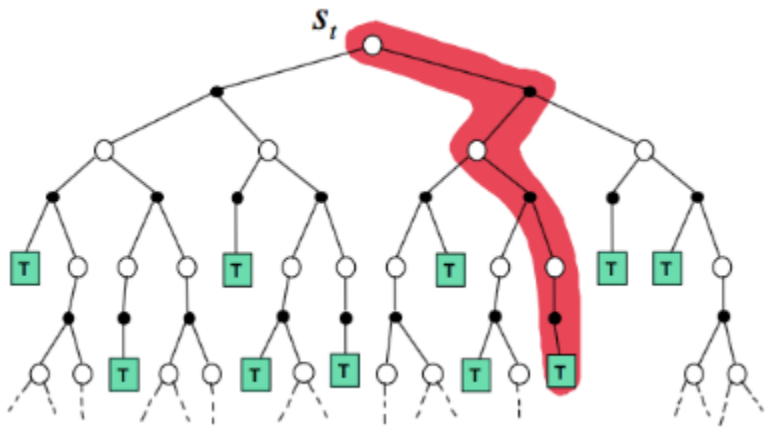
Value Function

- Functions of states (or state-action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state) under a policy π
  - *State-value* (v(s) value) function and *action value* (q(s,a) value) function
  - Many reinforcement learning algorithms are based on estimating value functions

- *State-value* function $v_\pi(s)$: expected discounted sum of future rewards in state s under a particular policy π

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | s]$$

  - Discount factor γ
  - Used to evaluate the goodness/badness of states and therefore to select between actions



Value Function

- The *action-value* function $q_\pi(s,a)$ is the expected return being in state *s* taken an action *a* and, thereafter, following a policy π

- Assuming a MDP problem structure, the action-value function is

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | s,a]$$

- Optimal value functions: maximum return instead of expected return
  - $v_*(s) = \max_\pi v_\pi(s)$
  - $q_*(s,a) = \max_\pi q_\pi(s,a)$

- Value function can also be a deep function mapping (value network)

Value Function

- Representation of value function

Tabular

state(s)

