Cornell Bowers C·IS
College of Computing and Information Science

Deep Learning

Week [8]: [MDPs/Q-Learning]

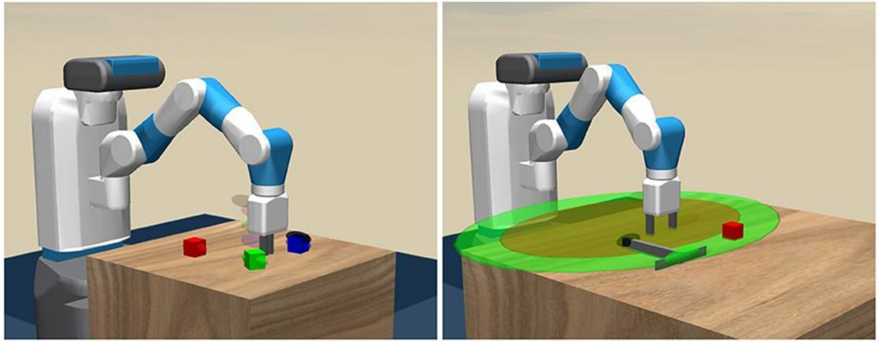
Logistics

- Assignment #4 Feedback Form due Friday
    - Will release this evening

Cornell Bowers C·IS

Limitations of Supervised Learning

Can a regression/classification algorithm learn to perform these tasks successfully?

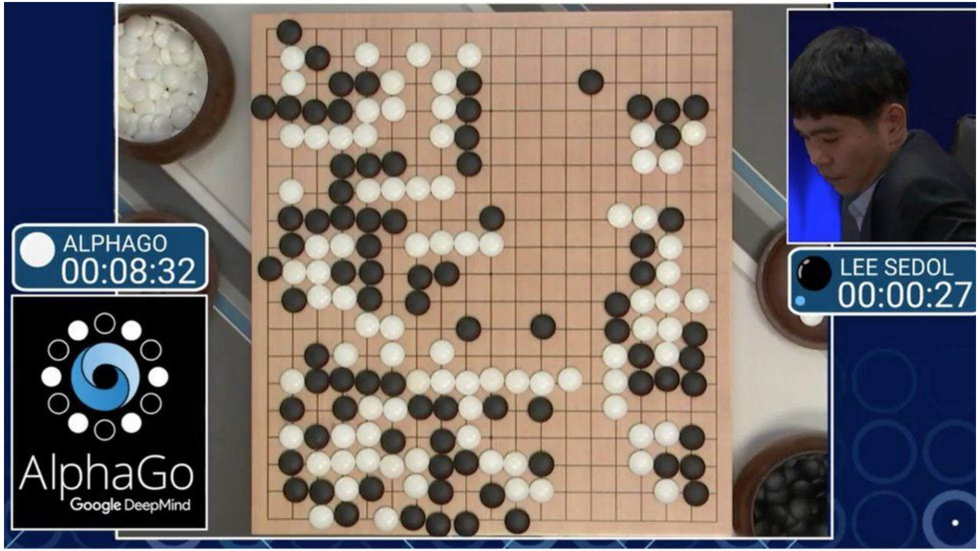


Robot learning to pick up blocks

Waymo self-driving car

Cornell Bowers C·IS

Using Reinforcement Learning to play games



Cornell Bowers C·IS

Supervised Learning

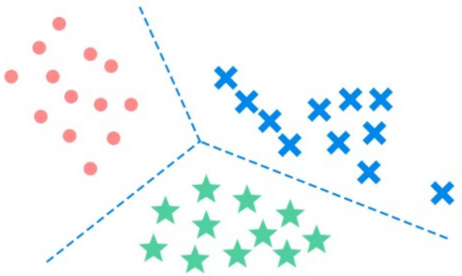
- Learns from a labeled dataset
- Classification
  Regression

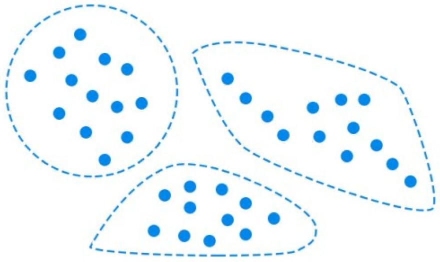
Unsupervised Learning

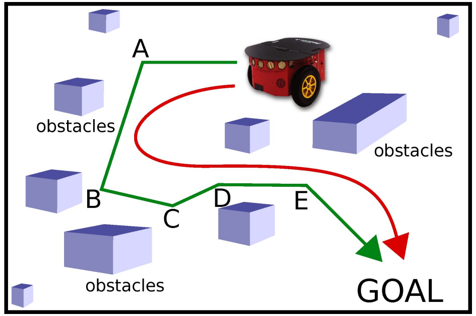
- Find patterns in unlabeled data
- Clustering
  Generative Models

Reinforcement Learning

- Agent interacts with an environment and learns to maximize a reward
- Game Playing
  Robot Navigation





