Reinforcement Learning: Introduction to RL

CSIP5403 – Research Methods and AI Applications

Dr. Nathanael L. Baisa

Lecture Content

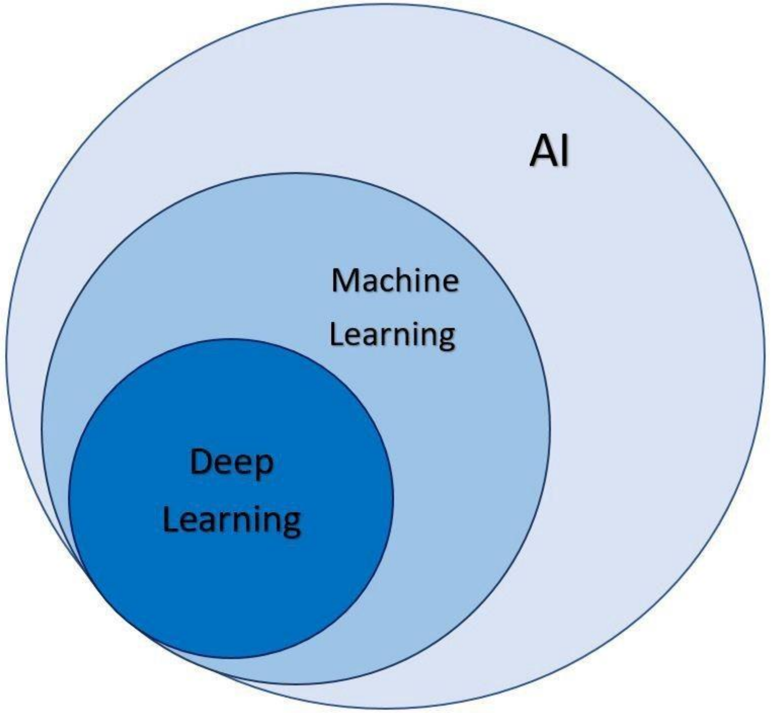
- Introduction
- Characteristics of Reinforcement Learning (RL)
- Sequential Decision Making
- Agent and Environment
- Fully and Partially Observable Environments
- Major Components of a RL Agent
- Categorizing RL Agents
- Exploration and Exploitation
- Learning and Planning
- Conclusion
- References

Session Outcomes

- Acquire basic knowledge of Machine Learning with focus on Reinforcement Learning (RL).
- Understand Agent and Environment in RL.
- Understand the major of components of a RL Agent and categorization of Agents.
- Understand what Learning and Planning mean in Robotics.