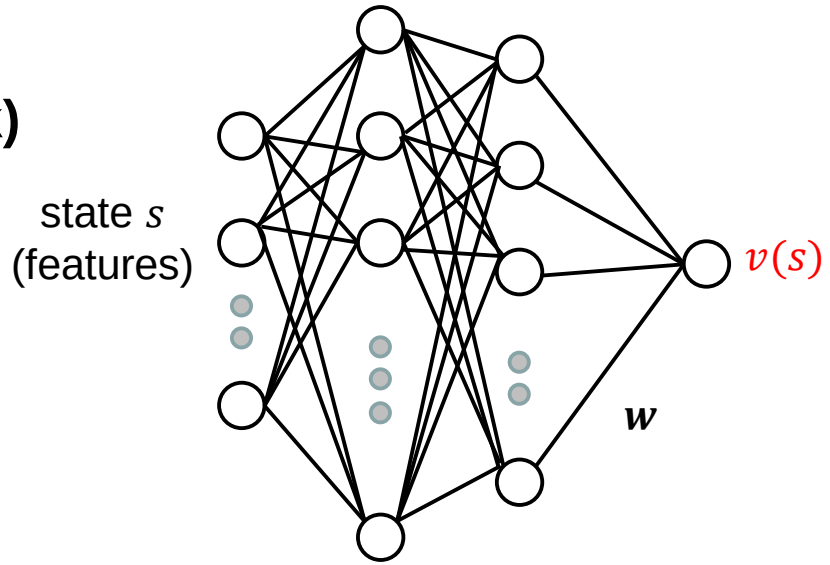
	$v(s)$
s_0	1.2
s_1	0.4
...	
s_m	2.3

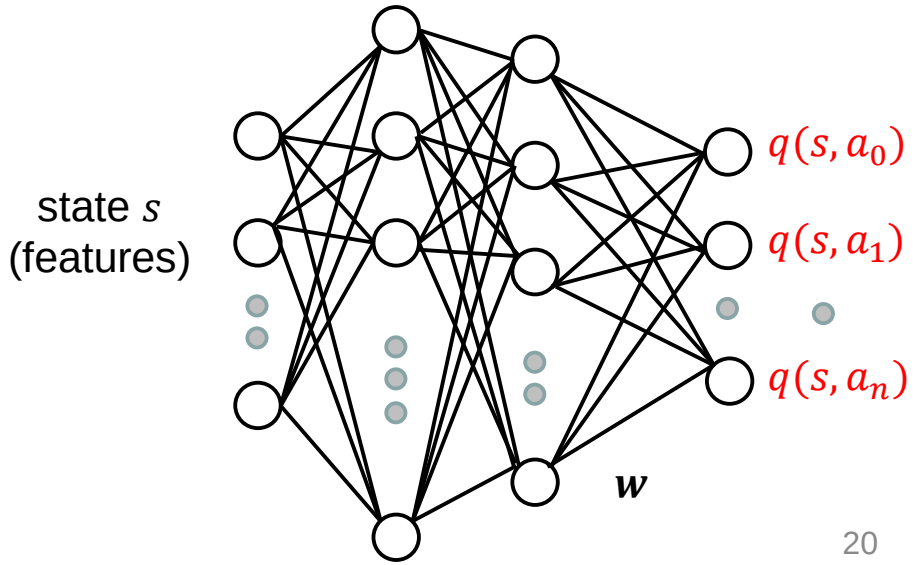
action(a)

state(s)

$q(s,a)$	a_0	a_1	a_2	a_3
s_0	0.5	0.3	0.2	0.4
s_1	1.5	1.0	0.6	0.8
...				
s_m	0.6	0.5	1.1	0.4

Function (neural network)





Optimal Value Function

- Optimal value function is the maximum value that can be obtained from a given state or action, *regardless of* the policy.
- Two types of optimal value functions: optimal state value function and optimal action value function.
- The optimal state value function is the maximum state value that the agent can achieve in the future from its current state, regardless of policy.
- The optimal state value is expressed as $v_*(s)$ which is the value at which $v_\pi(s)$ is maximized regardless of any policy.

$$v_*(s) = \max_\pi v_\pi(s)$$

- The optimal action value is also the maximum future action value that the agent can achieve given its current state, regardless of policy.
- The optimal action value is denoted by $q_*(s,a)$ which is the value at which $q_\pi(s,a)$ is maximized, regardless of policy.

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Model

- Agent's representation of how world changes given agent's action
- A model predicts what the environment will do next
  - Transition model(prob.) predicts next agent state

$$P_{ss'}^{a} = P[S_t = s' | S_t = s, A_t = a]$$

    - *P* predicts the next state (transition prob.)

  - Reward model predicts immediate reward

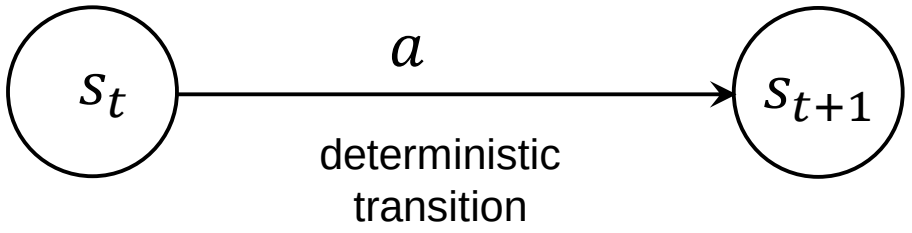
$$R(s,a) = R_s^a = E[R_{t+1} | S_t = s, A_t = a]$$

