Week 1: Introduction to RL and Sequential Decision Making

Intro to reinforcement learning,
k-armed bandits, MDPs

Go to our Linktree to follow our **socials**

Land Acknowledgement

Statement

We wish to acknowledge this land on which the University of Toronto operates. For thousands of years, it has been the traditional land of the Huron-Wendat, the Seneca, and the Mississaugas of the Credit.

Today, this meeting place is still the home to many indigenous people from across Turtle Island and we are grateful to have the opportunity to learn, work, and discuss on this land.

AI News!

- AI sweeps Nobel Prizes in Physics (Hopfield networks, Boltzmann machines), and Chemistry (AlphaFold)
- Reinforcement learning algorithm helps route robots to help study whales (link)

Boring (Administrative) Stuff

Workshop Details

Name: Jingmin Wang, Jessica Chen, guest speaker Dr. Michael Bowling

Emails: jingmin.wang@mail.utoronto.ca, jiajing.chen@mail.utoronto.ca

Time and Location: Wednesdays 7 - 9 pm in GB119, Zoom

Next Sessions: November 13, 20, 27

Topics Covered: Sequential decision making, the RL paradigm, tabular methods, finite MDPs, dynamic programming, iterative methods, Monte Carlo methods, temporal-difference learning

Prerequisites

(approximately)

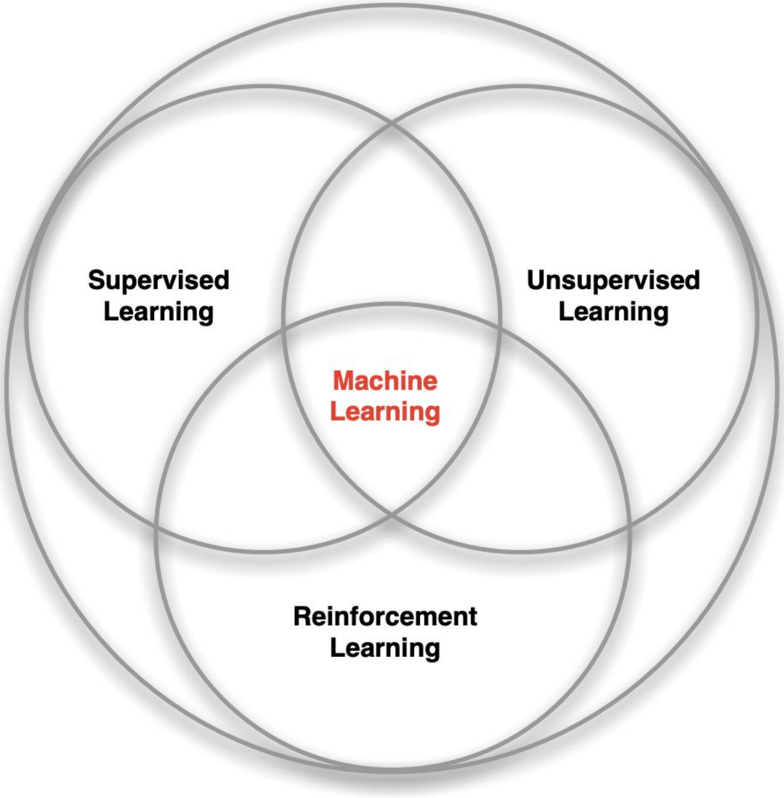
NONE

Most important:

A genuine passion for the subject

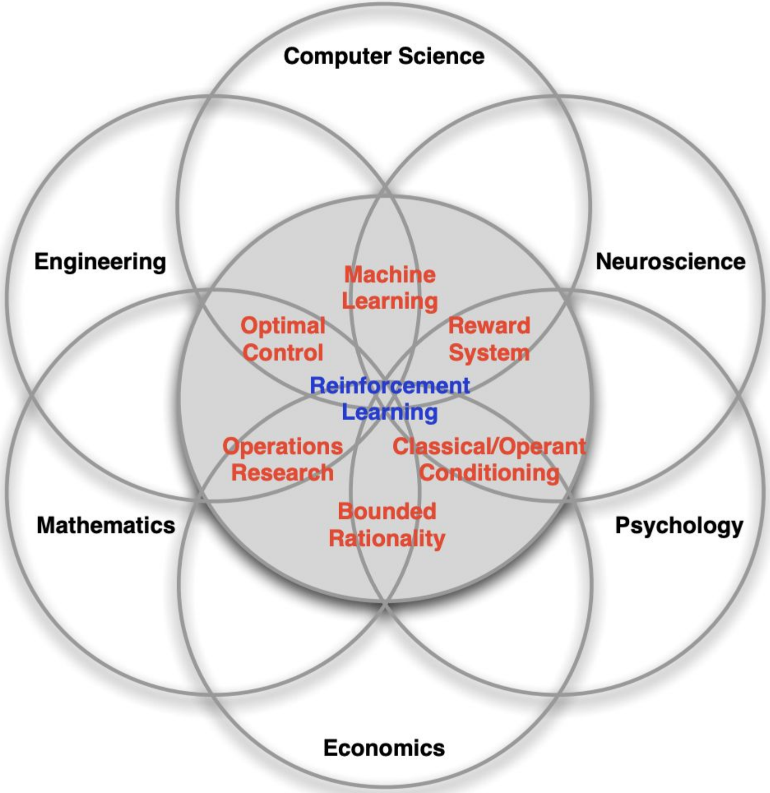
Overview of Reinforcement Learning

Branches of Machine Learning



Credit: David Silver

Many Faces of Reinforcement Learning



Credit: David Silver

What exactly is Reinforcement Learning?

A learning agent interacts overtime with its environment in order to achieve a goal. The agent should:

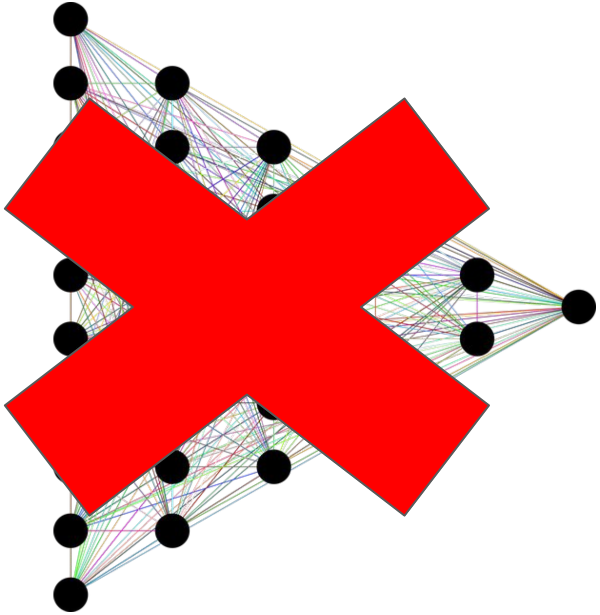
- ❑ Be able to sense the state of the environment
- ❑ Have clear goals relating to the state

Most often times, the goal will be to maximize return

(will be discussed later)

RL is a Paradigm!

You will **NOT** be learning some model architecture or a single algorithm. Rather, a way of framing problems



Examples of Reinforcement Learning

AlphaGo

Examples of Reinforcement Learning

Examples of Reinforcement Learning

How would you train a supervised/unsupervised model do to this?

Characteristics of Reinforcement Learning

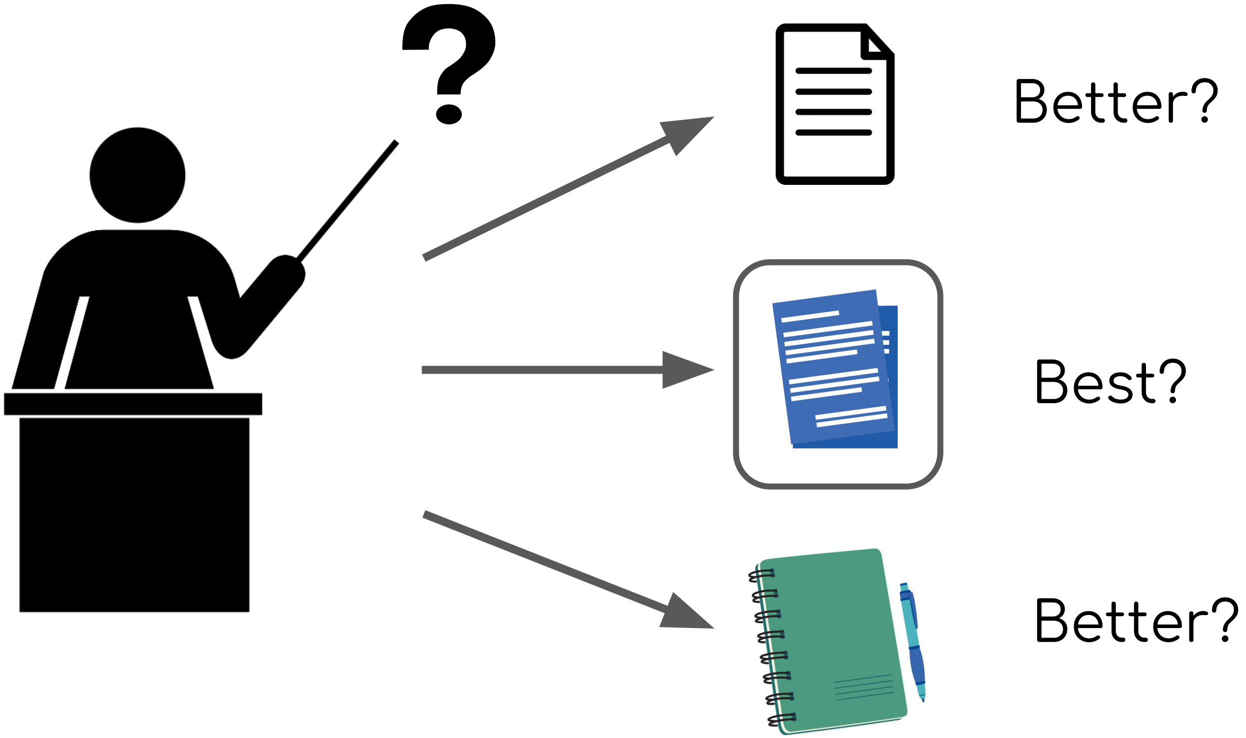
What makes reinforcement learning different from other machine learning paradigms?

