ECE 414/517 Reinforcement Learning

Electrical Engineering and Computer Science

Weizi Li

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Course Information

- Course website (bookmark it!)
- Canvas: announcements, slides, and lab tasks
- Prerequisites
  - Math: Linear Algebra, Calculus, and Probability Theory
  - Programming
    - Ideal: proficient with Python
    - Minimum: **Suggested Topics** under **Python Tutorials** on the course website
- No required textbook

Course Content

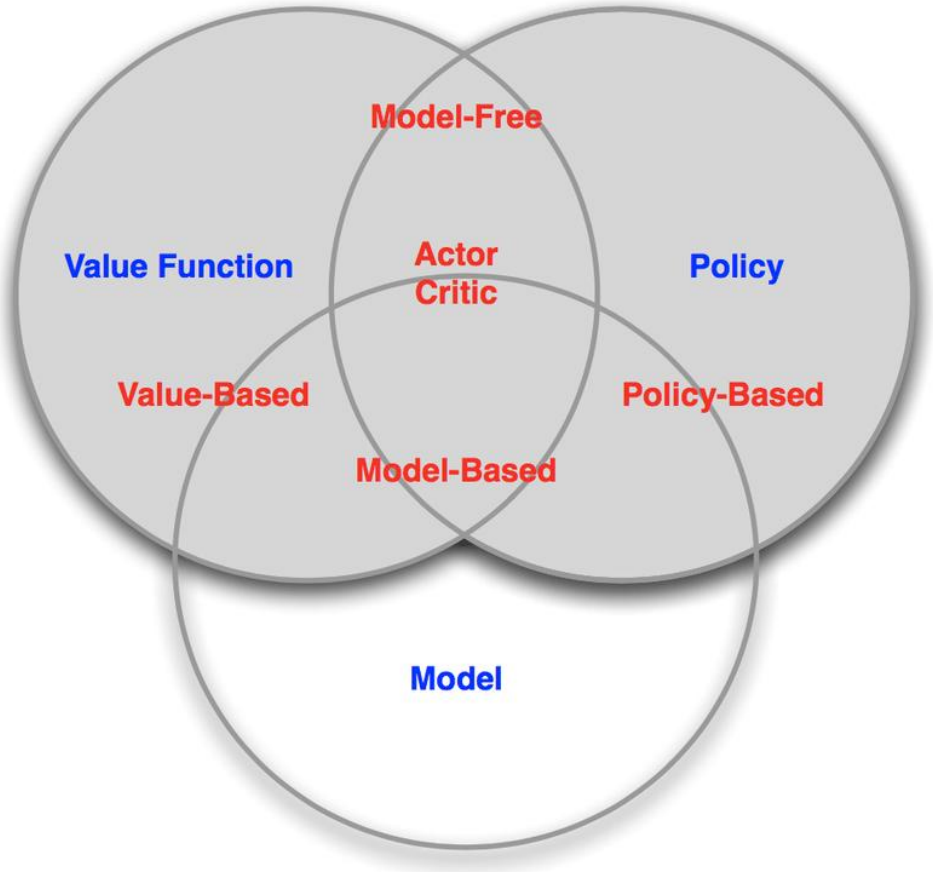
- RL Basics
- Multi-armed Bandits
- Markov Decision Process
- Dynamic Programming
- Model-free Prediction
- Model-free Control
- Deep Neural Networks
- Value Function Approximation
- Learning from Demonstrations
- Policy Gradient Methods
- Model-based RL
- Transformers in RL
- Exploration Strategies

Course Design

- Lectures
  - Introduce RL
  - Build intuition about how RL works
  - Introduce a variety of RL / Deep RL topics
- Labs (see details on the course website)
  - Hands-on experience with programming tasks
- Course Project (see details on the course website)
  - Explore RL in depth on an application of your interest

Course Design

- The curriculum is intended to be broad
- The project is intended for you to explore RL more deeply
- RL is rapidly developing—this course only scratches the surface
- Goal: motivate you to begin your RL journey



Reinforcement Learning Basics

Electrical Engineering and Computer Science

Weizi Li

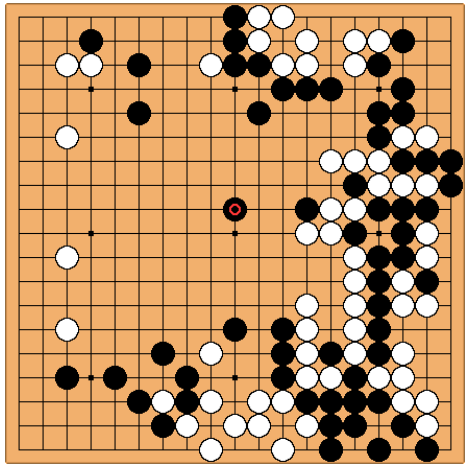
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

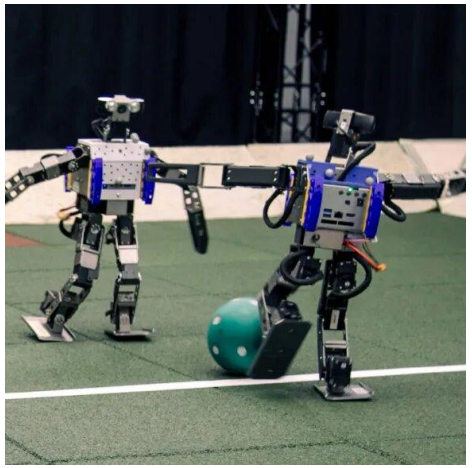
Class Content

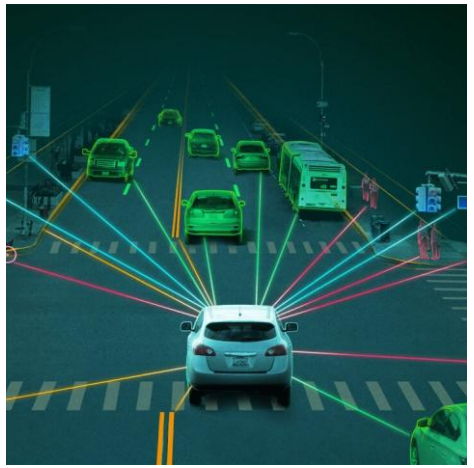
- What is RL
- Brief History of RL
- Key Concepts of RL
- Components of RL Agent
- RL vs. SL (Supervised Learning)

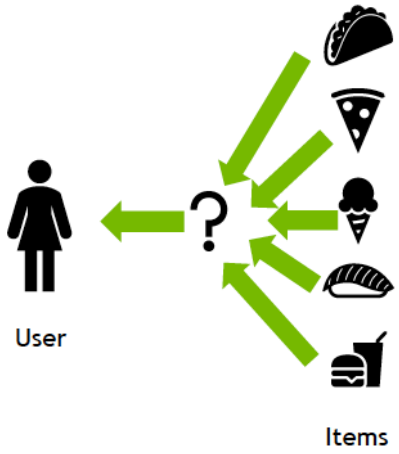
What is RL

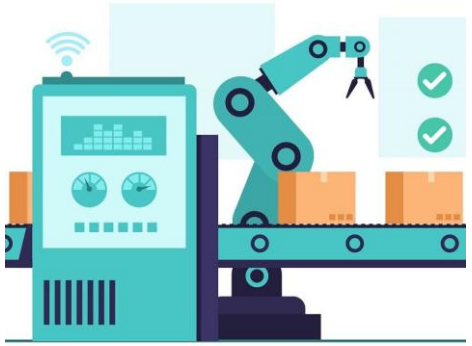
Sequential Decision Making



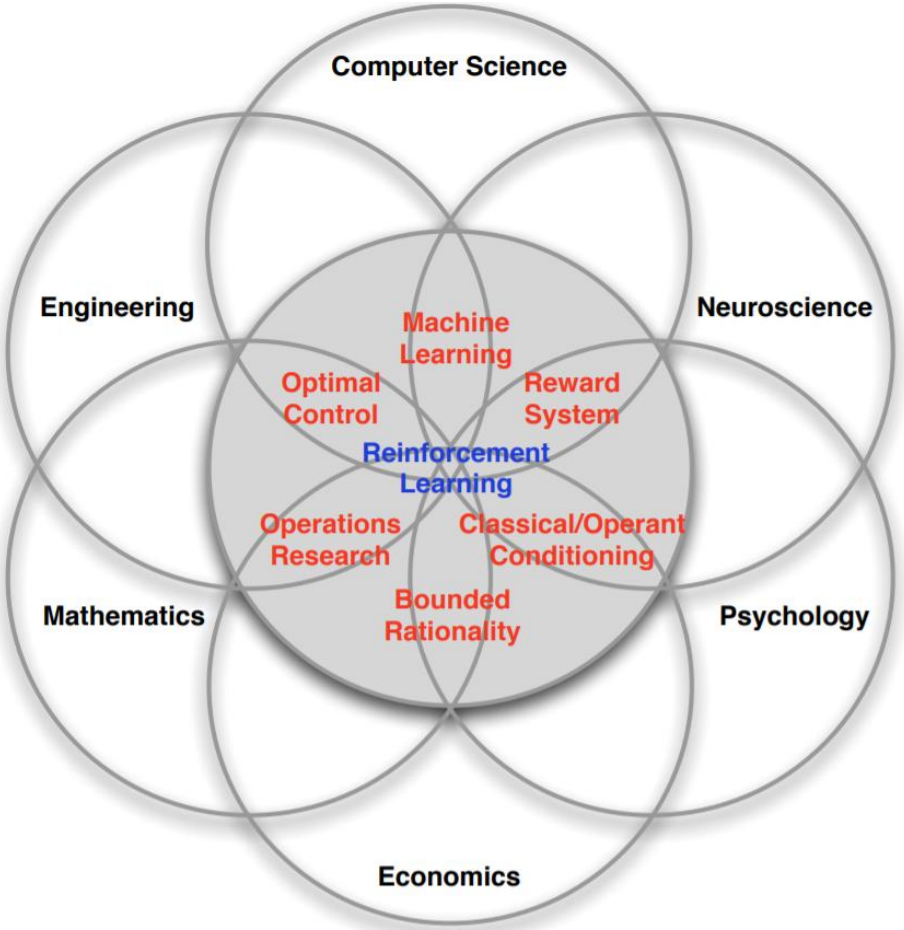






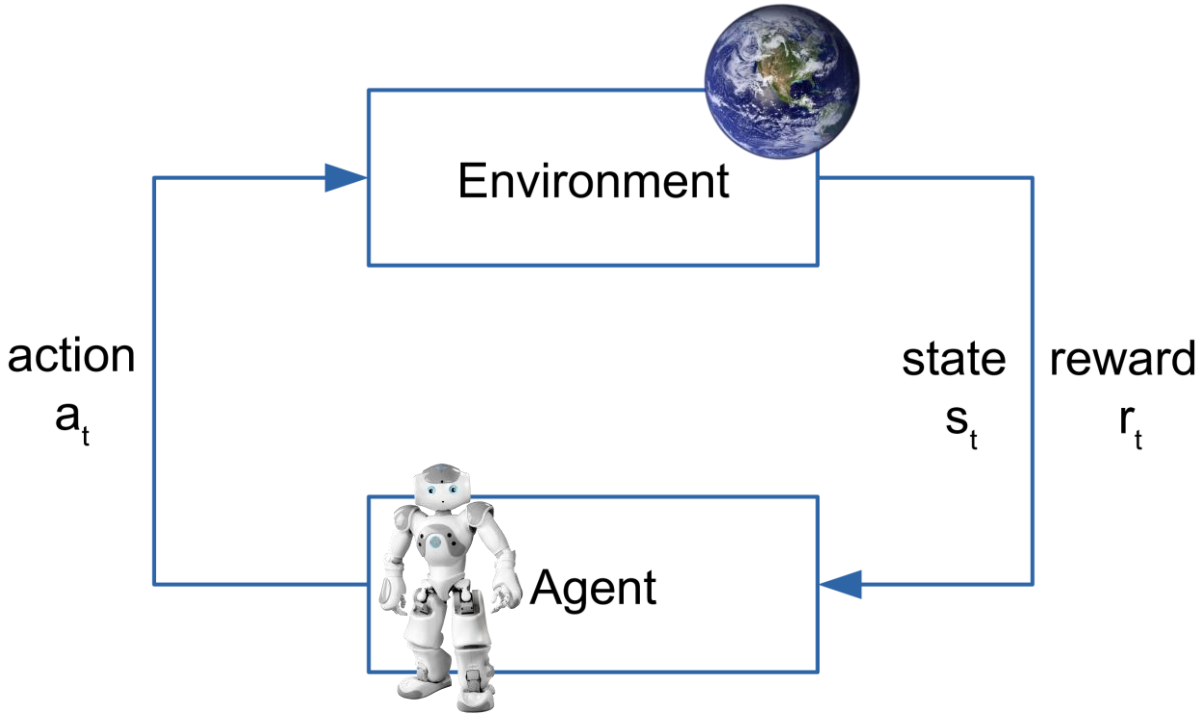


Sequential Decision Making



RL Task

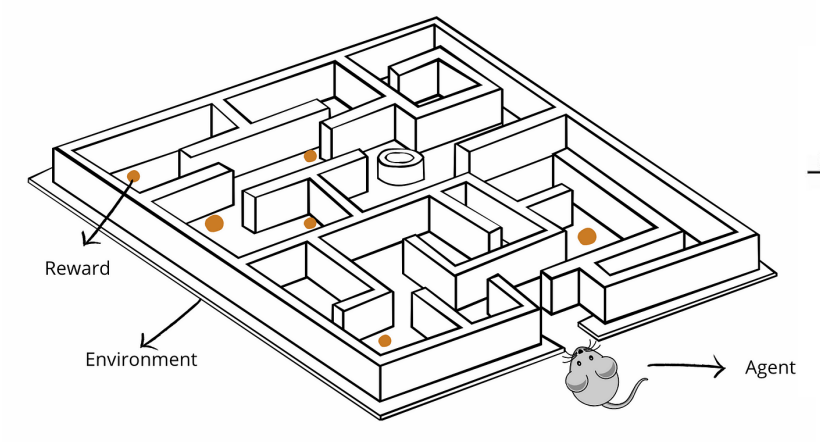
- How can an autonomous agent that senses and acts in its environment learn to choose optimal actions sequentially to maximize its cumulative reward

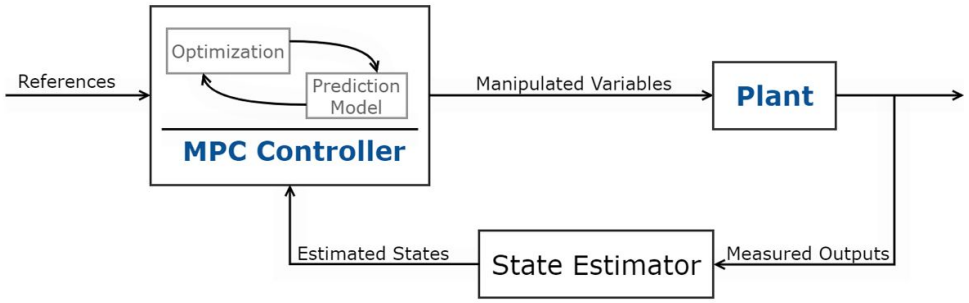


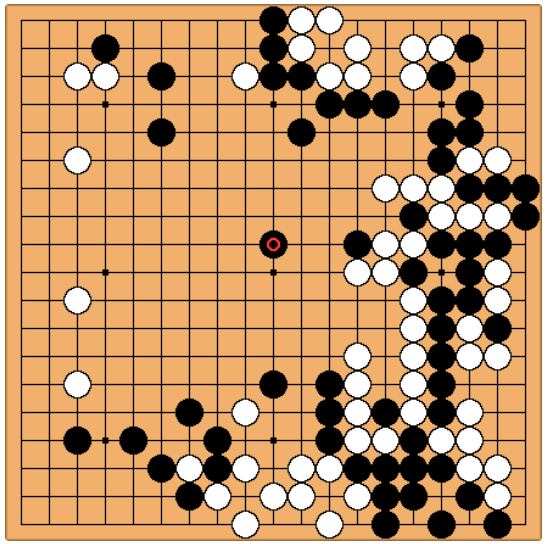
Brief History of RL

Brief History of RL

- Three threads joined in the '80s forming the modern RL
  - Animal learning (psychology)
  - Optimal control (engineering)
  - Artificial intelligence (computer science)

