Computational Data Analysis

Machine Learning

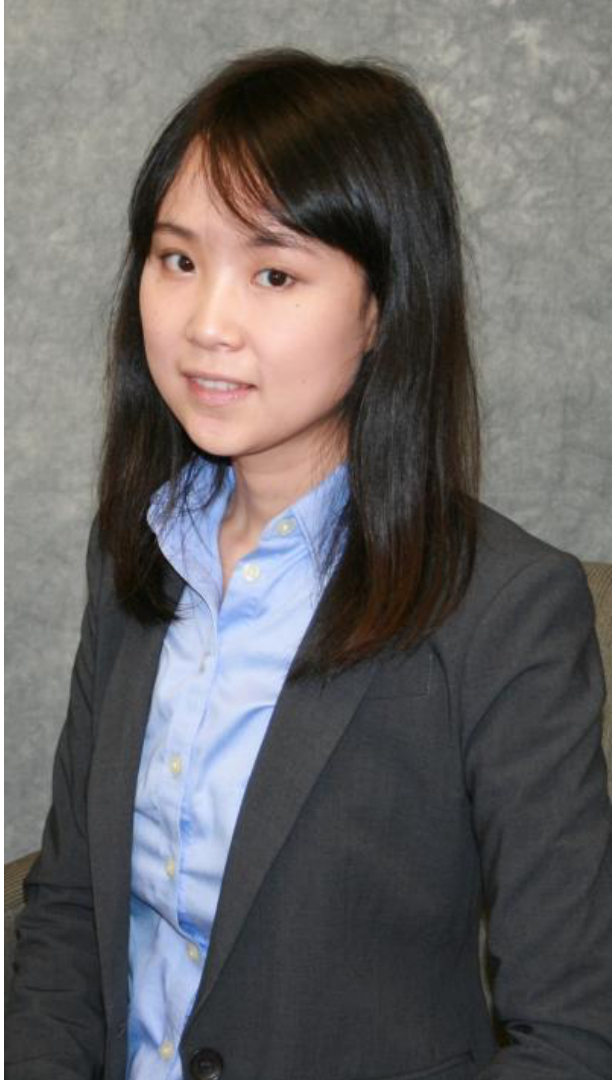
Yao Xie, Ph.D.

Associate Professor

Harold R. and Mary Anne Nash Early Career Professor

H. Milton Stewart School of Industrial and Systems Engineering

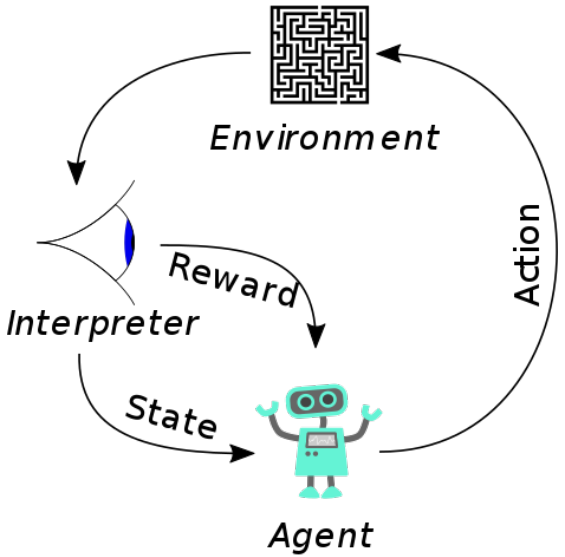
Introduction to reinforcement learning



Reinforcement learning (RL)

- Reinforcement learning (RL) is an area of machine learning concerned with how **agents** ought to take a **sequence** of **actions** in an environment in order to maximize the cumulative **reward**.

- Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

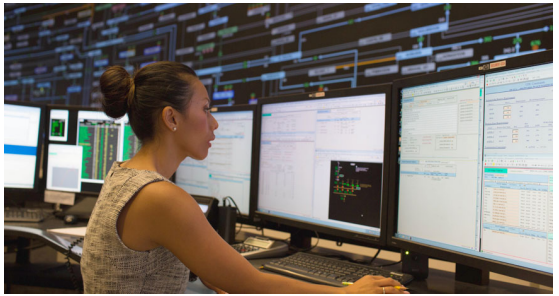


Examples of RL

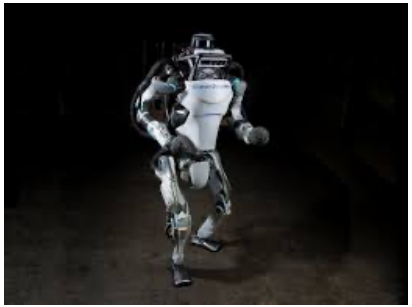
Self-driving car

Games: Chess/Go playing

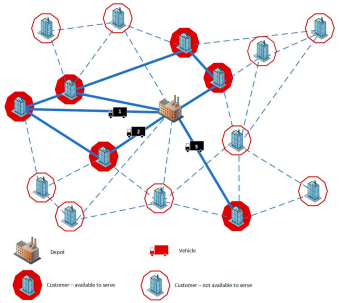
Dynamic portfolio management



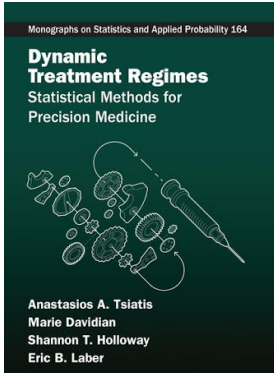
Robotics



Dynamic vehicles routing



Healthcare



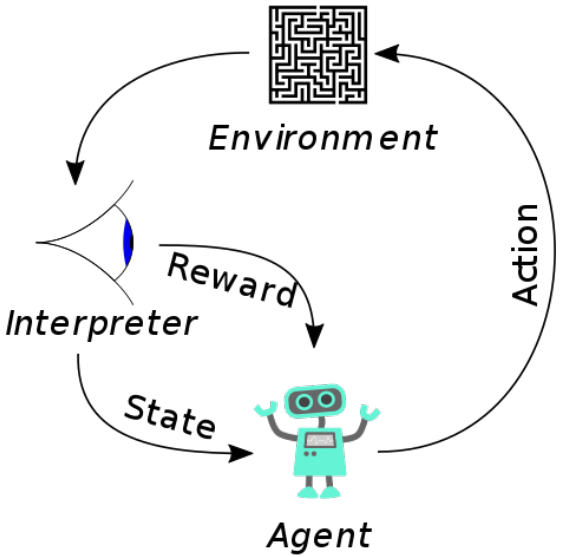
https://neptune.ai/blog/reinforcement-learning-applications

Characteristics of RL

- **Sequential** decision making:

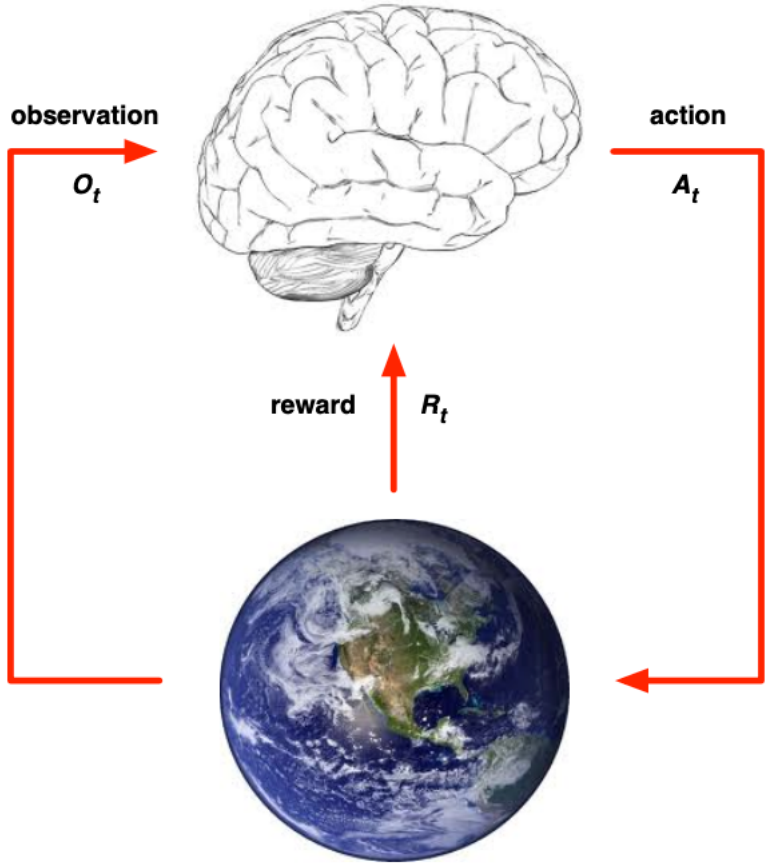
Goal: select actions to maximize total reward

