HEC MONTRÉAL

Machine Learning for Large-Scale Data Analysis and Decision Making (MATH80629A)
Winter 2023

Week #12- **Summary**

HEC MONTRÉAL

Announcement

- **Last Quiz, Quiz#6 on RL:** Next week
- **Project Presentation: in-person** on **April 14**
- **Project Report:** due **April 23**
- **Final exam: April 30**

Today

- Summary of Sequential decision making I
- Q&A
- Finish RSs case study
- Hands on session

Sequential decision making I

Reinforcement Learning

- Sometimes we need a model where **the learning** and **the decision making** interact closely
- Imagine building a robot that must navigate autonomously
    - The robot has wheels and a camera
- You think about using a **two-stage approach:**
    1. Use supervised learning to identify objects in scenes
    2. Given scene content have a decision-making module that controls its wheels

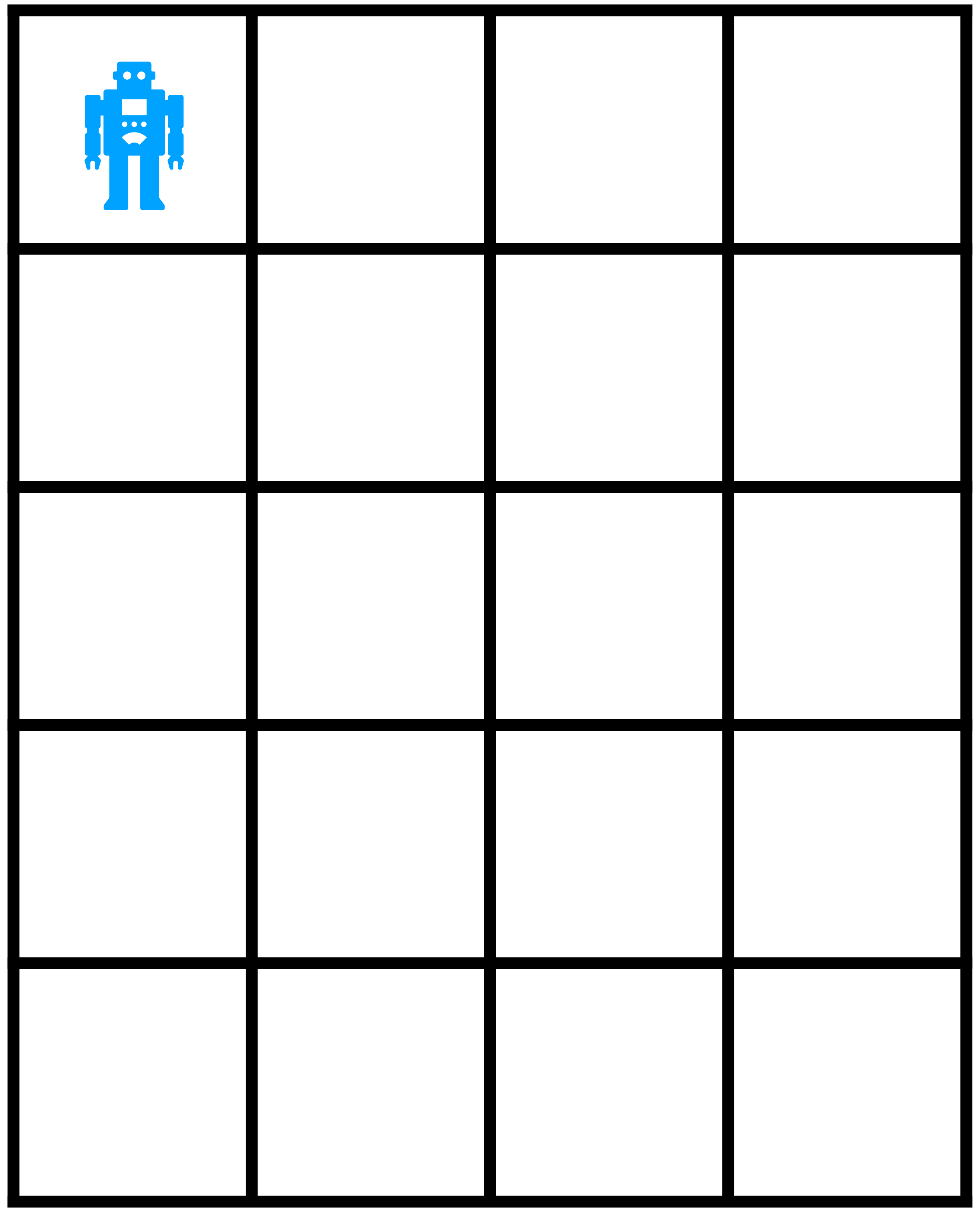
Limitations of two-stage approach

- **Supervised learning doesn't know about the decision-making**
  - Its objective is, for example, to maximize accuracy
- **For decision making, different errors have different costs**
  - E.g., missing the cliff could have dire consequences. missing sky less so.
  - Incorporating these costs into the learning objective is tough
- **Several other limitations:**
  - need labeled data
  - improvements in SL do not necessarily lead to improvements in decision making
  - …

Alternative: Reinforcement learning (RL)

- **Incorporates both stages in a single framework**
- Incorporates the ideas of:
  - **state (observation)**
  - **action**
  - **reward**

Initial example with grid world



- **Each cell is a state (S)**
- **Actions indicate which movements are possible:**

$$A := \{L, R, U, D\}$$

- **Rewards encode the task:** $R(s)$
- **Transition probabilities encode the outcome of an action:**