- ❑ Goal is to maximize a reward signal
- ❑ Feedback is mostly delayed, not instantaneous
- ❑ The actions of an agent affect the subsequent observations it receives
- ❑ Data is sequential, and not independently and identically distributed (i.i.d)

Sequential Decision Making under Uncertainty

K-armed bandit problem

Assigning Labs Example



The K-armed Bandit Problem

Agent = …

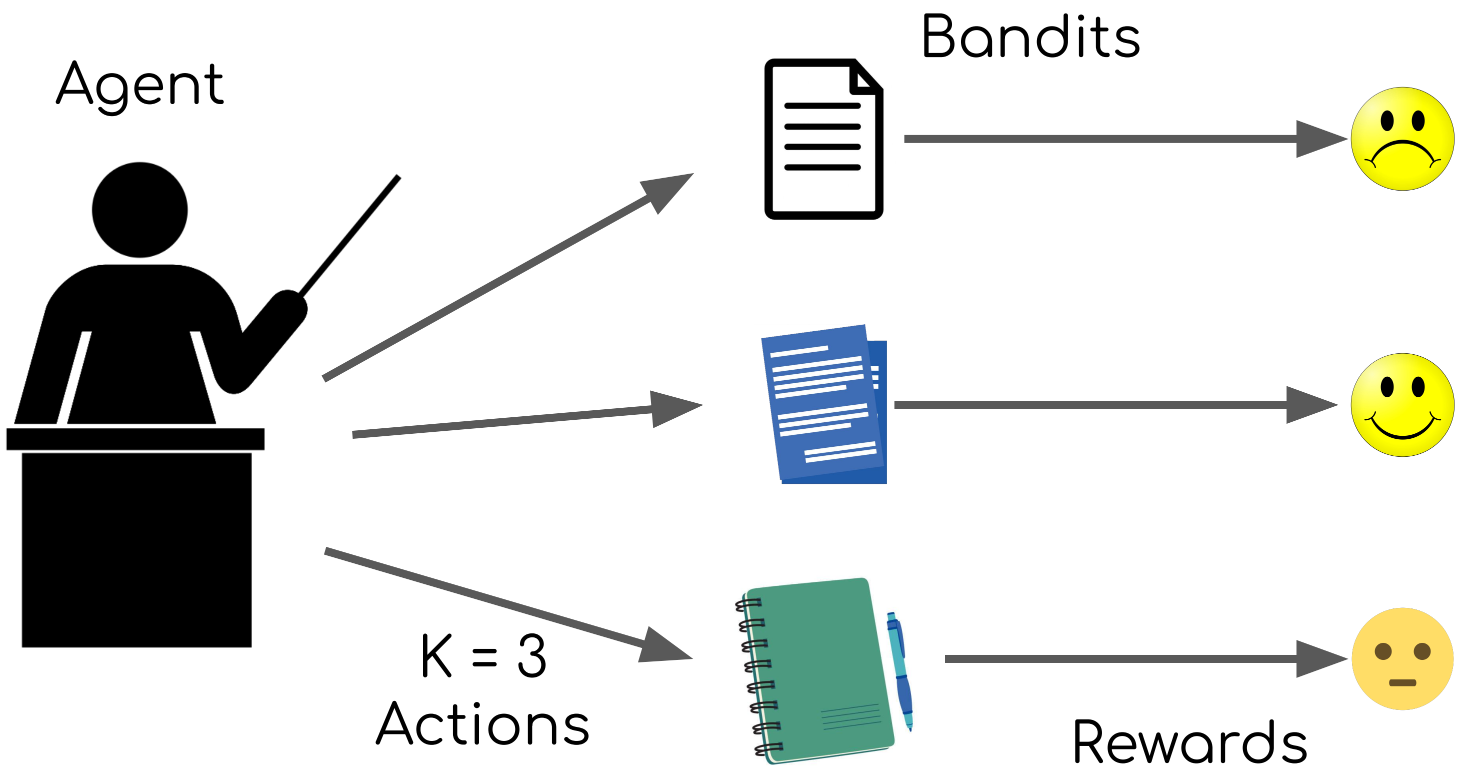
Environment = …

Reward = …

The K-armed Bandit Problem

In the k-armed bandit problem, we have an **agent** who chooses between k **actions** and receives a **reward** based on the action it chooses

Assigning Labs Example



Action Values

Action value is the expected reward of some action

$$q_*(a) = \mathbb{E}[R_t | A_t = a], \forall a \in \{1, \ldots, k\}$$

The goal is to maximize the expected reward

$$\arg\max_a q_*(a)$$

How do we estimate this?

Side Note: Expected Value

$$E[X] = \sum x_i p(x_i)$$

x_i = The values that X takes

$p(x_i)$ = The probability that X takes the value x_i

$$70(0.3) + 90(0.7) = 84$$

Estimating Action Values: Sample Average

- The value estimate at time *t* is the sum of rewards when *a* taken prior to *t* divided by number of times *a* taken prior to *t*

$$q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

$$\xrightarrow{t_1} 90$$

$$q_t(a) = \frac{90}{1} = 90$$

Estimating Action Values: Sample Average

- The value estimate at time *t* is the sum of rewards when *a* taken prior to *t* divided by number of times *a* taken prior to *t*

$$q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

$$\xrightarrow{t_1} 90 \xrightarrow{t_2} 90$$

$$q_t(a) = \frac{90 + 90}{2} = 90$$

Estimating Action Values: Sample Average

- The value estimate at time *t* is the sum of rewards when *a* taken prior to *t* divided by number of times *a* taken prior to *t*

$$q_t(a) = \frac{\sum_{i=1}^{t-1} R_i}{t-1}$$

Side note: the concept of estimating things with sampling will be key later on

$$\xrightarrow{t_1} 90 \xrightarrow{t_2} 90 \xrightarrow{t_3} 70$$

$$q_t(a) = \frac{90 + 90 + 70}{3} = 83.3$$

Incremental Sample Average

Recall:

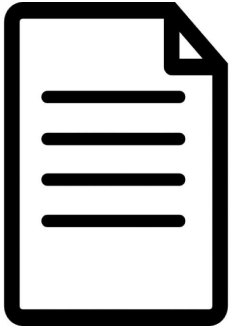
$$Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i$$

$$\begin{aligned} Q_{n+1} &= \frac{R_1 + \cdots + R_n}{n} \\ &= \ldots \\ &= Q_n + \frac{1}{n}(R_n - Q_n) \end{aligned}$$

NewEstimate ← OldEstimate + StepSize (Target - OldEstimate)

Exploration vs. Exploitation

- **Exploration:** Improve knowledge for long-term benefit
- **Exploitation:** Improve short-term benefits



$Q_n = 64.2$

Greedy

?

$Q_n = 83.3$

$Q_n = 79.1$

Epsilon-Greedy Action Selection

- Always go for the greedy choice, except for a small percentage of time

$$A_t \leftarrow \begin{cases} \underset{a}{\operatorname{argmax}}\ Q_t(a) & \textbf{with probability } 1-\epsilon \\ a \sim Uniform(\{a_1 \ldots a_k\}) & \textbf{with probability } \epsilon \end{cases}$$

$$\epsilon \in [0, 1)$$

Notebook: K-armed bandit

Epsilon-Greedy Action Selection

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

$$\boxed{A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}} \quad \text{(breaking ties randomly)}$$

$R \leftarrow bandit(A)$

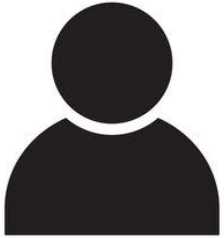
$N(A) \leftarrow N(A) + 1$

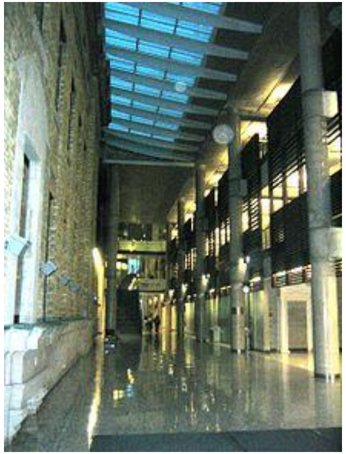
$$\boxed{Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]}$$

