Introduction to Reinforcement Learning: Part 1

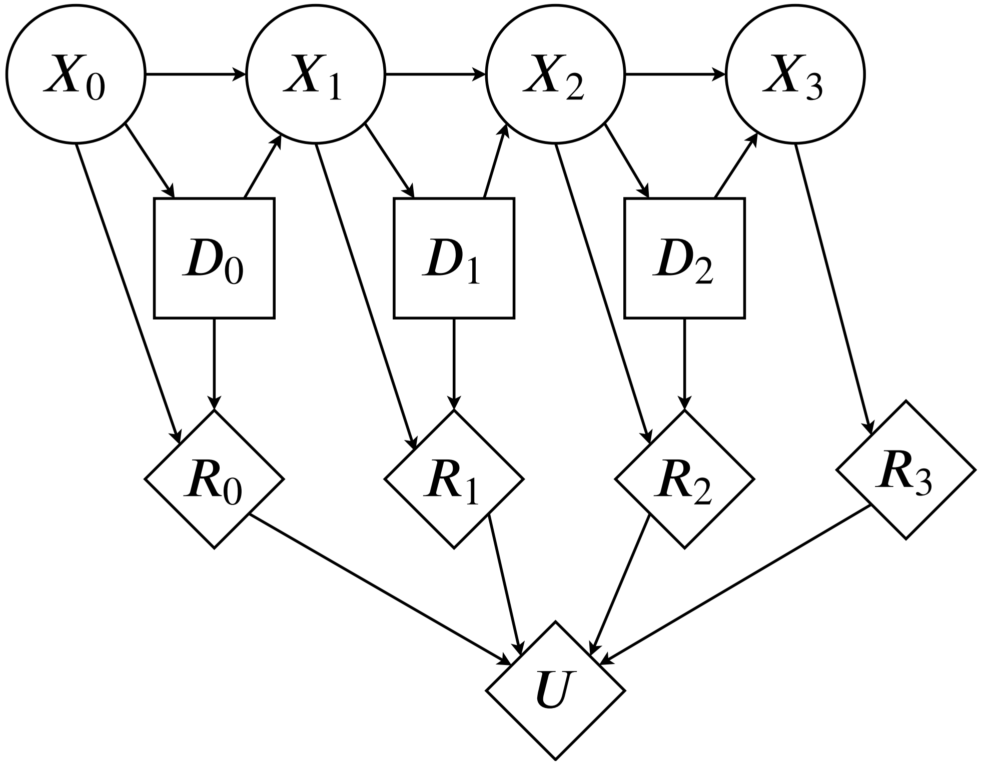
March 4, 2021

Today:

- Agent – Environment interaction
- Markov Decision Process
- Value iteration
- Policy Iteration