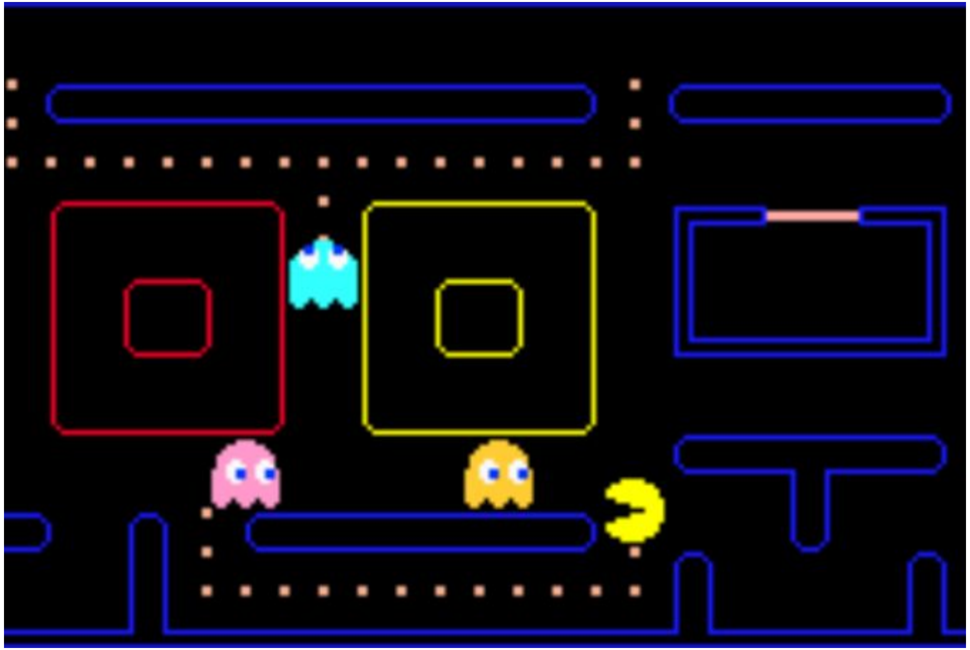


Pacman Example

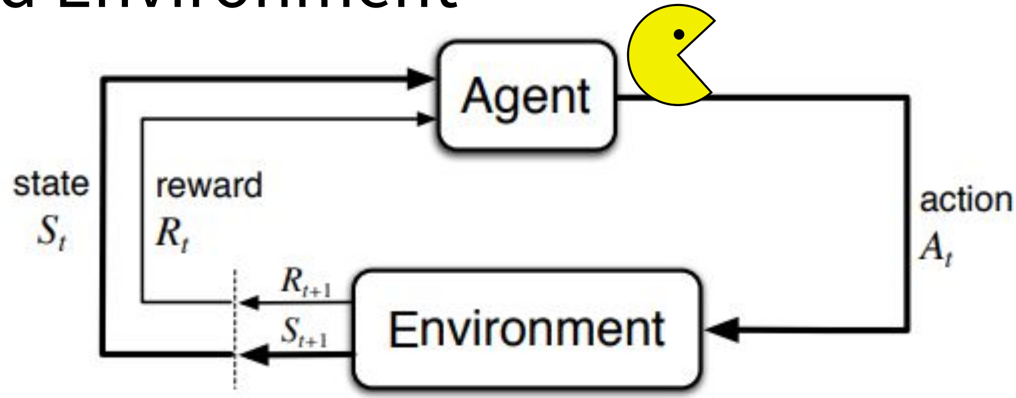
Objective of Game: Eat the most pellets

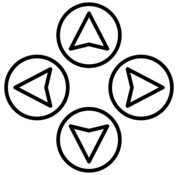
Want to **maximize reward** (pellets)

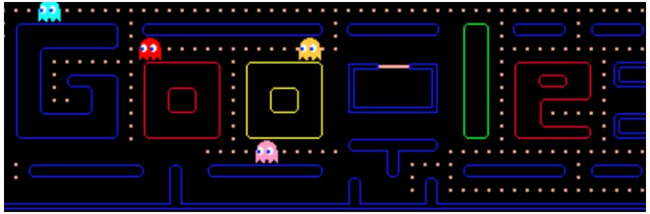
Not differentiable!



Agent and Environment







Agent:

- Perceives environment.
- Makes decisions.
- Aims to maximize reward.

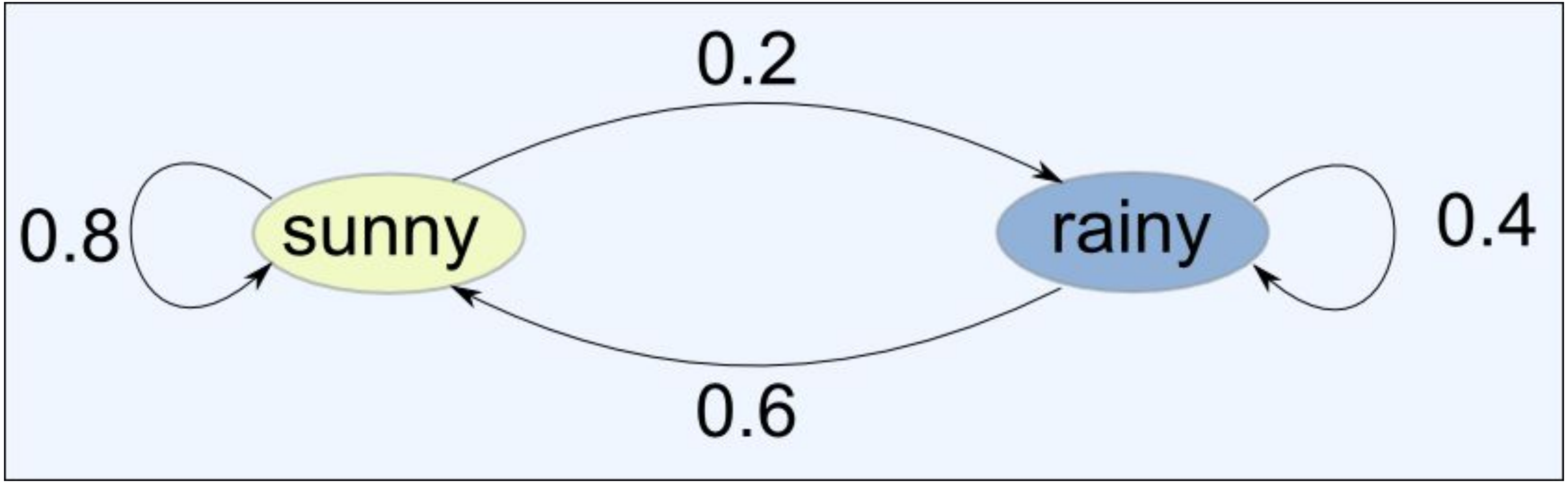
Environment:

- The external context in which an agent operates and interacts with
- Provides feedback to agent

Cornell Bowers C·IS

Markov Process

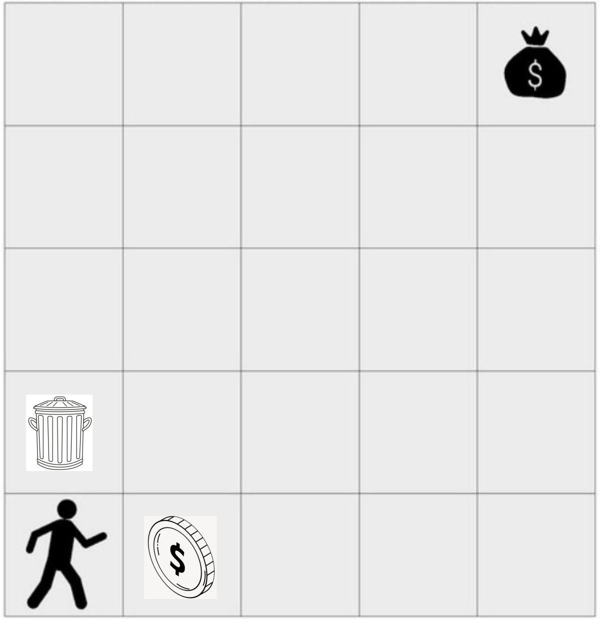
Probabilistic events in which state at time $t+1$ solely depends on state at time t



Cornell Bowers CIS

Markov Decision Process (MDP)