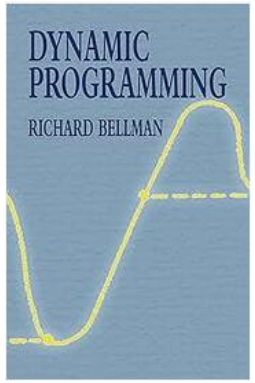


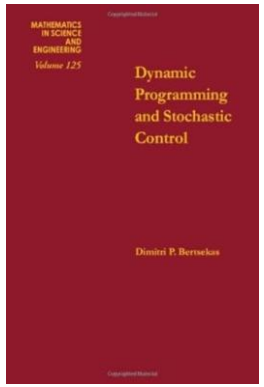


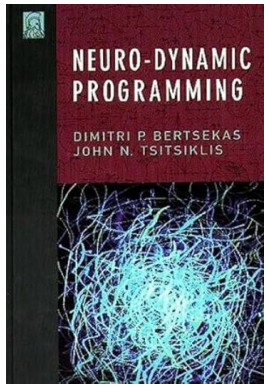


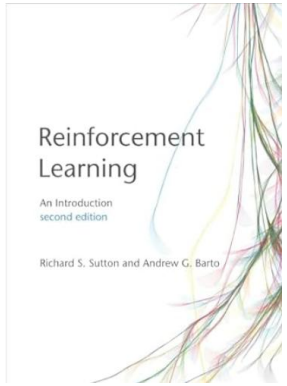
Seminal Books (Control & AI)

- *Dynamic Programming* by Bellman, 1957
- *Dynamic Programming and Stochastic Control* by Bertsekas, 1976
- *Neuro-Dynamic Programming* by Bertsekas and Tsitsiklis, 1996
  - Nowadays "deep reinforcement learning"
- *Reinforcement Learning: An Introduction* by Sutton and Barto, 1998



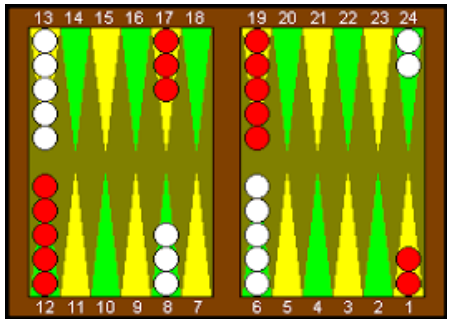






Before Deep Learning

- 1959: first successful checkers-playing program by Samuel
- 1970's: machine intelligence based on animal learning by Klopf
- 1980's: temporal-difference learning
- 1980's—1990's: approximate dynamic programming (via Neural Networks)
- 1992: TD-GAMMON by Tesauro (world's best backgammon player)
- 1980's—2012: winter of RL

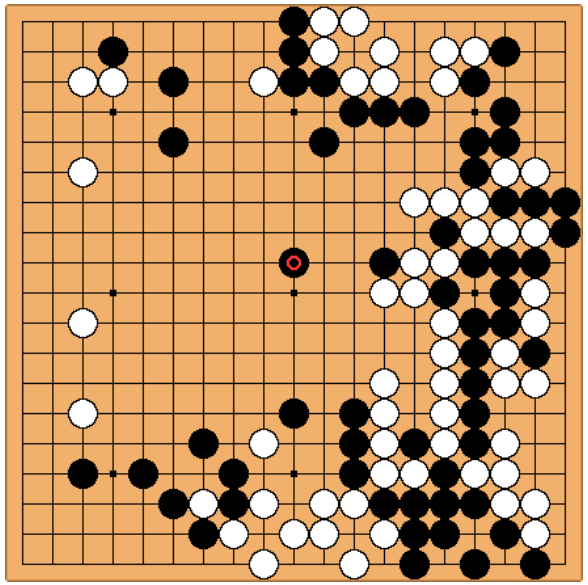


Deep Learning Era

- 2015: Atari games (superhuman on some games)
- 2017: AlphaZero (superhuman on Go and Chess)
- 2019: OpenAI Five (superhuman on Dota 2)
- 2019: AlphaStar (grandmaster on StarCraft II)
- 2019: Rubik's cube (real robot)
- 2020: Hide and Seek (multi-agent system)
- 2021: Atari games using Go-Explore (superhuman on all games)
- 2021: Chip floor planning (superhuman)
- 2022: Gran Turismo 7 (superhuman)

Go: A Game of Surrounding and Controlling Areas

- Two players, one plays black stones, the other white
- Players take turns placing stones
- If stones are completely surrounded by the opponent, they are removed from the board
- Goal: control more territory than your opponent
- Grand AI challenge for decades



- Deep Blue (IBM) defeats chess world champion in 1997
- AlphaGo (DeepMind) defeats Go world champion in 2016

What Made The Difference

- Increased computing power + deep neural networks + new algorithms + engineering practices
- Not much advancement on theory

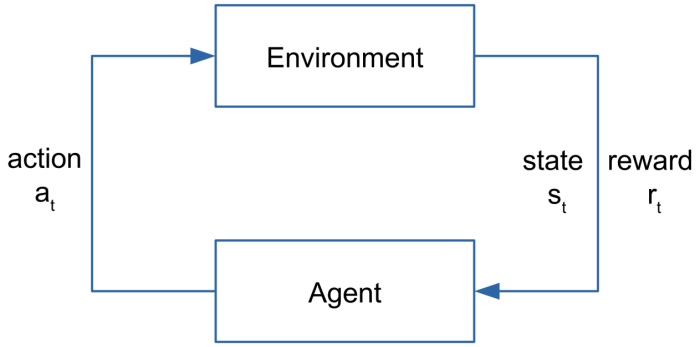
What Made The Difference

- Increased computing power + deep neural networks + new algorithms + engineering practices
- Not much advancement on theory

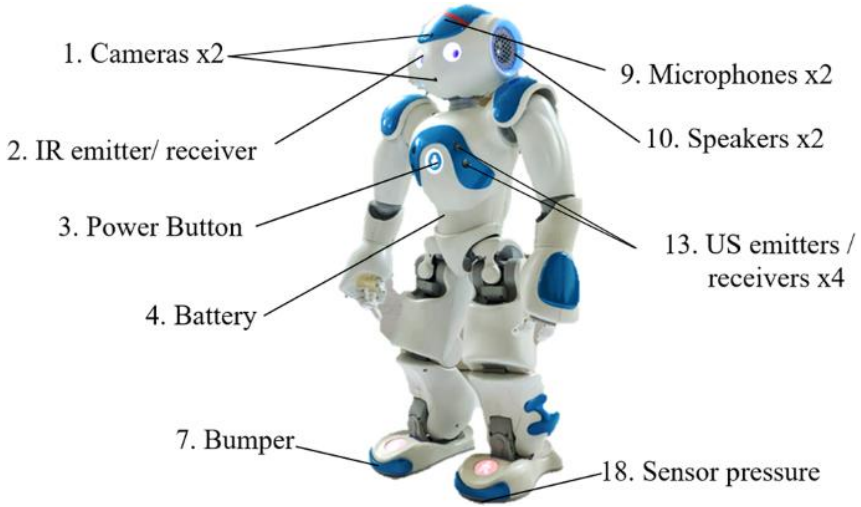
- Bellman: far more ability and sophistication is required to obtain a numerical solution than to establish the usual existence and uniqueness theorems

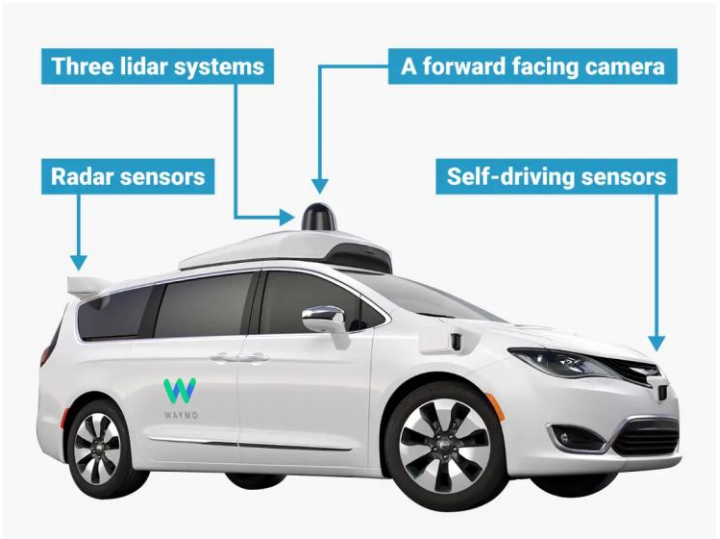
Key Concepts of RL

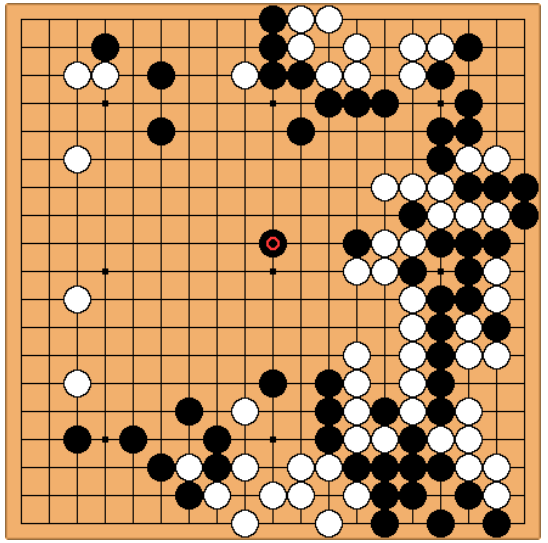
Key Concept: Agent



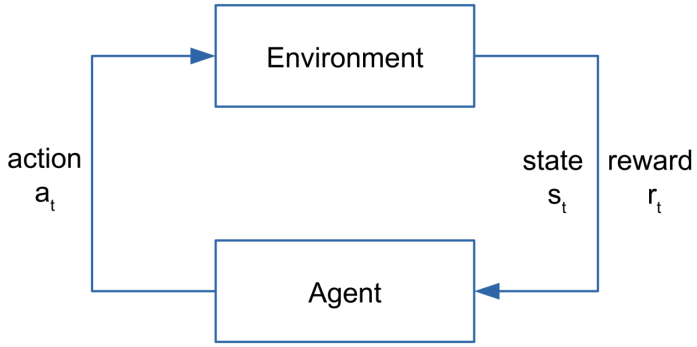
- Can sense the environment and choose actions to perform
- Common examples: robot, vehicle, software agent, etc.



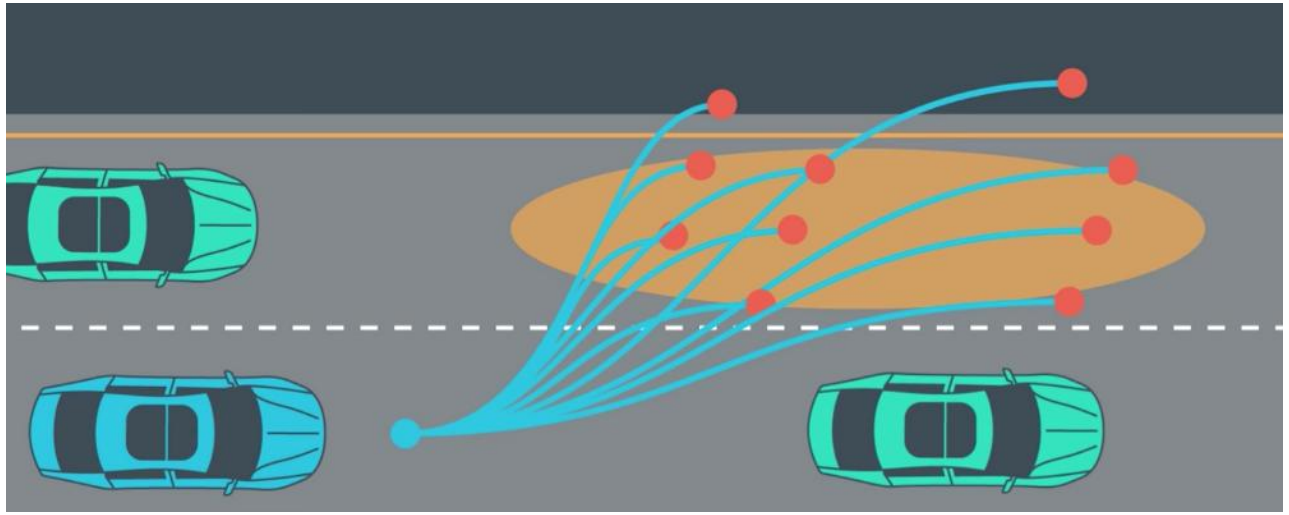


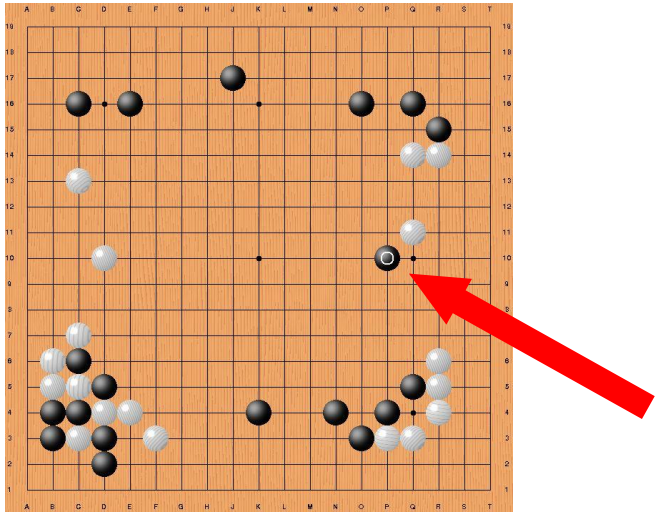


Key Concept: Action

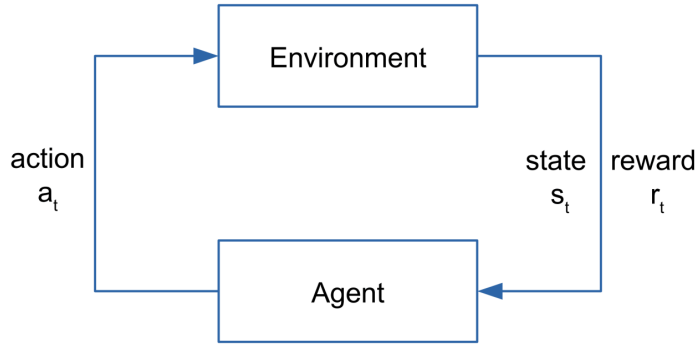


- Decisions available to the agent
- Action will affect the environment
- Action may have long-term consequences