Goal of RL

Immediate Scenario

Actions Influence Future Rewards

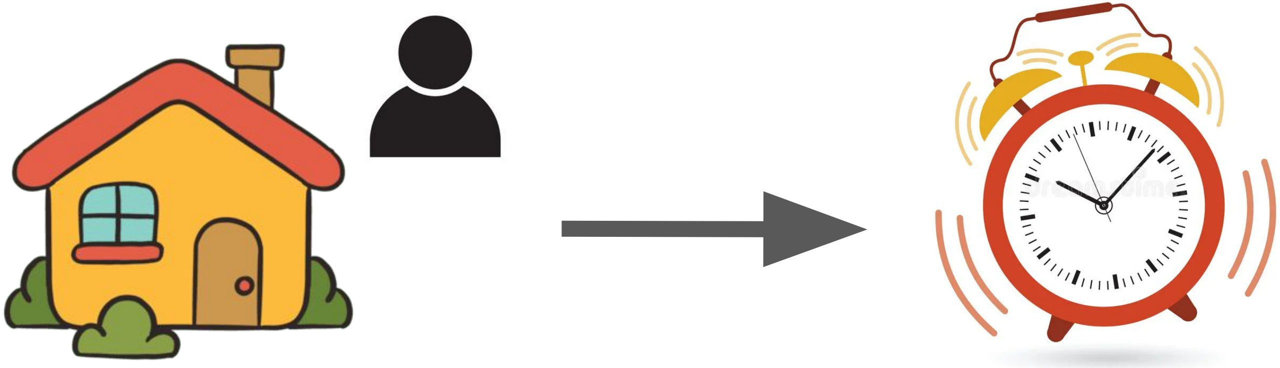




+10

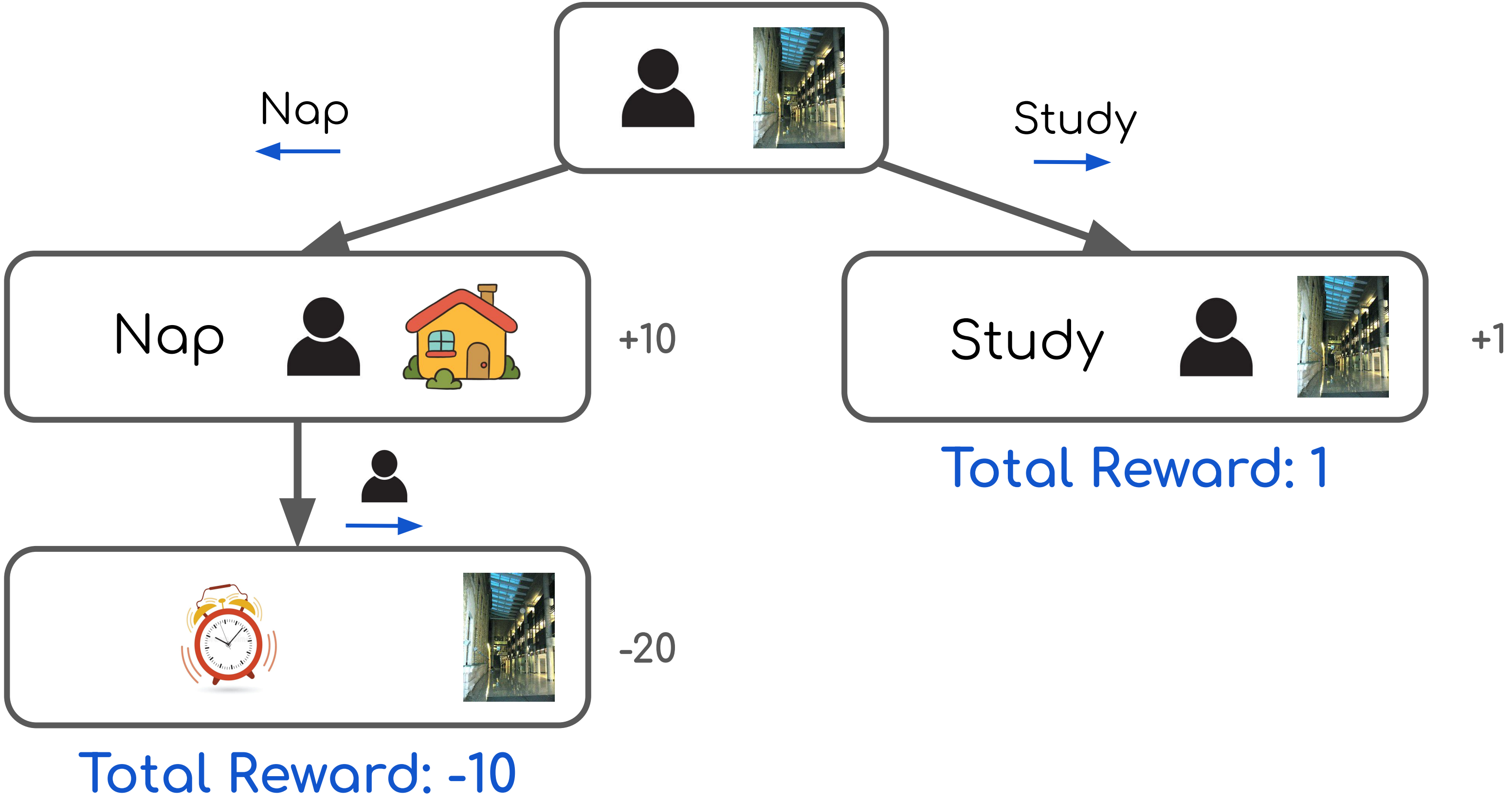
+1

Actions Influence Future Rewards



+10 -20

Full Scenario



States

Return

Don't want to maximize immediate reward, but the total expected reward from all timesteps. This is the **return**.

Definition

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Why the discount factor $\gamma \in [0, 1]$?

Discount Factor

Discount factor accounts for the priority of long term rewards vs short term rewards

- If γ = 1 (long-sighted):

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

- If γ = 0 (short-sighted):

$$G_t = R_{t+1}$$

Goal of Reinforcement Learning

Maximize **expected return**

$$\mathbb{E}[G_t] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots]$$

Finite Markov Decision Processes

We Need a Formal Definition

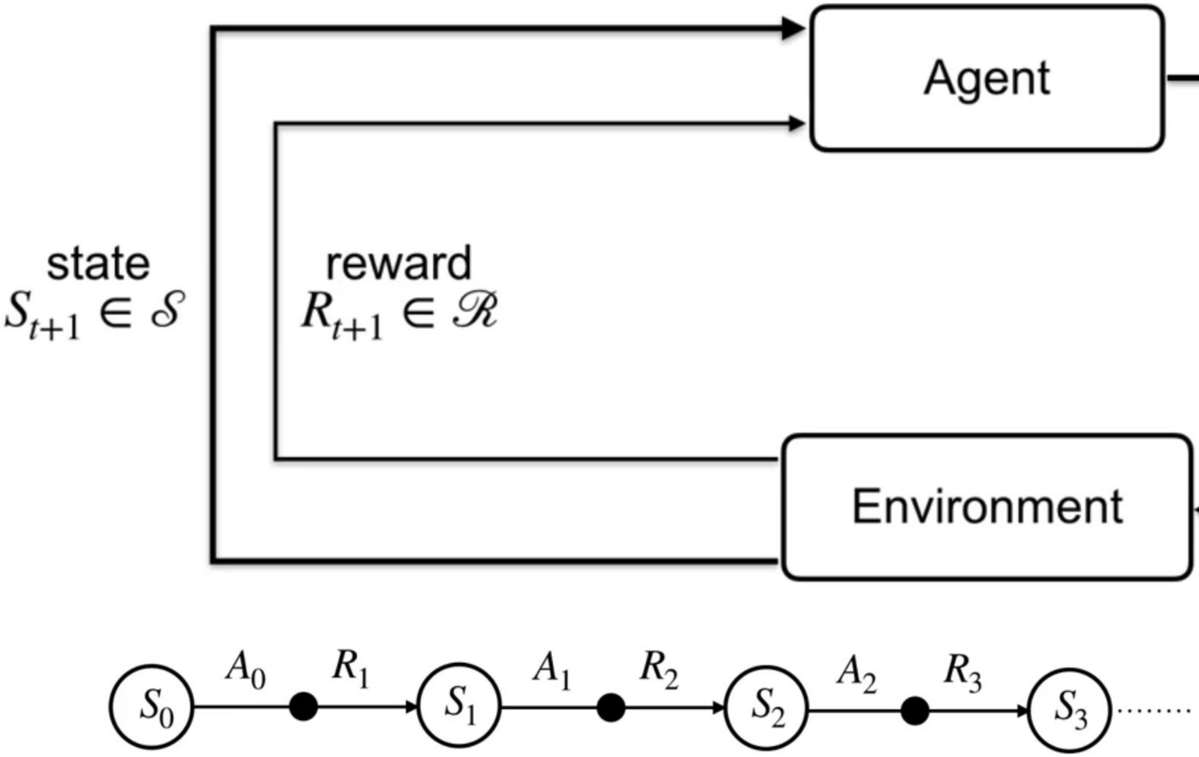
Examples are good and all, but we need a **formalization** of the RL decision making process

Why? So that we can apply this framework to almost any **decision making problem** (beyond just UofT labs and Bahen)

How can we do this?