    - *R* predicts the next (immediate) reward

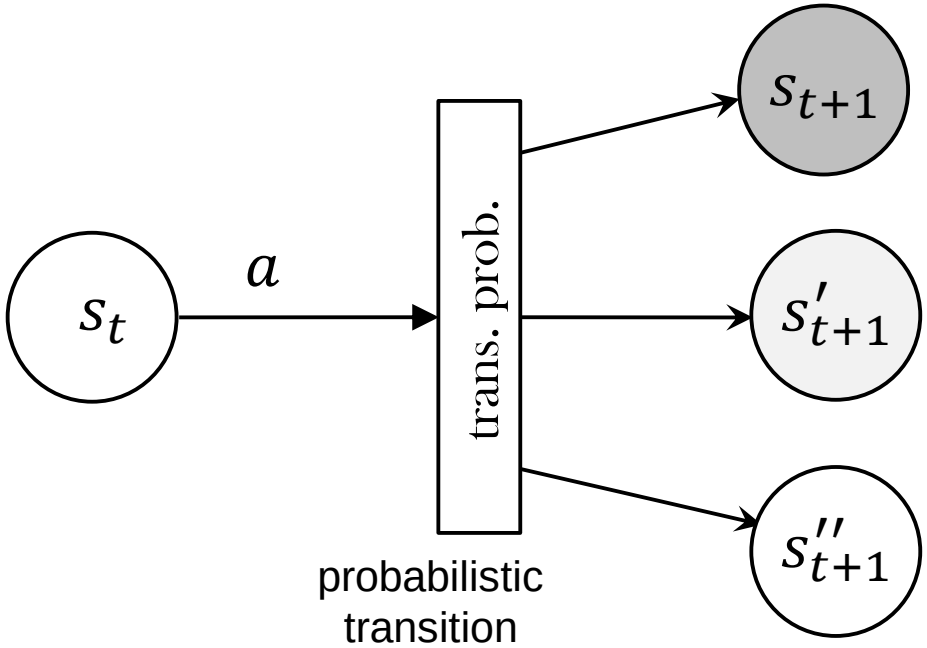
Transition Model

- In general, those models could be stochastic but in some problems relax to a deterministic form.

Deterministic transition



Probabilistic/Stochastic Transition



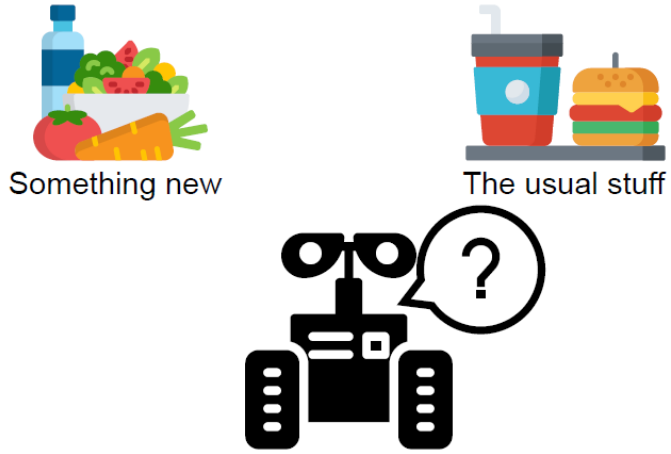
Model-free versus Model-based

- Given a state and an action to be taken, the model could predict the next state and the next reward
- *Model-based* methods use models and planning.
  - Have complete knowledge of environment
  - Transition prob. and rewards are given
- *Model-free* methods learn exclusively from trial-and-error (i.e. no modeling of the environment)
  - No transition prob. and no rewards are known
- Many RL algorithms mainly focus on model-free methods

Exploration and Exploitation

- Reinforcement learning is like trial-and-error learning
- Should discover a good policy from its experiences of the environment

- In most RL, the environment is initially unknown(model-free). How to find optimal policy?
  - *Exploration*: Find out more about the environment.
  - *Exploitation*: Exploits known information to maximize reward
    - Estimate value functions more correctly



- It is usually important to explore as well as exploit
- Trade-off problem: what's the best split between both strategies?

Finite Horizon

- When *h* (total number of steps) is finite
- Non-stationary optimal policy
- Best action different at each time step
- Intuition: best action varies with the amount of time left

