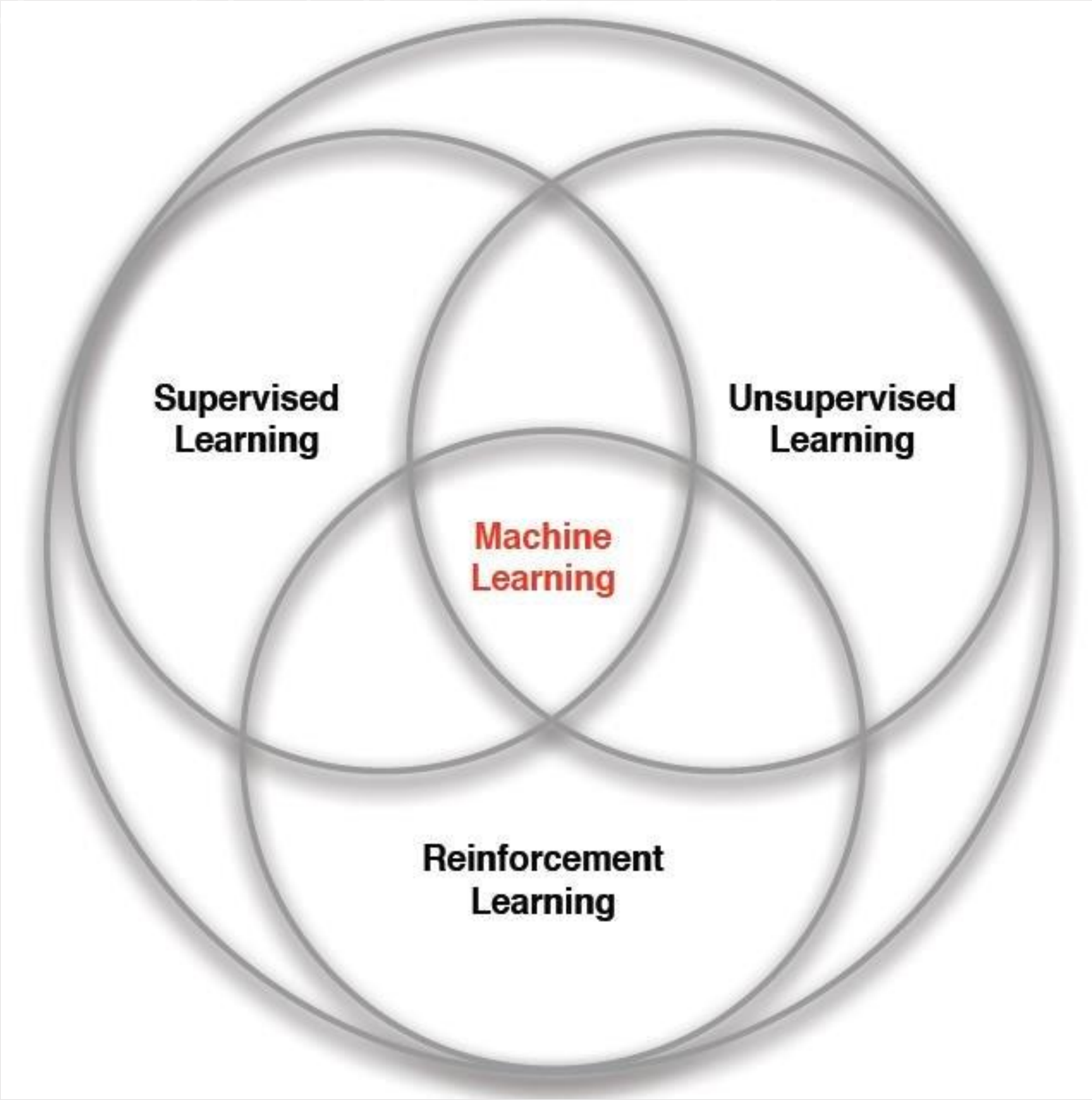
CS 540 Introduction to Artificial Intelligence