Markov Decision Processes

Markov Decision Process



Trajectory of experience across timesteps

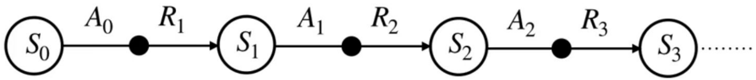
Markov Decision Process

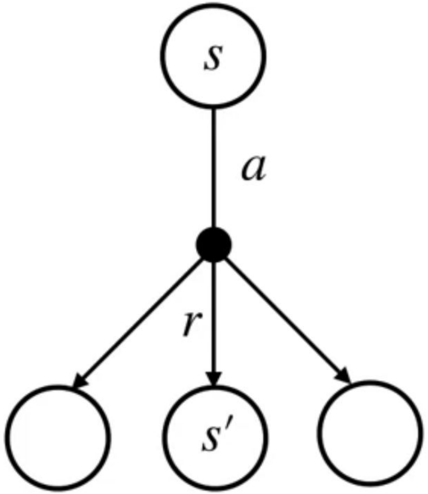
A Markov Decision Process (MDP) is defined by the following tuple: (S, A, P, R, γ)

- S is a finite set of states
- A is a finite set of actions
- P is state transition dynamics function, a probability distribution with $S \times A \times S \to [0, 1]$
- R is the reward function
- γ is the discount factor (from 0 to 1)

Dynamics of an MDP

Recall...





$$p(s', r | s, a)$$

Markov Property

All MDPs have the Markov property, or that the current state captures all the relevant information from history

Definition

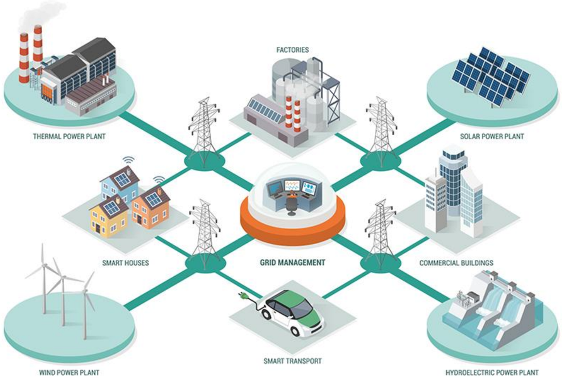
A state S_t is *Markov* if and only if

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

In our case, future states and rewards only depend on the current state and action $p(s', r|s, a)$

Episodic vs. Continuing tasks

- **Episodic Tasks:** Tasks that can be broken up into independent episodes, where each episode has a terminal state
- **Continuing Tasks:** Tasks that where there is no terminal state, and go on indefinitely



Summary of MDPs

- ❑ MDPs formally describe an environment for reinforcement learning
- ❑ Almost all reinforcement learning problems can be formalizes as MDPs
  - ❑ Bandits are special cases of MDPS with one state
- ❑ The MDP formulation can be extended to include continuous states and actions
- ❑ Partially observed problems can be turned into the MDP formulation described above

Inside an RL Agent

The RL Agent

A reinforcement learning agent consists of a combination of the following components:

- **Policy**: the agent's behavior function (it's brain)
- **Value function**: the agent's belief of how good each state and/or action is
- **Model (out of scope)**: the agent's representation of the dynamics of the MDP

Policy

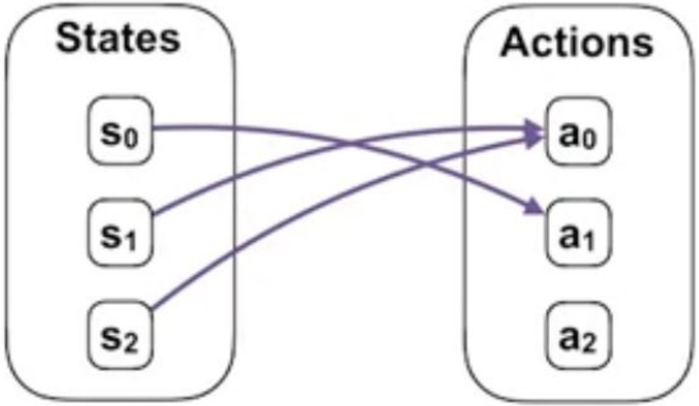
A policy is the agent's behavior, or its brain.
It determines the agent's action given a state

- ❑ An agent's policy <u>only</u> depends on its current state
- ❑ There are two kinds of policies: deterministic policies and stochastic policies

Deterministic Policy

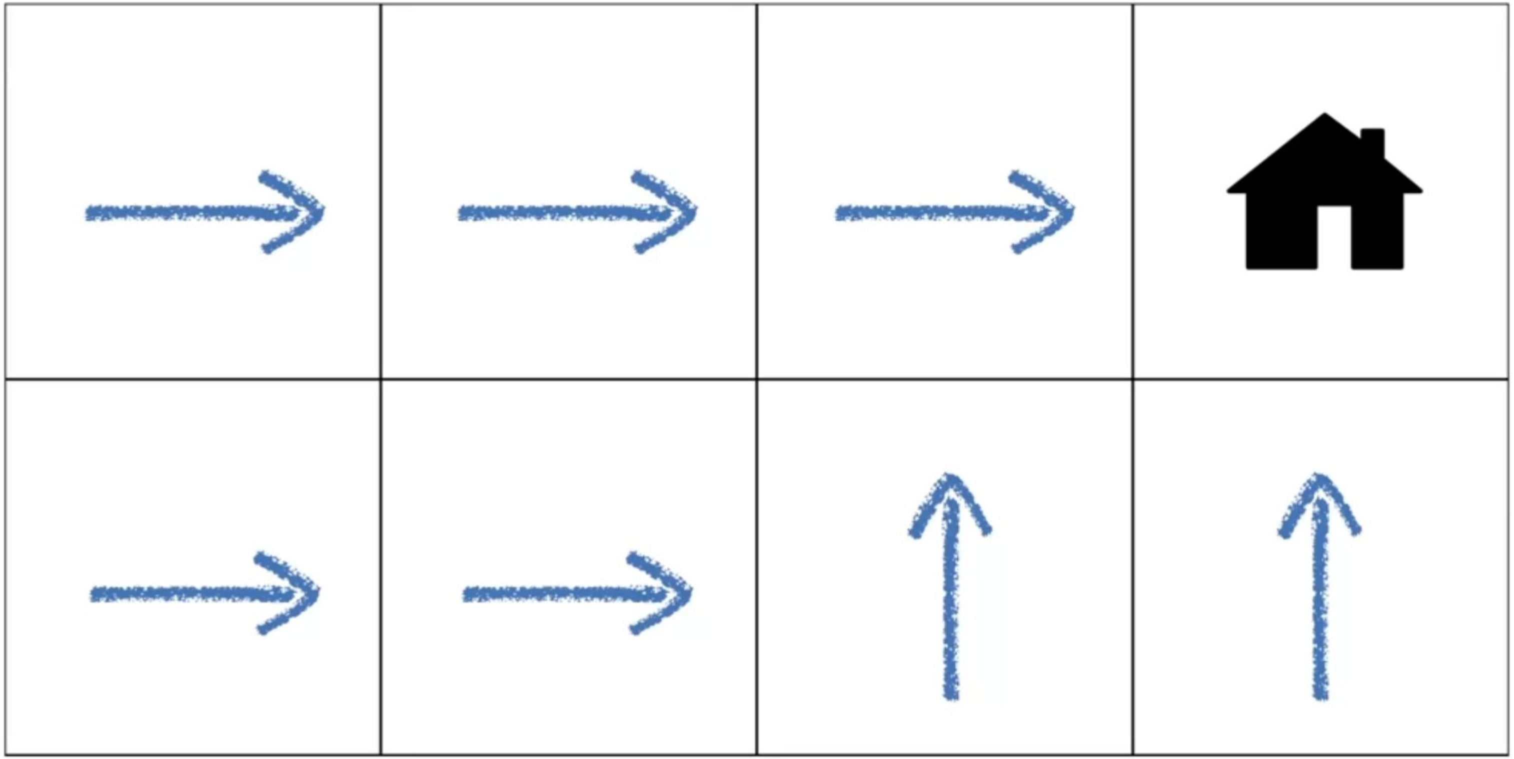
A deterministic policy maps a state to one action with 100% probability