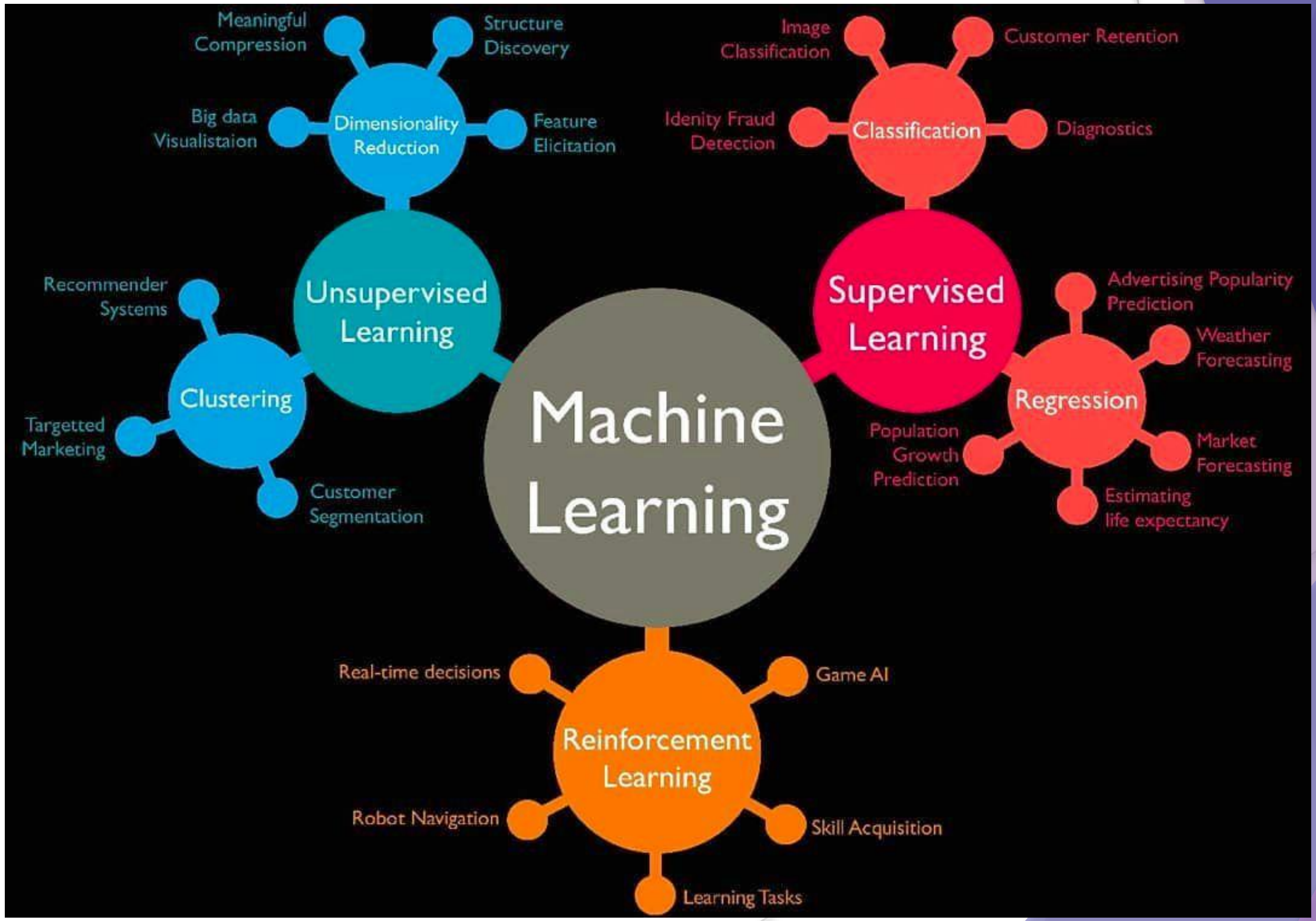
Introduction

- Machine learning (ML) is a core subfield of AI.
- ML is a study of algorithms that enable systems to learn and improve from experience without being explicitly programmed.



Introduction

- ML has 3 main branches for different applications:



Introduction

- **Supervised learning:** learns a model from a labelled training data.
  - Labelled training data: for each input sample data x_i, there is label y_i.
  - Regression – predicts continuous valued output. E.g. linear regression, non-linear regression, etc.
  - Classification - predicts discrete valued output. E.g. logistic regression, perceptron, neural networks, decision trees, SVM, etc.
- **Unsupervised learning:** Unlabelled training data.
  - Discovers hidden patterns in unlabelled data i.e. knowledge discovery. E.g. k-means clustering, principal component analysis (PCA), etc.
- **Reinforcement learning:** No training data.
  - Learns through trial and error. It learns from its mistakes. It uses input from the environment. E.g. Q-learning, Temporal Difference (TD) learning, etc.

Introduction

- Reinforcement learning (RL) allows an AI-driven system (sometimes referred to as an **agent**) to learn through trial and error using feedback from its actions.
- This feedback is either negative or positive, signalled as punishment or reward, with the aim of maximising the reward function.
- RL's goal is to find the most suitable action model to maximise total cumulative reward for the RL agent.
- With no training dataset, the RL problem is solved by the agent's own actions with input from the environment.
- RL mimics natural intelligence as it learns from its mistakes.
- RL algorithms are used in autonomous vehicles, in learning to play a game against a human opponent, etc.
  - Robot Learns to Flip Pancakes: https://www.youtube.com/watch?v=W_gxLKSsSIE

Characteristics of Reinforcement Learning

- What makes reinforcement learning different from other machine learning paradigms?
  - There is no supervisor, only a reward signal. A reward is scalar feedback signal.
  - Feedback is delayed, not instantaneous.
  - Time really matters (sequential, non independent and identically distributed data).
  - Agent's actions affect the subsequent data it receives.
  - The agent's job is to maximise cumulative reward.
- Assume you want to make a humanoid robot walk,
  - What is a positive reward?
  - What is a negative reward?