Reinforcement Learning

Asmit Nayak
UW-Madison

Based on slides by David Silver, Fred Sala, Yingyu Liang and R. Sutton

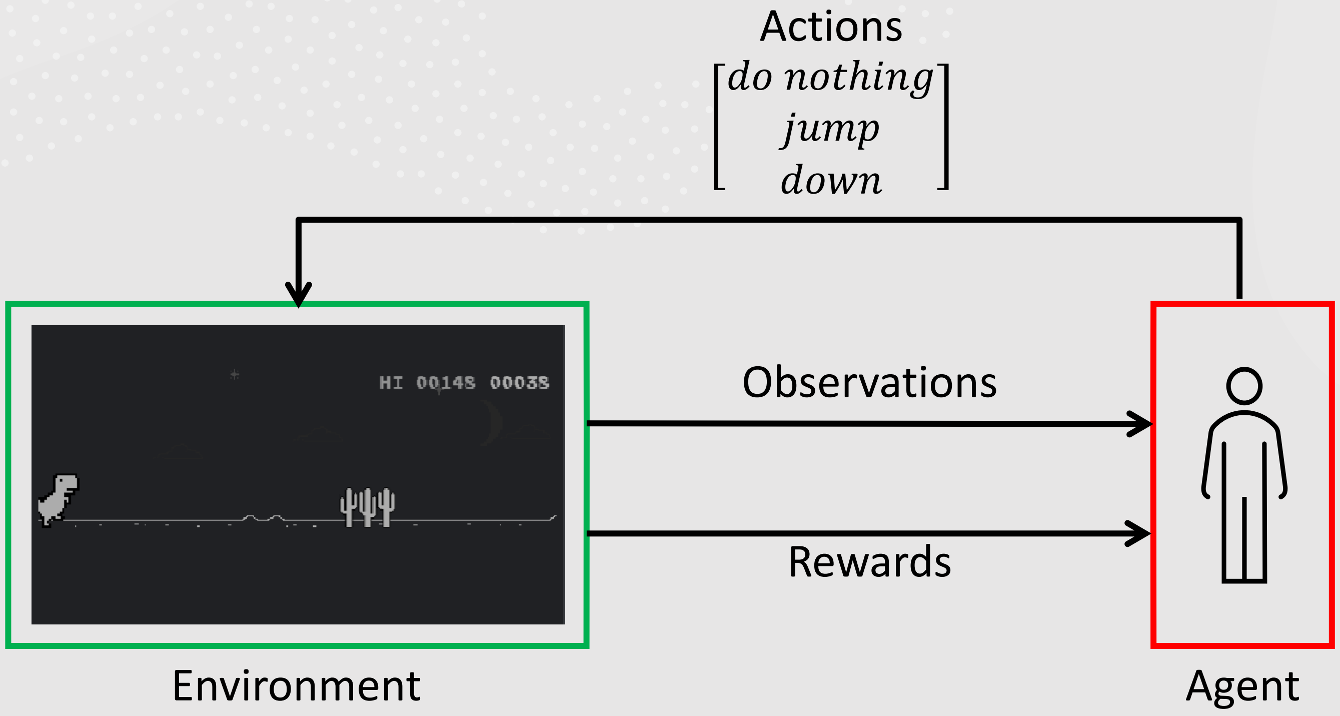
Branches of Machine Learning



What makes RL different?

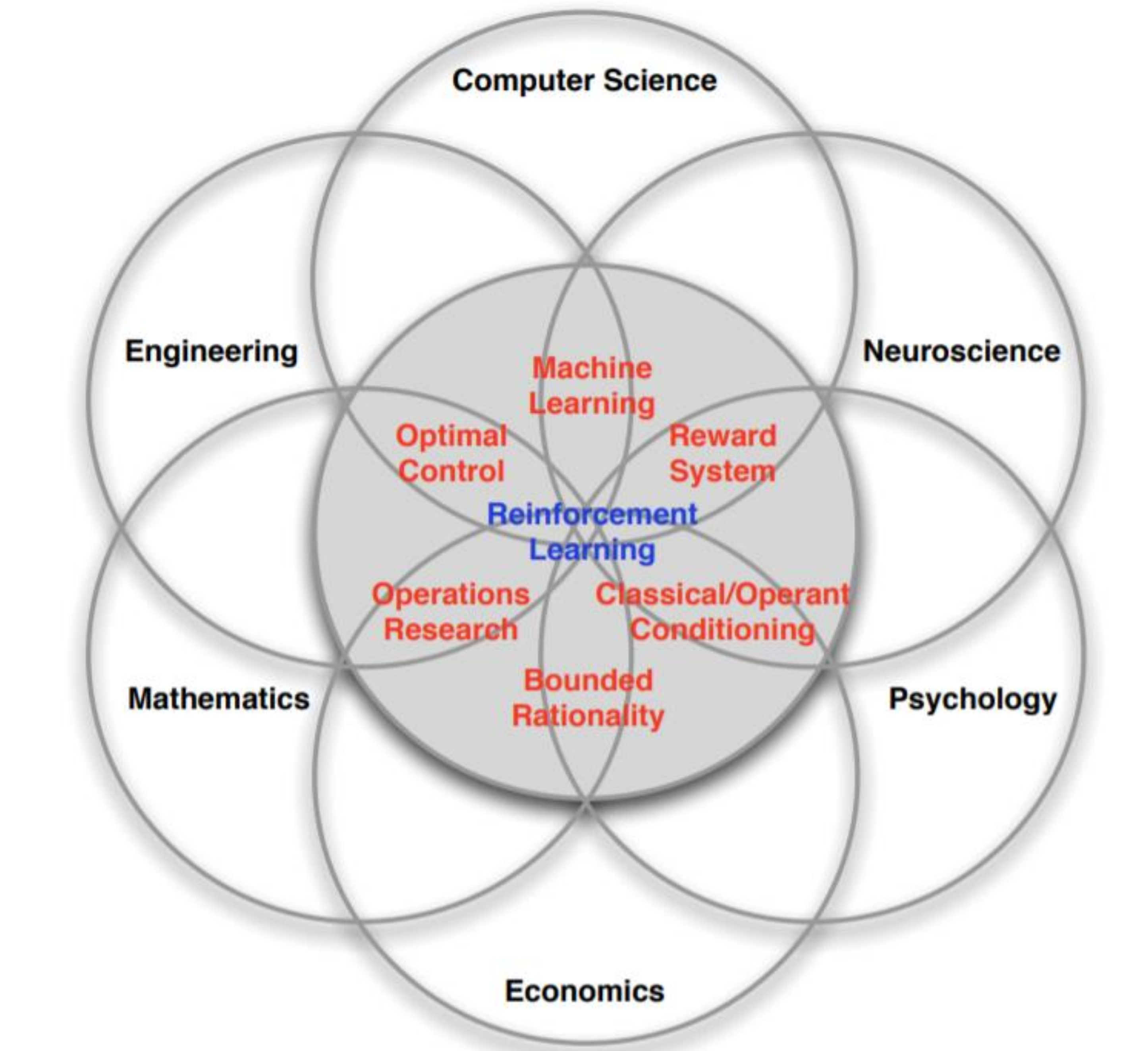
- No “supervisor”; only rewards
- Feedback is delayed
- Time (or, timesteps) matters!
- Actions affect the subsequent observations/next-actions

Basic construct of a RL model?



Goal: Maximize total reward

Uses of RL in the world



Examples of RL in wild.