$$\boxed{\pi(s) = a}$$



State	Action
s_0	a_1
s_1	a_0
s_2	a_0

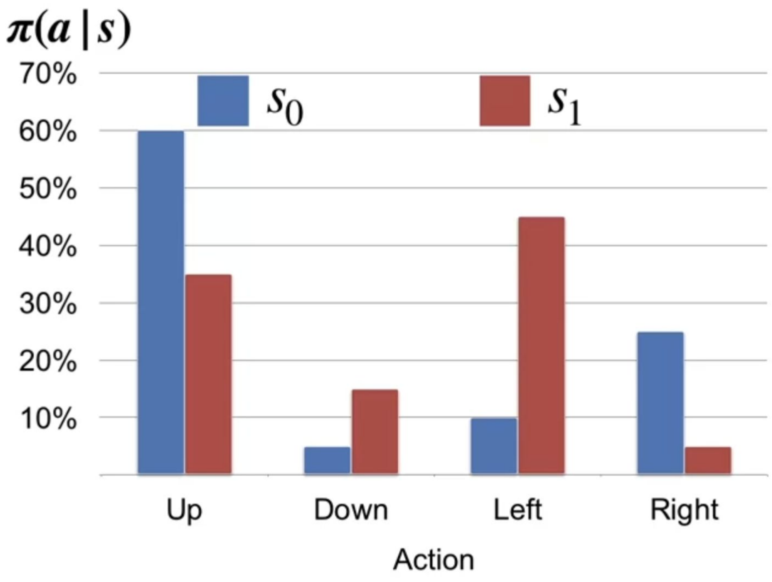
Deterministic Policy Example



Credit: University of Alberta

Stochastic Policy

Multiple actions may be selected with non-zero probability

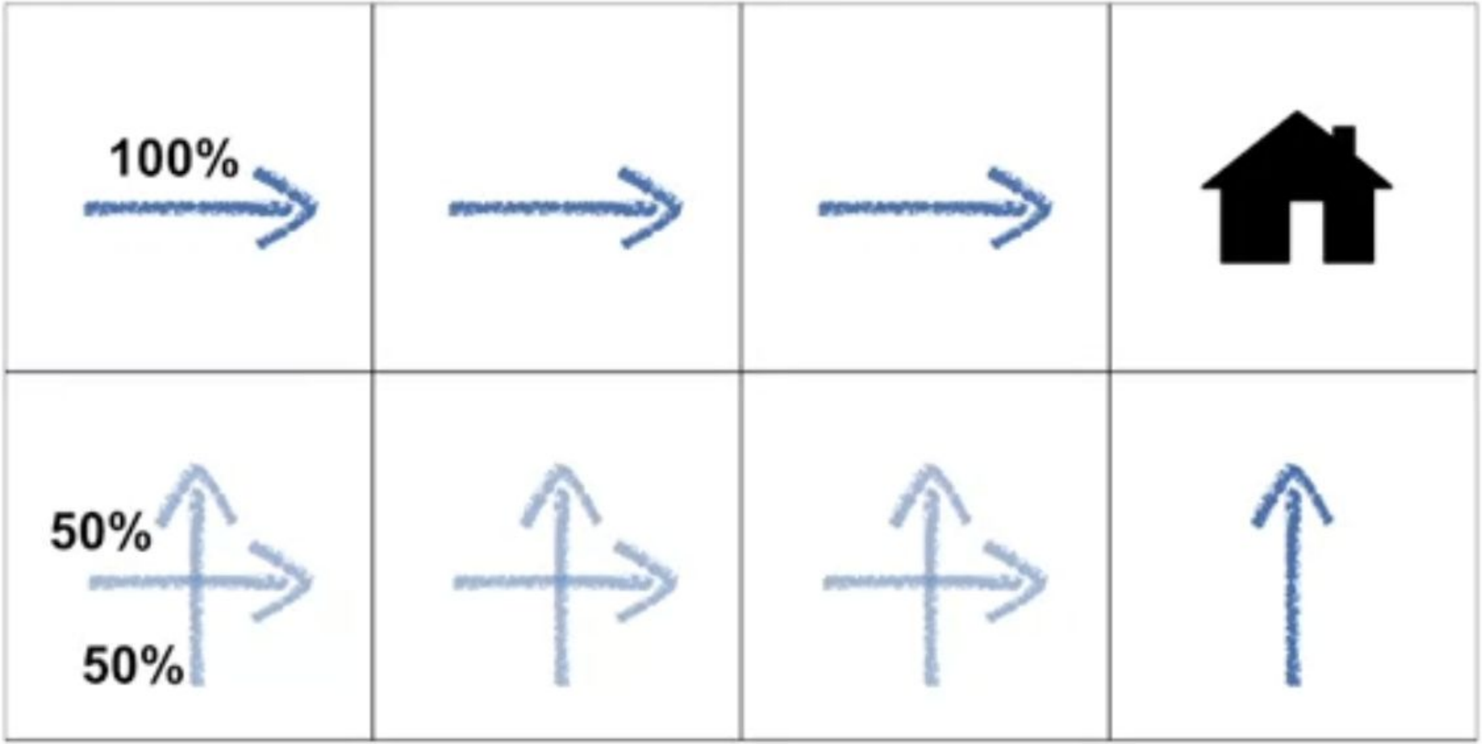


$$\boxed{\pi(a|s)}$$

$$\sum_{a \in A} \pi(a|s) = 1$$

$$\pi(a|s) \geq 0$$

Stochastic Policy Example



Credit: University of Alberta

State Value Function

- A numerical quantification of how good or bad a particular state is
- Defined as the expected return starting in that state and following policy π.

$$
\begin{aligned}
V_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s]
\end{aligned}
$$

State-Action Value Function

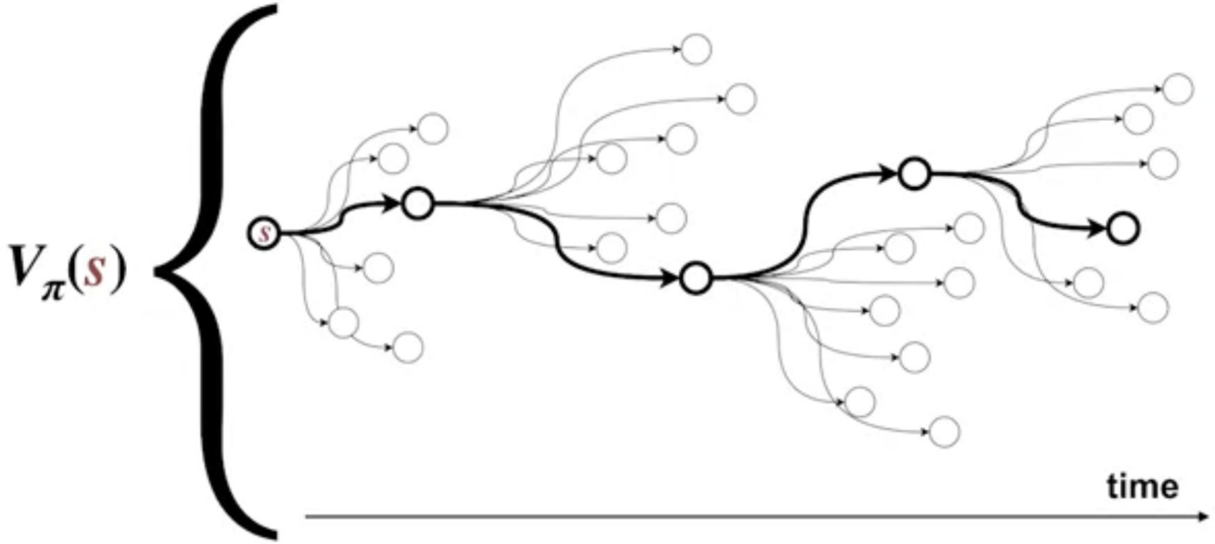
- A numerical quantification of how good or bad a particular state and action is
- Defined as the expected return starting in that state, taking that action, and following policy π.

$$
\begin{aligned}
Q_\pi(s,a) &= \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a]
\end{aligned}
$$

Different from $V_\pi(s)$!

Why Value Functions?

Summarizes (averages) all future choices into one concise representation

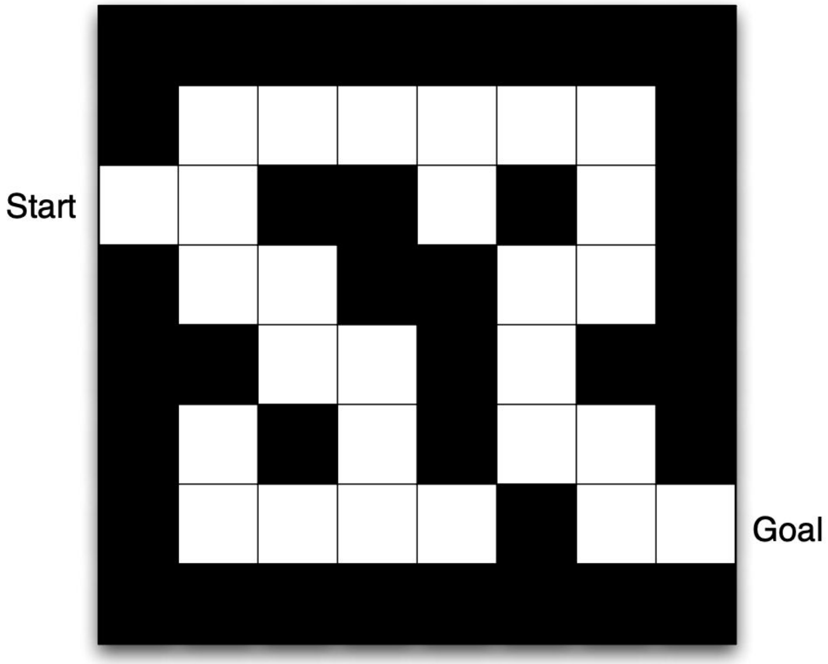


Credit: University of Alberta

Model (out of scope)