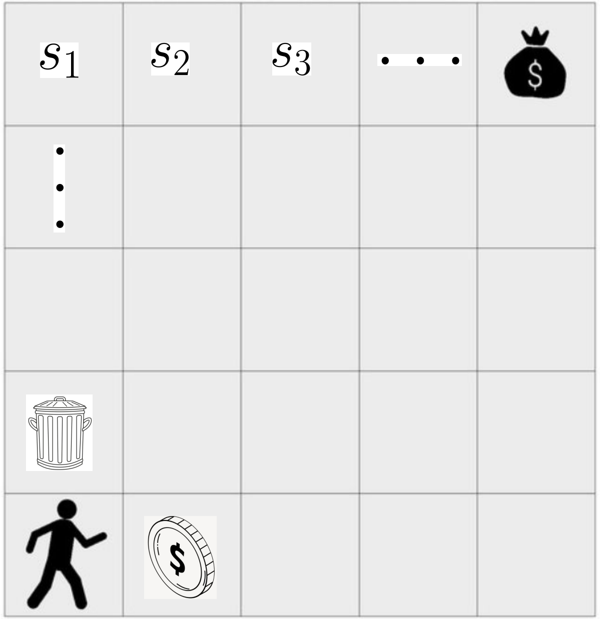
Framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision maker



State Space

$$S = \{s_1, s_2, s_3, \ldots\}$$

– $\mathcal{S}$: Set of states representing the environment.

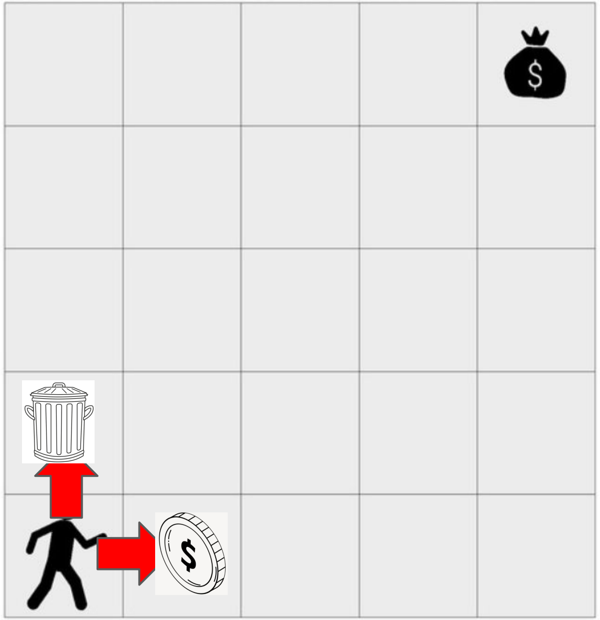


Cornell Bowers C·IS

Action Space

$$\mathcal{A} = \{\text{Up, Down, Left, Right}\}$$

– $\mathcal{A}$: Set of actions the agent can take.

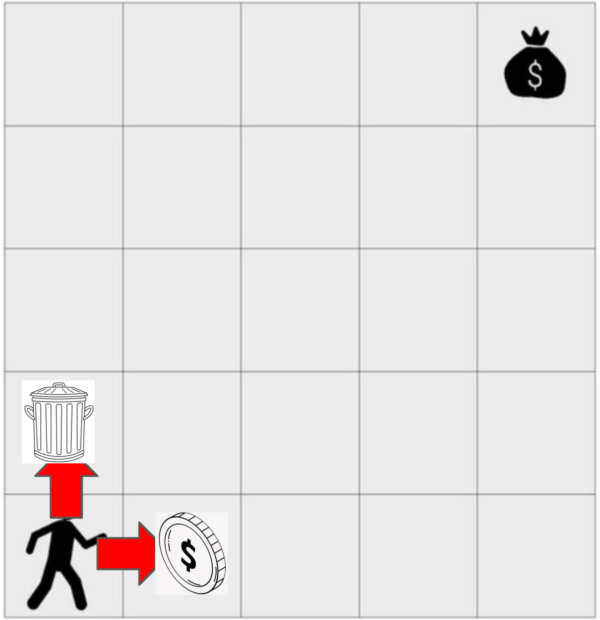


Cornell Bowers C·IS

Transition Function

– $\mathcal{P}$: Transition probability function, $\mathcal{P}(s'|s, a)$.

Probability distribution of the next state given the current state and action

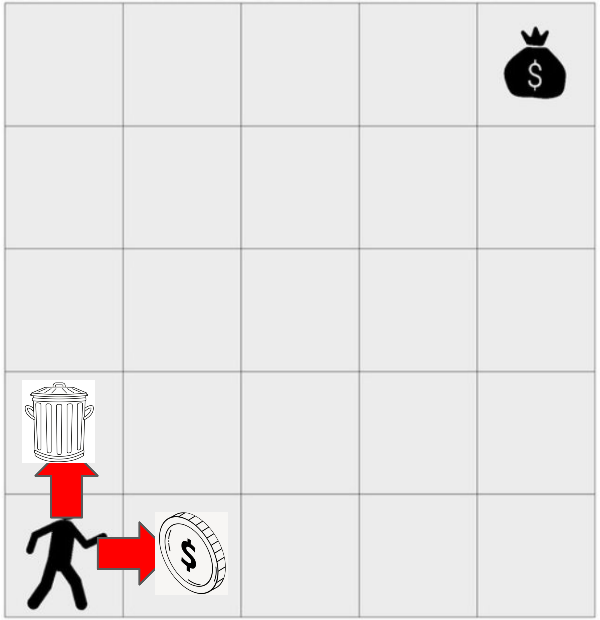


Cornell Bowers CIS

Reward Function

– $\mathcal{R}$: Reward function, $\mathcal{R}(s, a)$.

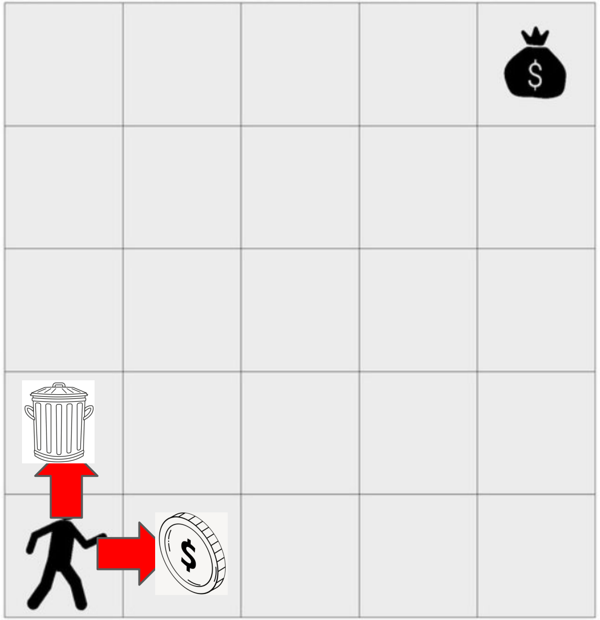
Action	Reward
Up	-5
Right	10



Discount Factor

– $\gamma \in [0, 1]$: Discount factor that balances immediate and future rewards.