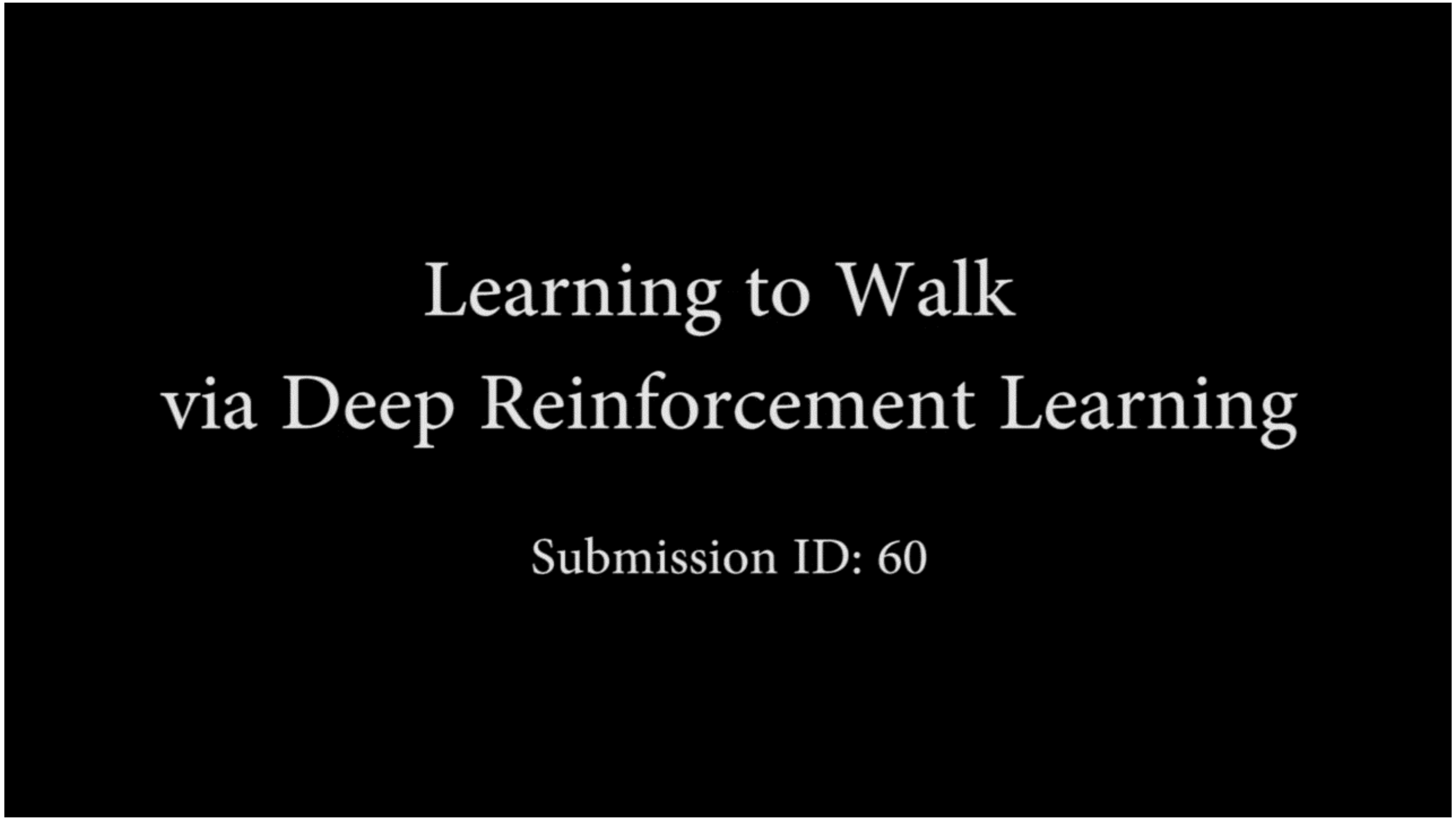
Playing strategy games like Dota 2. (Open-AI Five)

Defeat professional Go player. Go is *the* most challenging game for an AI. (Alpha Go)

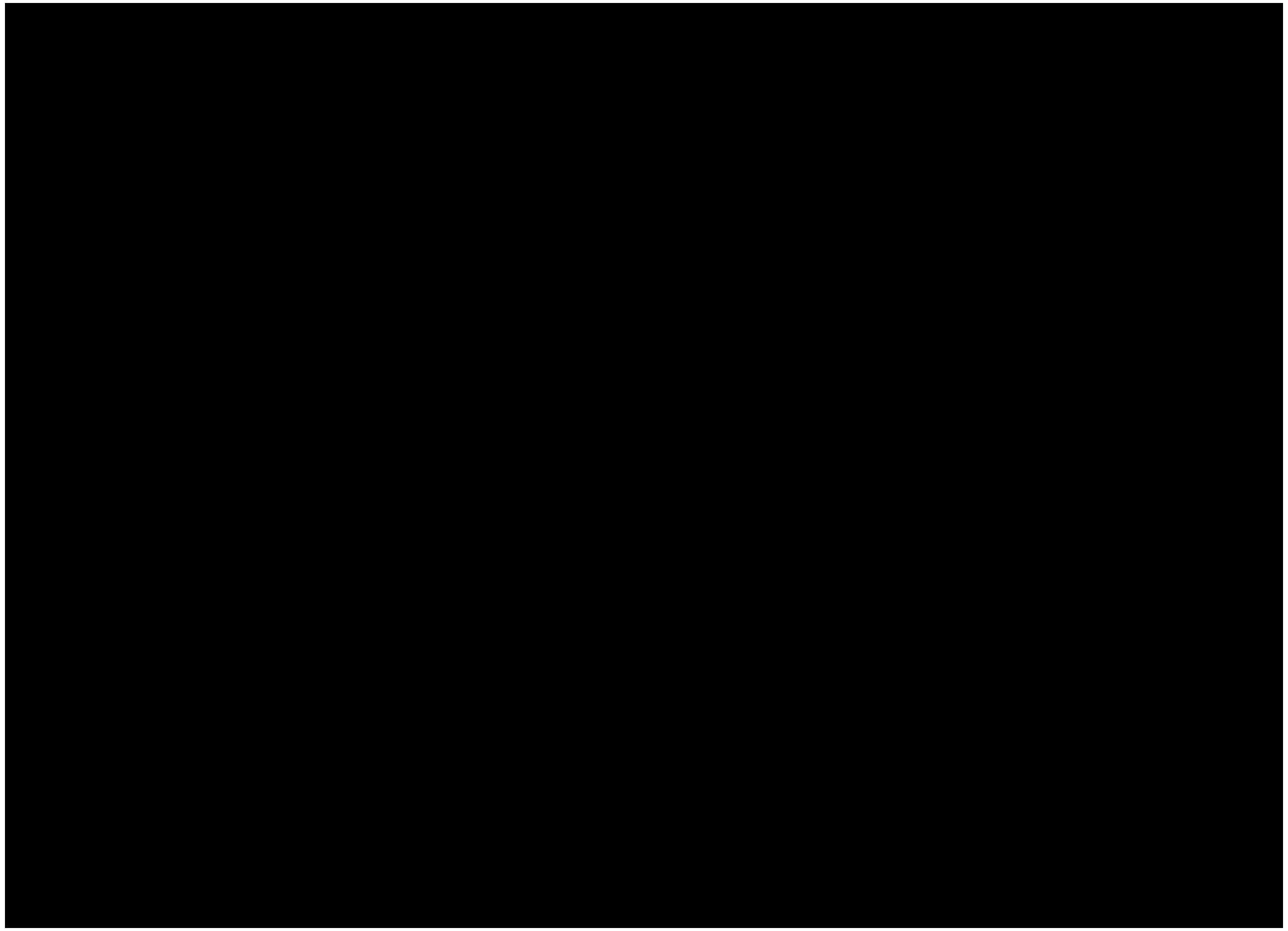
Make a humanoid robot walk

Fly a RC-helicopter

A robot learning to walk.