Characteristics of Reinforcement Learning

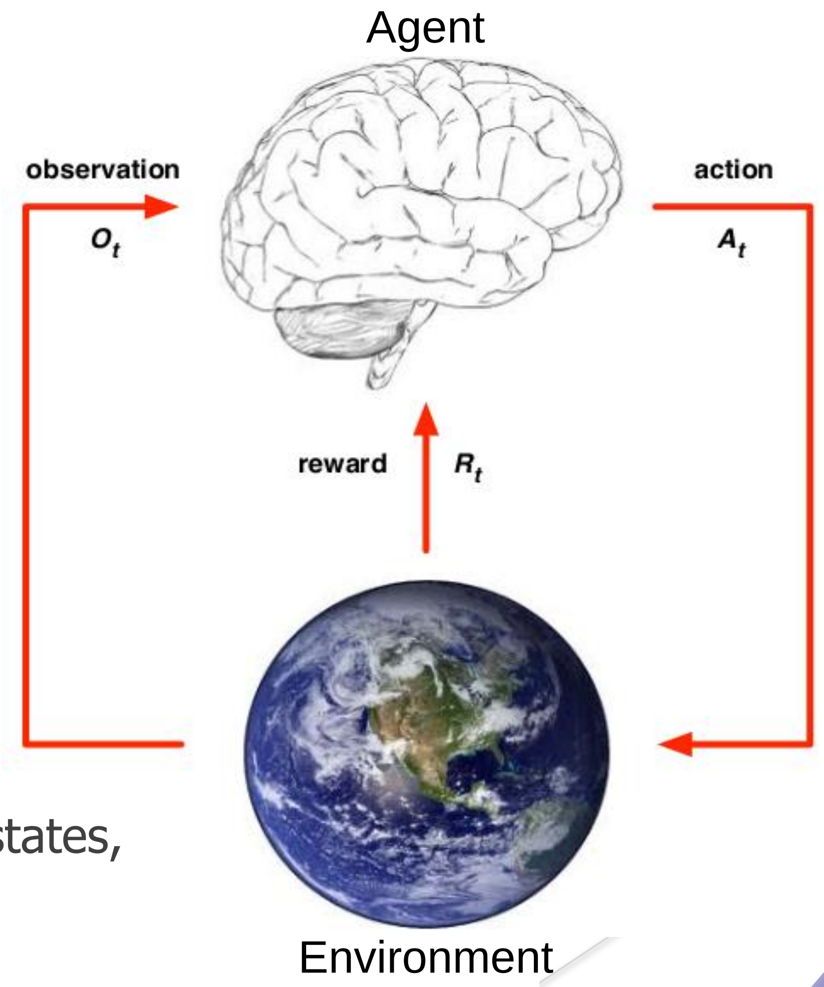
- What makes reinforcement learning different from other machine learning paradigms?
  - There is no supervisor, only a reward signal. A reward is scalar feedback signal.
  - Feedback is delayed, not instantaneous.
  - Time really matters (sequential, non independent and identically distributed data).
  - Agent's actions affect the subsequent data it receives.
  - The agent's job is to maximise cumulative reward.
- Assume you want to make a humanoid robot walk,
  - What is a positive reward? Positive reward for forward motion.
  - What is a negative reward? Negative reward for falling over.

Sequential Decision Making

- **Goal:** select actions to maximise total future reward.
- 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 financial investment (may take months to mature).
  - Refuelling a helicopter (might prevent a crash in several hours).

Agent and Environment

- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at environment step.
- Agent interacts with an environment via states, actions and rewards.



History and State

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

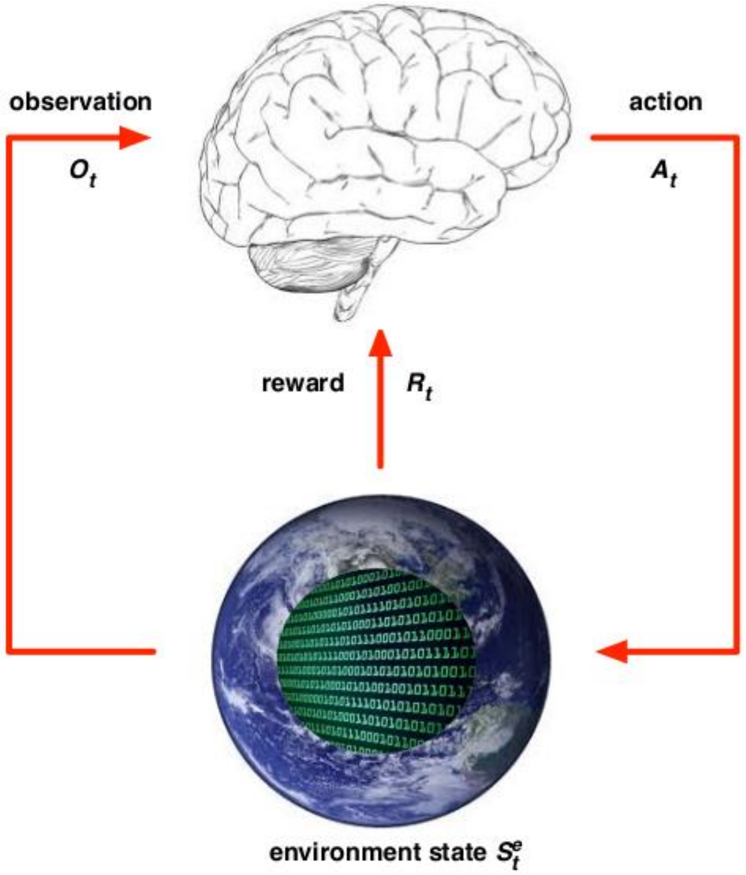
$$H_t = O_1 , R_1 , A_1, ..., A_t , O_t , R_t$$