- There is no supervisor, only a **reward** signal
- Feedback is delayed, not instantaneous (have to perform optimization by anticipating unknown "future")
- It may be better to sacrifice immediate reward to gain more long-term reward
- Time really matters (sequential, not i.i.d. data)
- Agent's actions affect the subsequent data

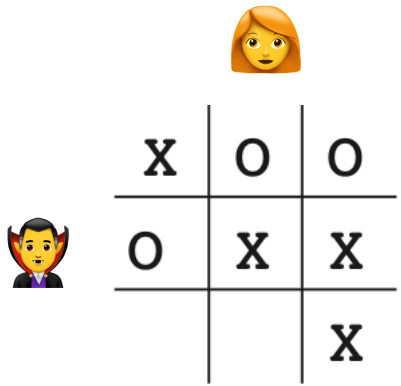


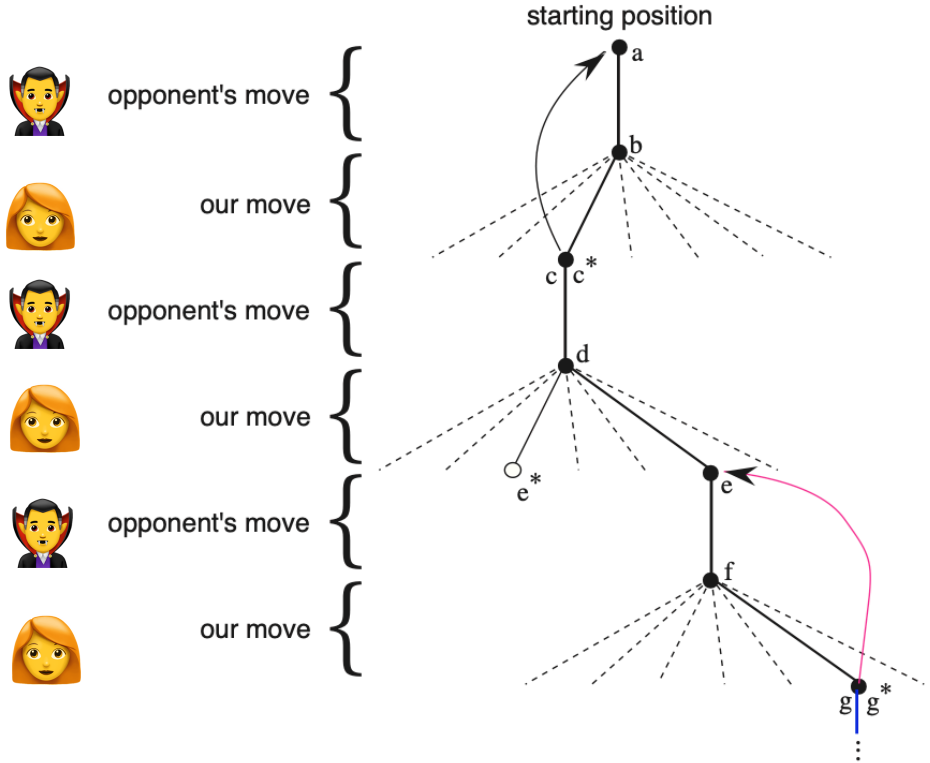
Agent and environment

- At each step t the agent
  - Execute action A_t
  - Receives observation O_t
  - Receives scalar reward R_t, indicates how well agent is doing at step t
- The environment
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar R_{t+1}
- Given observation and reward, the agent's job is to choose action to maximize cumulative reward