DeepMind AI Learning to walk.



DeepMind playing Atari Breakout.

Building an RL "Model"

Basic setup:

- Set of states; S (Note: observations $\subseteq S$)
- Set of actions; A
- Information: at time t, state is $s_t \in S$, reward r_t, and the <u>history</u> till then
- Agent makes a choice of action (a_t) based on the information at time t moving on to state s_{t+1}

<u>*POLICY*</u>

GOAL:

Find <u>*a map*</u> of state-action pairs to maximize rewards

History & States

Q. What is ***history***?

Ans: The history is the sequence of observations, actions, rewards, i.e., all observable variables up to time t

The action taken by agent at t is determined based the history at time t

Q. What is ***state***?

Ans: State is the information used to determine what happens next

State is the function of history:

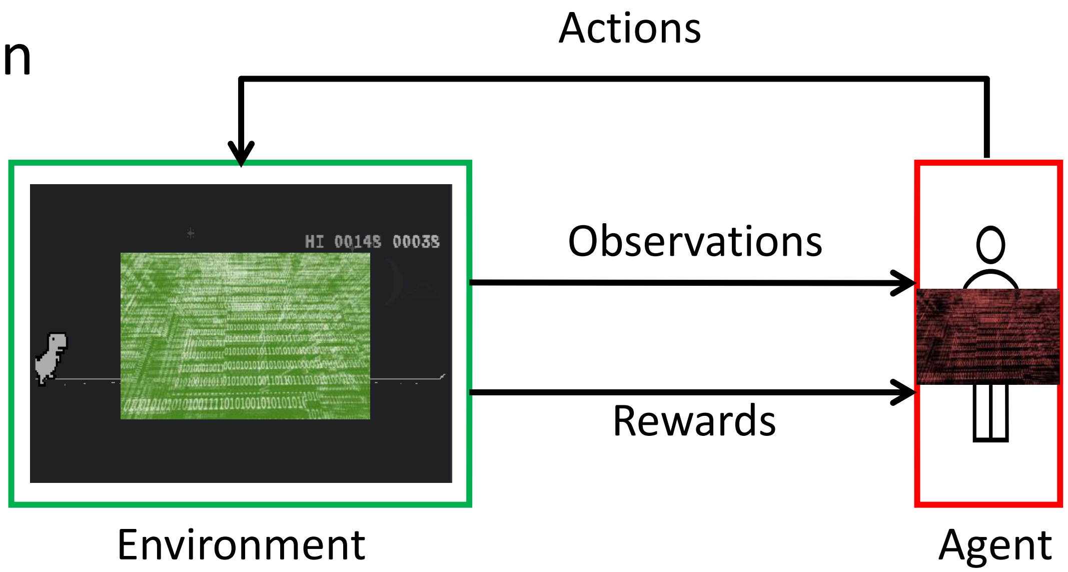
$$S_t = f(H_t)$$

States

Environment State: S_t^e

The private representation of the environment, used to get the next set of observations and rewards

May contain irrelevant information



Agent State: S_t^a

The internal representation of the agent used to predict the next set of actions

Contains information and is a subset of S_t^e

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

States

The <u>*information state*</u> (Markov state) contains <u>**all the useful information**</u> from the history

Definition:

A state S_t is a <u>*Markov State*</u> iff:

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

- "The future is independent of the past given the present"
- Once the state is known, history can be discarded
- S_t^a is Markov iff it contains <u>**all the necessary infos**</u>
- S_t^e is Markov; H_t is Markov

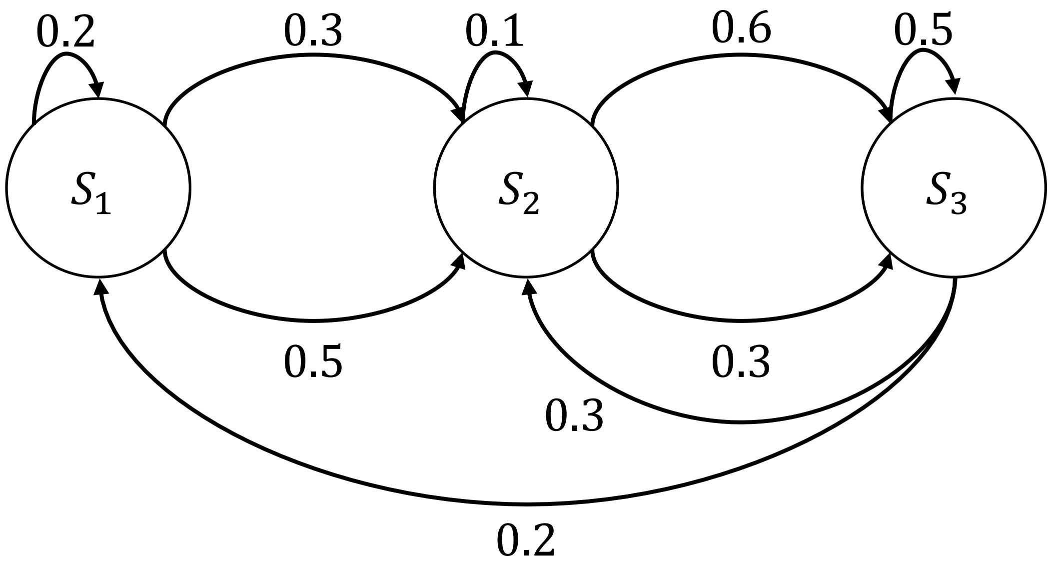
Markov States

$\mathbb{P}(S_t = S_1 | S_{t-1} = S_1, S_{t-2} = S_3) = 0.2$

$\mathbb{P}(S_t = S_1 | S_{t-1} = S_1, S_{t-2} = S_1) = 0.2$

$\mathbb{P}(S_t = S_3 | S_{t-1} = S_2, S_{t-2} = S_1) = 0.6$

$\mathbb{P}(S_t = S_3 | S_{t-1} = S_2, S_{t-2} = S_2) = 0.6$



Minions Example

(adapted from David Silver)

- What if S_t^a = last 3 items of the sequence?
- What if S_t^a = count of lights, bulbs and levers?
- What if S_t^a = exact sequence?