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

$$S_t = f(H_t)$$

Environment State

- The **environment state** S_t^e is the environment's private representation.
- i.e. whatever data the environment uses to pick the next observation/reward.
- The environment state is not usually visible to the agent.
- Even if S_t^e is visible, it may contain irrelevant information.

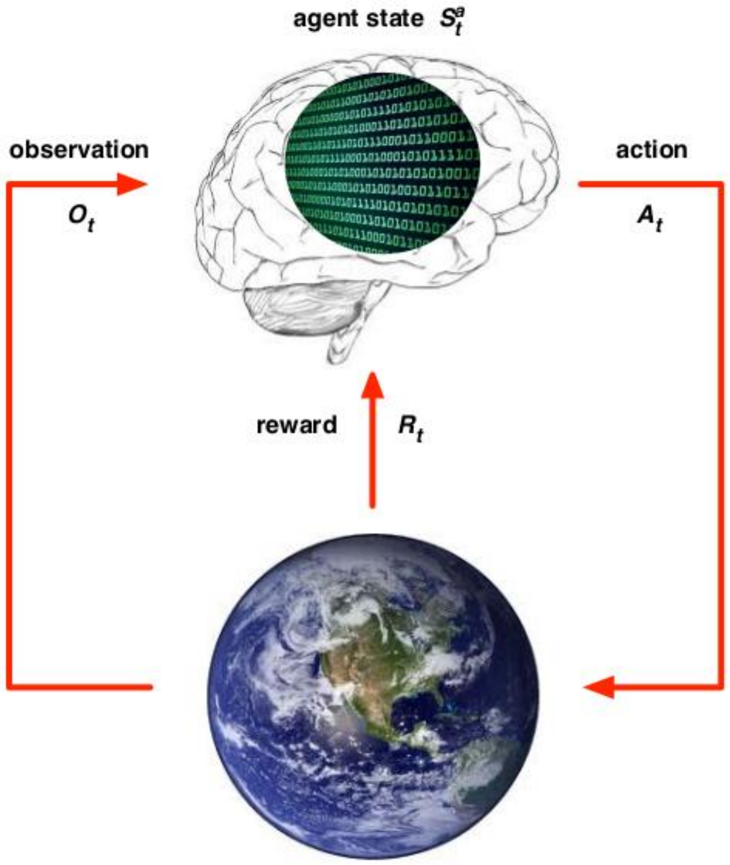


Agent State

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

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

- By performing an action A_t, the agent transitions from state to state.



Information State

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

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

- The future is independent of the past given the present.

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

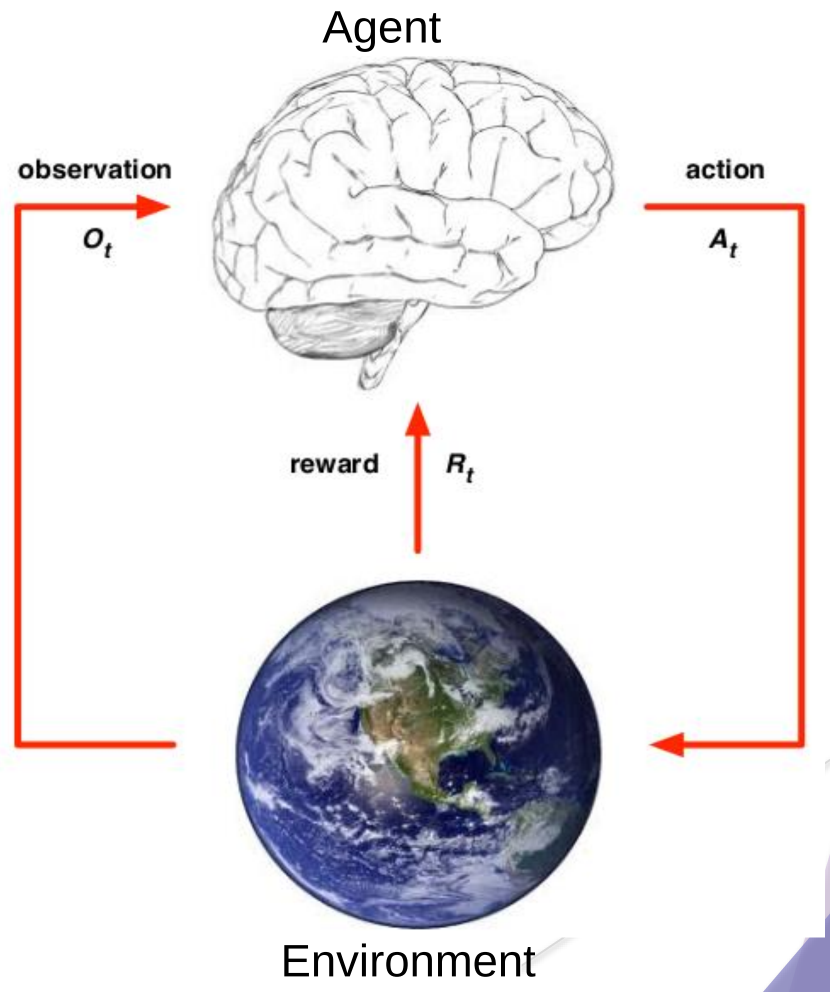
- Once the state is known, the history may be thrown away.
- i.e. the state is a sufficient statistic of the future.
- The environment state S_t^e is Markov.
- The history H_t is Markov.