A sequence of tic-tac-toe game





History and state

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

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

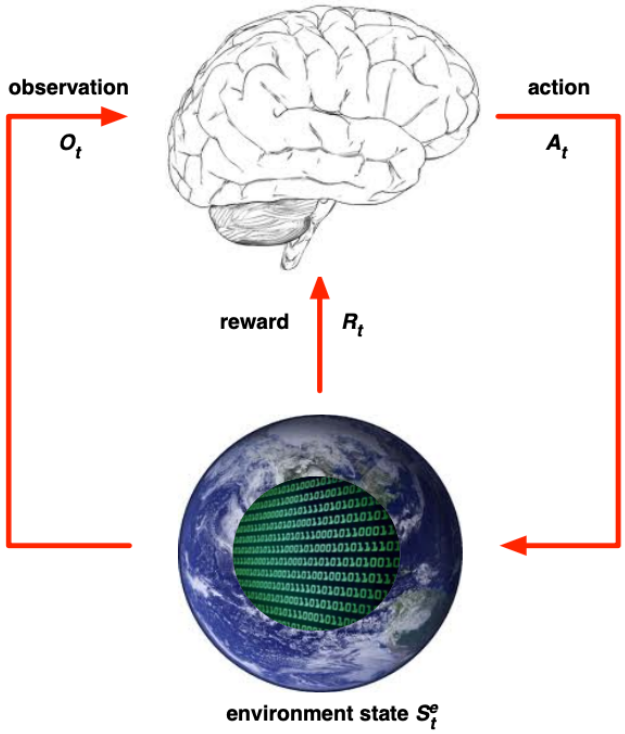
- What happens next depends on the history
  - The agent selects actions
  - The environment selects observations/rewards

- State is the information used to determine what happens next

$$S_t = f(H_t)$$

Environment state

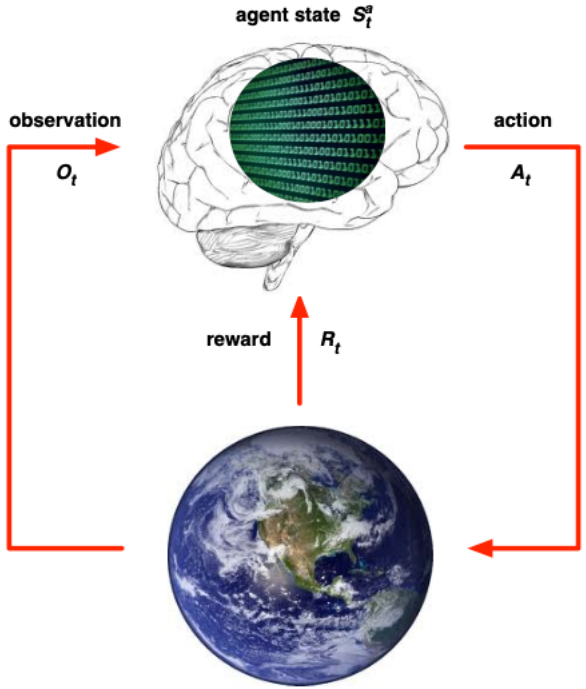
- The environment state S_t^e is the environment's private representation
- The environment state is not usually observable to the agent



Agent state

- The agent state S_t^a is the agent's internal representation
- i.e., whatever information the agent used to pick the next action
- It is the information used by reinforcement learning algorithm
- It can be any function of the history

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

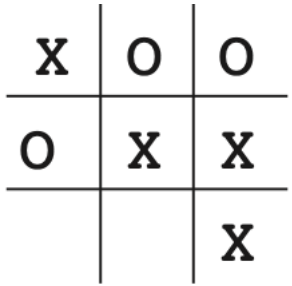


Information state

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

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

The future is independent of the past given the present



- Assume the environment state S_t^e is Markov
- Assume the history H_t is Markov

Fully observable environment

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)

Partial observability:

- Agent state ≠ environment state

- Formally, this is Partially Observable Markov Decision Process (POMDP)

- Agent construct its own belief:

$$S_t^a = \left(P(S_t^e = s^1), \dots, P(S_t^e = s^n)\right)$$

e.g., Recurrent neural networks (RNN):

$$S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$$

Major component of an RL agent

An 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: $\pi(a|s) = P(A_t = a|S_t = s)$