Markov Decision Process

Given,

1. Set of States: S, and a set of Actions: A
2. Markov State Transitions model: $P(s_{t+1} \mid s_t, a_t)$
3. Reward functions: $r(s_t)$

Find the optimal ***policy***: $\pi(s_t): S \rightarrow A$

RL Agent

RL Agent has one or more of the following features:

- Policy: Agent’s behavior function (mapping of states and actions)
- Value Function: How good a state and/or action is
- Model: Agent’s representation of the environment

Policy

- Policy: Agent's behavior function
- It's a mapping from states to actions
- Deterministic Policy: $a = \pi(s)$
- Stochastic Policy: $\pi(s|a) = \mathbb{P}[A_t = a \mid S_t = s]$
    - Good for exploration

Value Function

- Predicts the future rewards from a state, <u>for a given policy</u>
- Used to evaluate how good/bad a state is
- Is used to select between actions.

$$V_\pi(s_t) = \sum_{t=t}^{T} r(s_t) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots]$$

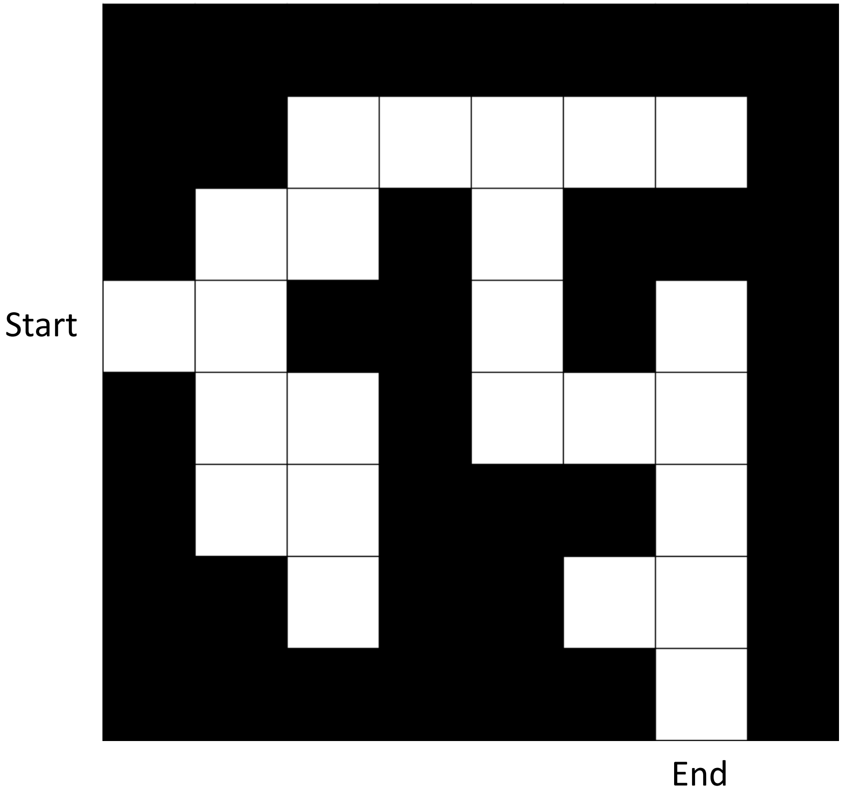
Model

- Predicts the environment would do next
- $\mathcal{P}$ predicts the next state
- $\mathcal{R}$ predicts the immediate rewards

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

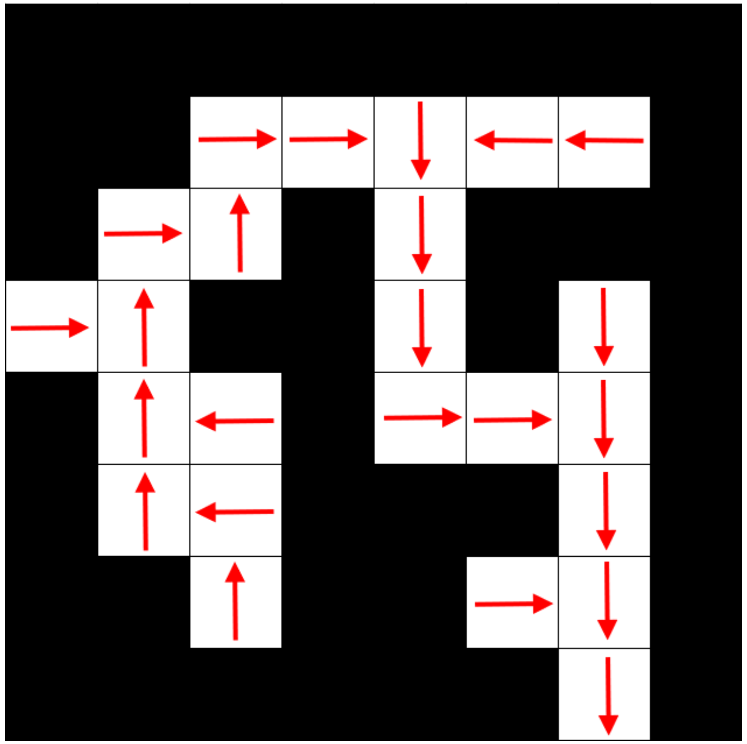
$$\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

Maze Example:



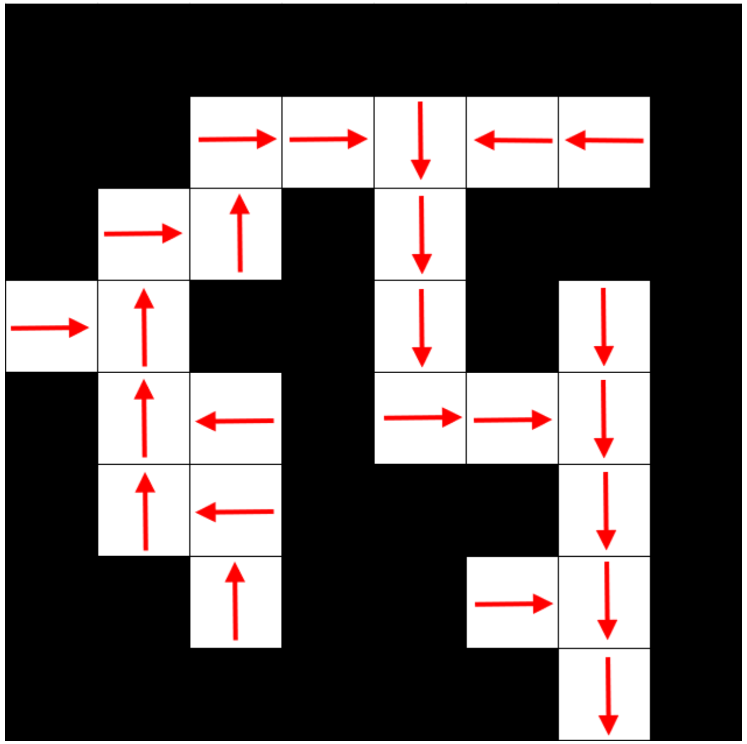
- Reward = -1 per timestep
- Actions = N,E,W,S
- States = Agent’s location

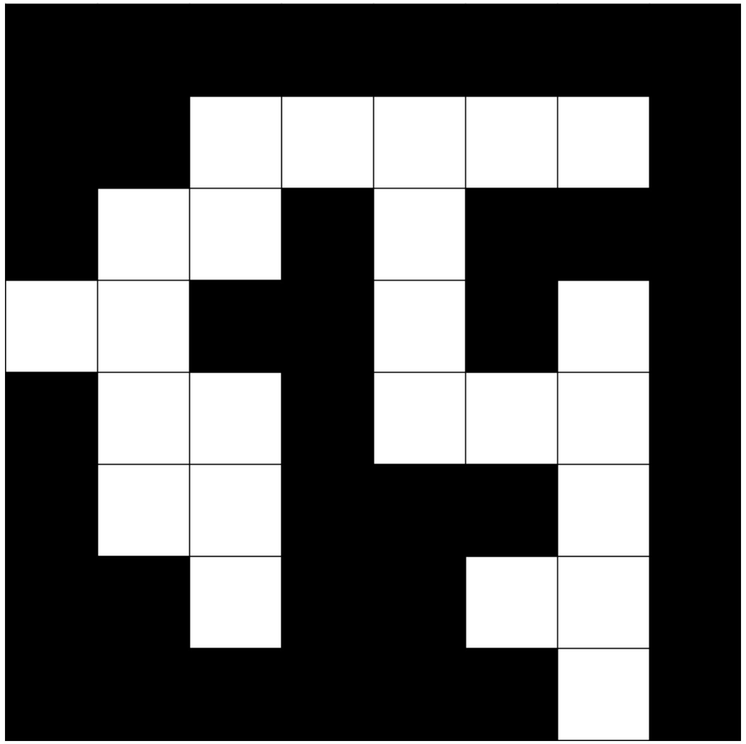
Maze Example:



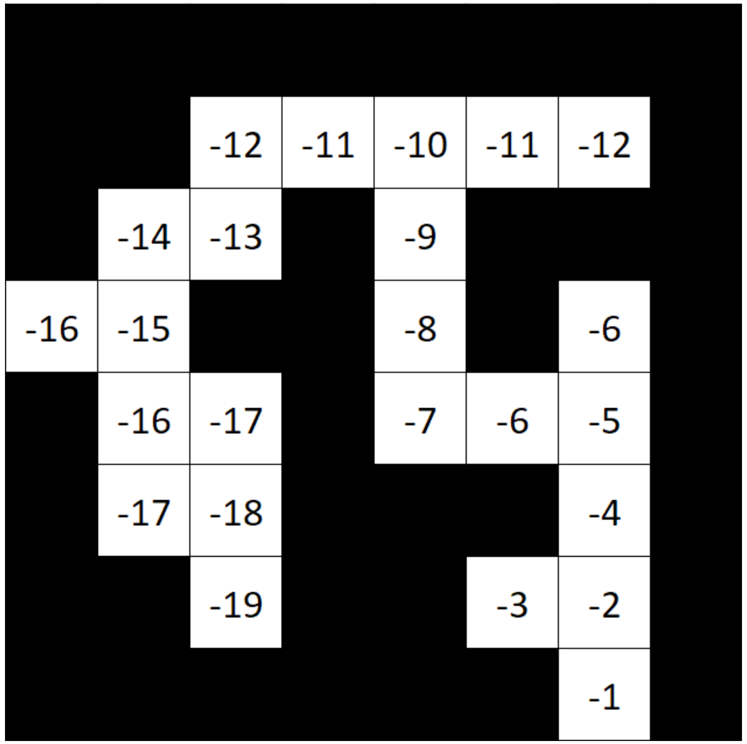
Arrows represent the policy $\pi(s)$ at each state s

Maze Example:





Maze Example:



Values represent $V_{\pi}(s)$ for each state s following policy π.

Value Function → Policy

Value Function:

$$V_\pi(s_t) = \sum_{t=t}^{T} r(s_t)$$

… but this doesn't converge!

Discounted Rewards: $0 \leq \gamma < 1$

$$V_\pi(s_t) = r(s_t) + \gamma r(s_{t+1}) + \gamma^2 r(s_{t+2}) + \cdots = \sum_{t \geq 0} \gamma^t r(s_t)$$

Value Function → Policy

So now that we have a value function V_π for policy π how do we get the optimal policy?