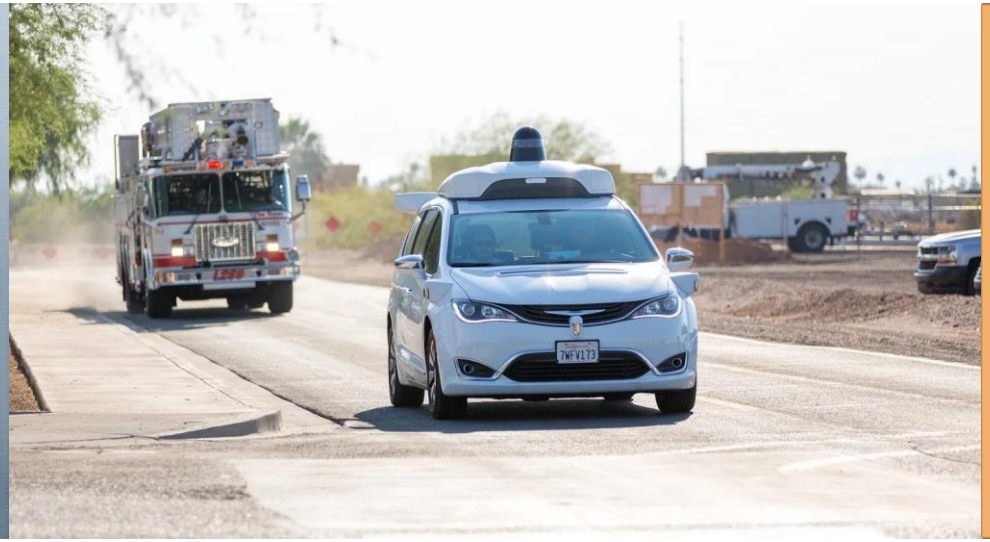


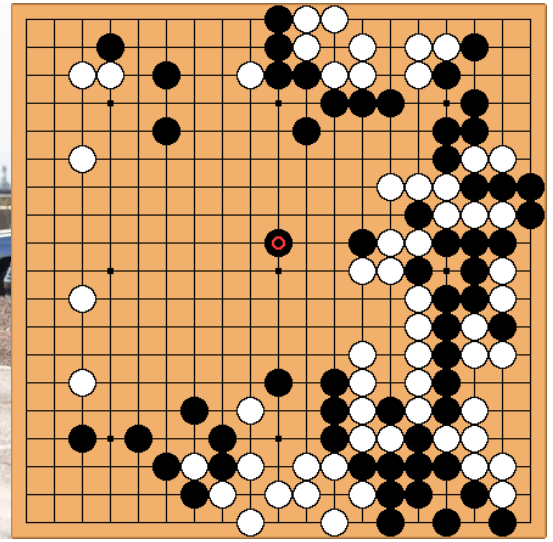


Key Concept: Environment

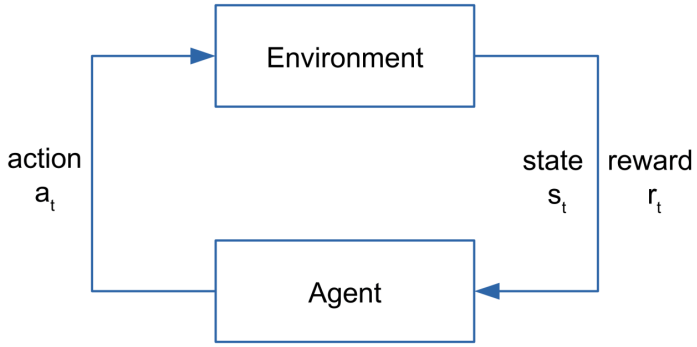


- The world that the agent lives in and operates

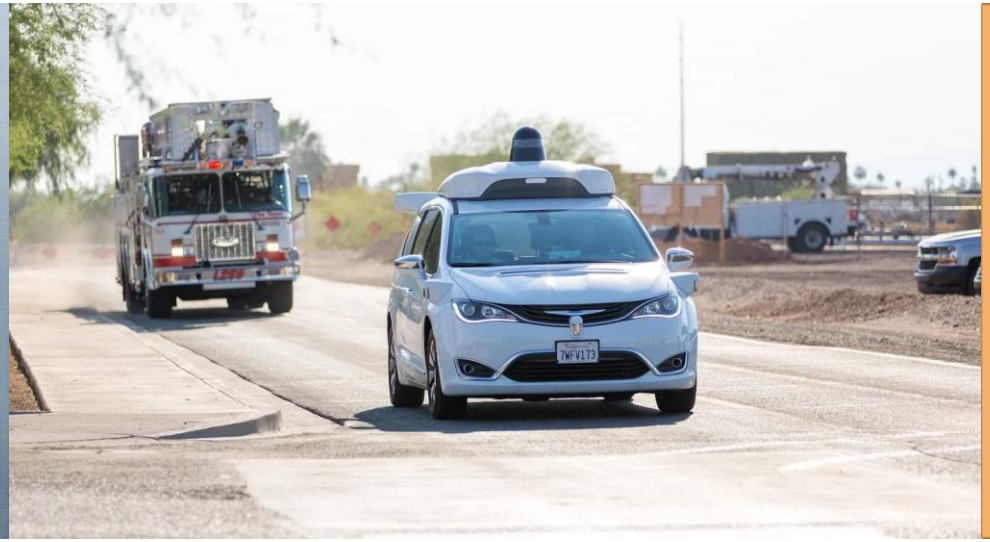


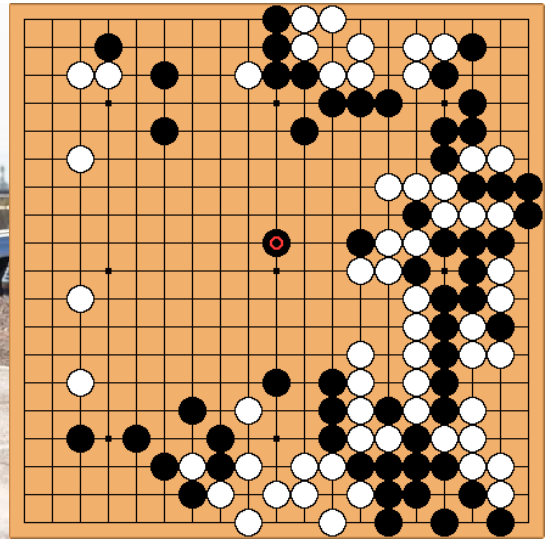


Key Concept: State



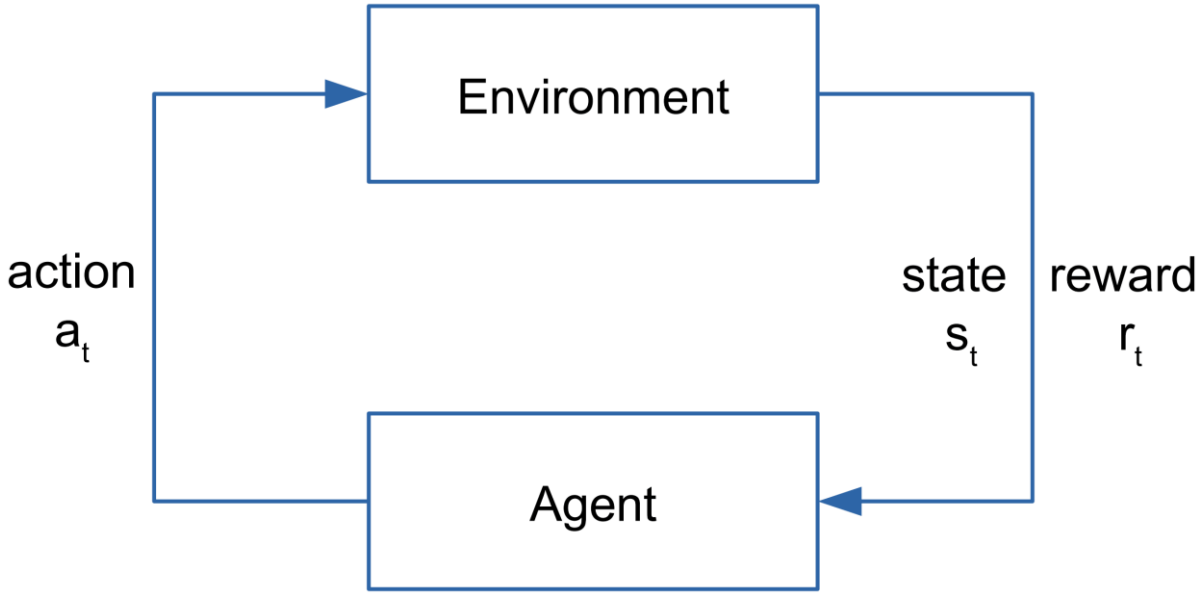
- Snapshot of the status of the environment
- Contains all relevant information for agent decision-making





Key Concept: Reward

- Immediate payoff from the environment to each action of the agent
- Reward is expressed as a single number (scalar)



Example: Atari Breakout

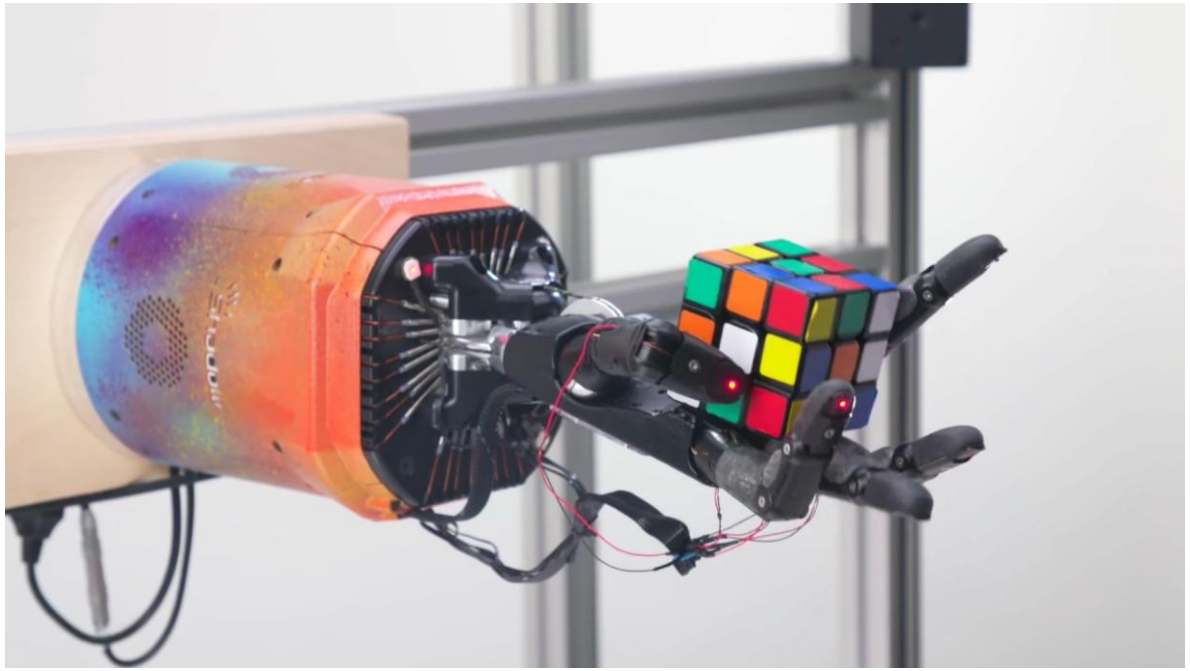
- Agent
- Action
- Environment
- State
- Reward

Example: Go

- Agent
- Action
- Environment
- State
- Reward

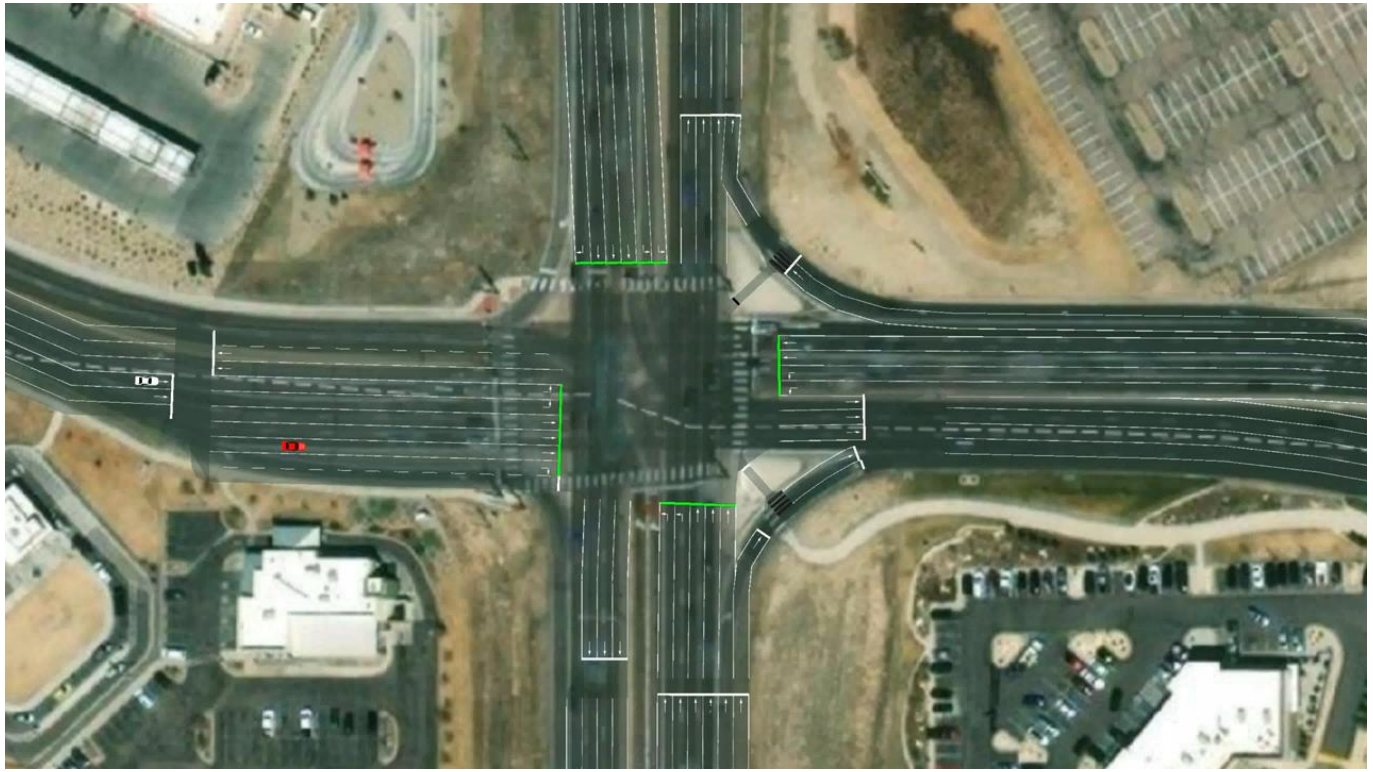
Example: Rubik's Cube

- Agent
- Action
- Environment
- State
- Reward



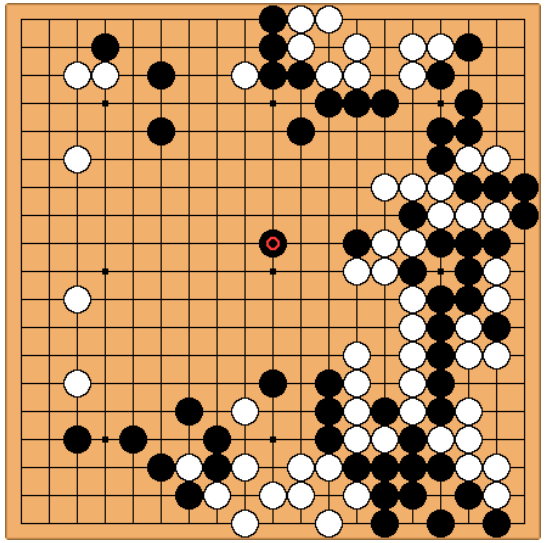
Example: Mixed Traffic Control

- Agent
- Action
- Environment
- State
- Reward



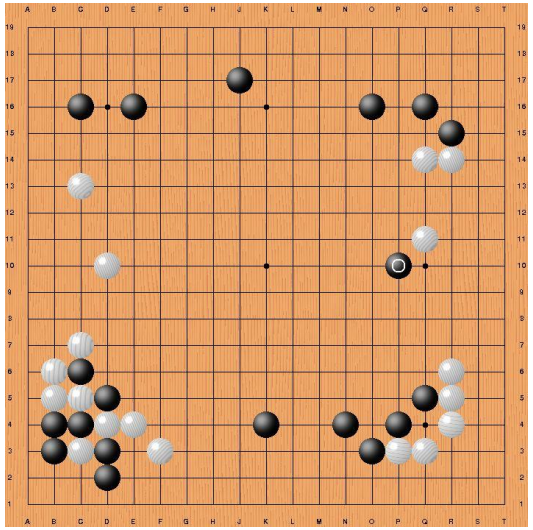
Key Concept Revisited: Agent

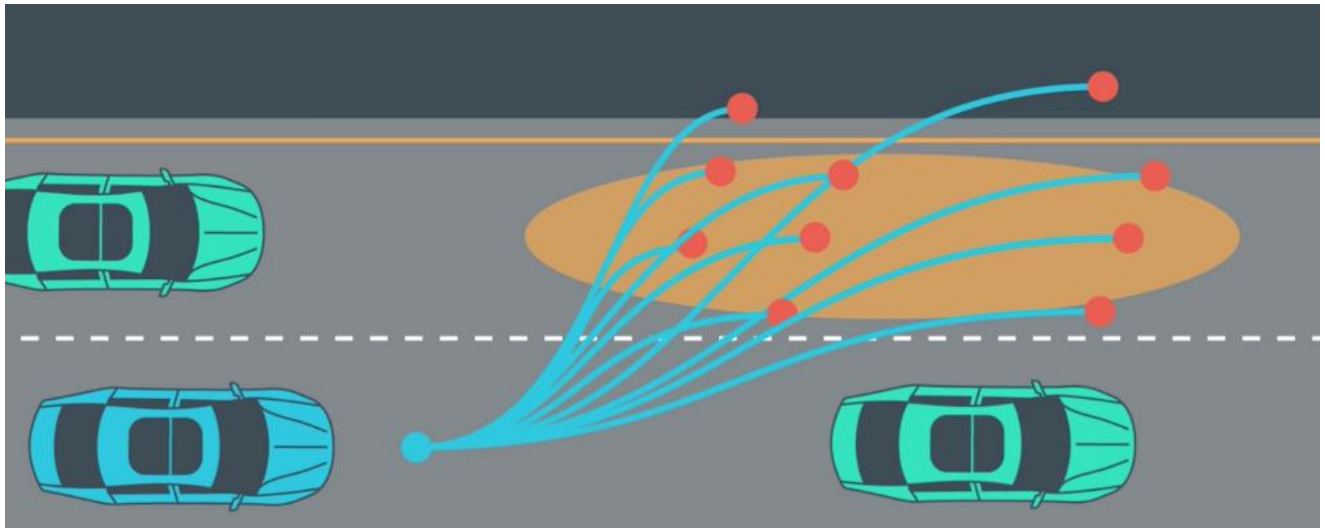
- Only the decision-making process
- Anything beyond agent's complete control (thus subject to environment's manipulation) is part of the environment



Key Concept Revisited: Action

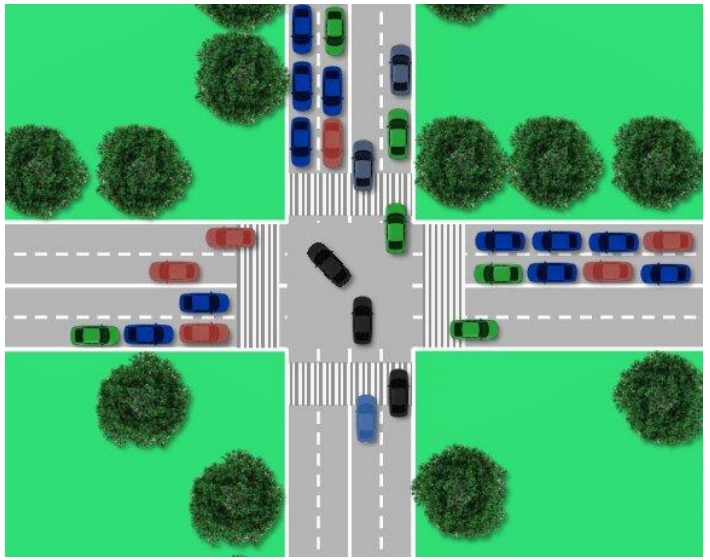
- Discrete vs. Continuous
  - If the actions can be enumerated, the action space is discrete; otherwise, the action space is continuous