Infinite Horizon

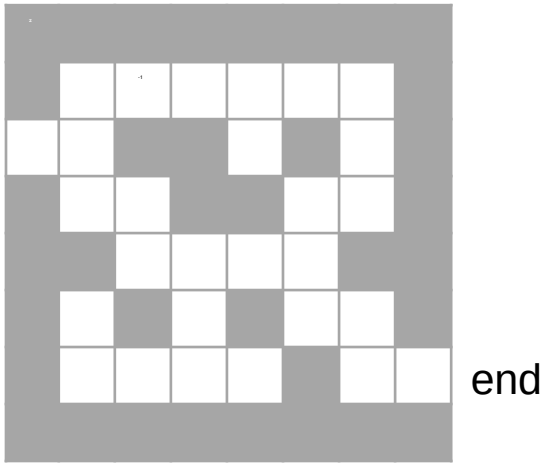
- When *h* is infinite
- Stationary optimal policy
- Same best action at each time step
- Intuition: same (infinite) amount of time left at each time step, hence same best action
- Problem: value iteration does an infinite number of iterations

Prediction and Control

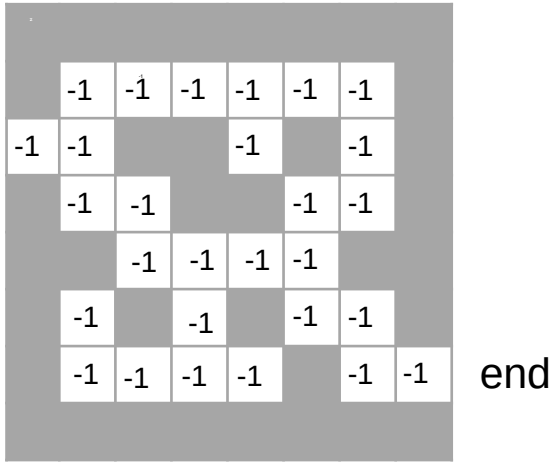
- Prediction: evaluate the performance of a certain policy
  - Given a policy, compute state (or action) values of the policy

- Control: Find the optimal state(action) values and/or optimal policy
  - optimize the future

MAZE Example

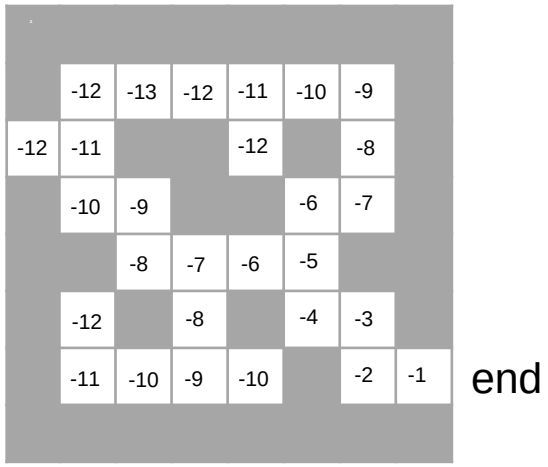


- Actions: N, E, S, W
- States: Agent's location

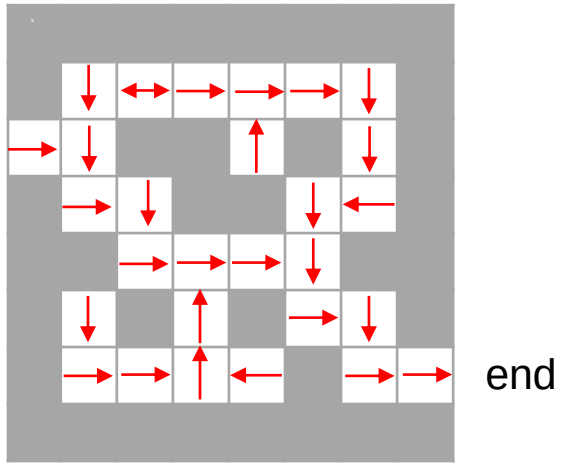


- Model
  - Grid layout represents transition model $P_{ss'}^{a}$,
  - Rewards: how much reward from each state (-1 per time-step)
  - Numbers represent immediate reward R_{s}^{a} from each state s (same for all a)

MAZE Example



- Value function: Numbers represent value $V_\pi(s)$ of each state s



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

On-policy vs Off-policy

- An important distinction in RL methods about on-policy vs. off-policy learning
- In the *on-policy* setting (e.g. SARSA algorithms), you are learning the value of the policy π that you are following
  - Target policy = behavior policy
- In the *off-policy* setting (e.g. Q-learning algorithm) you are learning the value of a *different* policy than the one you are following
  - Target policy ≠ behavior policy
- Virtually all learning guarantees apply to the on-policy case; there exist some convergence proofs for off-policy case, but only for very limited cases