 now versus  later

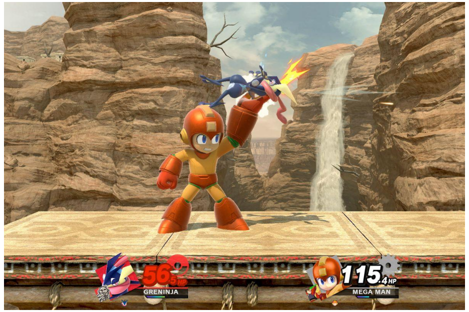


Markov Decision Process (MDP)

- MDPs provide a framework for modeling sequential decision-making problems.
- An MDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$:
  - $\mathcal{S}$: Set of states representing the environment.
  - $\mathcal{A}$: Set of actions the agent can take.
  - $\mathcal{P}$: Transition probability function, $\mathcal{P}(s'|s, a)$.
  - $\mathcal{R}$: Reward function, $\mathcal{R}(s, a)$.
  - γ: Discount factor, $\gamma \in [0, 1]$.

Discussion: Identify the **agent**, **environment** and **reward** in each game.

Tetris



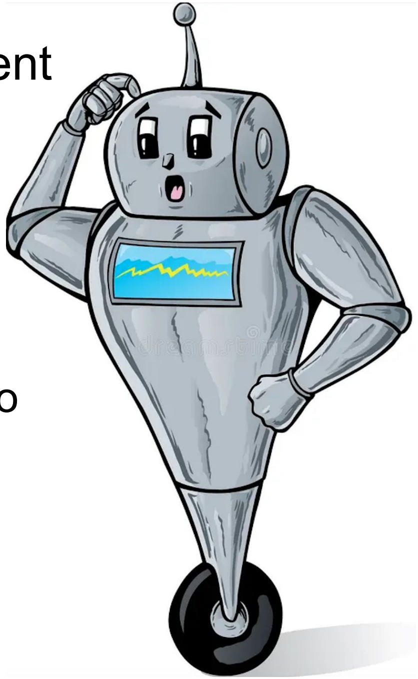
Super Smash Bros

Chess

Markov Decision Process (MDP)

An MDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

Framework for modeling agent in environment



How can we find a **policy** to maximize reward?

Cornell Bowers C·IS

Policy

- A policy π is a mapping from states to actions, defining the agent's behavior.
  - Formally, a policy is a function $\pi : \mathcal{S} \rightarrow \mathcal{A}$.
- The policy determines which action the agent takes in each state.
- The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected cumulative reward.

Policy

- The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected cumulative reward.
- The expected cumulative reward is defined as:

$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$$

where:

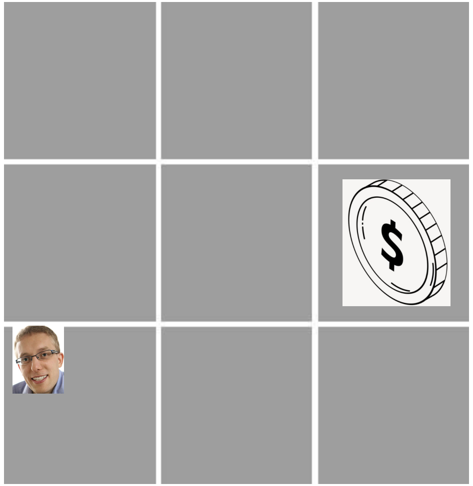
- $\mathcal{R}(s_t, a_t)$ is the reward obtained at time step t for taking action a_t in state s_t.
- $\gamma \in [0, 1]$ is the discount factor that determines the importance of future rewards.

Cornell Bowers C·IS

Discounting Example

$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$$

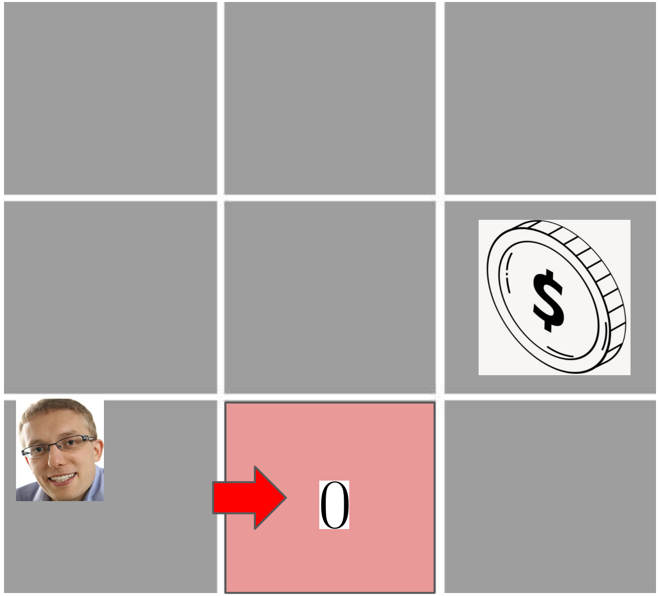
Kilian gets reward r for finding the coin



Cornell Bowers C·IS

Discounting Example

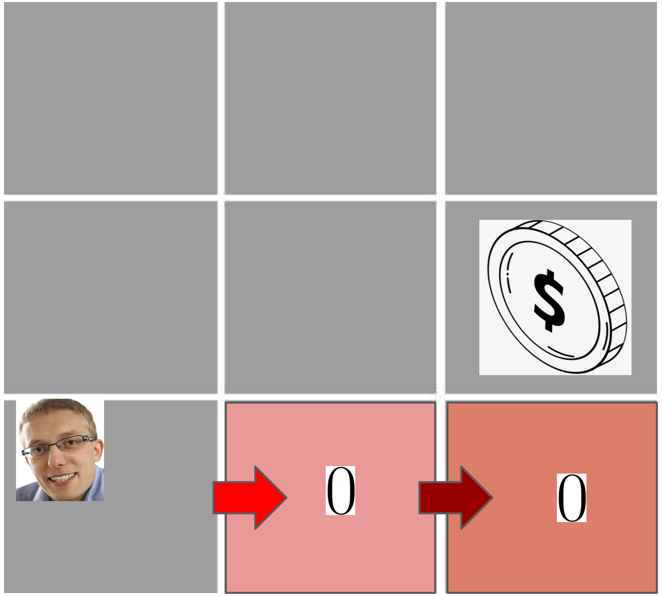
$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$$