Fully Observable Environments

- **Full observability:** agent directly observes environment state.

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

- Agent state = environment state = information state.
- Formally, this is a Markov decision process (MDP).



Partially Observable Environments

- **Partial observability:** agent indirectly observes environment:
  - A robot with camera vision isn't told its absolute location.
- Now agent state is not equal environment state.
- Formally, this is a partially observable Markov decision process (POMDP).
- Agent must construct its own state representation S_t^a , e.g.
  - Complete history: $S_t^a = H_t$ OR
  - Beliefs of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], \dots, \mathbb{P}[S_t^e = s^n])$ OR
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Major Components of a RL Agent

- A **RL agent** may include one or more of these components:
  - Policy: agent's behaviour function.
  - Value function: how good is each state and/or action.
  - Model: agent's representation of the environment.

Major Components of a RL Agent

- **Policy:**
  - A policy is the agent's behaviour. It is the strategy which dictates the actions the agent takes as a function of the agent's state as well as the environment.
  - It is a map from state to action, e.g.
    - Deterministic policy: $a = \pi(s)$
    - Stochastic policy: $\pi(a|s) = P[A_t = a|S_t = s]$

Major Components of a RL Agent

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

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

$0 < \gamma < 1$ is discount factor

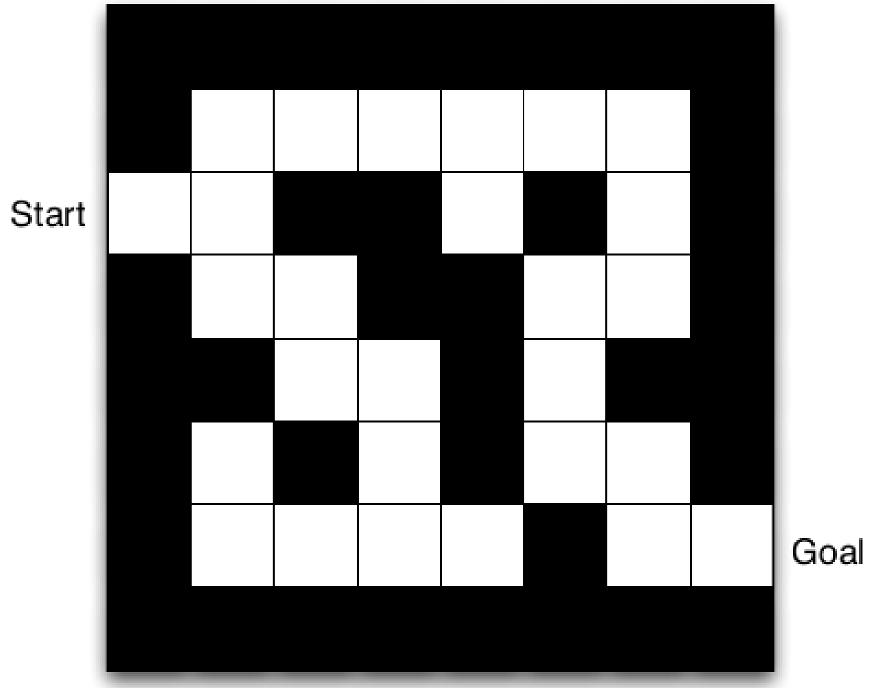
Major Components of a RL Agent

- **Model:**
  - A model predicts what the environment will do next.
  - Transitions: $\mathcal{P}$ predicts the next state.
  - Rewards: $\mathcal{R}$ predicts the next (immediate) reward, e.g.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