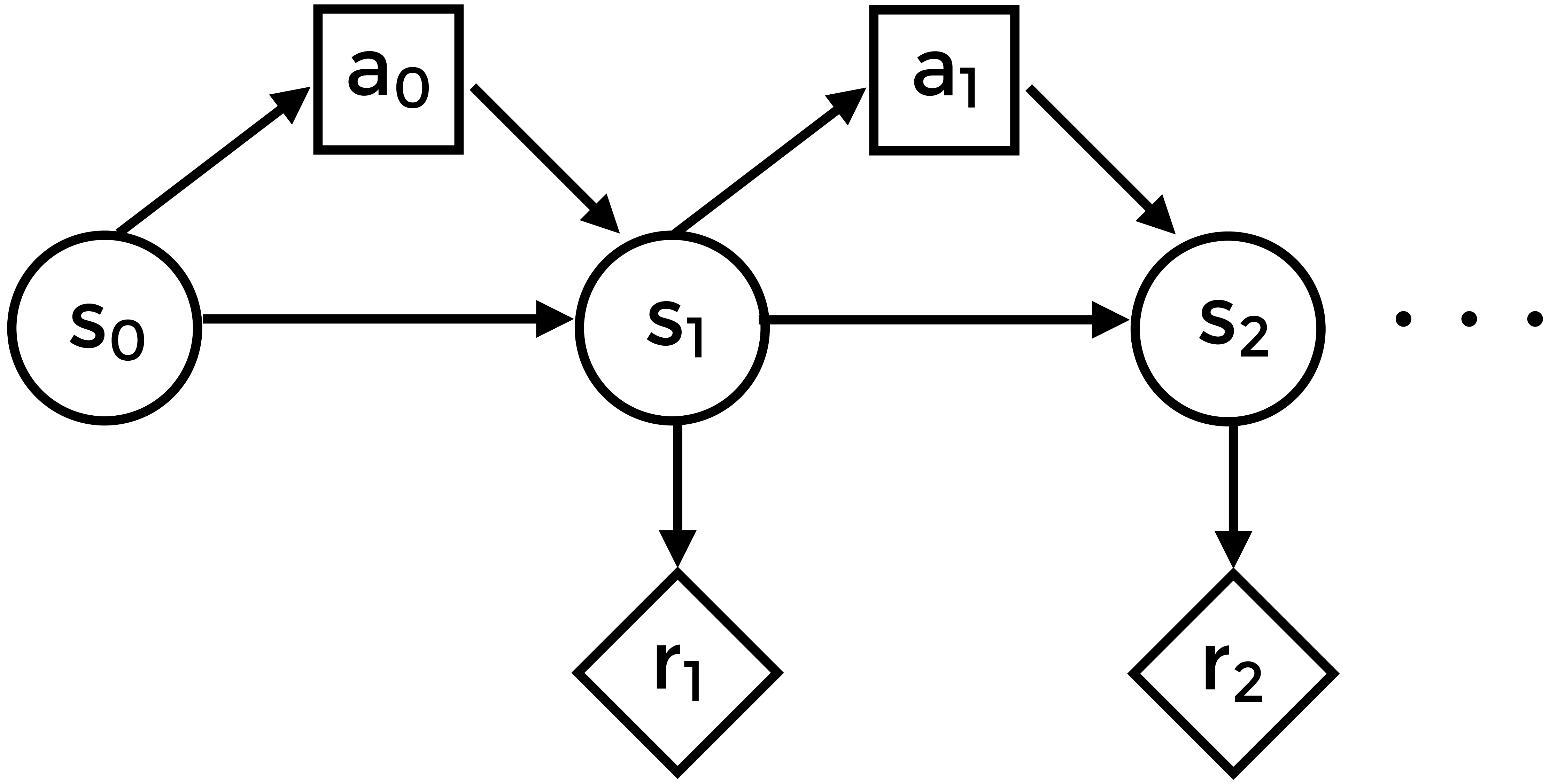
$$P(s' \mid s, a)$$

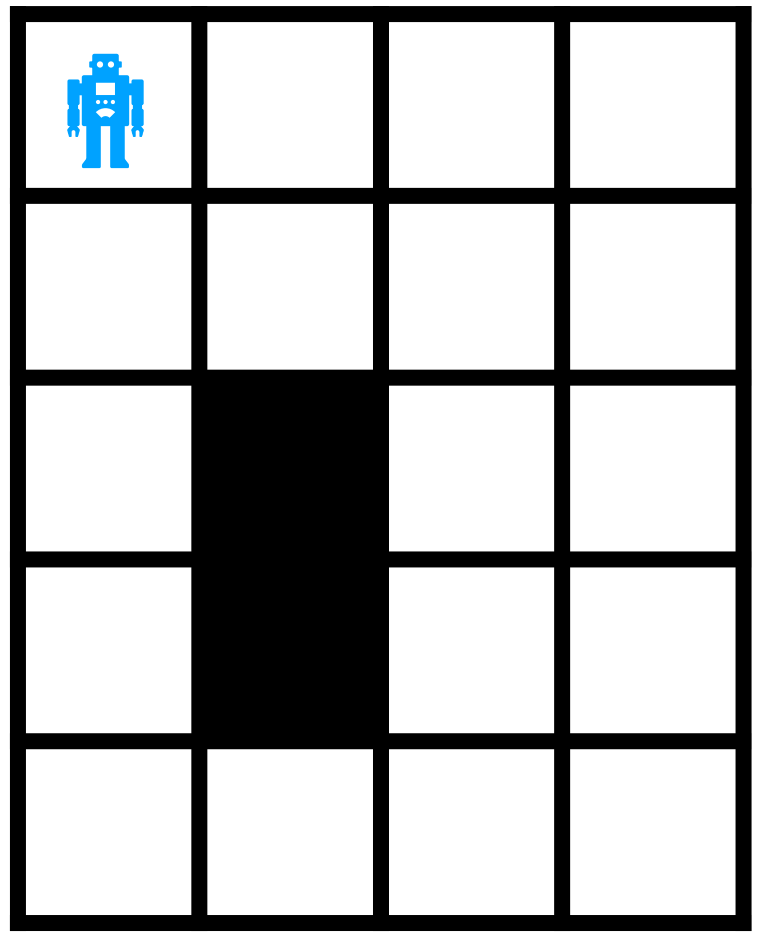
This week we discuss a version of RL where these are observed

Markov Decision Process (MDP)

- **Provide a framework for decision-making under uncertainty**
  - **Markov process with decisions and utilities**
  - **Assumes stationarity (i.e., transitions are fixed across time)**