Let's the optimal policy be π^* and its corresponding value function be V^*

So, what will be the expected value of an action for an agent be?

$$\sum_{s'} \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] V^*(s')$$

All possible states

Transition Probability

Expected Rewards

Value Function → Policy

Therefore, the optimal policy would be:

$$\pi^*(s) = \text{argmax}_a \sum_{s'} \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] V^*(s')$$

BUT!!

We defined V^* in terms of π^*

Value Function → Policy

How do we get v^* then?

$$V_\pi(s_t) = r(s_t) + \gamma r(s_{t+1}) + \gamma^2 r(s_{t+2}) + \cdots = \sum_{t \geq 0} \gamma^t r(s_t)$$

$$\Rightarrow V_\pi(s_t) = \sum_a \pi(a|s_t) \sum_{s_{t+1},r} \mathbb{P}[s_{t+1}, r|s_t, a][r + \gamma V_\pi(s_{t+1})]$$

$$\boxed{\therefore\ V^*(s) = \sum_a \pi(a|s) \sum_{s',r} \mathbb{P}[s', r|s, a][r + \gamma V_\pi(s')]}$$

(Bellman's Equation)

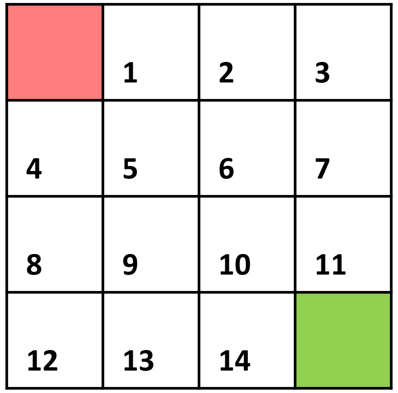
Value Iterations

We know:

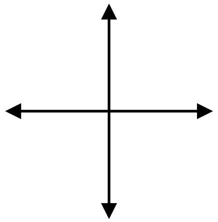
- $r(s)$ and transition probabilities $\mathbb{P}[s'|s, a]$
- V^* satisfies the Bellman equation, as it's recursive
- Therefore, we will use the above property and start with $V_0(s) = 0$, and update the value per-iteration as:

$$V_{i+1}(s) = \sum_{a} \pi(a|s) \sum_{s',r} \mathbb{P}[s', r|s, a][r + \gamma V_\pi(s')]$$

Value Iterations Example



Grid World



Actions

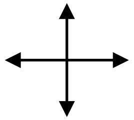
$R_t = -1$
on all transitions

$\gamma = 0.9$
$\alpha = 1$

$\pi_0 \Rightarrow \mathbb{P}[a] = 0.25 \; \forall \; a$

For terminal states $s' = s$

Value Iterations Example



$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

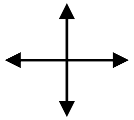
$k = 2$

0	-1.675	-1.9	-1.9
-1.675	-1.9	-1.9	-1.9
-1.9	-1.9	-1.9	-1.675
-1.9	-1.9	-1.675	0

$$V_k = 3\big(0.25 * (-1 + 0.9 * (-1))\big) + 0.25 * (-1 + 0) = -1.675$$

$$V_k = 4\big(0.25 * (-1 + 0.9 * (-1))\big) = -1.9$$

Value Iterations Example



$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 2$

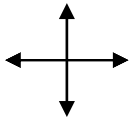
0	-1.675	-1.9	-1.9
-1.675	-1.9	-1.9	-1.9
-1.9	-1.9	-1.9	-1.675
-1.9	-1.9	-1.675	0

$k = 3$

0	-2.23	-2.67	-2.71
-2.23	-2.61	-2.71	-2.66
-1.8	-2.71	-2.61	-2.23
-2.71	-2.66	-2.23	0

$$\begin{aligned} V_k = {} & 0.25 * \big(-1 + 0.9 * (-1.675)\big) \\ & +0.25 * \big(-1 + 0.9 * (-1.9)\big) \\ & +0.25 * \big(-1 + 0.9 * (-1.9)\big) \\ & +0.25 * (-1 + 0) = -2.23 \end{aligned}$$

Value Iterations Example



$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 3$

0	-2.23	-2.67	-2.71
-2.23	-2.61	-2.71	-2.66
-1.8	-2.71	-2.61	-2.23
-2.71	-2.66	-2.23	0

$k = \infty$

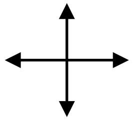
0	-5.28	-7.13	-7.65
-5.28	-6.60	-7.18	-7.13
-7.13	-7.17	-6.60	-5.28
-7.65	-7.13	-5.28	0

Policy Iterations

For a policy π:

- Evaluate $V_\pi(s)$
- Update $\pi \longleftarrow \pi'$
- Keep updating till $\Delta \longrightarrow 0$, where $\Delta = \max(\Delta,\ V_\pi(s) - V_\pi'(s))$

Value Iterations Example



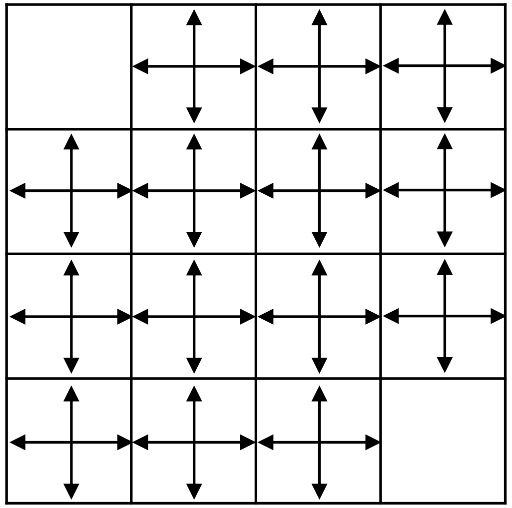
$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 0$

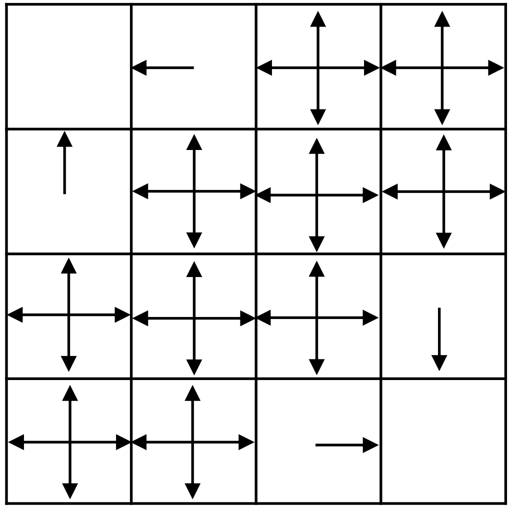
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0