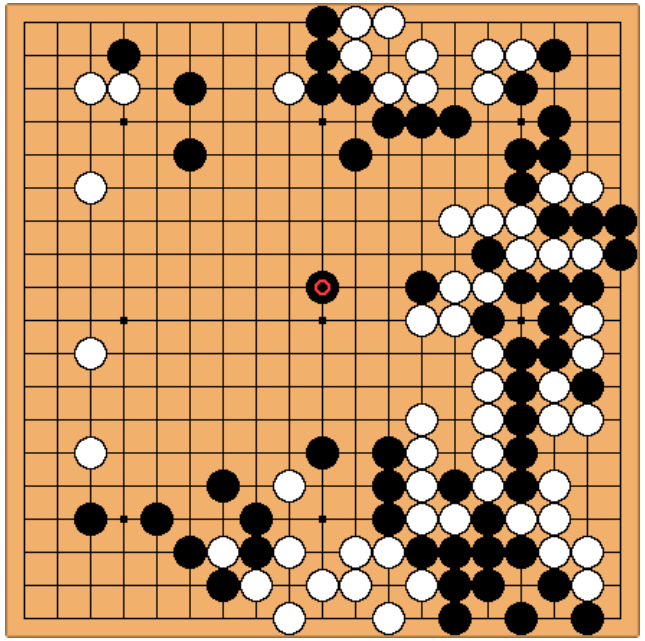
Key Concept Revisited: Environment

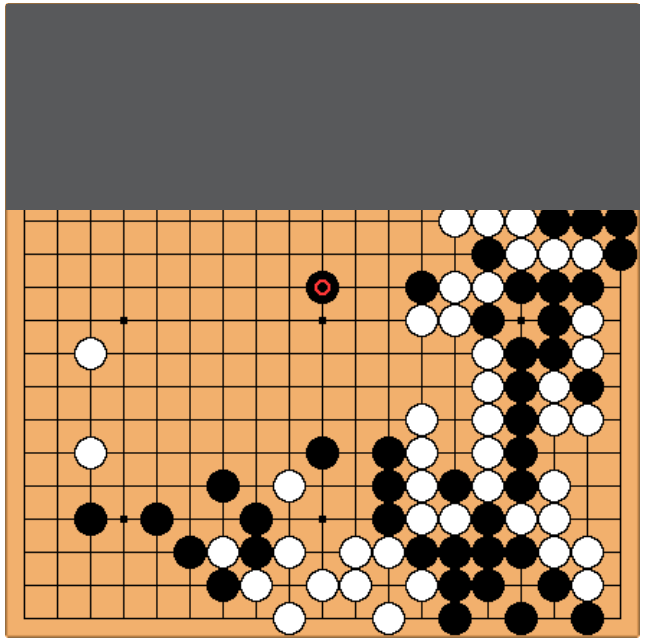
- Single-agent vs. Multi-agent
  - How many agents are being optimized
  - Multi-agent: cooperative vs. competitive



Key Concept Revisited: Environment

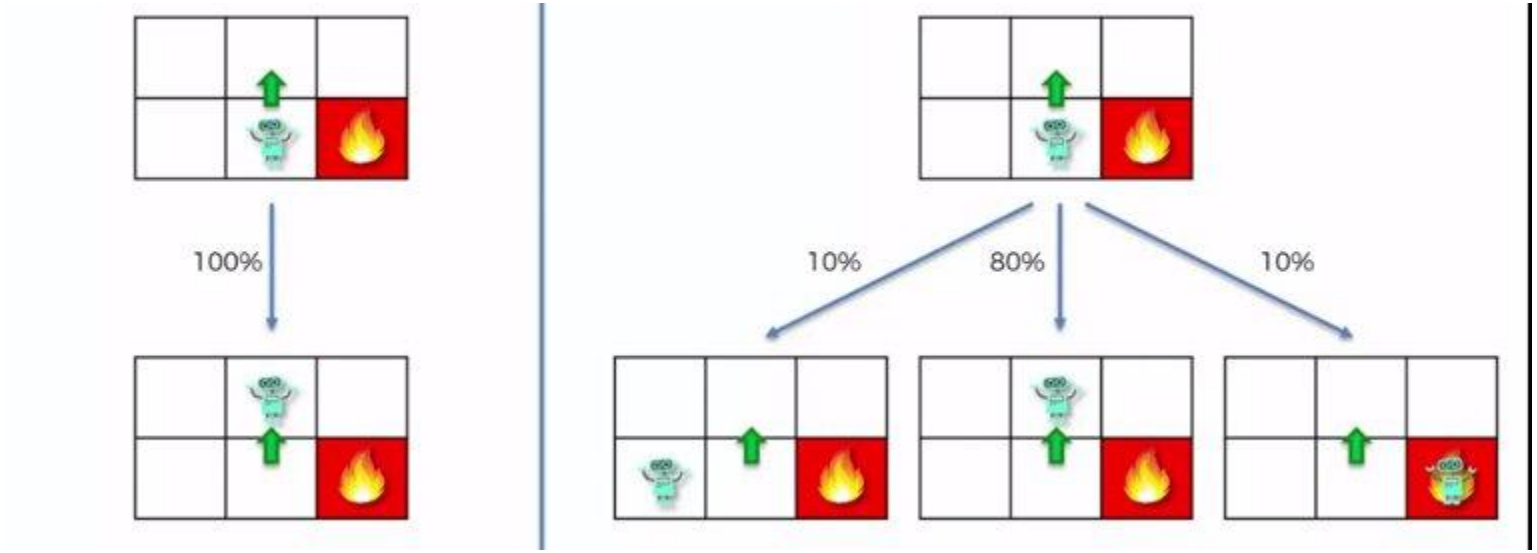
- Fully observable vs. Partially observable
  - Whether agent can sense the complete state of the environment





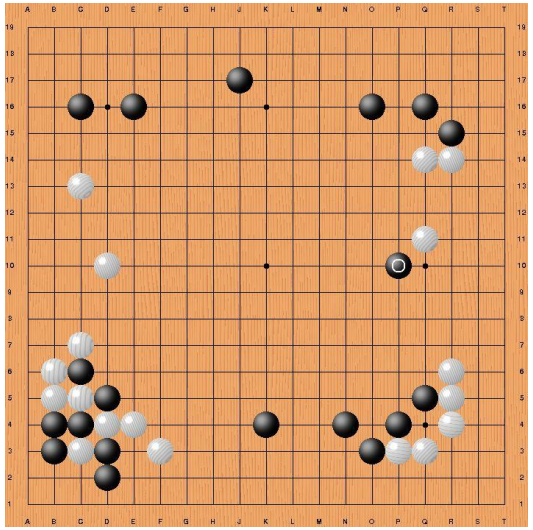
Key Concept Revisited: Environment

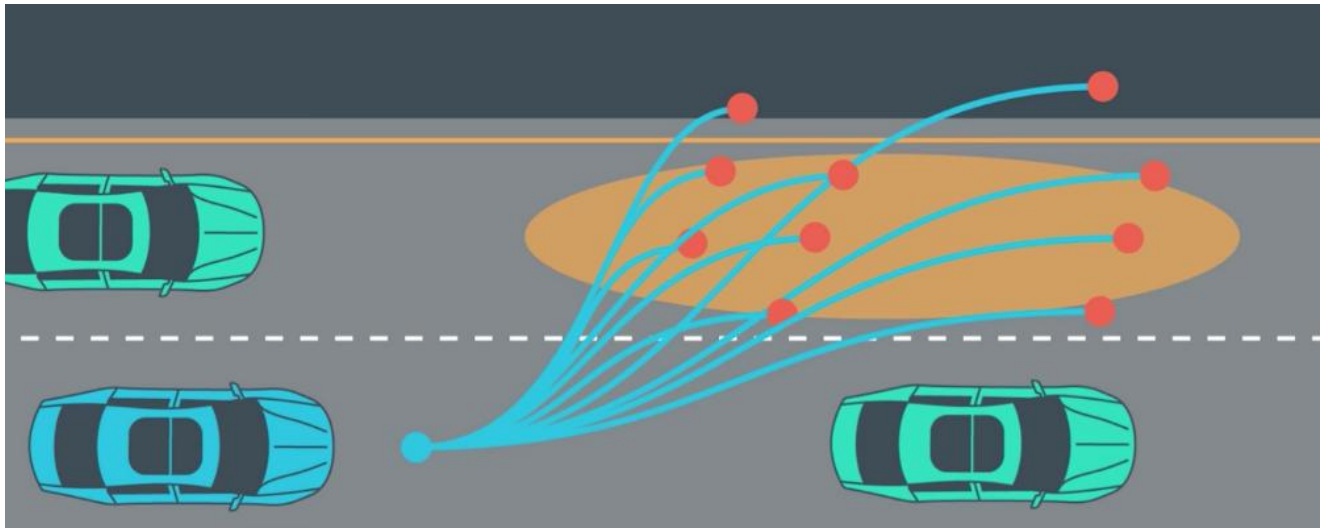
- Deterministic vs. Nondeterministic
  - If the next state is completely determined by the current state and the action of the agent, the environment is deterministic



Key Concept Revisited: Environment

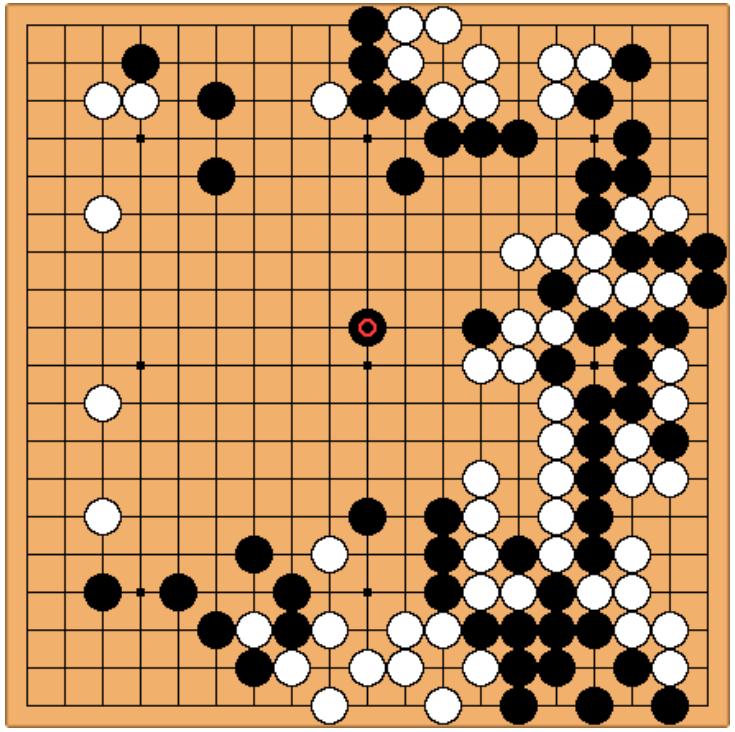
- Static vs. Dynamic
  - If the environment doesn’t change during an agent's decision-making, it’s static; otherwise, it’s dynamic





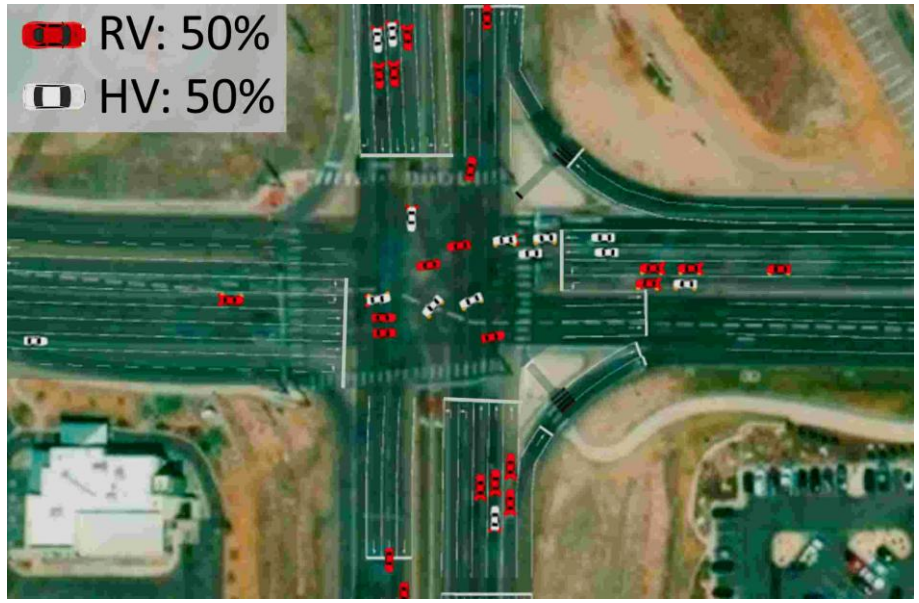
Example: Go

- Action
  - Discrete vs. Continuous
- Environment
  - Fully observable vs. Partially observable
  - Single-agent vs. Multi-agent
  - Deterministic vs. Nondeterministic
  - Static vs. Dynamic



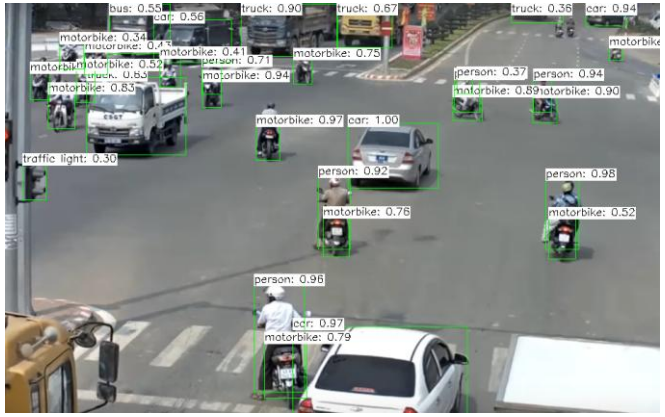
Example: Mixed Traffic Control

- Action
  - Discrete vs. Continuous
- Environment
  - Fully observable vs. Partially observable
  - Single-agent vs. Multi-agent
  - Deterministic vs. Nondeterministic
  - Static vs. Dynamic



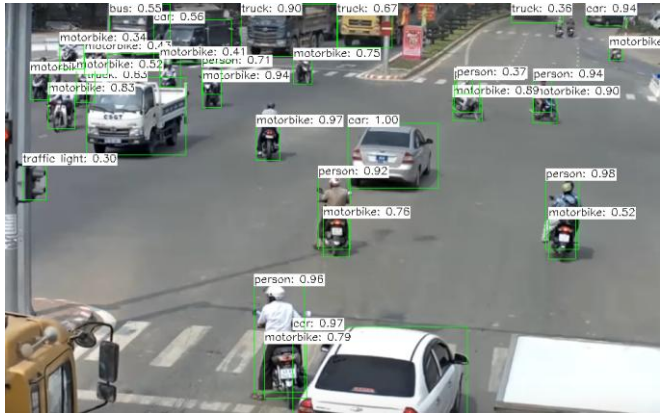
Key Concept Revisited: State

- Recall: state contains all relevant information for decision-making



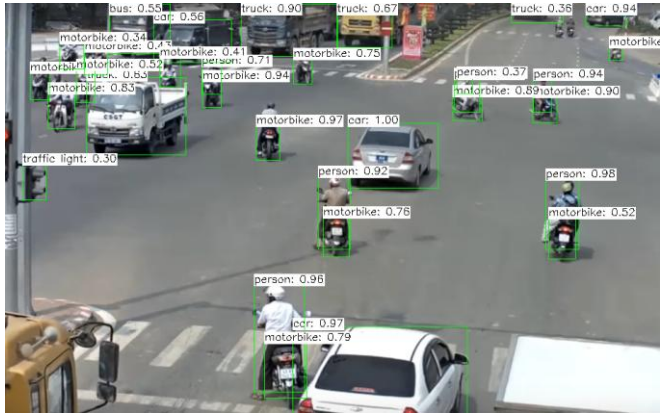
Key Concept Revisited: State

- Recall: state contains all relevant information for decision-making
- State (exact information) can be missing or unknown