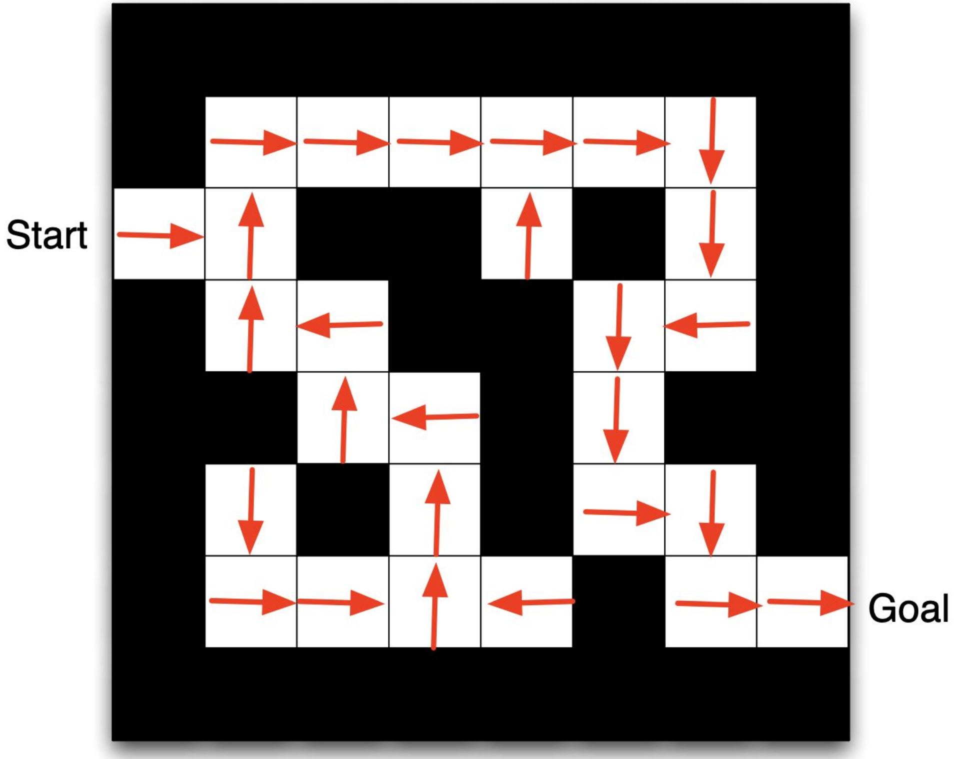
- An agent's representation of the environment dynamics, allowing it to predict what the world does next
- Can be designed to predict the next state only or the next state and reward
- NOT a statistical model (i.e. neural network), but the environment model

Example: Maze

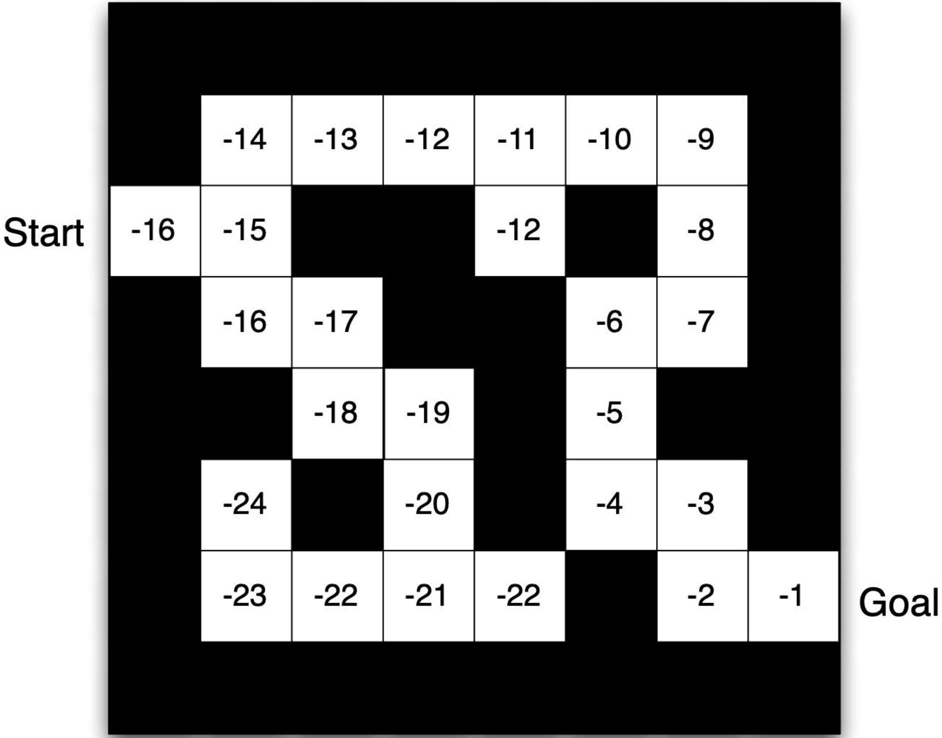


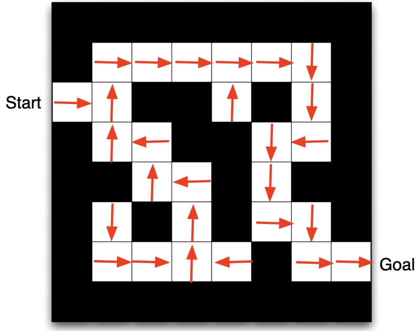
- ❑ Rewards: -1 per time step (escape ASAP!)
- ❑ Actions: North, South, East, West
- ❑ States: The agent's location in the maze

Example: Optimal Maze Policy



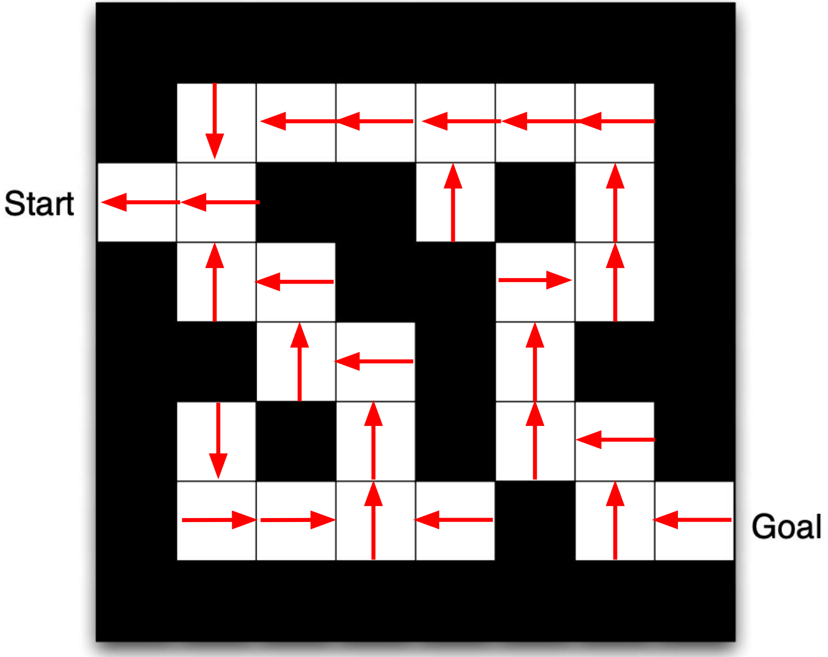
Example: Optimal State Value Function



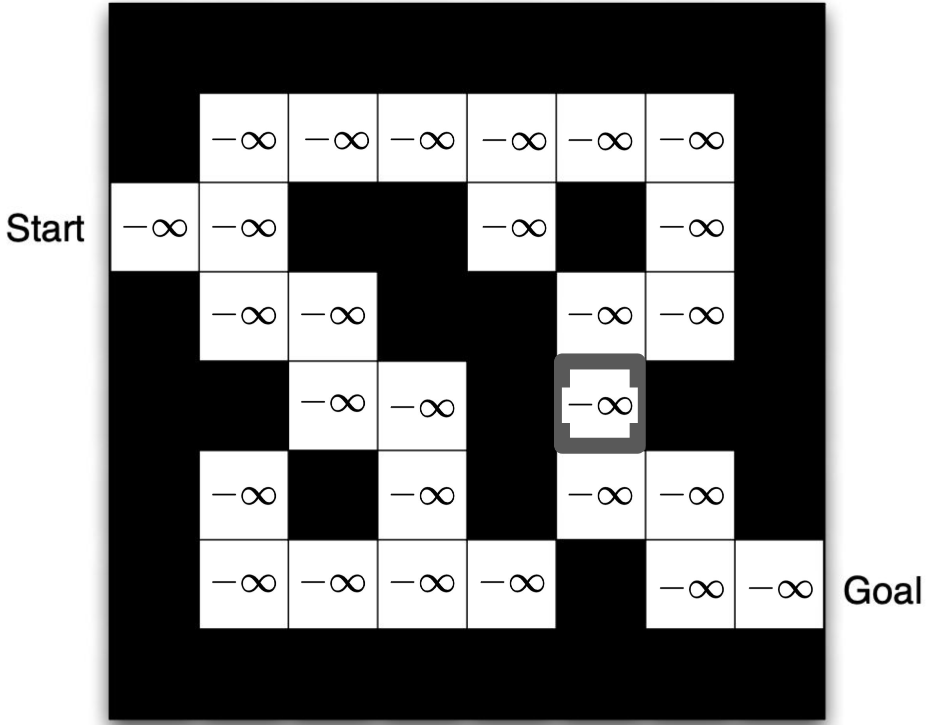


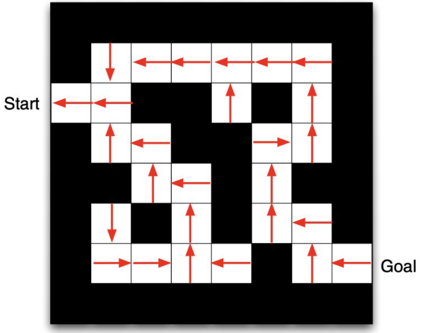
Recall: value $V_\pi(s)$ function depends on the policy!