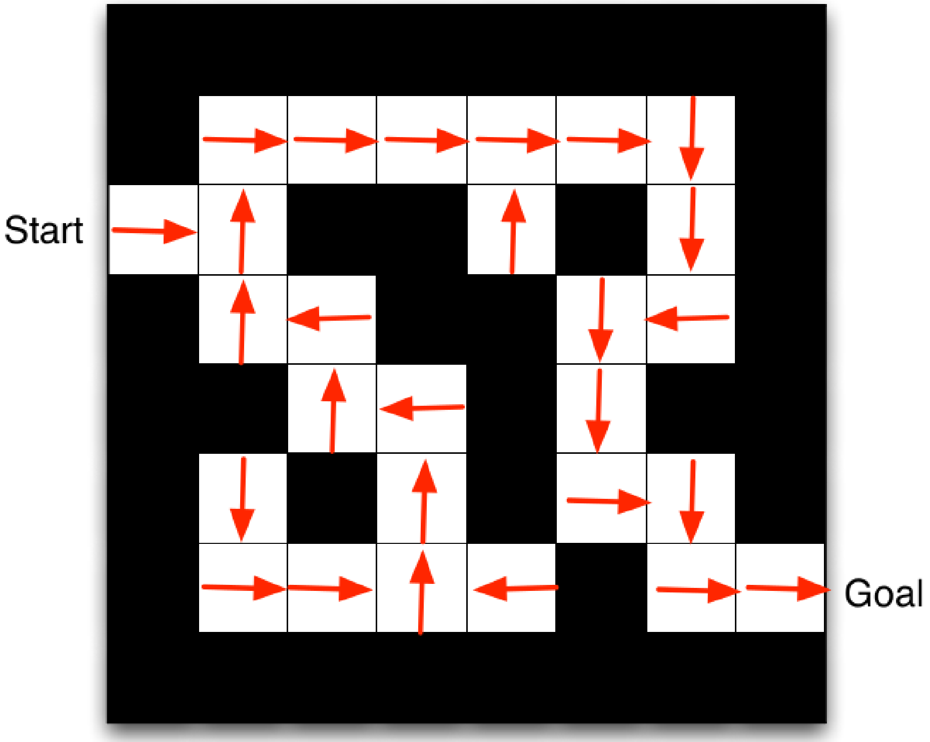
Major Components of a RL Agent

- **Maze Example:**
  - A **maze** is a path or collection of paths, typically from an entrance to a goal.
  - Rewards: -1 per time-step.
  - Actions: N, E, S, W.
  - States: Agent's location.



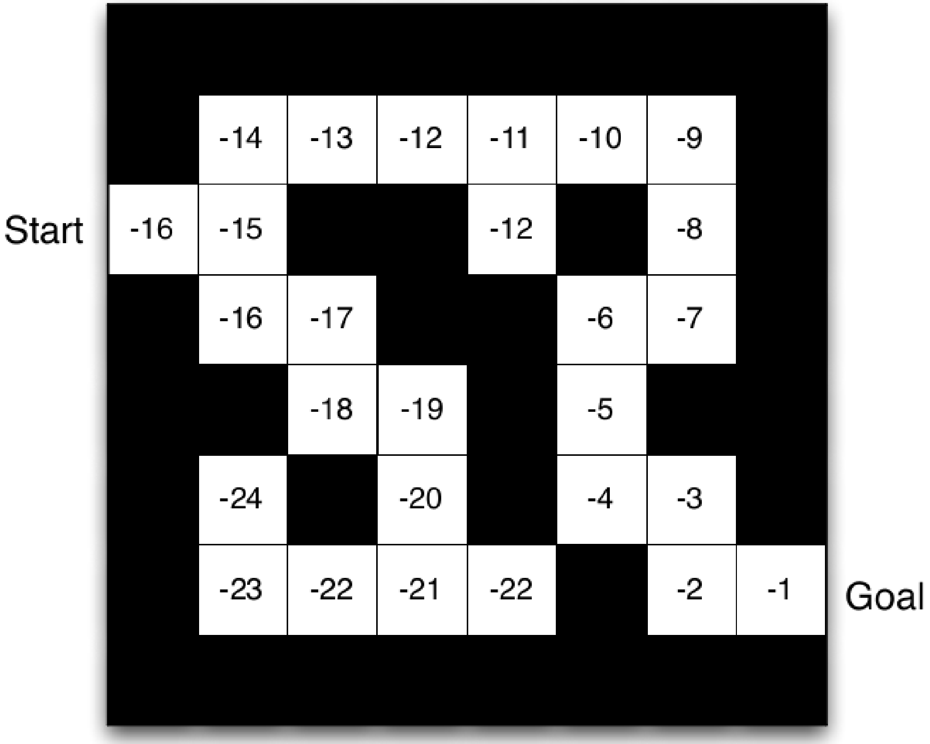
Major Components of a RL Agent

- **Maze Example: Policy**
  - Arrows represent policy π(s) for each state s.