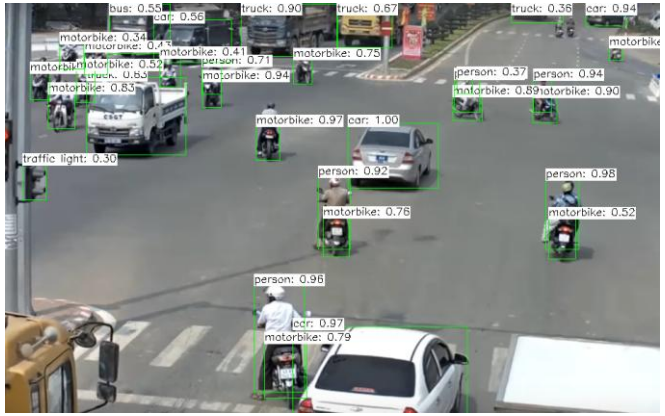
Key Concept Revisited: State

- Recall: state contains all relevant information for decision-making
- State (exact information) can be missing or unknown
- Agent usually obtains observations of the environment via sensors



Key Concept Revisited: State

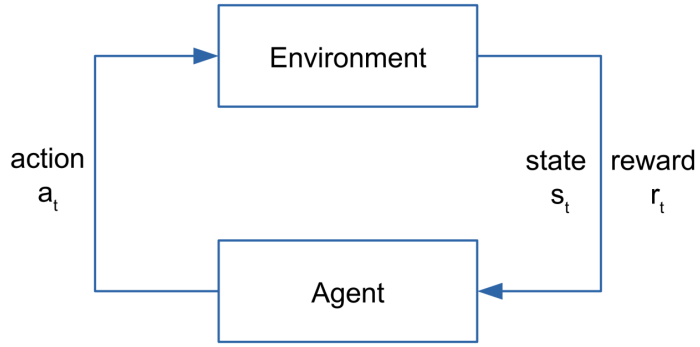
- Recall: state contains all relevant information for decision-making
- State (exact information) can be missing or unknown
- Agent usually obtains observations of the environment via sensors
- Observations can be distorted due to sensor or processing errors



Key Concept Revisited: State

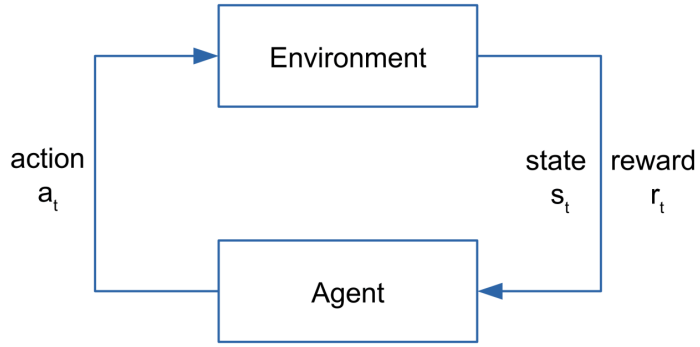
- Recall: state contains all relevant information for decision-making
- State (exact information) can be missing or unknown
- Agent usually obtains observations of the environment via sensors
- Observations can be distorted due to sensor or processing errors
- Designing what to observe (if you can) to enable an RL task can be challenging

Key Concept Revisited: Reward



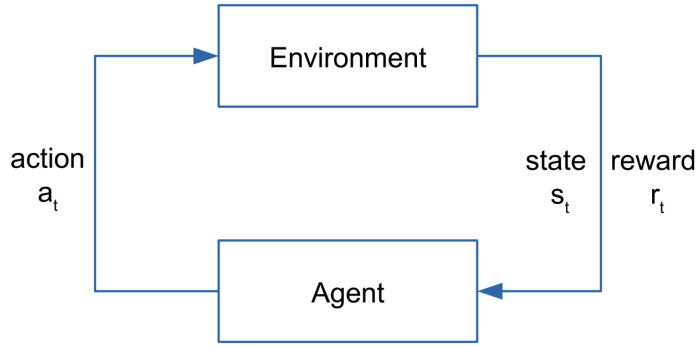
- Recall: reward is the immediate payoff from the environment to each action of the agent

Key Concept Revisited: Reward



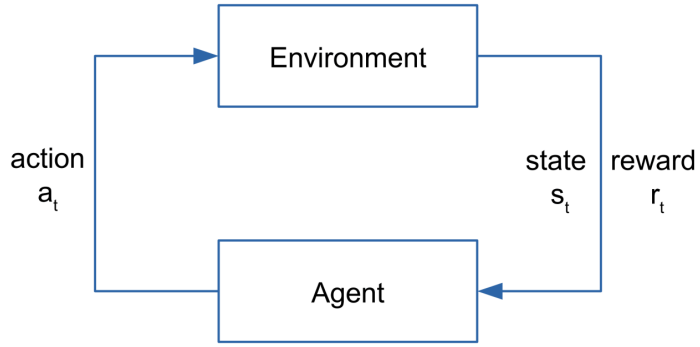
- Recall: reward is the immediate payoff from the environment to each action of the agent
- The immediate payoff can be no reward or even penalty

Key Concept Revisited: Reward



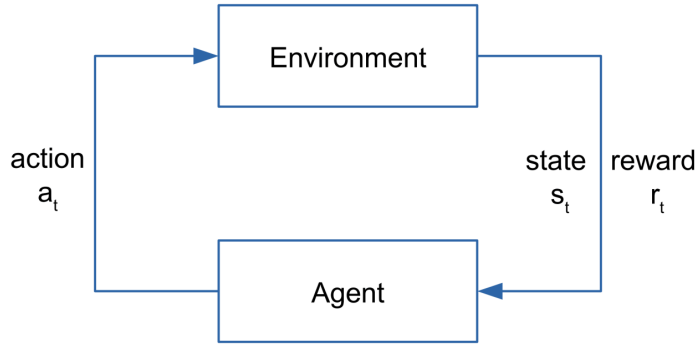
- Recall: reward is the immediate payoff from the environment to each action of the agent
- The immediate payoff can be no reward or even penalty
- The actual rewards are often sparse and delayed

Key Concept Revisited: Reward



- Recall: reward is the immediate payoff from the environment to each action of the agent
- The immediate payoff can be no reward or even penalty
- The actual rewards are often sparse and delayed
- Thus, the meaning of "reward" needs to be interpreted in context

Key Concept Revisited: Reward

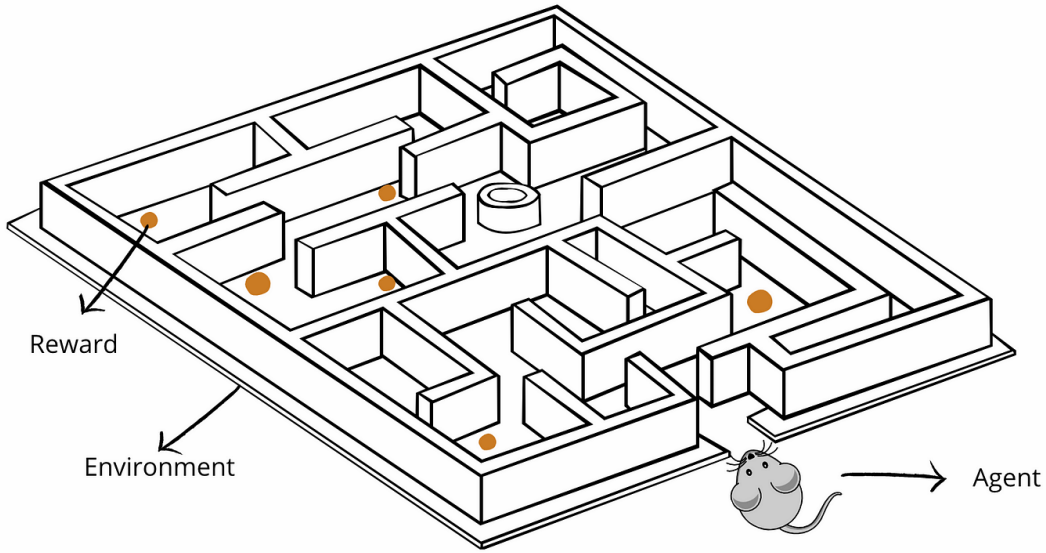


- Recall: reward is the immediate payoff from the environment to each action of the agent
- The immediate payoff can be no reward or even penalty
- The actual rewards are often sparse and delayed
- Thus, the meaning of "reward" needs to be interpreted in context
- Reward design can be challenging (examples)

Components of RL Agent

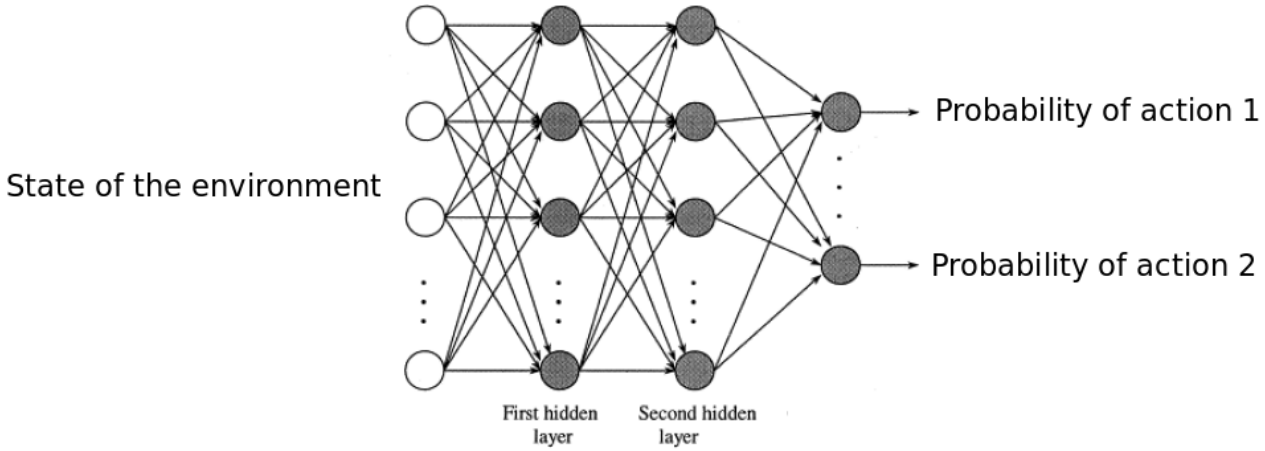
RL Agent

- RL Task: how can an agent that senses and acts in its environment learn to choose optimal actions sequentially to maximize its cumulative reward
- Goal of RL Agent
  - Perform sequences of actions;
  - Observe their rewards; and
  - Learn an optimal policy



Components of RL Agent: Policy

- A mapping from states (observations) of the environment to agent actions
- Deterministic: $\pi(s) = a$
- Stochastic: $\pi(a|s) = P[a_t = a|s_t = s]$

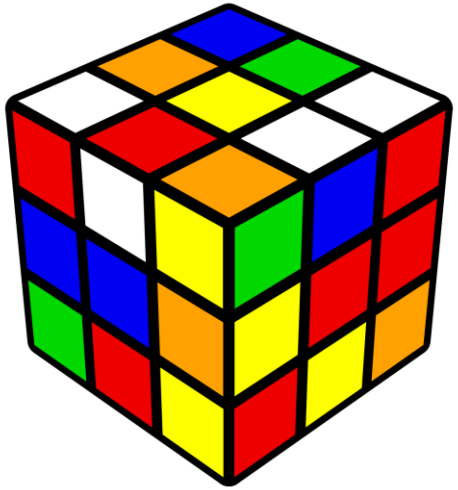


Components of RL Agent: Value Function

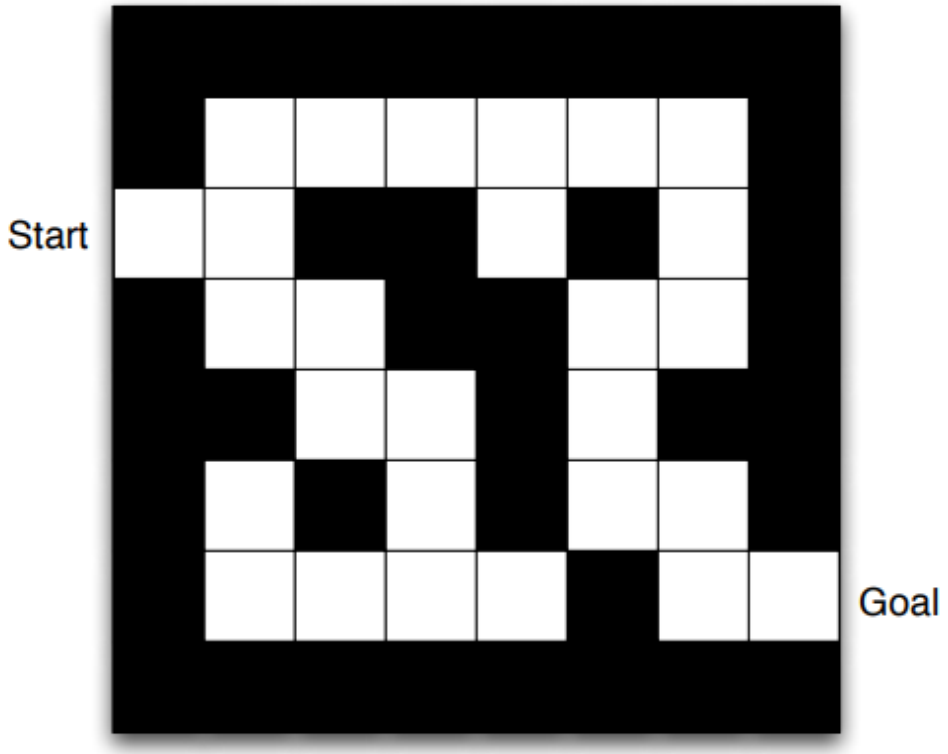
- Value function
  - Estimated cumulative reward starting from a state or a state-action pair
- Example of State Value Function $V(s)$
  - State 1: having MS in CS
  - State 2: not having MS in CS
  - V(State 1): estimated total income *n* years after having MS in CS
  - V(State 2): estimated total income *n* years after not having MS in CS
  - Note: in theory, *n* = infinite

Components of RL Agent: Model

- Agent's knowledge about rules, dynamics, and outcomes of its actions
- Model is optional for RL agent
- Model-Free learning: agent learns without knowing the effects of its actions

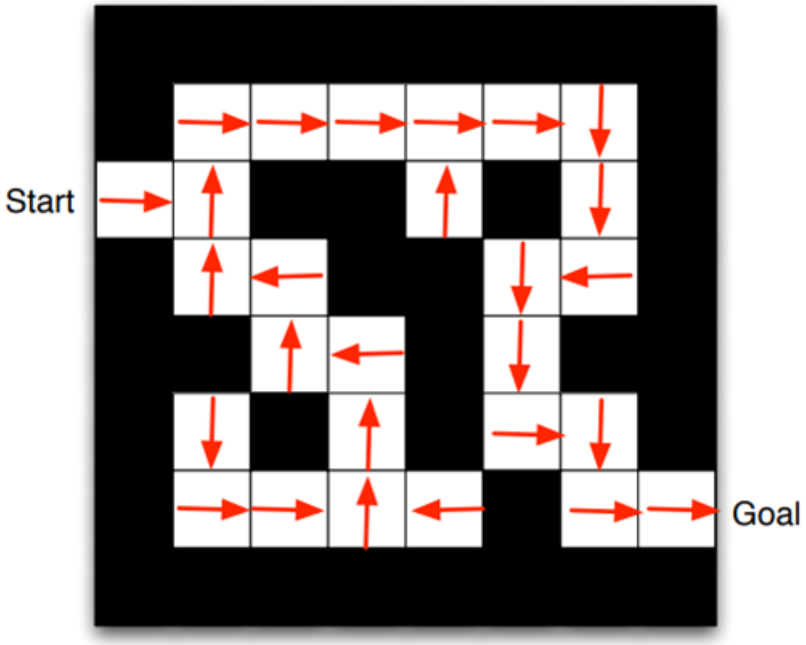


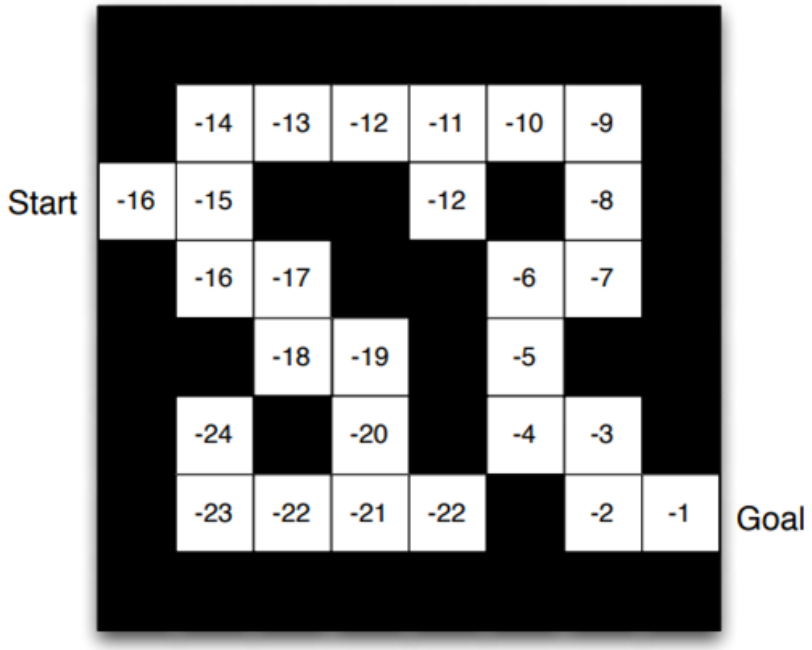
Example: Maze

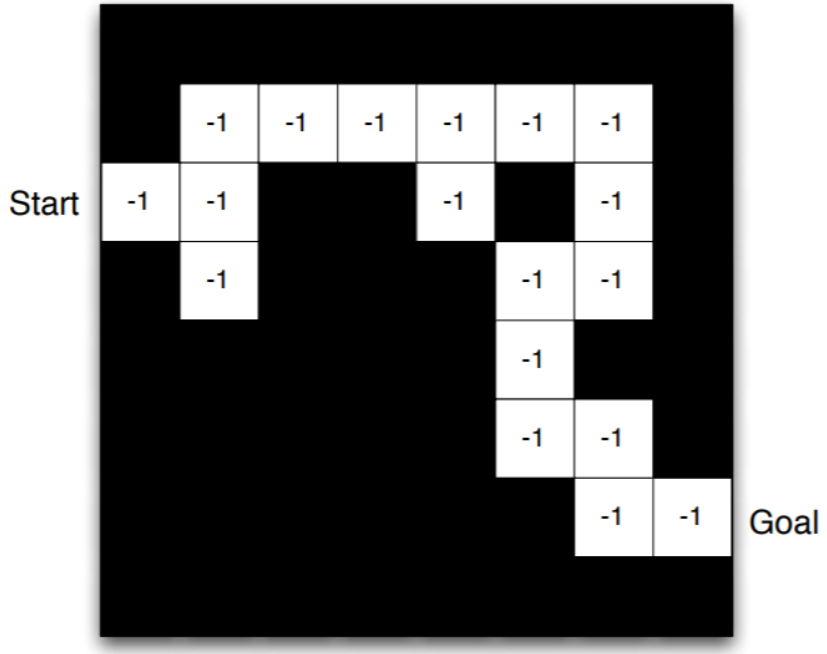


- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Example: Maze (Policy, Value Function, Model)