- Square nodes: decisions
- Circle nodes: States
- Diamond nodes: utility





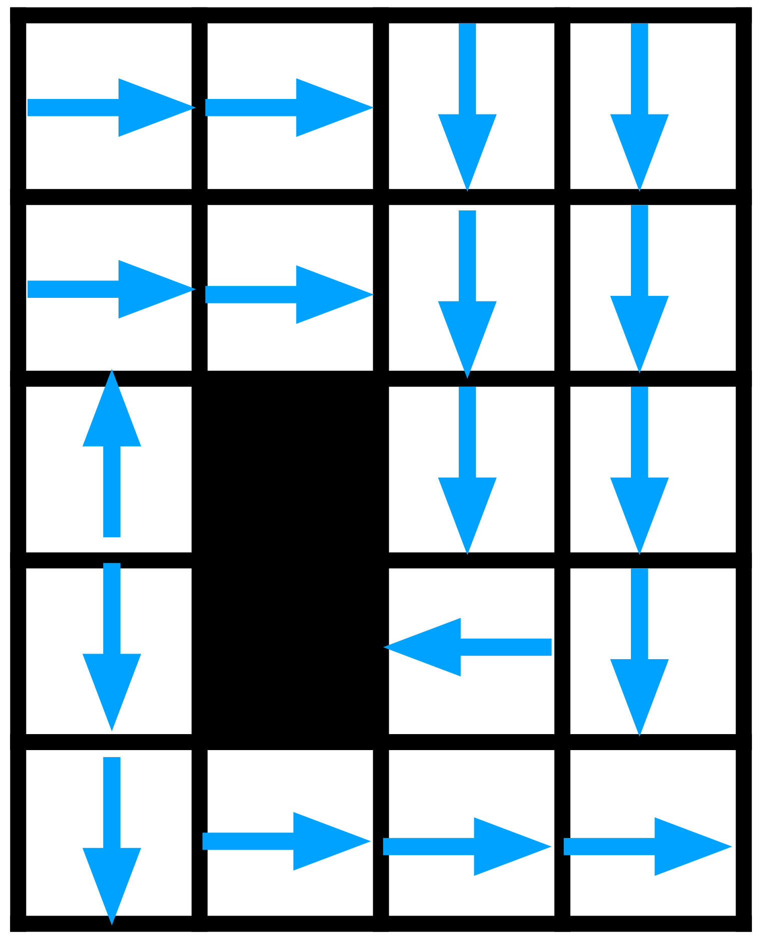
Markov Decision Process (MDP)

$\langle A, S, P, R, \gamma \rangle$

- **A: set of actions**
- **P(S' | S,A): transition probabilities**
- **R(S): reward function**
- γ **: discount factor** $\in [0, 1]$

- **A policy:** $\pi : S \to A$
- **Goal: find the optimal policy**

π



Optimal policy?

- Agent is trying to maximize its rewards (utility)
  - Utility simply assigns a real value to a state
  - Typically combine rewards with an additive function

$$\sum_t R(s_t)$$

Discounting (γ)

- **The sum of rewards could be infinite/unbounded**

$$\lim_{T \to \infty} \sum_{t}^{T} R(s_t)$$

- **A typical solution is to use a discount factor** $0 \leq \gamma \leq 1$

$$\lim_{T \to \infty} \sum_{t}^{T} \gamma^t R(s_t)$$

  - **Geometric series. Bounded by:** $\frac{R_{\max}}{1-\gamma}$
  - **Intuition: would rather have rewards sooner**

Solving an MDP

- **Find the optimal policy of an MDP**

$$\pi^*(s) \ \ \forall s$$

- **Policies are evaluated using their expected utility:**

$$EU(\pi) = \boxed{\sum_{t=0}^{\infty} \gamma^t} \sum_{s_{t+1}} P(s_{t+1} \mid s_t, \pi(s_t)) \boxed{R(s_{t+1})}$$

- **The optimal policy is the one with highest expected utility:** $EU(\pi^*) \geq EU(\pi) \ \ \forall \pi$

Solving an MDP

1. **Value iteration**
2. **Policy Iteration**

Value Function

- $V(s_t)$: **The value of being in state s at time t**

$$V(s_t) := \text{expected sum of rewards of being in s}$$

Finite horizon

- **Assume that the process has T steps**
- **The value at step T is** $V(s_T) = R(s_T)$
- **The value at step T-1 is**

$$V(s_{T-1}) = \max_{a_{T-1}} \left\{ R(s_{T-1}) + \gamma \sum_{s_T} P(s_T \mid s_{T-1}, a_{T-1}) R(s_T) \right\}$$

- **The value at step t is** $(0 \leq t \leq T)$

$$V(s_t) = \max_{a_t} \left\{ R(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V(s_{t+1}) \right\}$$

Bellman equation

- Value of state s

$$V(s_t) = \max_{a_t} \left\{ R(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V(s_{t+1}) \right\} \quad \forall s$$

- Recursive equations
- The value of a state only depends on the state's reward and the neighbours' value
- This is also known as a dynamic programming equation

Value iteration (VI)

- Iteratively update V(s) for each state until convergence
- (Initialize V(s) for every state)
- For i=1,2,3,…
  - For s=1,…,S

$$V(s) = \max_a \left\{ R(s) + \gamma \sum_{s'} P(s' \mid s, a) V(s') \right\}$$

- The policy is implicit
  - Once converged: $\pi^*(s) = \arg\max_a \left\{ R(s) + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \right\} \; \forall s$