Cornell Bowers C·IS

Discounting Example

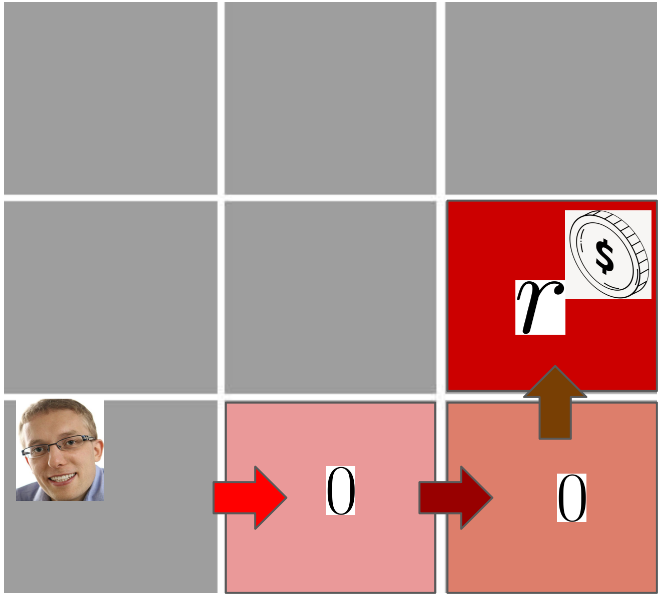
$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$$



Cornell Bowers C·IS

Discounting Example

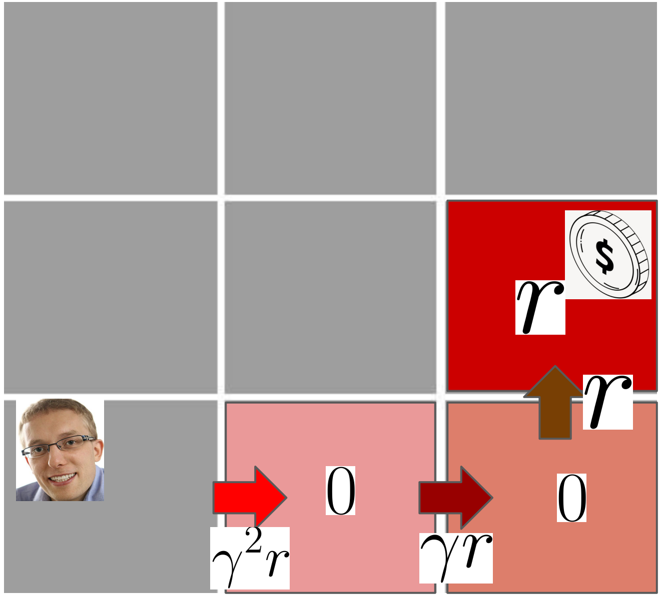
$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$$



Cornell Bowers C·IS

Discounting Example

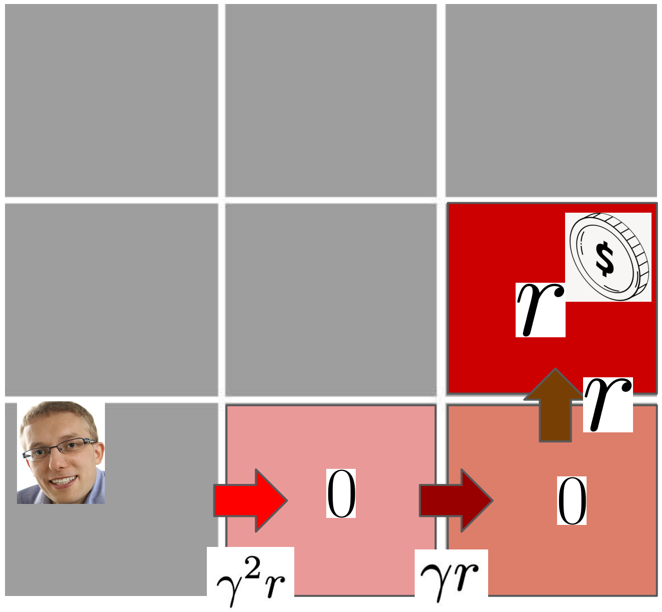
$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$$