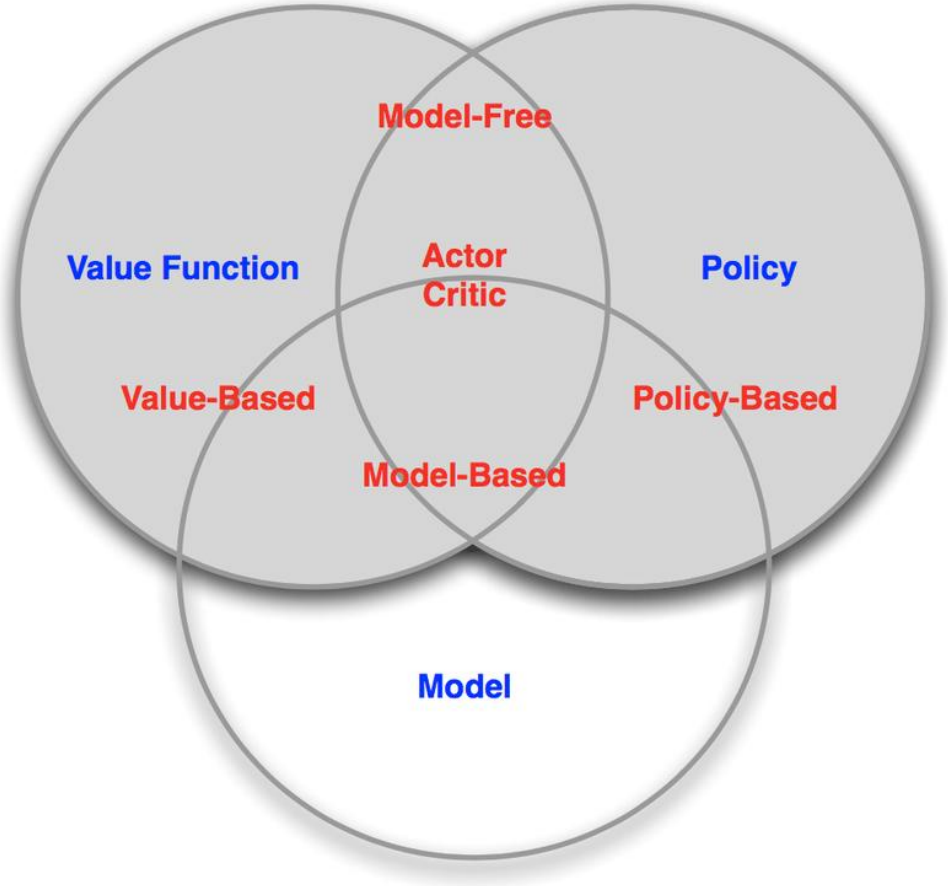




Types of RL Agent (Classes of RL Algorithms)

- Model-Free vs. Model-Based
- Value-Based vs. Policy-Based

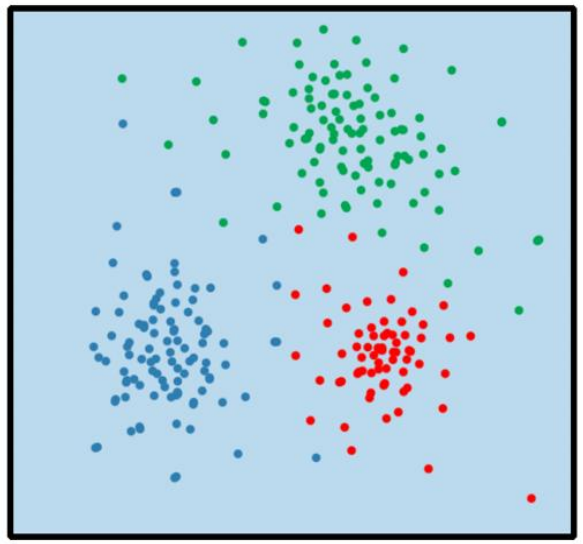
- Actor Critic: hybrid of value-based and policy-based methods



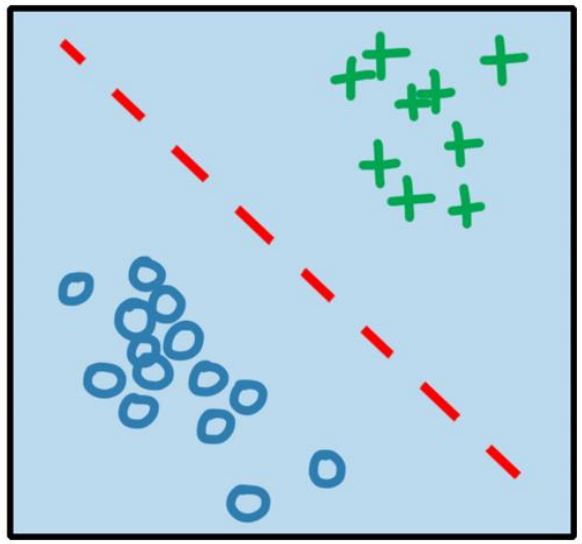
RL vs. SL (Supervised Learning)

Machine Learning Branches

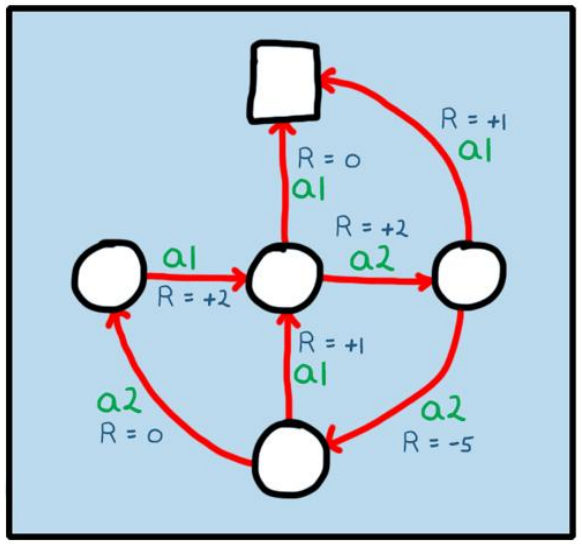
unsupervised learning



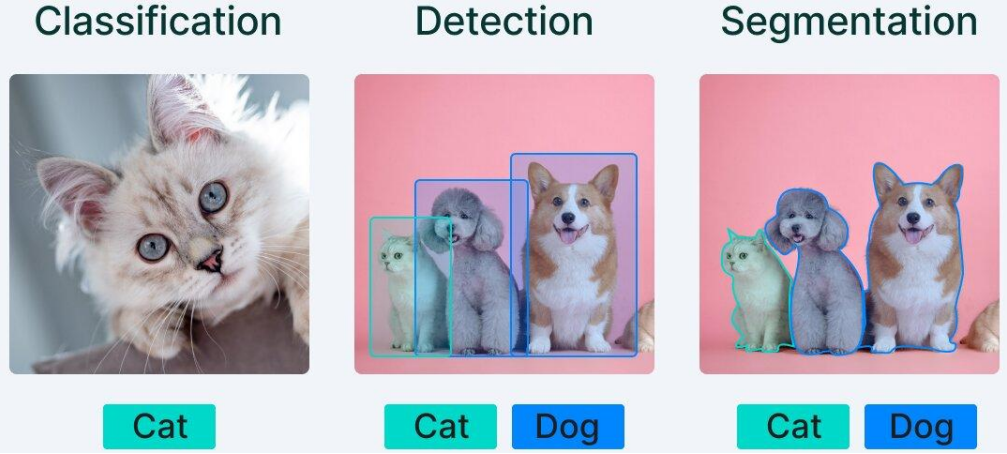
supervised learning

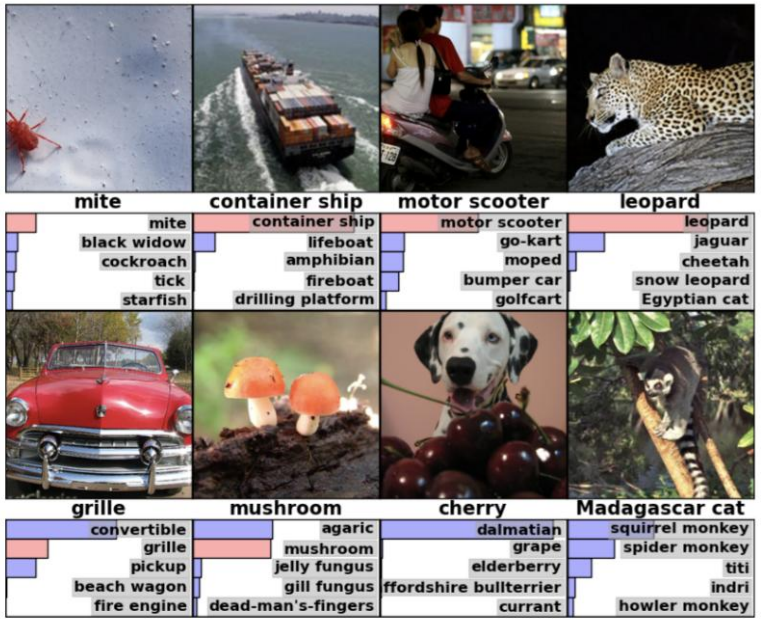


reinforcement learning



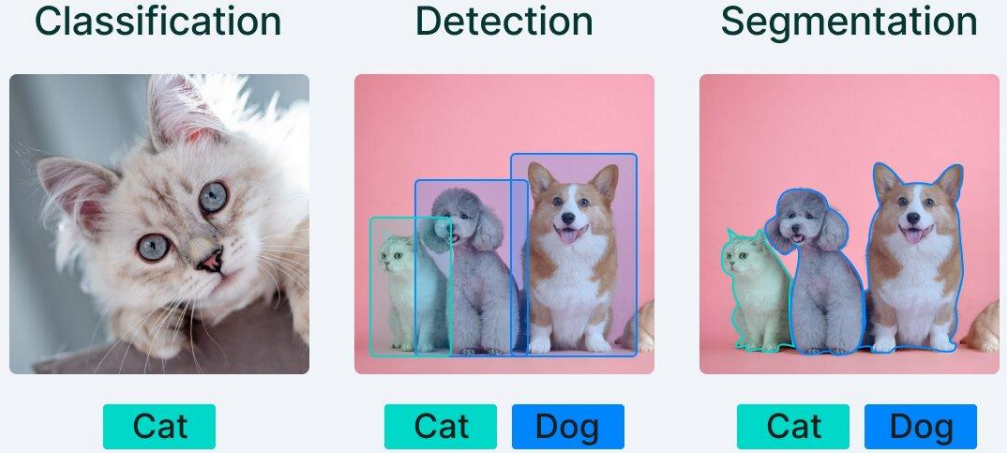
Supervised Learning (SL): Recognition Problem

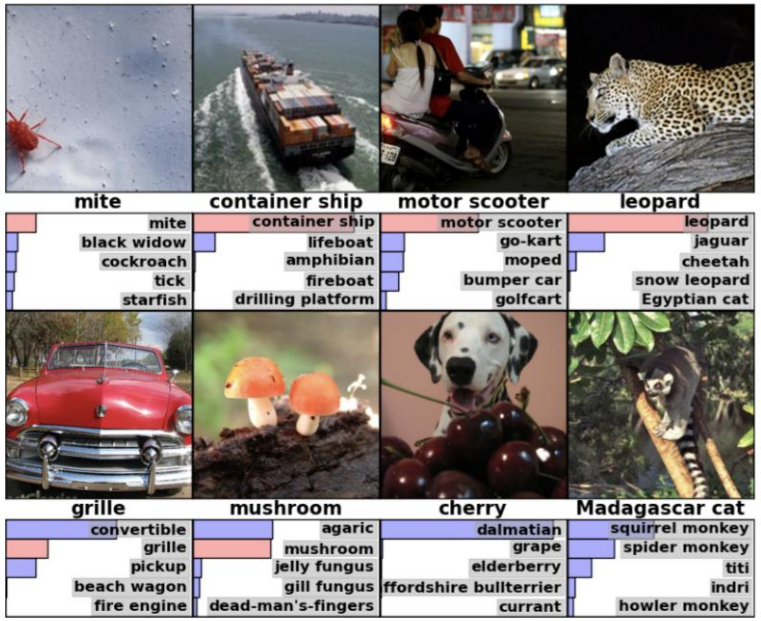




Supervised Learning (SL): Recognition Problem

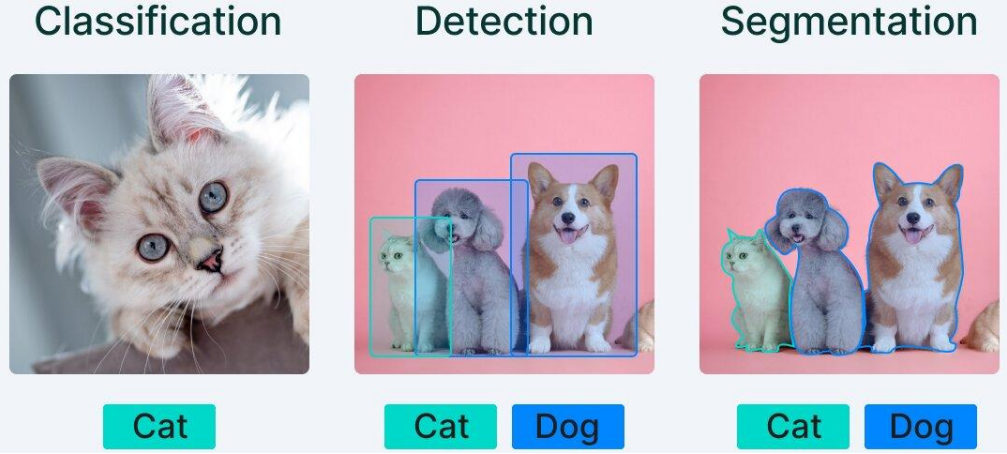
- Task: relies on accurate labels provided by the supervisor