Degree of Observability

- **Full observability**
  - Agent directly observes environment state (e.g. environment state = agent state)
  - If agent state is Markov: Markov decision process (MDP).
  - Majority of this course

- **Partial observability**
  - Agent indirectly observes environment state (e.g. environment state ≠ agent state)
  - If agent state is Markov: partial observable MDP (POMDP).
  - Agent may reconstructs state information (belief, estimate).
    - Technical systems with limited sensors (cutting costs)
    - Poker game (unknown opponents' cards)
    - Human health status

Reinforcement learning vs. Supervised learning

- RL is a form of active learning:
  - The agent gets the chance to collect her own data by acting in the world, querying humans, and so on.
  - The data changes over time, it depends on the policy of the agent.
  - To query the environment effectively, the agent needs to keep track of its uncertainty

- Supervised learning is a form of passive learning:
  - The data does not depend on the agent in anyway, it is provided by external labelers.
  - The data is static throughout learning.
  - In some sense, supervised learning is a special case of RL: each data contains rewards

Approaches to Reinforcement Learning

- Value-based RL
  - Estimate the optimal value function $v_*(s)$ or $q_*(s,a)$
  - This is the maximum value achievable under ANY policy
  - No policy

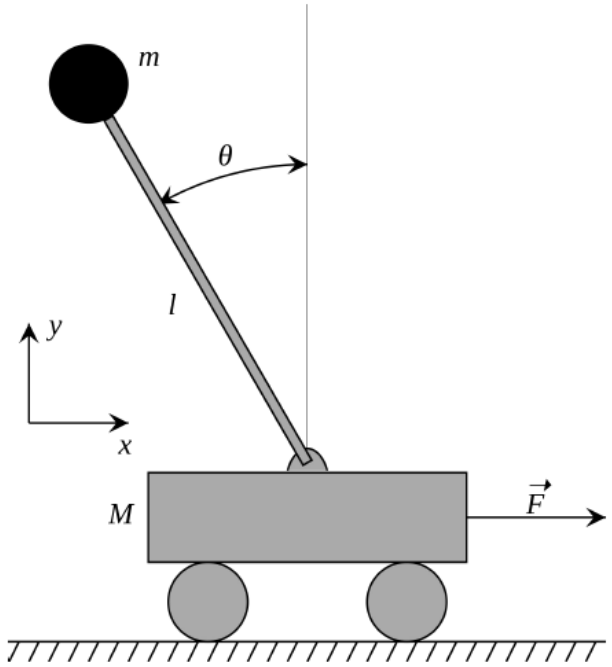
- Policy-based RL
  - Search directly for the optimal policy π^*
  - This is the policy achieving maximum future reward
  - No value function

- Actor Critic
  - Policy Function
  - Value Function

Examples of RL

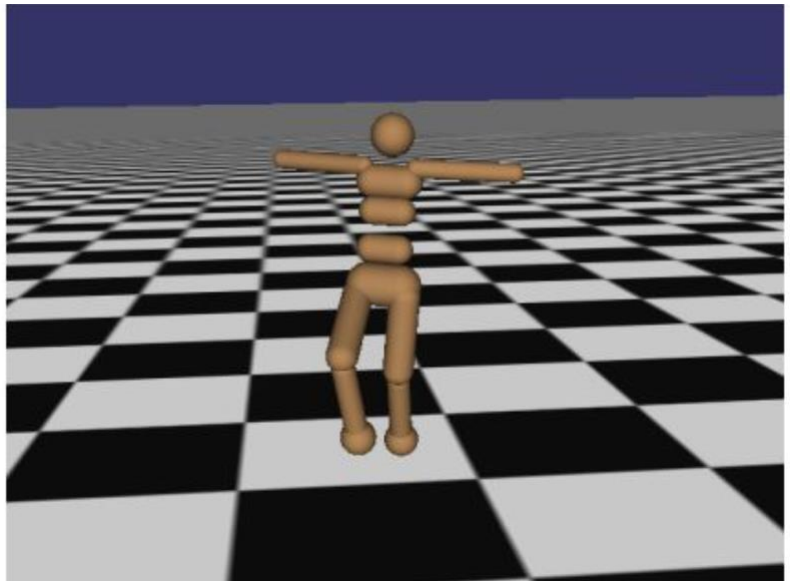
Cart-Pole Problem

Objective: Balance a pole on top of a movable cart
State: angle, angular speed, position, horizontal velocity
Action: horizontal force applied on the cart
Reward: 1 at each time step if the pole is upright