Policy Iteration (PI)

- **Improve policy explicitly.**

Start with any (e.g., random) policy π

Iterate until convergence:

Policy Evaluation

1. **Given current policy get the value of each state**

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) V^{\pi}(s') \qquad \forall s$$

Policy Update

2. **Update the current policy**

$$\pi'(s) = \arg\max_{a} \left\{ R(s) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi}(s') \right\} \quad \forall s$$

PI vs. VI

- **Value iteration is faster per iteration**
- **Policy iteration converges in fewer iterations**

MDP Real-world Examples

Examples are taken from:

https://towardsdatascience.com/real-world-applications-of-markov-decision-process-mdp-a39685546026

MDP framework

- To express a problem using MDP, we need to define the followings
- **states** of the environment
- **actions** the agent can take on each state
- **rewards** obtained after taking an action on a given state
- **state transition** probabilities.

Salmon Fishing

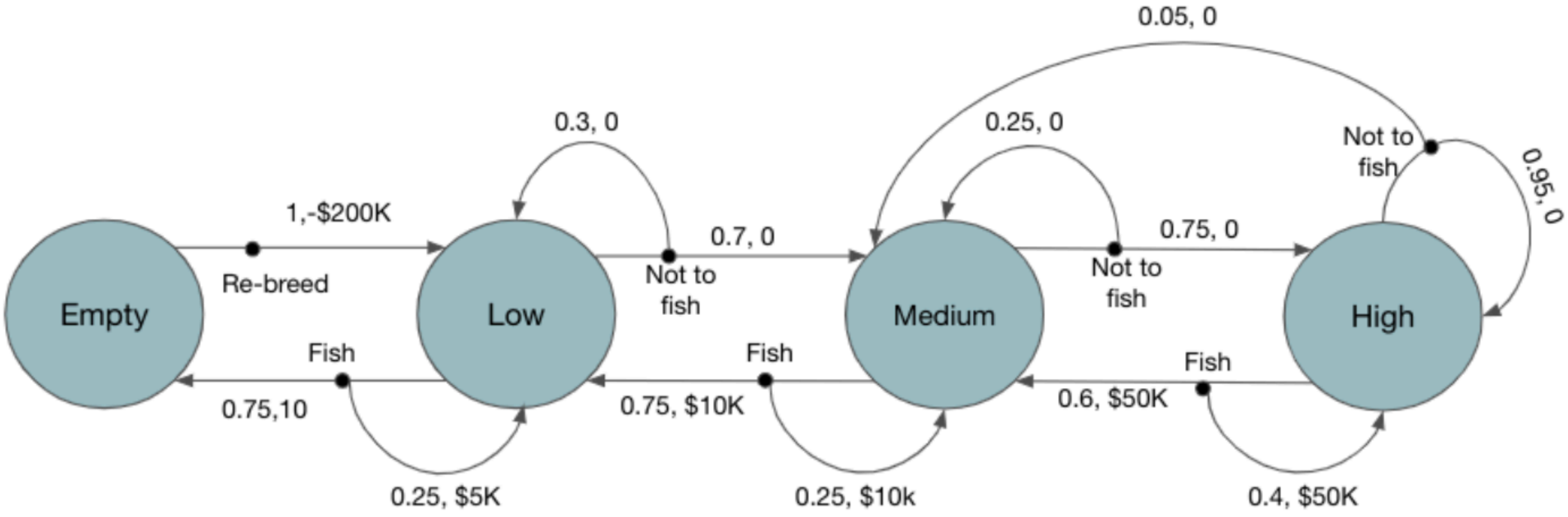
- **States:** The number of salmons available in that area in that year. E.g., four states; **empty**, **low**, **medium**, **high**.

- **Actions: fish** and **not_to_fish**. Fish means catching certain proportions of salmon. For the state empty the only possible action is not_to_fish.

- **Rewards**: Fishing at certain state generates rewards, let's assume the rewards of fishing at state low, medium and high are $5K, $50K and $100k respectively. If an action takes to empty state then the reward is very low -$200K as it require re-breeding new salmons which takes time and money.