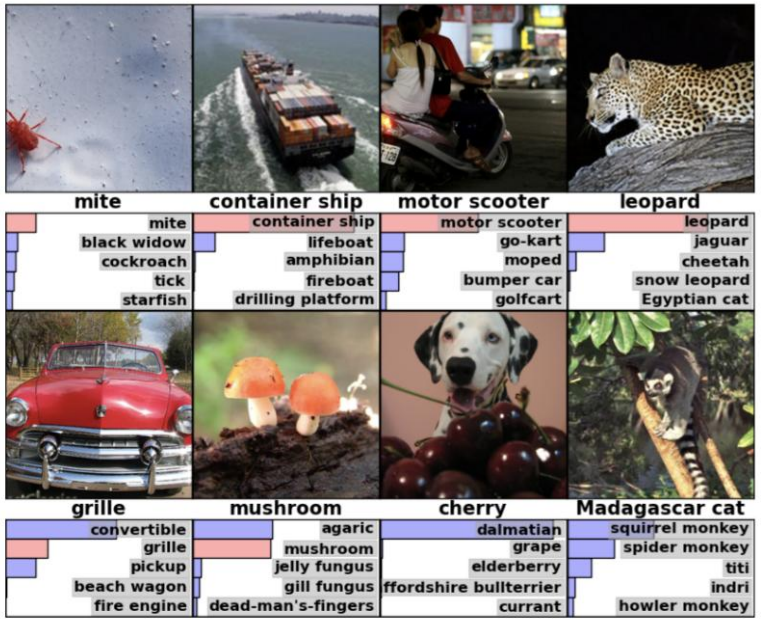




Supervised Learning (SL): Recognition Problem

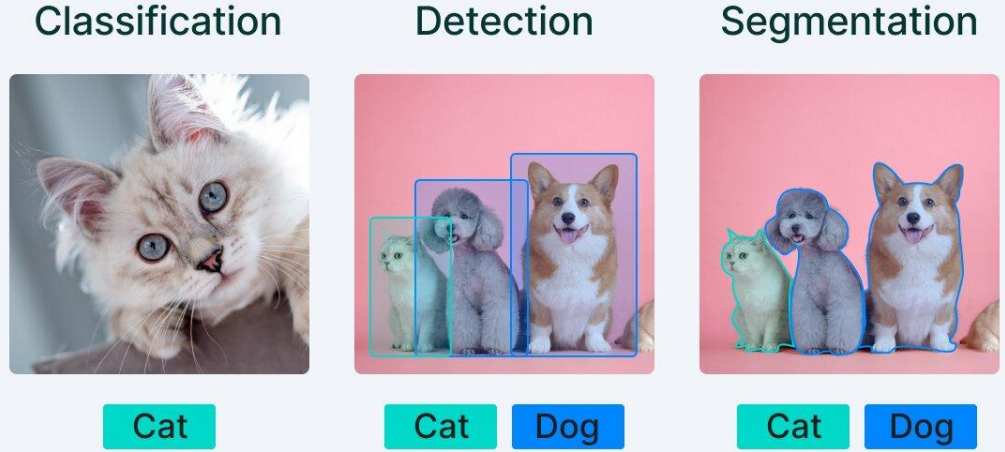
- Task: relies on accurate labels provided by the supervisor
- Assumption: independent, identically distributed (i.i.d.) data

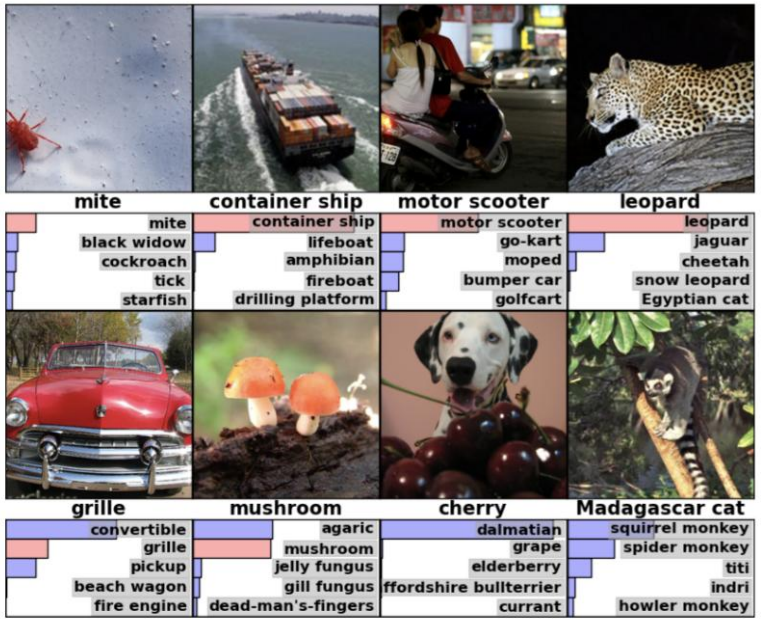




Supervised Learning (SL): Recognition Problem

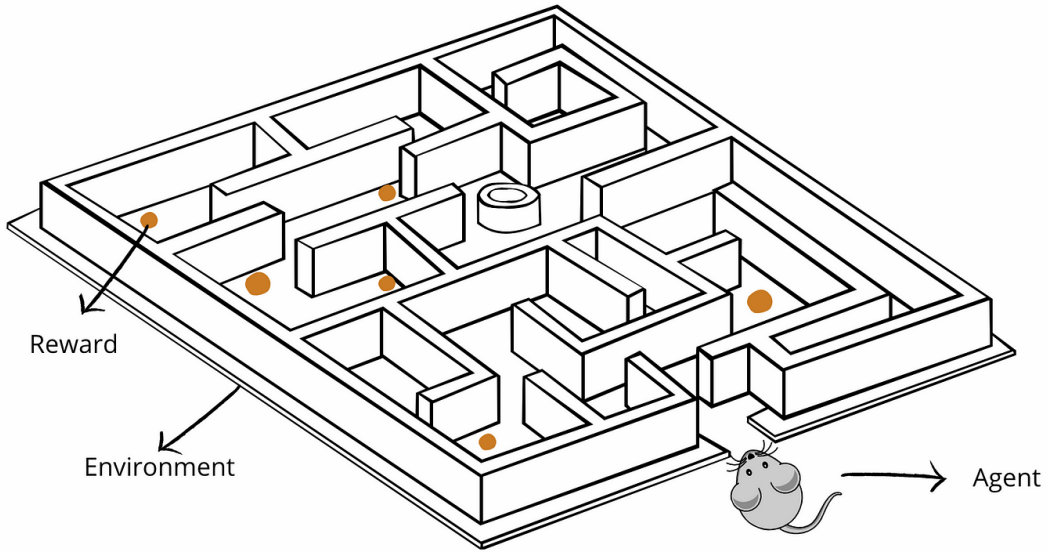
- Task: relies on accurate labels provided by the supervisor
- Assumption: independent, identically distributed (i.i.d.) data
- Challenge: obtain accurately labelled data





Reinforcement Learning (RL): Decision Problem

- Task: learns via trial and error, no supervisor, only (sparse and delayed) rewards



Reinforcement Learning (RL): Decision Problem

- Task: learns via trial and error, no supervisor, only (sparse and delayed) rewards
- Assumption: non-i.i.d. data, action order matters

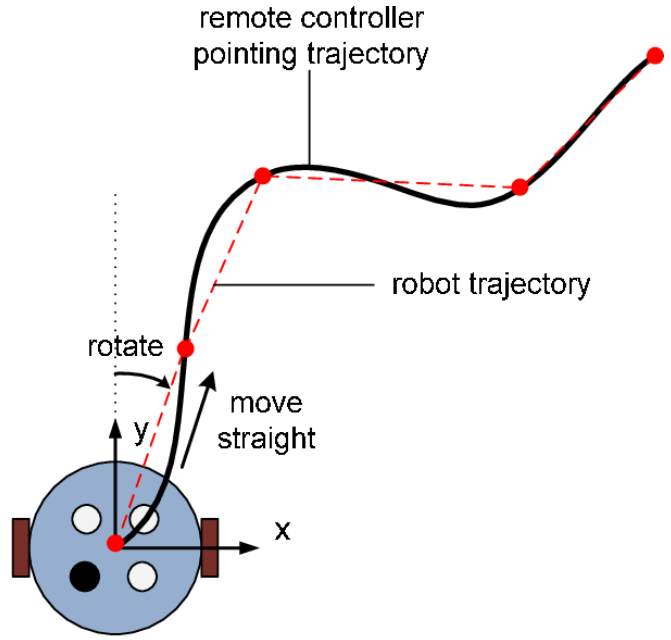


Fig. 5. Mobile robot control algorithm.

Reinforcement Learning (RL): Decision Problem

- Task: learns via trial and error, no supervisor, only (sparse and delayed) rewards
- Assumption: non-i.i.d. data, action order matters
- Challenge: design of action, observation, reward

Reinforcement Learning (RL)

- RL: superhuman potential
- SL: no such potential

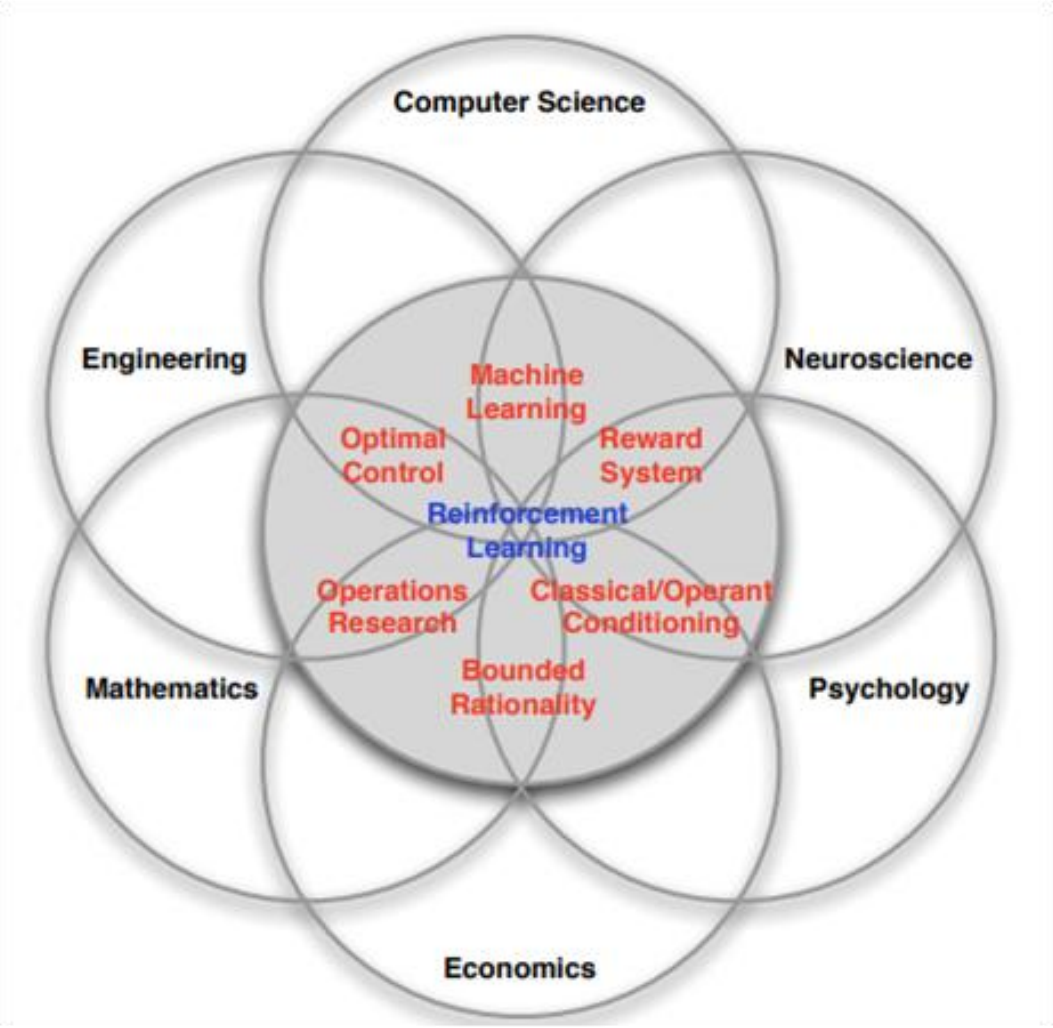
Summary of Class

- What is RL
  - Tool for sequential decision making
- Brief History of RL
  - Books, applications, before and after deep learning
- Key Concepts of RL
  - Agent, Action, Environment, State, Reward
- Components of RL Agent
  - Policy, Value Function, Model
- RL vs. SL
  - Decision task vs. recognition task

End

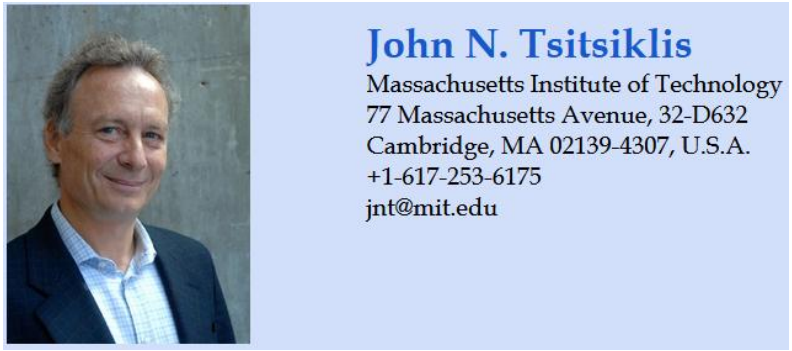
Sequential Decision Making

- Games
- Robotics
- Natural Language Processing
- Recommendation Systems
- Finance, Advertising
- Industrial Automation, Manufacturing
- Logistics, Transportation, Supply Chain
- Healthcare
- Education



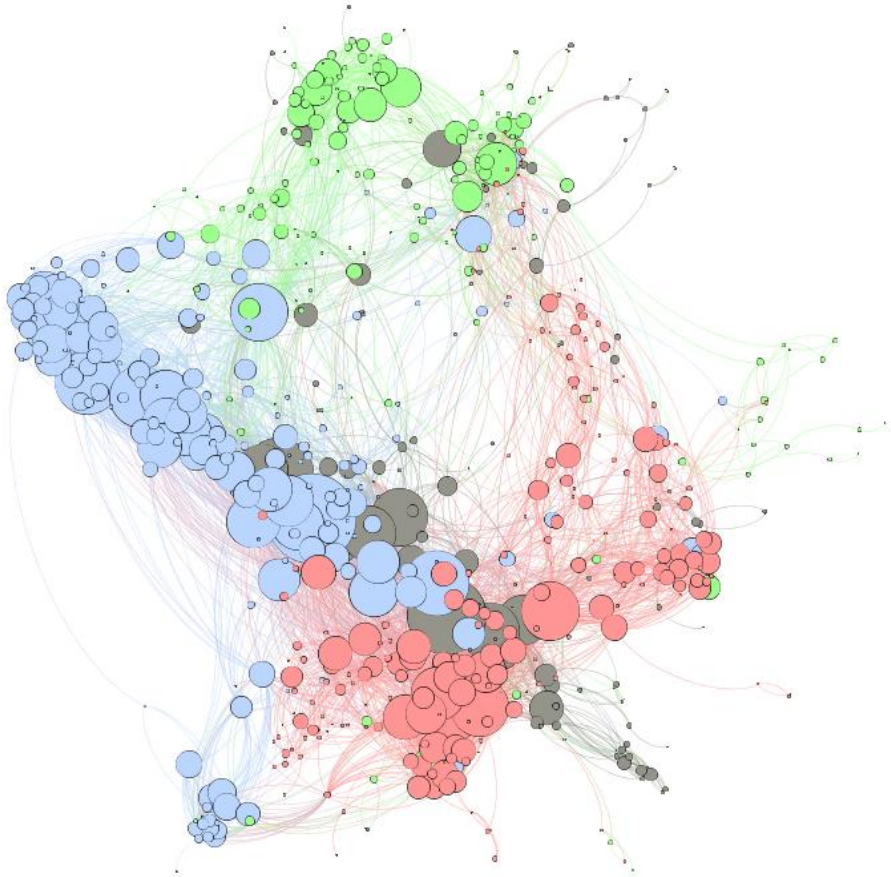
Control Thread

- "Dynamic Programming" by Bellman, 1957
- "Dynamic Programming and Stochastic Control" by Bertsekas, 1976
- "Neuro-Dynamic Programming" by Bertsekas and Tsitsiklis, 1996
  - Also known as "approximate dynamic programming"
  - Now "deep reinforcement learning"



Example: Graph Sparsification

- Agent
- Action
- Environment
- State
- Reward



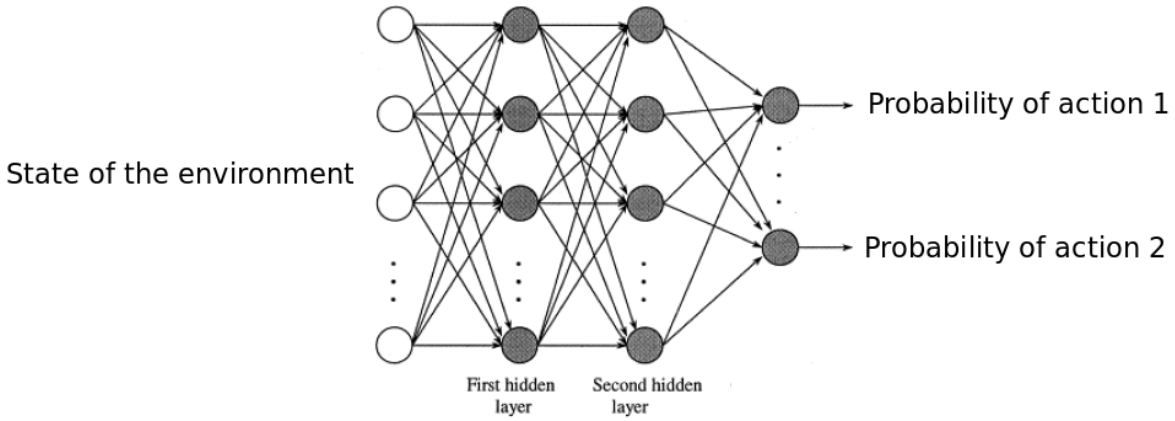
Components of RL Agent

RL Agent

- RL Task: how can an autonomous agent that senses and acts in its environment learn to choose optimal actions sequentially to maximize its cumulative reward
- Goal of RL Agent: perform sequences of actions, observe their rewards, and learn an optimal policy

Components of RL Agent: Policy

- A mapping from states (observations) of the environment to actions that the agent should take
- Deterministic: $\pi(s) = a$
- Stochastic: $\pi(a|s) = P[a_t = a|s_t = s]$



Components of RL Agent: Value Function

- Your friend: should I get MS in CS from UT?
- Actions after graduation
  - Action 1: become software engineer (likelihood 80%)
  - Action 2: become hardware engineer (likelihood 20%)
- Annual salary (fixed)
  - Software engineer: 100K
  - Hardware engineer: 90K
- Value: expected total income 1 year after graduation
- What is the value of having MS in CS from UT?