Example: Bad Maze Policy



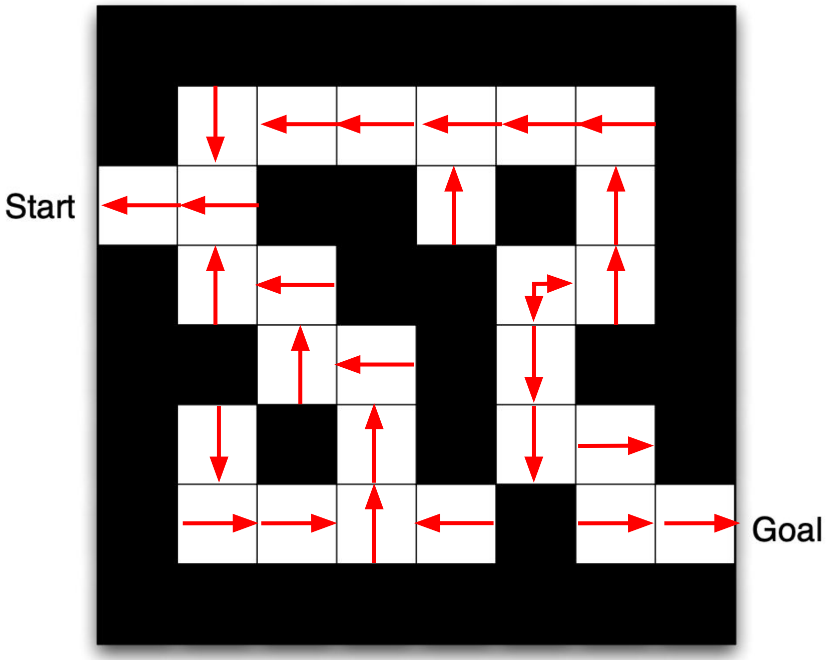
Example: Bad Policy State Value Function





What is the value of the highlighted state?

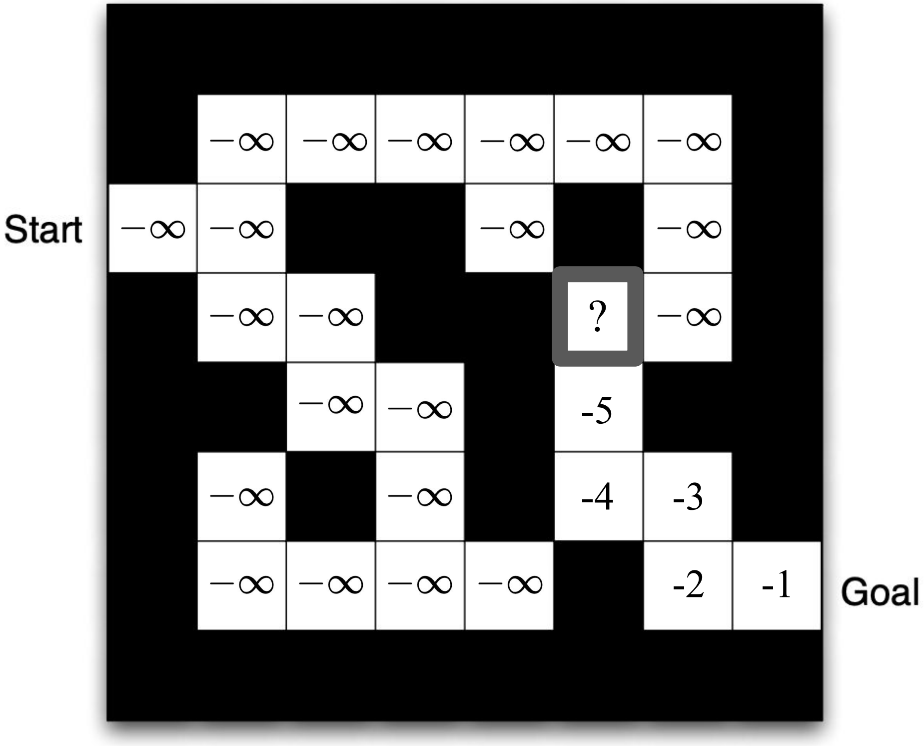
Example: Another Policy

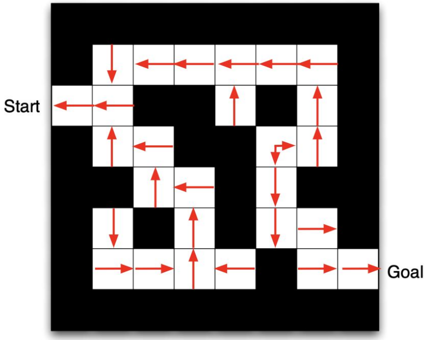


At highlighted state, 50% chance of going right, 50% change of going down

What type of policy is this?

Example: Stochastic Policy Value Function



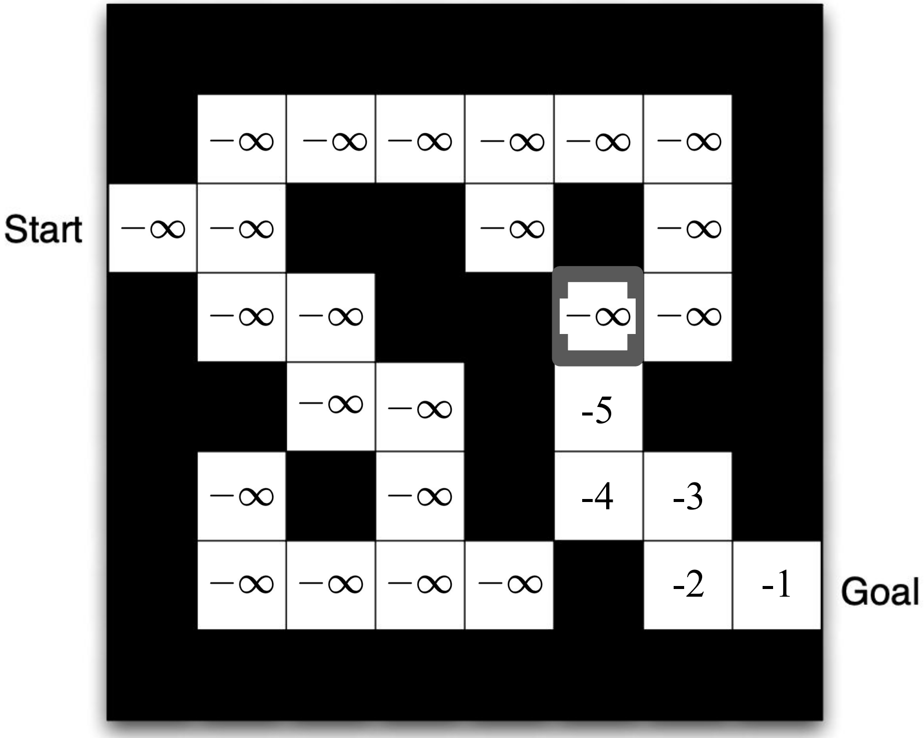


What is the value of that state?

Recall:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

Example: Stochastic Policy Value Function



Recall: Expected Value

$$E[X] = \sum x_i p(x_i)$$

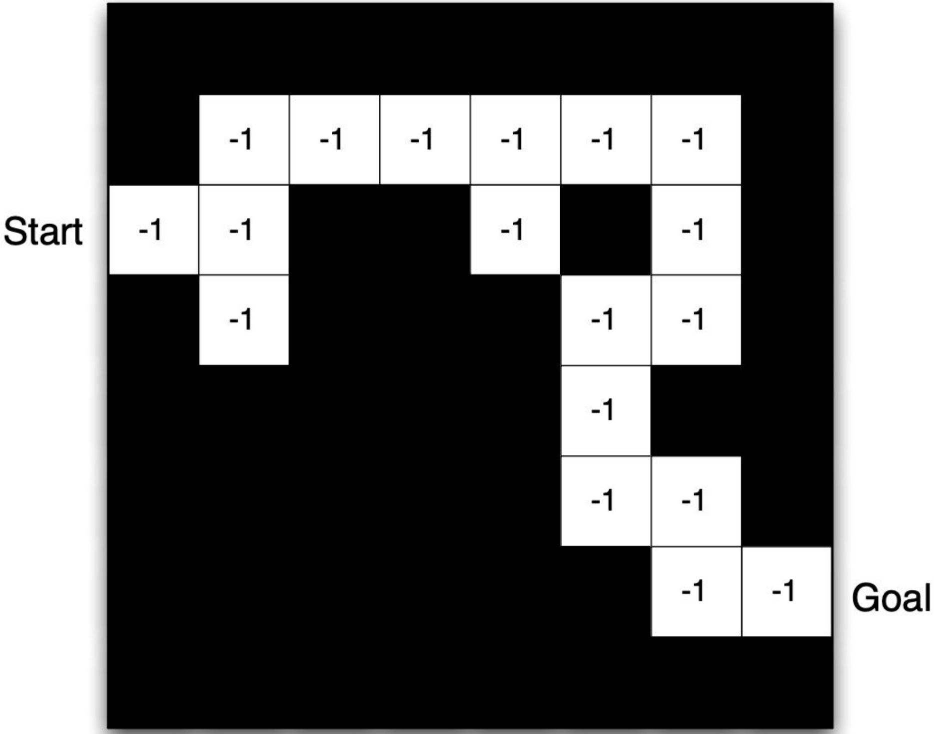
x_i = The values that X takes

$p(x_i)$ = The probability that X takes the value x_i

Take the expected value (average) of the possible returns:

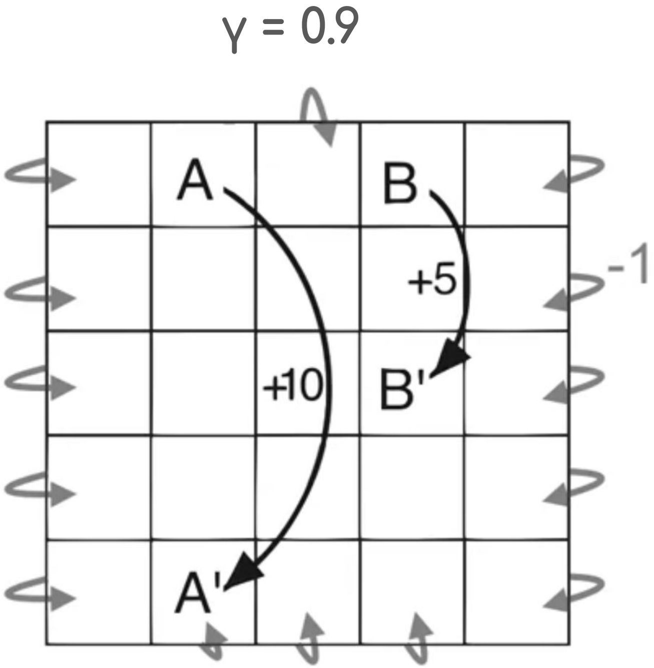
$$V_\pi(?) = 0.5(-6) + 0.5(-\infty) = -\infty$$

Example: Maze Model



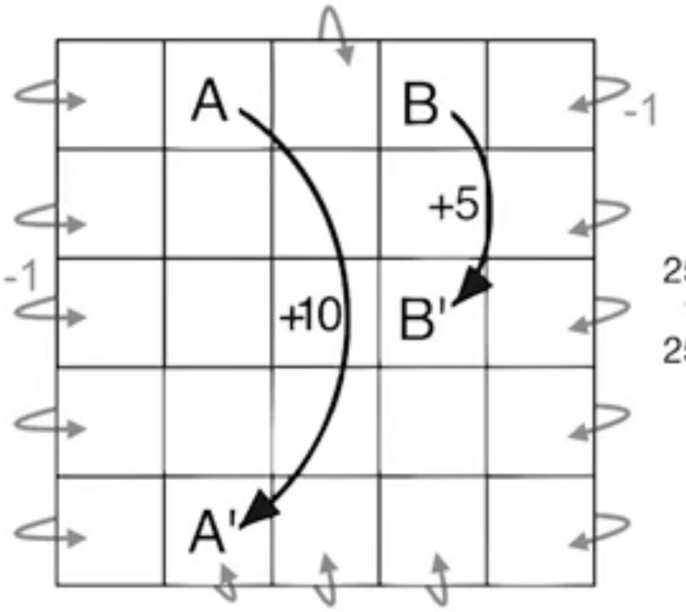
The -1s represent the immediate reward from each state, which is the same, as defined

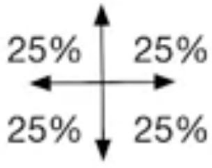
Another Value Function Example



- Rewards: 0 for any transitions, -1 for bumping into walls, +10 for any action in A and +5 for any action in B
- Actions: North, South, East, West
- States: The agent's location in the world

Another Value Function Example

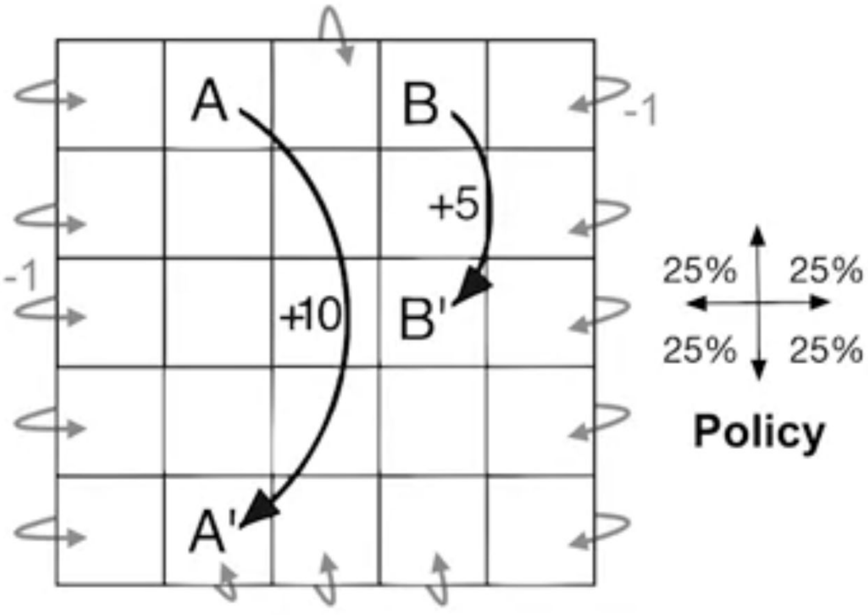




Policy

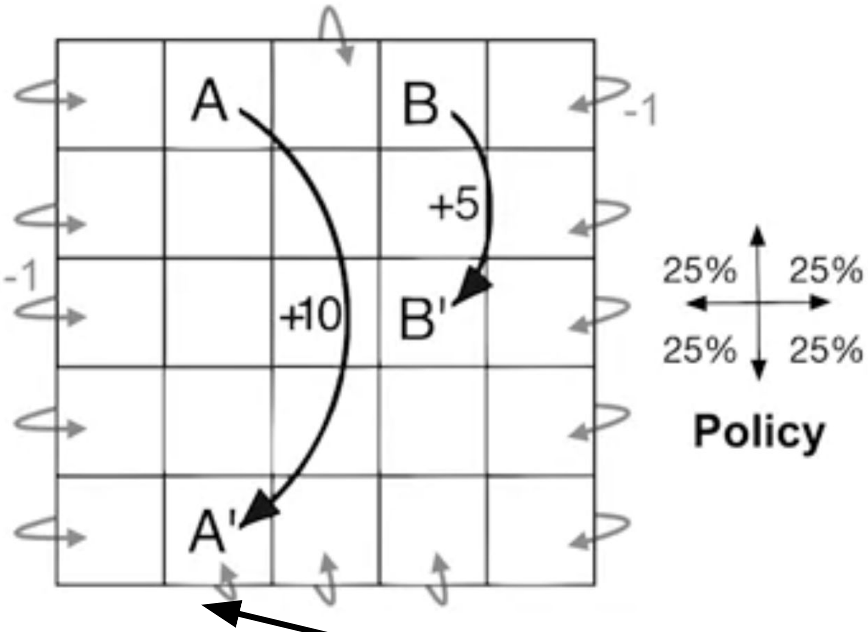
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Another Value Function Example



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

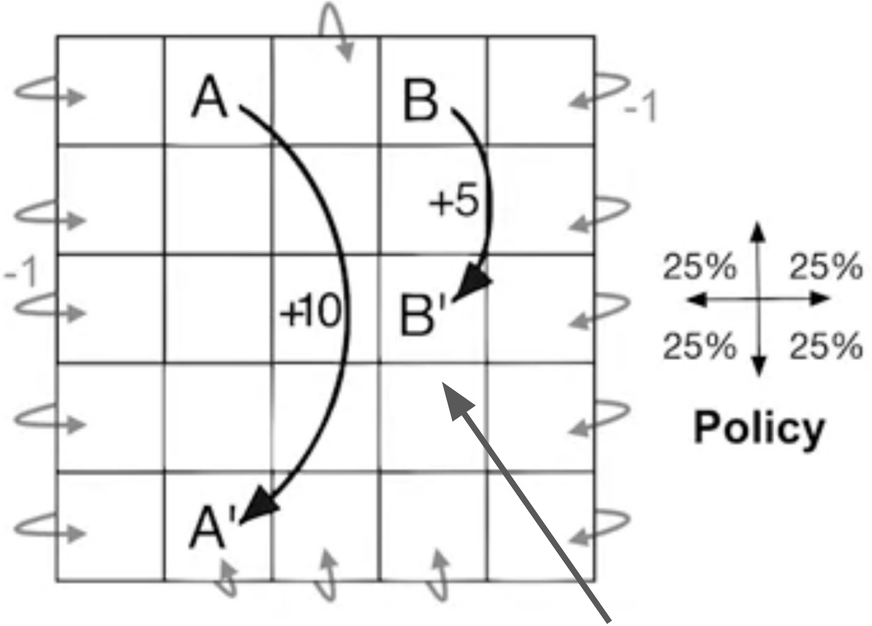
Another Value Function Example



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Bad spot to be

Another Value Function Example



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Better spot to be

Classifying RL Agents

As mentioned previously, RL agents may contain a mixture of a policy, value function(s), and model:

Value-based: Value function, no explicit policy

Policy-based: Explicit policy, no value function

Actor-Critic: Policy and value function

Model-free: Policy and/or value function, no model

Model-based: Policy and/or value function, model

End of Week 1 Questions?

Please Fill Out the Feedback Survey