Salmon Fishing

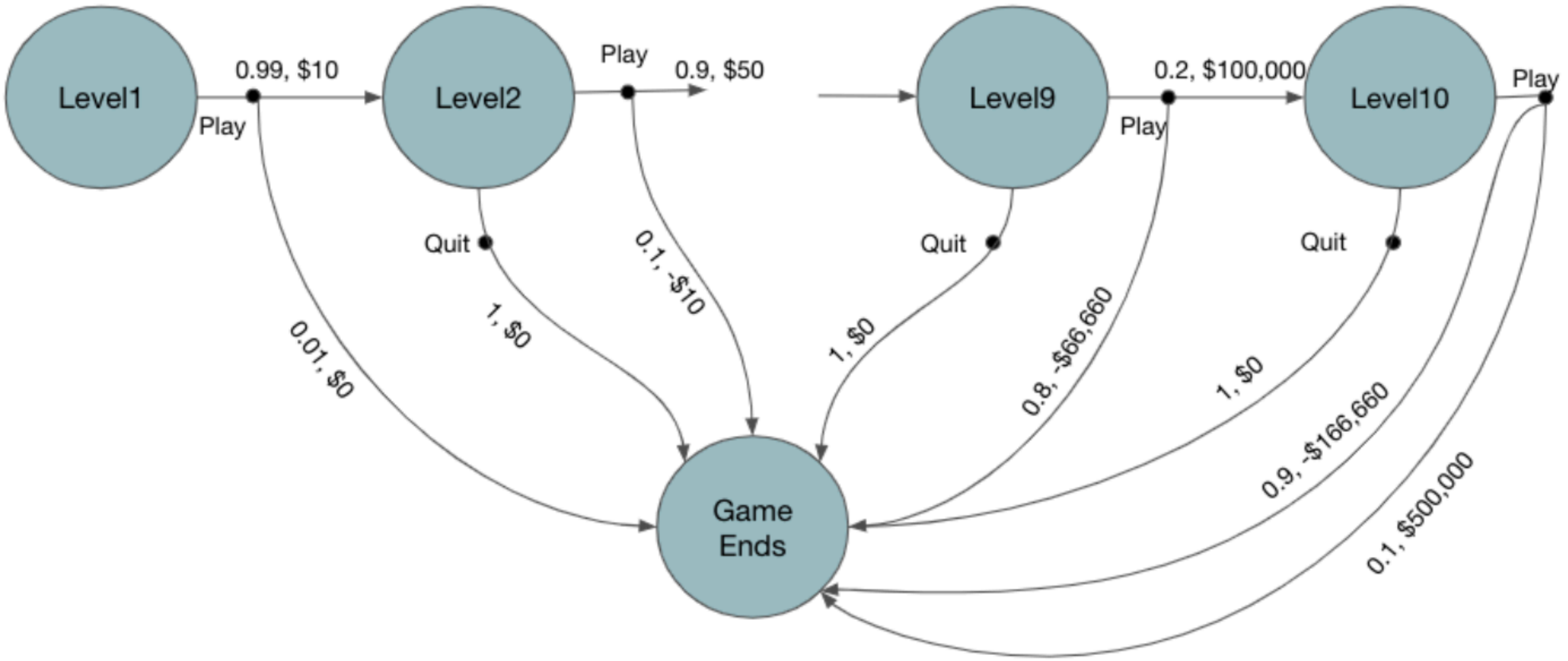
State Transitions: Fishing in a state has higher a probability to move to a state with lower number of salmons. Similarly, not_to_fish action has higher probability to move to a state with higher number of salmons (excepts for the state high).



Quiz Game Show

- **States:** {level1, level2, …, level10}
- **Actions:** Play at next level or quit
- **Rewards**: Play at level1, level2, …, level10 generates rewards \$10, \$50, \$100, \$500, \$1000, \$5000, \$10000, \$50000, \$100000, \$500000 with probability p = 0.99, 0.9, 0.8, …, 0.2, 0.1 respectively. The probability here is a the probability of giving correct answer in that level. At any level, the participant losses with probability (1- p) and losses all the rewards earned so far.

Quiz Game Show



HEC MONTRÉAL

Other MDP examples

- **Harvesting**: how much members of a population have to be left for breeding.
- **Agriculture**: how much to plant based on weather and soil state.
- **Water resources**: keep the correct water level at reservoirs.
- **Inspection, maintenance and repair**: when to replace/inspect based on age, condition, etc.
- **Purchase and production**: how much to produce based on demand.
- **Queues**: reduce waiting time.
- **Finance**: deciding how much to invest in stock.
- **Robotics**: navigator, dialogue system, etc.

Other MDP examples

- Interested to find more? Check White, D.J. (1993) that mentions a large list of applications.