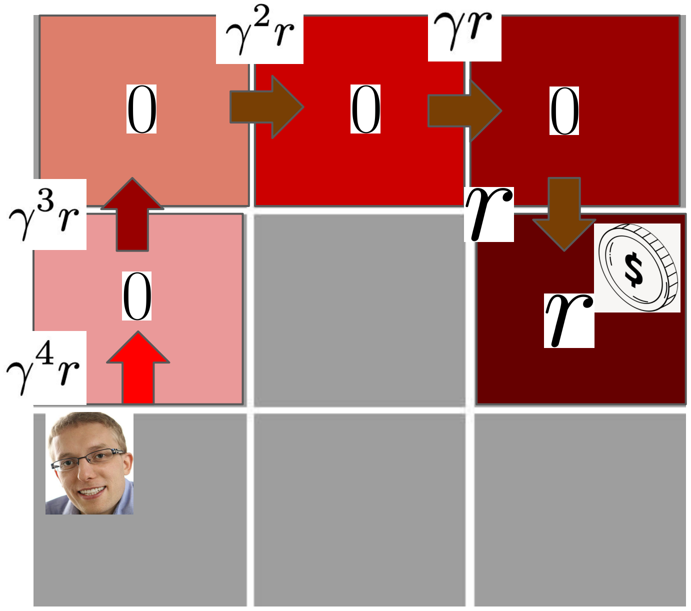


Cornell Bowers C·IS

Discounting Example

$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$$





Cornell Bowers C·IS

Policy

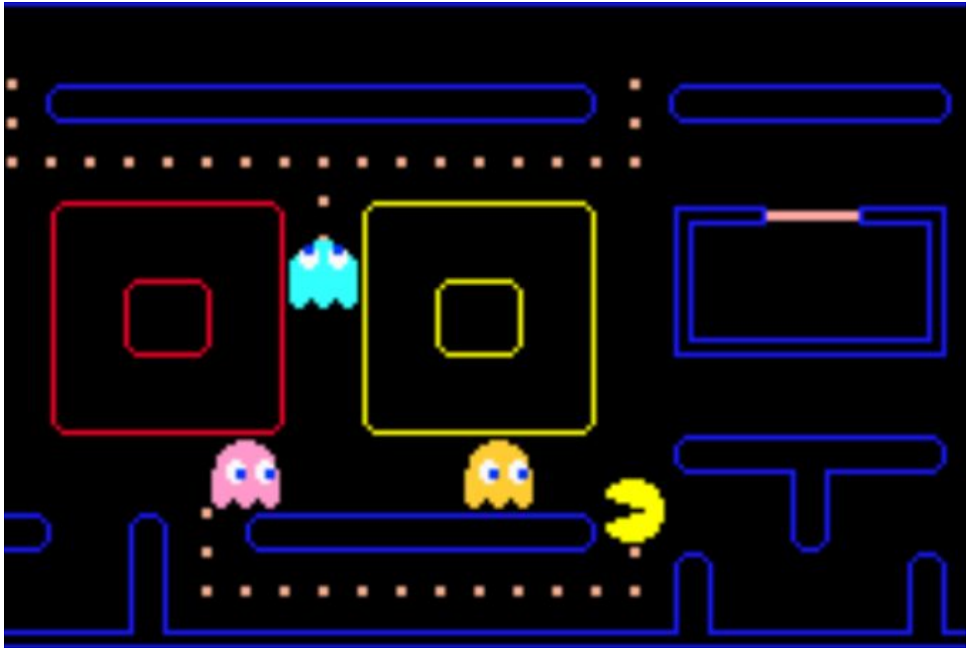
- Different types of policies:
  - Deterministic policy: Maps each state to a single action.
  - Stochastic policy: Defines a probability distribution over actions for each state.

Cornell Bowers C·IS

Pacman Example

Discussion: Identify the **state space S** and **action space A** of Pacman

Is the **policy** below good (based on our current location)?
What would a good policy look like?



Move	Probability
Left	.85
Right	.00
Up	.15
Down	.00

Value Function

- The value function $V^{\pi}(s)$ represents the expected cumulative reward an agent can obtain starting from state s and following policy π.
- Mathematically, the value function is defined as:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s, \pi\right]$$

  - $\gamma \in [0, 1]$: Discount factor that balances immediate and future rewards.
  - $\mathcal{R}(s_t, a_t)$: Reward obtained at time step t for taking action a_t in state s_t.
  - π: Policy that maps states to actions.

Value Function

- Mathematically, the value function is defined as:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s, \pi\right]$$

- The optimal value function $V^*(s)$ represents the maximum expected cumulative reward achievable from state s by following the optimal policy π^*.

- The optimal value function is defined as:

$$V^*(s) = \max_\pi V^* \pi(s)$$

Cornell Bowers C·IS

Action-Value (Q) Function

- Mathematically, the value function is defined as:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s, \pi\right]$$

- Can we use a value function to choose actions?
  - Optimal action: $\arg\max_a \left[r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)}\left[V^{\pi}(s')\right]\right]$
  - Problem: Requires taking the expectation with respect to the environment's dynamics, which we don't have direct access to!

Cornell Bowers C·IS

Action-Value (Q) Function

- The action-value function, or Q-function, represents the expected return if you take action a in state s and then follow policy π.
- Defined as:

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{k+1} \mid s_0 = s, a_0 = a\right]$$

- Optimal action using Q-function:

$$\arg\max_a Q^\pi(s,a)$$

Cornell Bowers C·IS

Action-Value (Q) Function

- Relationship between value function and Q-function:

$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) Q^{\pi}(s, a)$$

- Can recover the value function from the Q-function by marginalizing over all possible actions.

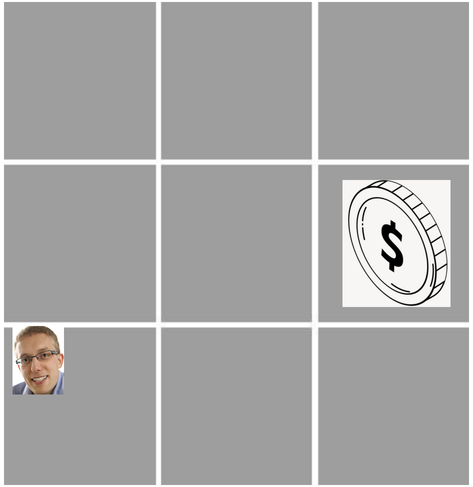
Q-Table

- In Q-Learning, the Q-function is typically represented using a Q-table.
  - A table that stores the estimated Q-values for state-action pairs.
- Q-table has dimensions $|\mathcal{S}| \times |\mathcal{A}|$, where:
  - $|\mathcal{S}|$ is the number of states in the state space.
  - $|\mathcal{A}|$ is the number of actions in the action space.
- The Q-table is initialized with arbitrary values and iteratively updated based on the agent's experiences during the learning process.

Cornell Bowers C·IS

Q-Table Example

- 9 states and 4 actions
- Initialize valid (s,a) tuples to 0s



	Up	Down	Right	Left
Bottom Left	0	-	0	-
Bottom Middle	0	-	0	0
Bottom Right	0	-	-	0
Mid Left	0	0	0	-
Mid Middle	0	0	0	0
Mid Right	0	0	-	0
Top Left	-	0	0	-
Top Middle	-	0	0	0
Top Right	-	0	-	0

Cornell Bowers CIS

Bellman Equation

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{k+1} \mid s_0 = s, a_0 = a\right]$$

- The Bellman equation for Q-Learning provides a recursive relationship between the Q-value of a state-action pair and the Q-values of the successor state-action pairs.
- Bellman equation for the optimal action-value function $Q^*(s,a)$:

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^*(s',a')$$

- Optimal Q-value $Q^*(s,a)$ can be expressed in terms of the immediate reward and the discounted maximum Q-value of the next state.

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^*(s',a')$$

Bellman Error

- The Bellman error measures the difference between the current Q-value estimate and the target Q-value based on the recursive Bellman equation.
- Bellman error for a state-action pair (s, a):

$$\delta = \mathcal{R}(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)$$

- The Bellman error represents the discrepancy between the current Q-value estimate and the target Q-value derived from the recursive Bellman equation.
  - Goal is to minimize the Bellman error and bring the current Q-value estimates closer to the optimal Q-values.

Q-Learning Update Rule

- The Q-Learning update rule adjusts the Q-value estimate towards the target Q-value using the Bellman error:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta$$

  where:

  - α: Learning rate that controls the step size of the update.
  - δ: Bellman error.

Cornell Bowers C·IS

Q-Learning Update Rule

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^*(s',a')$$

- The Q-Learning update rule can be expanded as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R}(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

- The update rule adjusts the current Q-value estimate $Q(s,a)$ in the direction of the target Q-value based on the Bellman error.
  - Q-values are gradually improved and converge towards the optimal Q-function.