Model

- A model predicts what the environment will do next
- P predicts the next state
- R predicts the next (immediate) reward, e.g.,

$$R_{ss'}^{a} = P(S_{t+1} = s' | S_t = s, A_t = a)$$
$$R_{s}^{a} = E[R_{t+1} | S_t = s, A_t = a]$$

Value function

- Value function is a prediction of **future** reward (cost-to-go)
- Used to evaluate the goodness/badness of states
- Enable us to select between actions
- State-value function

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

- Action-value function

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

Bellman equation

The value function can be decomposed into immediate reward plus discounted value of the future

- Bellman equation for state-value function

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

Immediate reward cost-to-go

- Bellman equation for action-value function

$$q_\pi(s, a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

Optimization

- Optimal value function specifies the best possible performance

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

- Similar definition for optimal action-value function

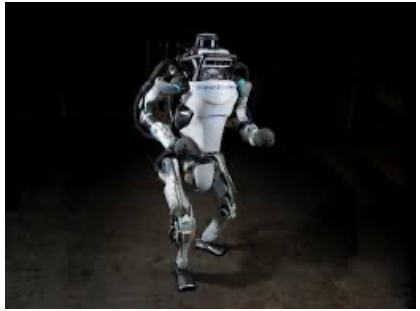
$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

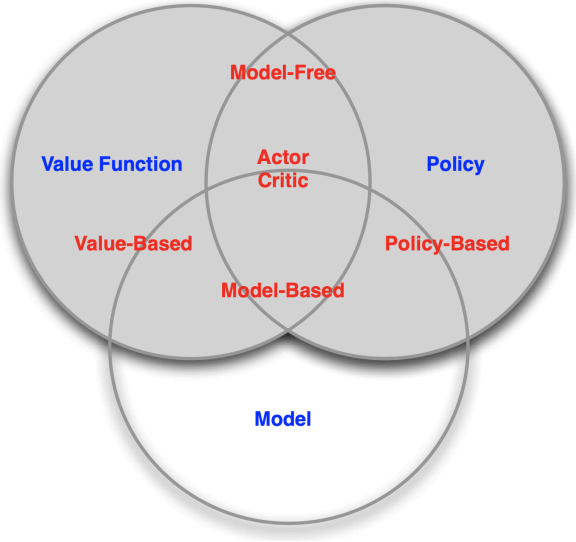
- Optimal policy can be found by maximizing over $q_*(s, a)$

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in A} q_*(s, a), \\ 0 & \text{otherwise.} \end{cases}$$

Types of agent

	Policy	Value function
Value-based	(implicit)	✓
Policy-based	✓	
Actor critic	✓	✓
Model free	✓	✓
Model-based	✓	✓