$\pi_{k=0}$

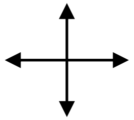
$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0



$\pi_{k=1}$

Value Iterations Example



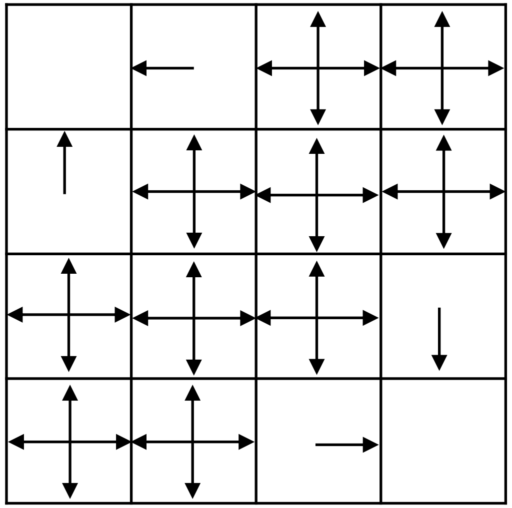
$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 1$

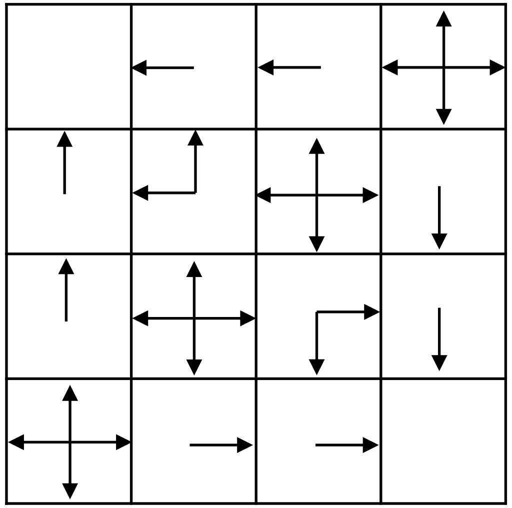
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0



$\pi_{k=1}$

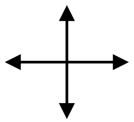
$k = 2$

0	-2.23	-2.67	-2.71
-2.23	-2.61	-2.71	-2.66
-1.8	-2.71	-2.61	-2.23
-2.71	-2.66	-2.23	0



$\pi_{k=2}$

Value Iterations Example



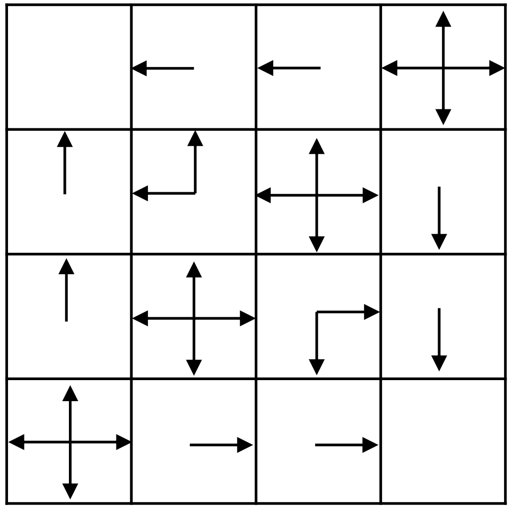
$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 2$

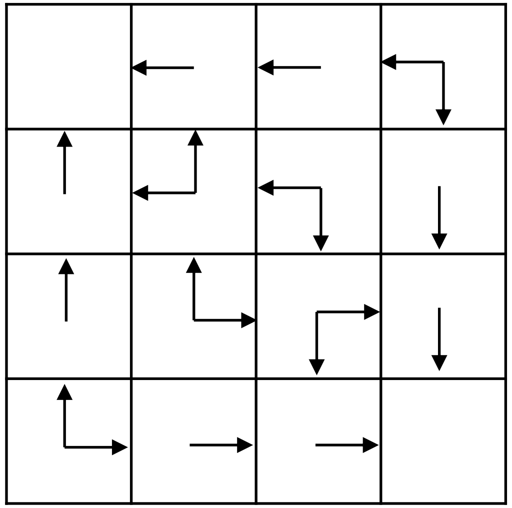
0	-2.23	-2.67	-2.71
-2.23	-2.61	-2.71	-2.66
-1.8	-2.71	-2.61	-2.23
-2.71	-2.66	-2.23	0



$\pi_{k=2}$

$k = \infty$

0	-5.28	-7.13	-7.65
-5.28	-6.60	-7.18	-7.13
-7.13	-7.17	-6.60	-5.28
-7.65	-7.13	-5.28	0



$\pi_{k=\infty}$

Q-Learning

For the previous value iteration, we knew $\mathbb{P}[s'|s,a]$. What if we didn't?

We will use Q-Learning!

Q-Learning tells us the value of doing a in state s

$$Q(s_t, a_t) = \mathbb{E}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots \mid s, a]$$

We follow a similar iterative approach as value iterations and get:

$$Q(s'_t, a'_t) \leftarrow Q(s_t, a_t) + \alpha \left[r(s_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

This is off-policy!

Exploration in Q-Learning

With some $0 \leq \epsilon \leq 1$ probability we choose to either take a random action at any given state or go with the highest $Q(s,a)$ value

$$a = \begin{cases} \text{argmax}_{a \in A} Q(s,a) & \epsilon \\ \text{random action } a \in A & 1-\epsilon \end{cases}$$

SARSA (*State – Action – Reward – State – Action*)

Alternative to Q-Learning, instead of choosing the best possible action we chose the next action according to the policy

$$Q(s_t', a_t') \leftarrow Q(s_t, a_t) + \alpha[r(s_t) + \gamma \sum_a \pi(a|s_{t+1})Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

This is on-policy!