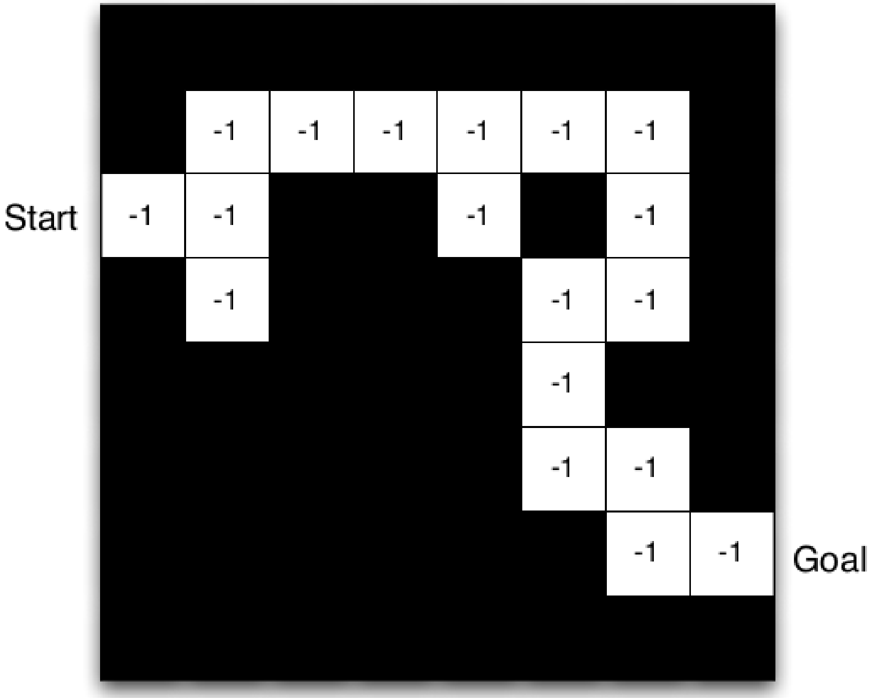
Major Components of a RL Agent

- **Maze Example:** **Value Function**
  - Numbers represent value $v_\pi(s)$ of each state s.



Major Components of a RL Agent

- **Maze Example: Model**
  - Agent may have an internal model of the environment.
  - Dynamics: how actions change the state.
  - Rewards: how much reward from each state.
  - The model may be imperfect.
  - Grid layout represents transition model $\mathcal{P}^a_{ss'}$
  - Numbers represent immediate reward $\mathcal{R}^a_s$ from each state s (same for all a).