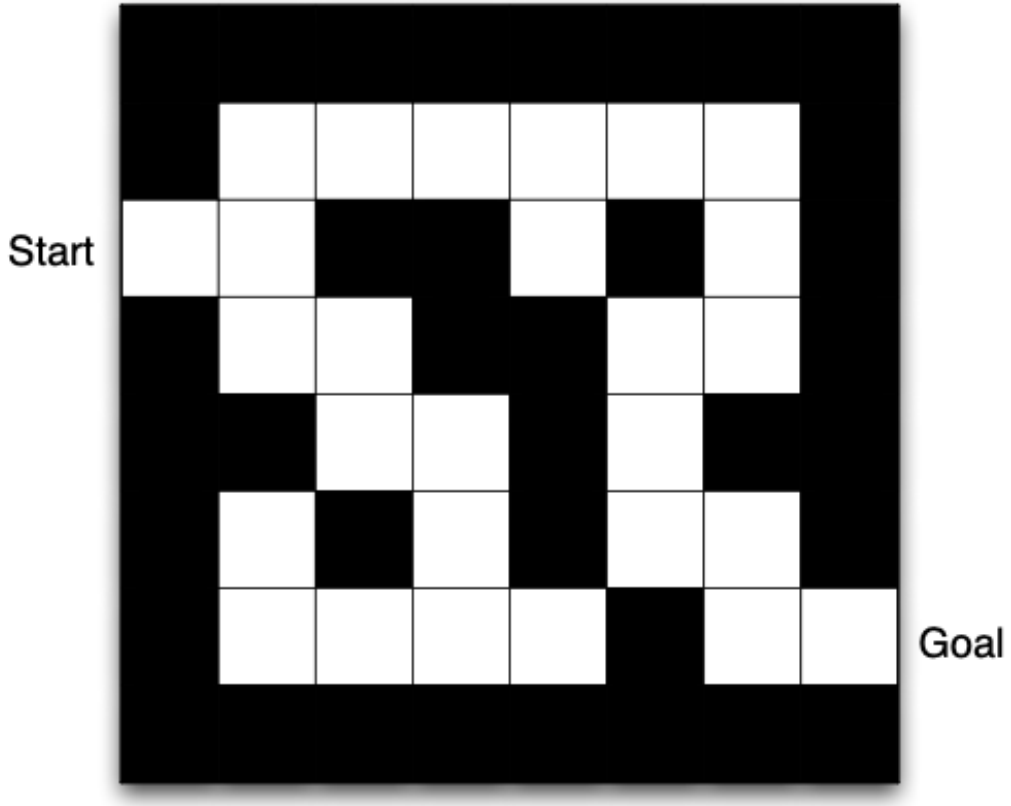


Maze example

Regard: -1 per time-step

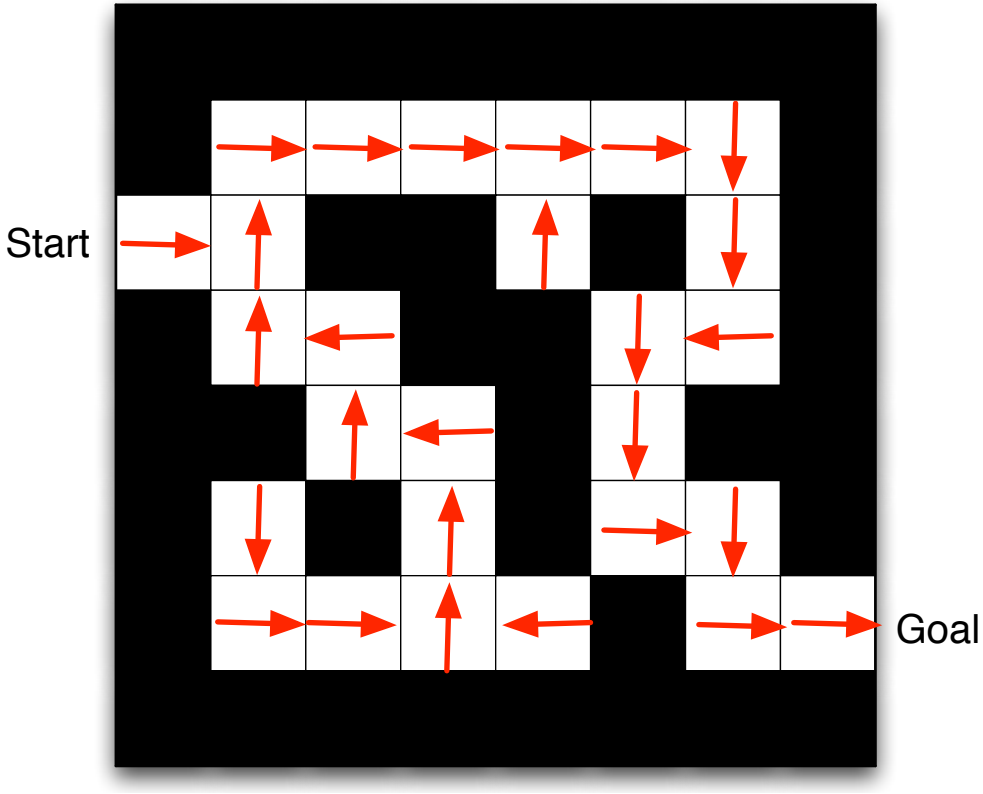
Action: N, E, S, W

State: Agent's location



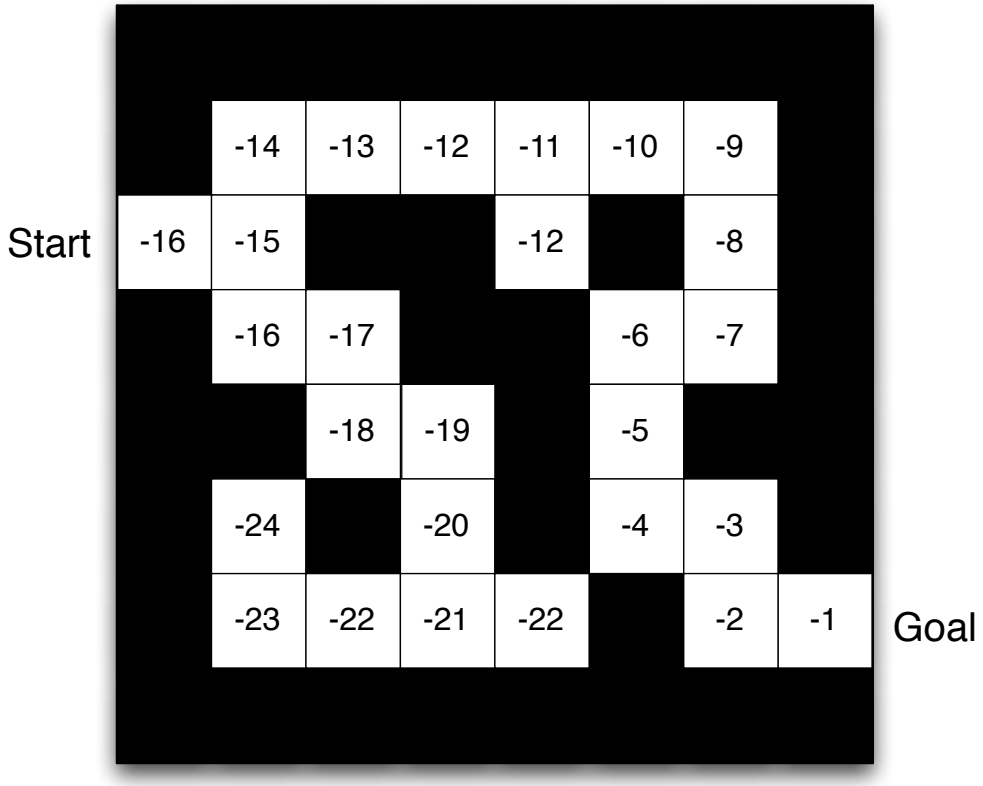
Maze example: Policy

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



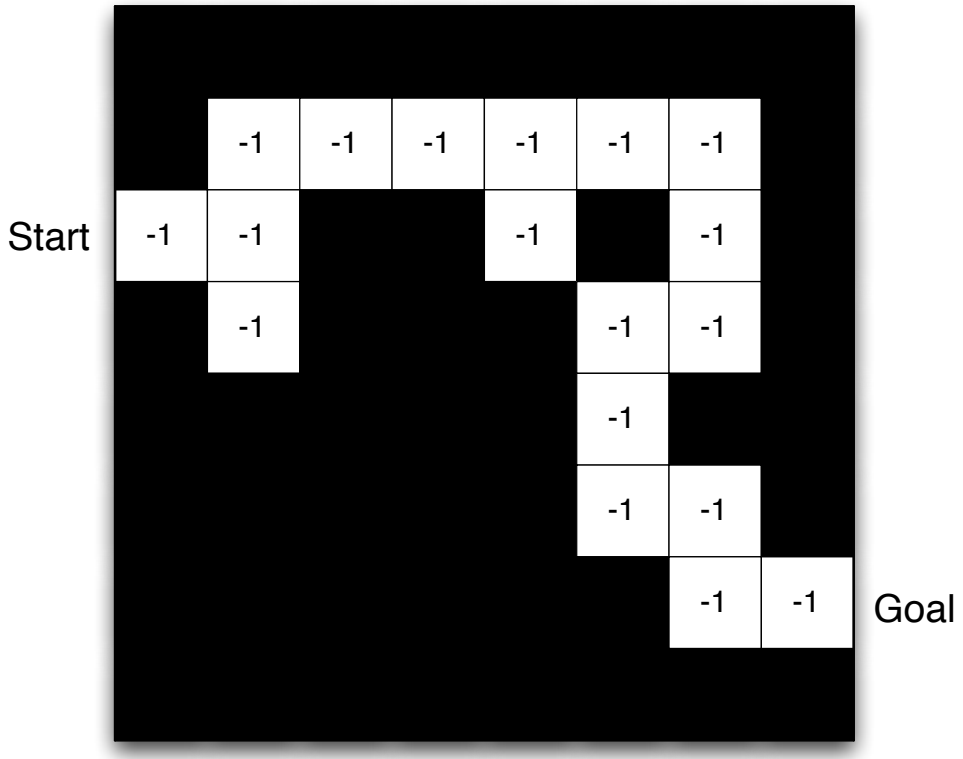
Maze example: State value function

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



Maze example: Model

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



Learning and planning

Two fundamental problems in sequential decision making

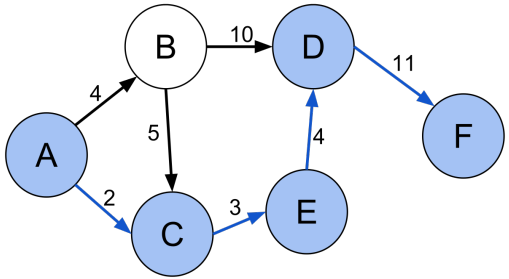
Reinforcement learning:

- The environment is initially unknown
- Agent interacts with the environment