Cornell Bowers C·IS

Q-Learning Algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
        $S \leftarrow S'$;
    until S is terminal

Cornell Bowers C·IS

Exploration-Exploitation Tradeoff

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta$$

- Q-Learning only learns about the states and actions it visits.
- Exploration-exploitation tradeoff: The agent should sometimes pick suboptimal actions to visit new states and actions.
- Balancing exploration and exploitation is crucial for effective learning:
  - Exploration: Trying new actions to gather information about the environment.
  - Exploitation: Using the current knowledge to make the best decisions based on the learned Q-values.

Exploration-Exploitation Tradeoff

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta$$

- Insufficient exploration may lead to suboptimal policies and getting stuck in local optima.
- Too much exploration may slow down the learning process and hinder the agent from converging to the optimal policy.
- Finding the right balance between exploration and exploitation is essential for efficient and effective learning in Q-Learning.

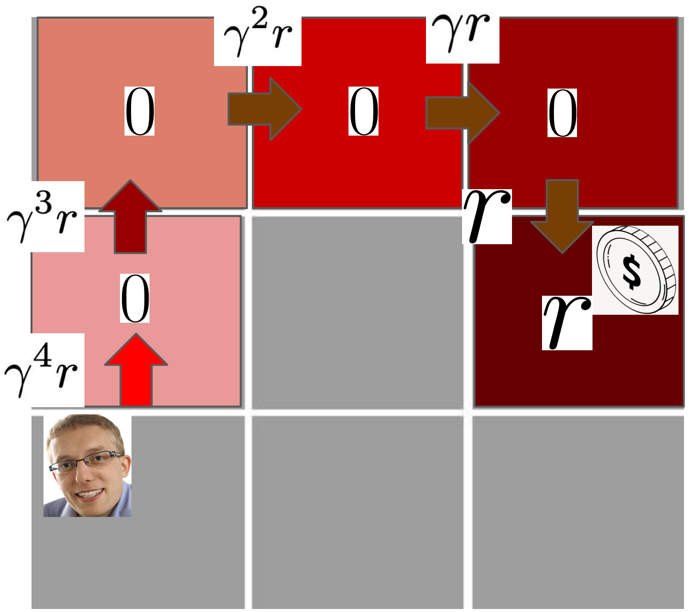
Cornell Bowers C·IS

Exploration-Exploitation TradeOff

$$\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$$

No Exploration:

- Begin by making random moves
- Find some way to obtain reward
- Stick with that solution



ε-Greedy Policy

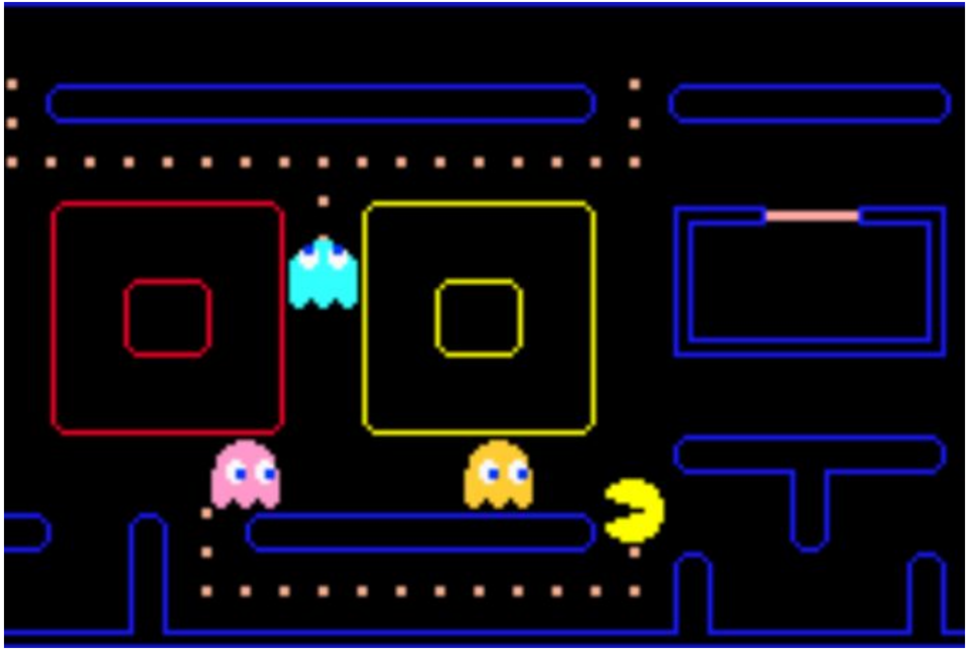
- Simple solution to balance exploration and exploitation: ϵ-greedy policy
- ϵ-greedy policy:
  - With probability $1 - \epsilon$, choose the optimal action according to the learned Q-values.
  - With probability ϵ, choose a random action.
- The ϵ-greedy policy ensures that the agent explores the environment while still exploiting the learned knowledge.
- Despite its simplicity, ϵ-greedy is still widely used in practice and often yields good results.

Cornell Bowers C·IS

Discussion: Calculate the size of the State Space for Pacman in the following environment

Assume there are 100 positions, 4 ghosts, and pacman

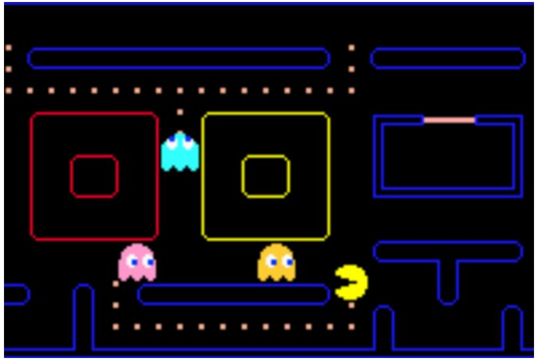
(Can ignore pellets for simplicity)



Discussion: Calculate the size of the State Space for Pacman in the following environment

4 ghosts and Pacman

$$100 \cdot 100 \cdot 100 \cdot 100 \cdot 100 = 10^{10}$$



What if we consider pellets? Assuming there are 100 pellets:

$$2^{100} \cdot 10^{10} > 10^{40}$$

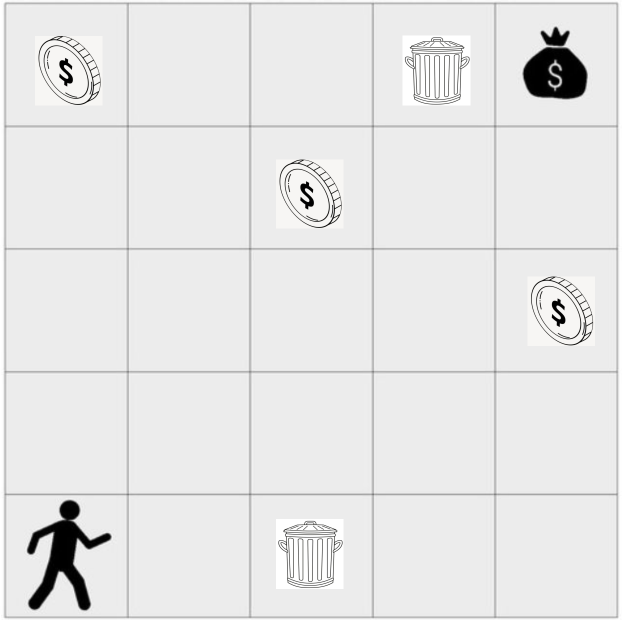
Cornell Bowers C·IS

What are the issues with Q Learning?