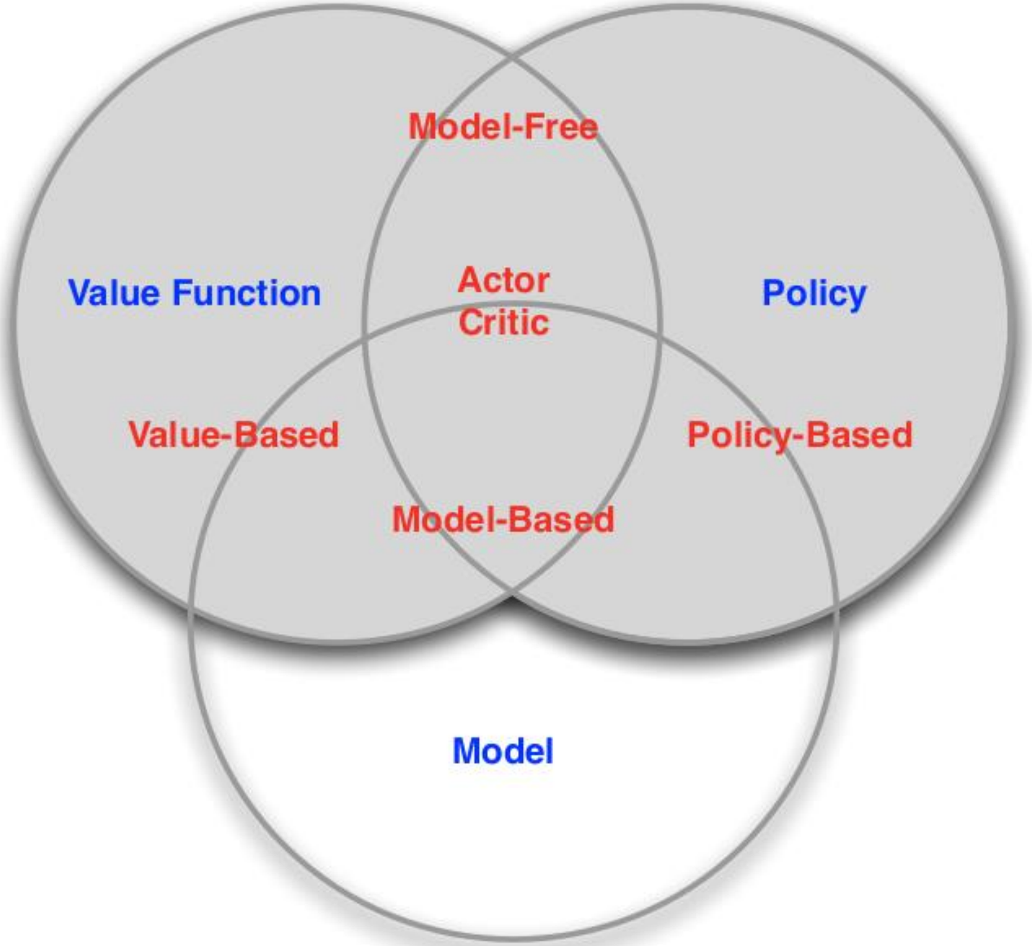
Categorizing RL Agents

- Based on policy and value function:
  - Value Based
    - No Policy (Implicit)
    - Value Function
    - E.g. Q-Learning, TD Learning, Monte Carlo, SARSA (State–action–reward–state–action), etc.
  - Policy Based
    - Policy
    - No Value Function
    - E.g. REINFORCE, Proximal Policy Optimization (PPO), etc.
  - Actor Critic
    - Policy
    - Value Function
    - E.g. Asynchronous Advantage Actor-Critic (A3C), etc.

Categorizing RL Agents

- Based on existence of model of the environment:
  - Model Free
    - Policy and/or Value Function.
    - No Model.
    - E.g. Q-Learning, TD Learning, Monte Carlo, SARSA, A3C, etc.
  - Model Based
    - Policy and/or Value Function.
    - Model.
    - Has limited applications in practice.
    - E.g. Dynamic Programming (DP), etc.

RL Agent Taxonomy



Learning and Planning

- **Two fundamental problems** in sequential decision making.
  - Reinforcement Learning:
    - The environment is initially unknown.
    - The agent interacts with the environment.
    - The agent improves its policy.
  - Planning:
    - A model of the environment is known.
    - The agent performs computations with its model (without any external interaction).
    - The agent improves its policy.
    - a.k.a. deliberation, reasoning, introspection, pondering, thought, search.

Exploration and Exploitation

- Reinforcement learning is like trial-and-error learning.
- The agent should discover a good policy.
  - From its experiences of the environment.
  - Without losing too much reward along the way.
- Exploration finds more information about the environment.
- Exploitation exploits known information to maximise reward.
- It is usually important to explore as well as exploit.
- Example:
  - Oil drilling
    - Exploitation: Drill at the best known location.
    - Exploration: Drill at a new location.

Prediction and Control

- **Prediction:** evaluate the future.
  - Given a policy.
- **Control:** optimise the future.
  - Find the best policy.

Conclusion

- There are 3 main branches of machine learning: supervised, unsupervised and reinforcement learning.
- Reinforcement learning (RL) allows an **agent** to learn through trial and error using feedback from its actions.
- RL's goal is to find the most suitable action model to maximise total cumulative reward for the RL agent.
- RL mimics natural intelligence as it learns from its mistakes.
- The major components of a RL agent are policy, value function and model.
- RL algorithms are used in autonomous vehicles, in learning to play a game against a human opponent, etc.

References

- R. Sutton and A. Barto, 'Reinforcement Learning: An Introduction, 2nd Edition', MIT press, 2018.
- C. Szepesvari, 'Algorithms for Reinforcement Learning', Morgan & Claypool Publishers, 2010.
- https://online.york.ac.uk/what-is-reinforcement-learning/
- https://github.com/dennybritz/reinforcement-learning
- Awesome RL: https://github.com/aikorea/awesome-rl