- Agent improves its policy

https://www.jmir.org/2020/7/e18477/

Planning

- A model of the environment is known
- Agent performs computations with its model
- Agent improves its policy



Finding shortest-path

Offline and online RL

Offline RL:

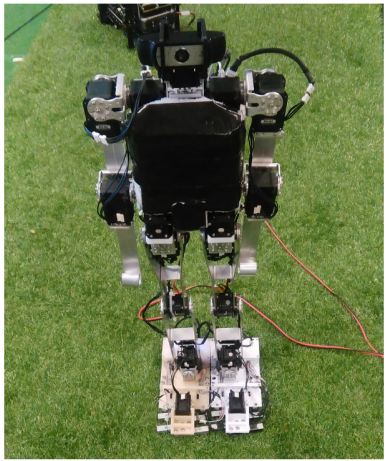
Utilize previously collected data, without additional online data collection

Large-scale robotic grasping data collection

https://arxiv.org/pdf/2005.01643.pdf

Online RL:

Collect more data on-the-fly and use them in updating the policy



Grosban robot

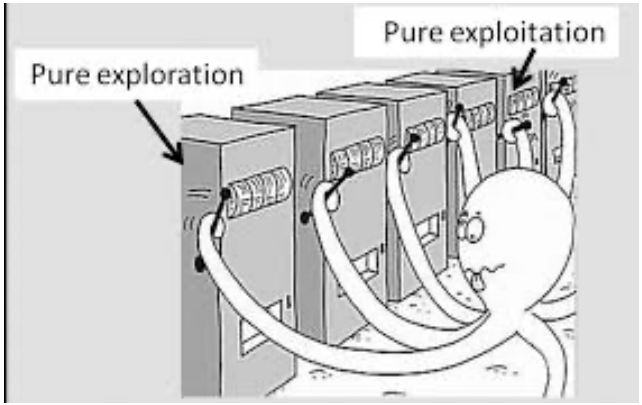
https://arxiv.org/pdf/1612.03780.pdf

Fundamental Exploration and exploitation tradeoff

- Reinforcement learning is trial-and-error learning
- Agent discover a good policy from its interaction with environment, without losing too much reward along the way

- Exploration: find more information about the environment
- Exploitation: exploits known information to maximize reward

- It is usually important to explore and exploit



Examples

	exploitation	exploration
Restaurant selection 	Go to your favorite restaurant	Try a new restaurant
Recommender system 	Recommend customer's favorite	Recommend something they have not tried before
Oil drilling 	Drill at best location	Drill at a new location
Game playing 	Play the best move	Try a new experimental move

References: David Silver’s lecture notes, Wikipedia