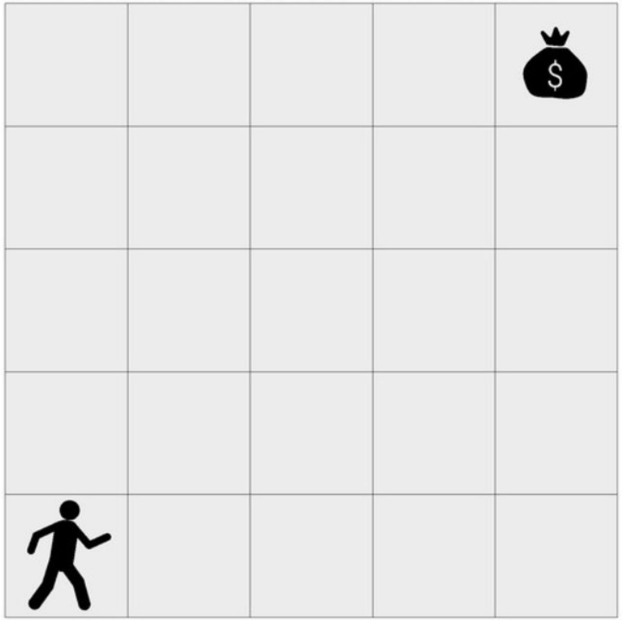
- As the state/action space increases, the likelihood of reaching a specific state and action decreases
- Continuous spaces are impossible to model with a table

Cornell Bowers C·IS

Dense Rewards



Sparse Rewards

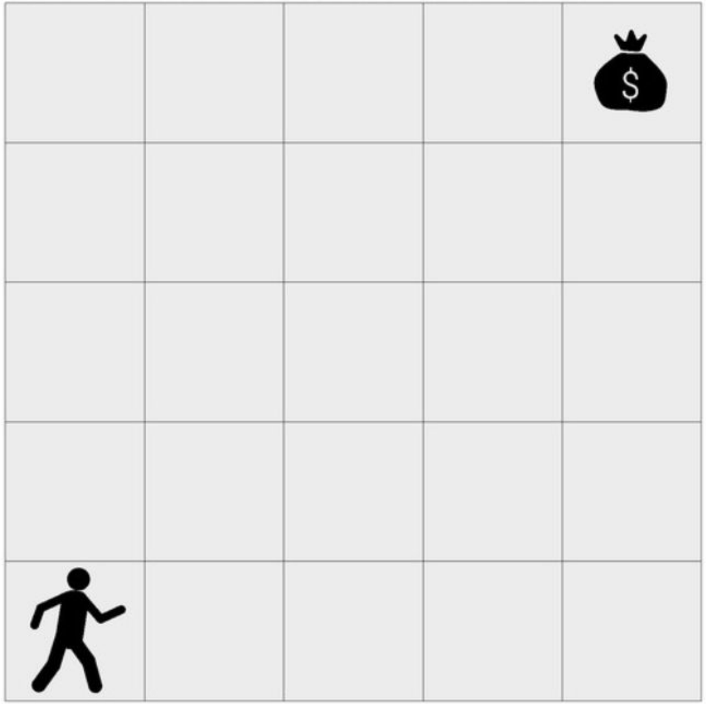


Cornell Bowers C·IS

Discussion: if a long sequence of actions results in just one reward, how do you know which states/action(s) are responsible?

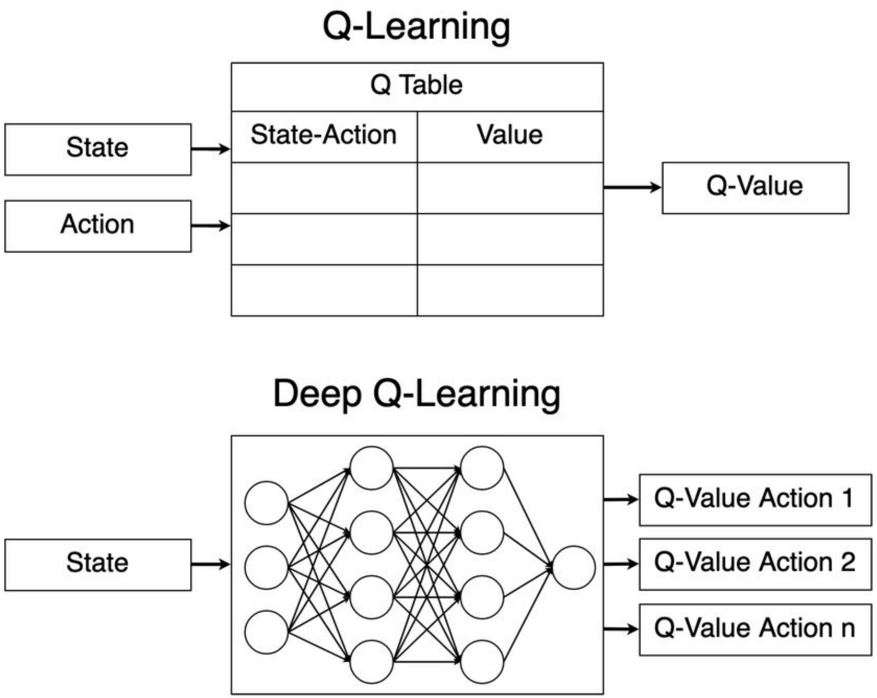
How do you learn a good policy in the presence of sparse rewards?

- In general, credit attribution is hard!
- Can introduce intermediate rewards
  - E.g. distance to money
  - Not always possible!
- Hard to learn with sparse rewards



Cornell Bowers C·IS

Next Time: How can we use deep learning to improve Q-Learning?



Cornell Bowers CIS

Recap

- Markov Decision Processes (MDPs)
  - Framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision maker
- Value function
  - Expected cumulative reward from state s following some policy
- Action-value (Q) function
  - Expected cumulative reward if you take action a in state s and follow some policy
- Q-Learning iteratively updates a Q-Table to minimize the Bellman Error
  - Need to balance exploration-exploitation tradeoff
  - Often use epsilon-greedy policy for exploration