Components of RL Agent: Value Function

- Your friend: should I get MS in CS from UT?
- Actions after graduation
  - Action 1: become software engineer (likelihood 80%)
  - Action 2: become hardware engineer (likelihood 20%)
- Annual salary (fixed)
  - Software engineer: 100K
  - Hardware engineer: 90K
- Value: expected total income 2 years after graduation
- What is the value of having MS in CS from UT?

Components of RL Agent: Value Function

- Your friend: should I get MS in CS from UT?
- Actions after graduation
  - Action 1: become software engineer (likelihood 80%)
  - Action 2: become hardware engineer (likelihood 20%)
- Annual salary (fixed)
  - Software engineer: 100K
  - Hardware engineer: 90K
- Value: expected total income 5 years after graduation
- What is the value of having MS in CS from UT?

Components of RL Agent: Value Function

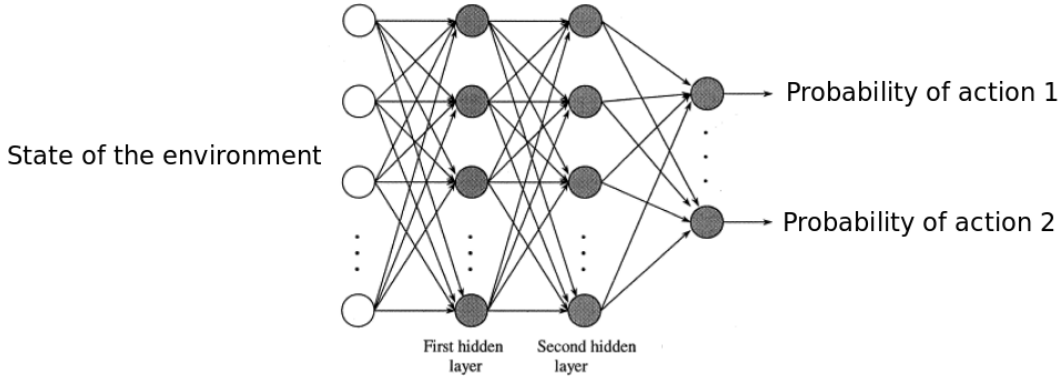
- Value function
  - Estimated cumulative reward starting from a state or a state-action pair
- An example of State Value Function $V(s)$
  - State 1: having MS in CS from UT
  - State 2: not having MS in CS from UT
  - V(State 1): estimated total income x years after having MS in CS from UT
  - V(State 2): estimated total income x years after not having MS in CS from UT

Components of RL Agent: Value Function

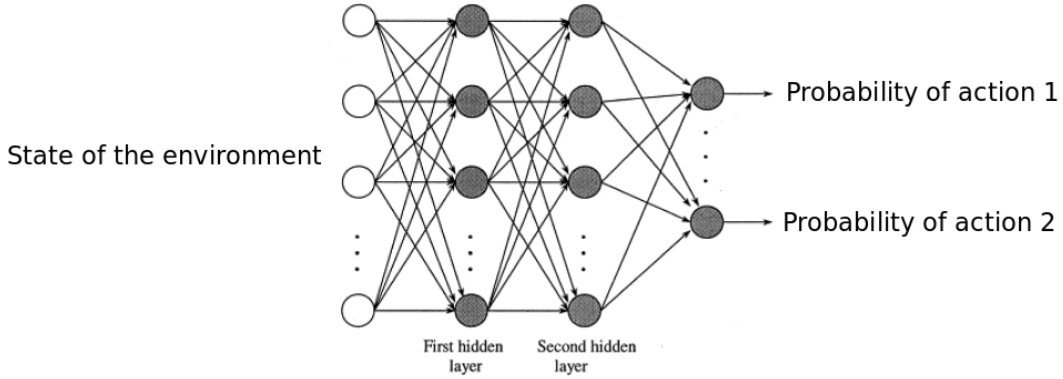
- Value function
  - Estimated cumulative reward starting from a state or a state-action pair
- An example of Action Value Function $Q(s, a)$
  - State + Action 1: having MS in CS from UT + become software engineer
  - State + Action 2: having MS in CS from UT + become hardware engineer
  - Q(State, Action 1): estimated total income x years after being at State and following Action 1
  - Q(State, Action 2): estimated total income x years after being at State and following Action 2

Value Function Connections

- $V(s) = \sum_{a \in A} \pi(a|s) Q(s,a)$



Value Function Connections

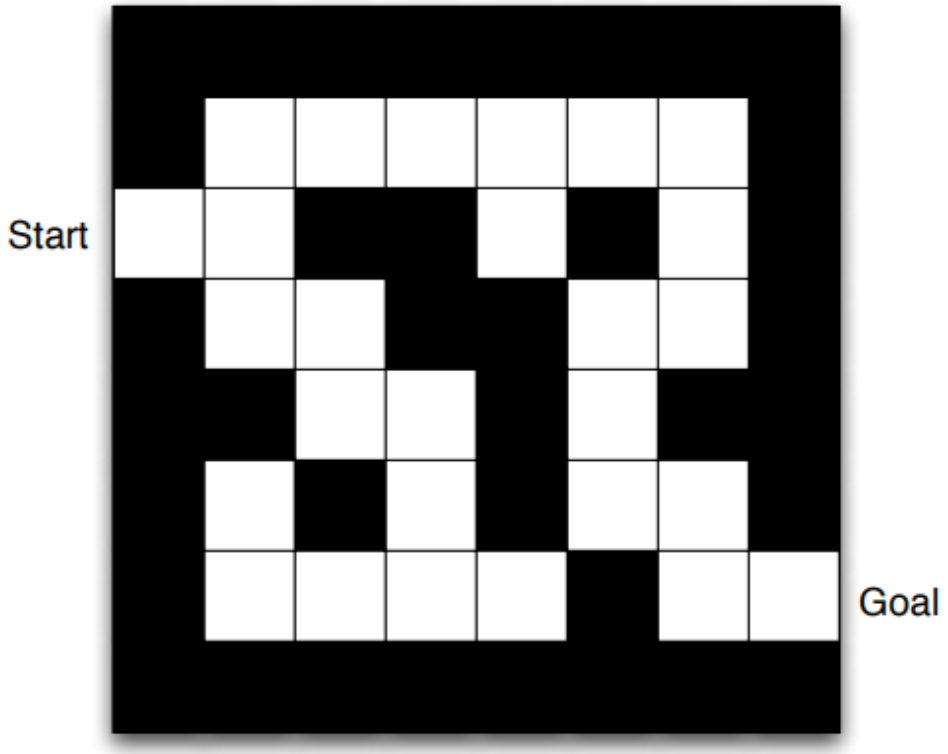


- $V(s) = \sum_{a \in A} \pi(a|s) Q(s,a)$
- Actions
  - Action 1: become software engineer (likelihood 80%); annual salary 100K
  - Action 2: become hardware engineer (likelihood 20%); annual salary 90K
- State value function
  - V(State): estimated total income x years after having MS in CS from UT
- Action value function
  - Q(State, Action 1): estimated total income x years after being at State and following Action 1
  - Q(State, Action 2): estimated total income x years after being at State and following Action 2

Components of RL Agent: Model

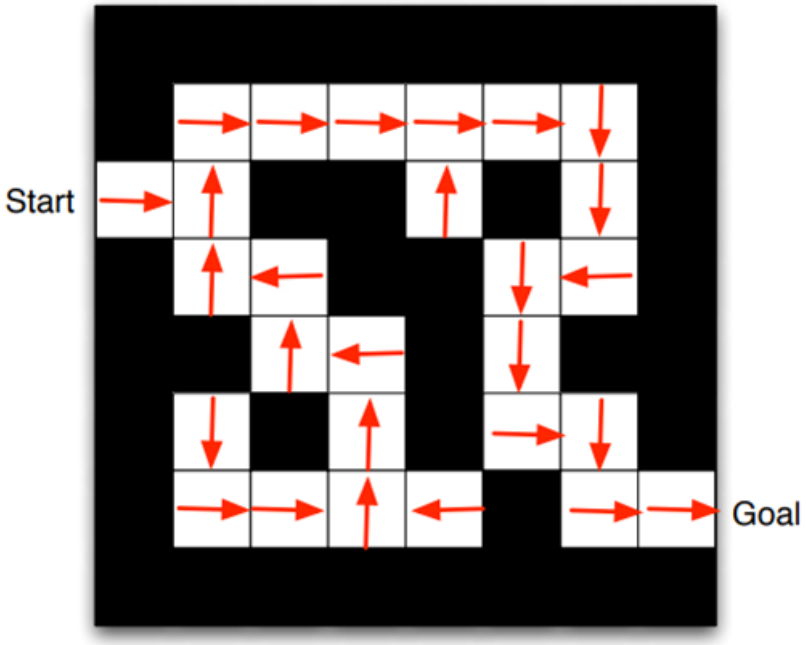
- Agent's knowledge about rules, dynamics, and outcomes (or outcome probabilities) of actions (accurate and complete model = known environment)
- Model is an optional component of RL agent
- Model-free learning: agent doesn't know effects of its actions on the environment (unknown environment)

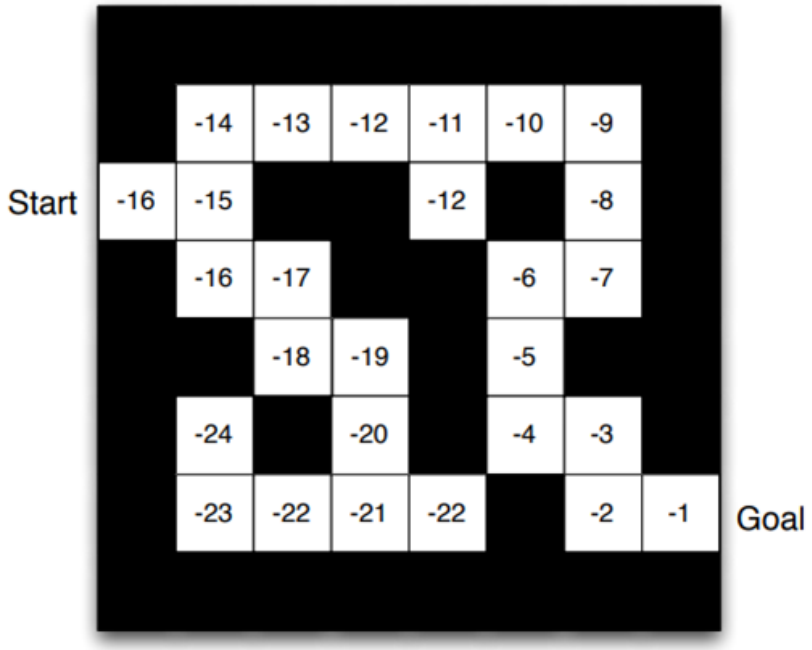
Example: Maze

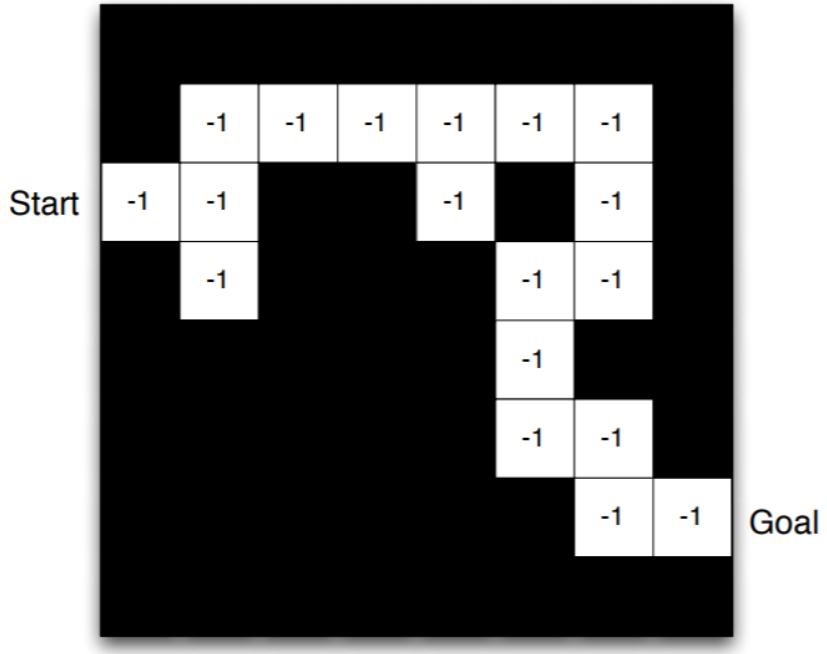


- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Example: Maze (Policy, Value Function, Model)

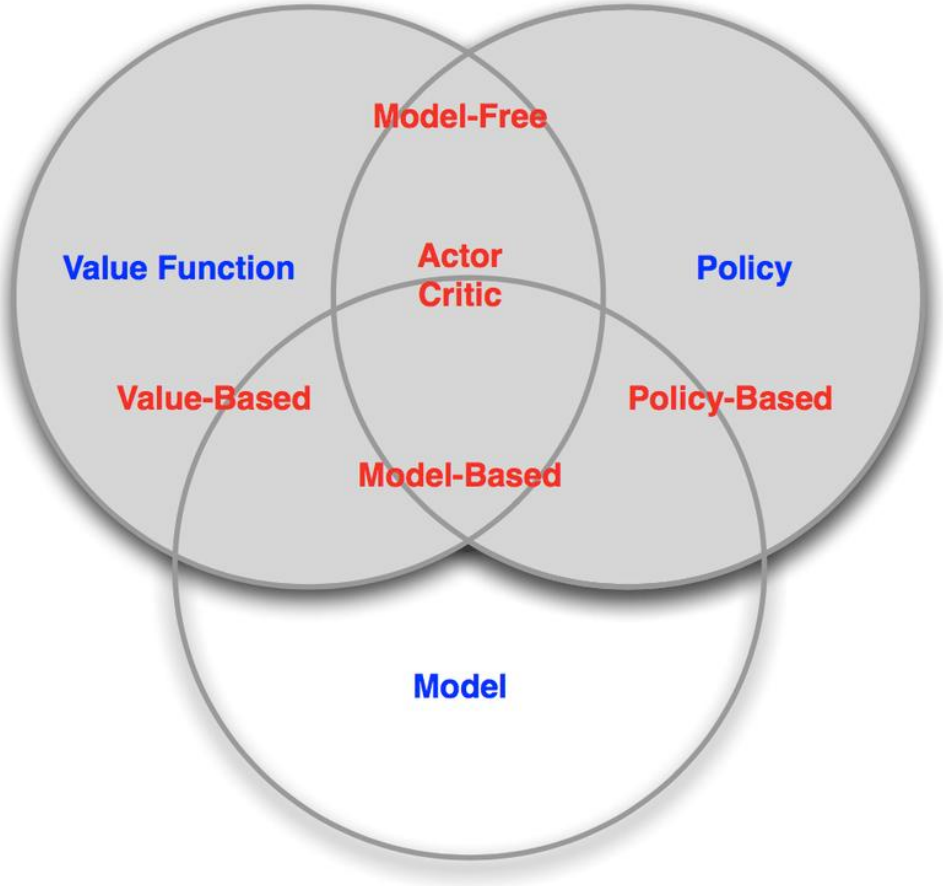






Types of RL Agent

- Model-Free
- Model-Based
- Value-Based
- Policy-Based
- Actor Critic



References

- I. Islam, W. Li, X. Wang, S. Li, and K. Heaslip, “Heterogeneous Mixed Traffic Control and Coordination,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2025.
- D. Wang, W. Li, L. Zhu, and J. Pan, “Learning to control and coordinate mixed traffic through robot vehicles at complex and unsignalized intersections,” International Journal of Robotics Research (IJRR), 2024.
- D. Wang, W. Li, and J. Pan, “Large-scale mixed traffic control using dynamic vehicle routing and privacy-preserving crowdsourcing,” IEEE Internet of Things Journal (IoT-J), vol. 11, no. 2, pp. 1981–1989, 2024.
- B. Poudel, W. Li, and K. Heaslip, “Endurl: Enhancing safety, stability, and efficiency of mixed traffic under real-world perturbations via reinforcement learning,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024.
- B. Poudel, W. Li, and S. Li, “Carl: Congestion-aware reinforcement learning for imitation-based perturbations in mixed traffic control,” International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), 2024.