Robot Locomotion

Objective: Make the robot move forward
State: Angle and position of the joints
Action: Torques applied on joints
Reward: 1 at each time step upright + forward movement

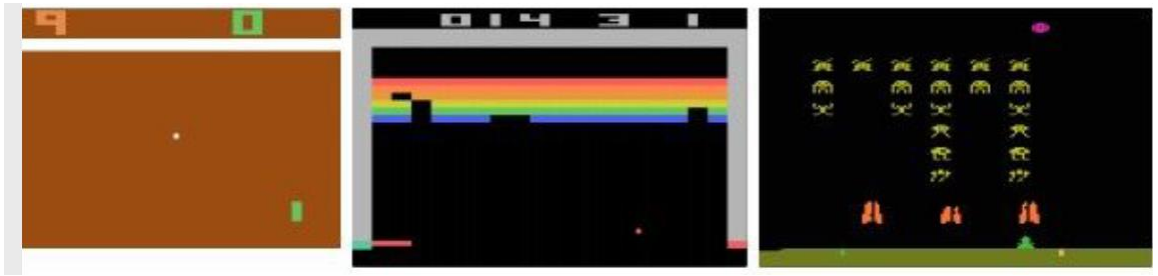


Source: cs231n Reinforcement learning, Stanford

Examples of RL

Atari Games

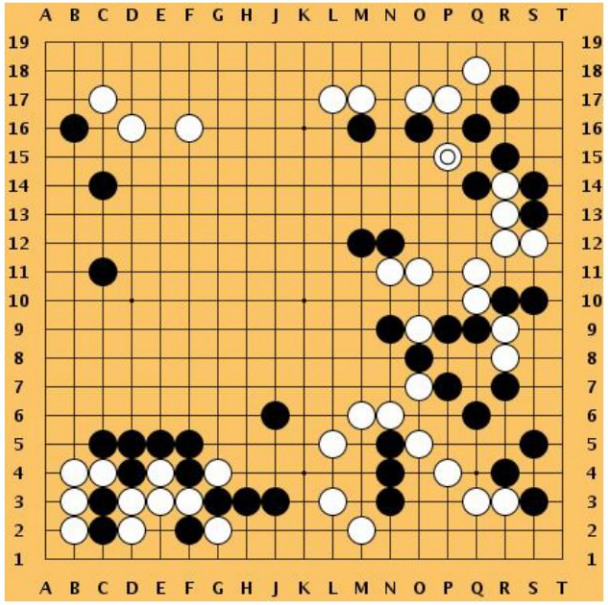
Objective: Complete the game with the highest score
State: Raw pixel inputs of the game state
Action: Game controls e.g. Left, Right, Up, Down
Reward: Score increase/decrease at each time step



Source: cs231n Reinforcement learning, Stanford

AlphaGo

Objective: Win the game!
State: Position of all pieces
Action: Where to put the next piece down
Reward: 1 if win at the end of the game, 0 otherwise



Examples of RL

Operations research

- Example: vehicle routing
- **Agent:** vehicle routing software
- **Environment:** stochastic demand
- **State:** vehicle location, capacity and depot requests
- **Action:** vehicle route
- **Reward:** - travel costs

Conversational agent

- **Agent:** virtual assistant
- **Environment:** user
- **State:** conversation history
- **Action:** next utterance
- **Reward:** points based on task completion, user satisfaction, etc.

Examples of RL

Computational Finance

- Example: how to purchase a large # of shares in a short period of time
- Automated trading
- **Agent:** trading software
- **Environment:** other traders
- **State:** price history
- **Action:** buy/sell/hold
- **Reward:** amount of profit