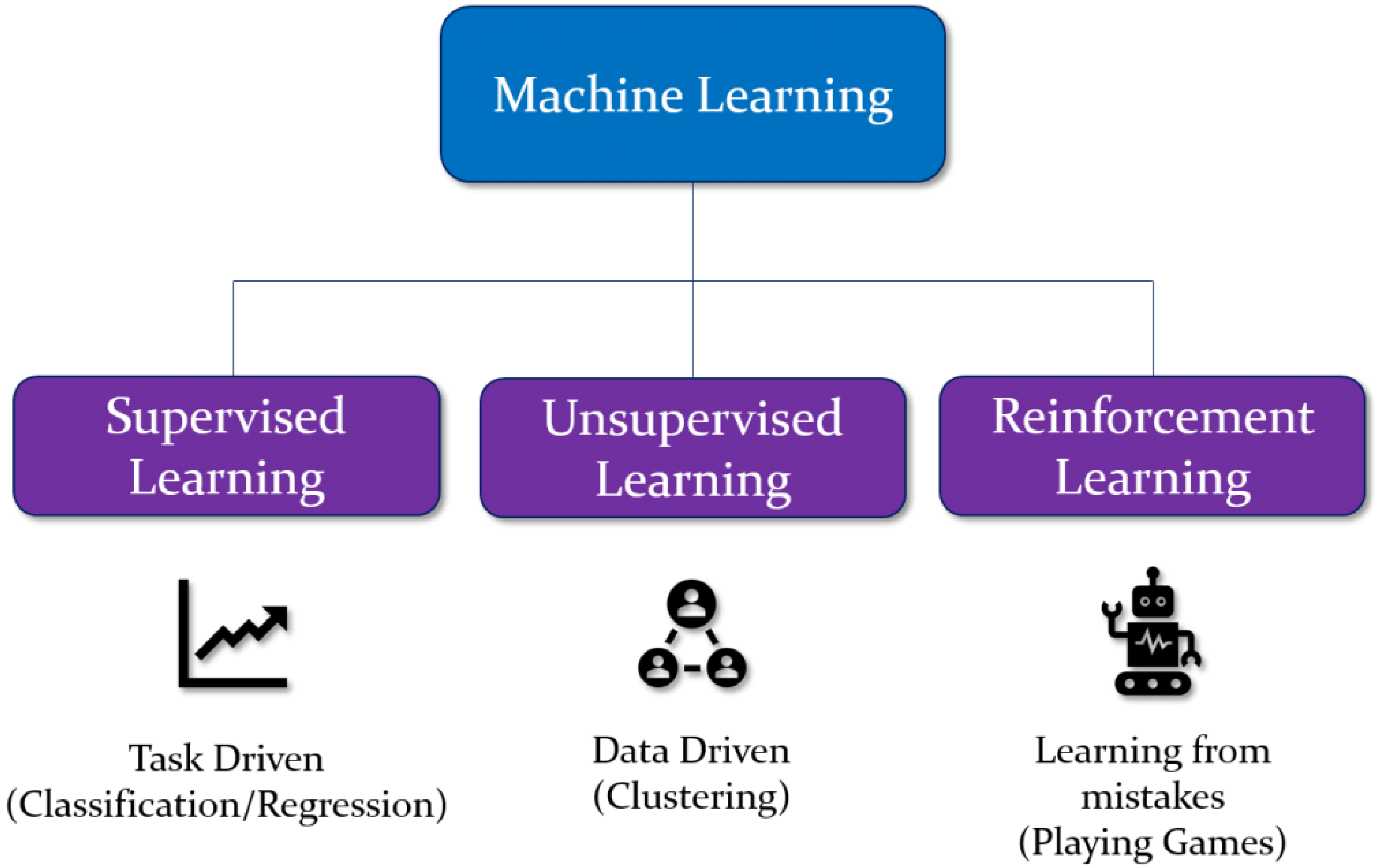
Recap

- Markov Chains (modeling sequential data)
- Hidden Markov Model (incorporate evidence)
    - Inference, e.g. hindsight, monitoring
- Dynamic Bayesian Networks (generalization with state variables)
- Decision Networks/Influence diagrams (incorporate decisions & utility)
- Principle of maximum expected utility (rational agent)
- Utility function (subjectivity)



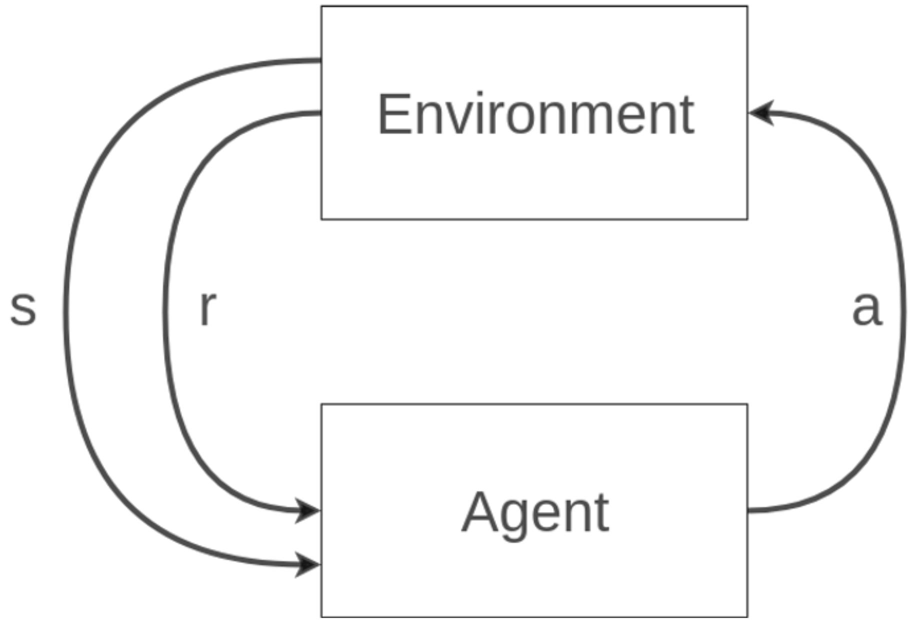
A decision network over time

Types of Machine Learning



source:newtechdojo.com

Agent – Environment interaction in RL



- a: action signal
- r: reward signal
- s: state signal

Before diving into RL let's train an agent

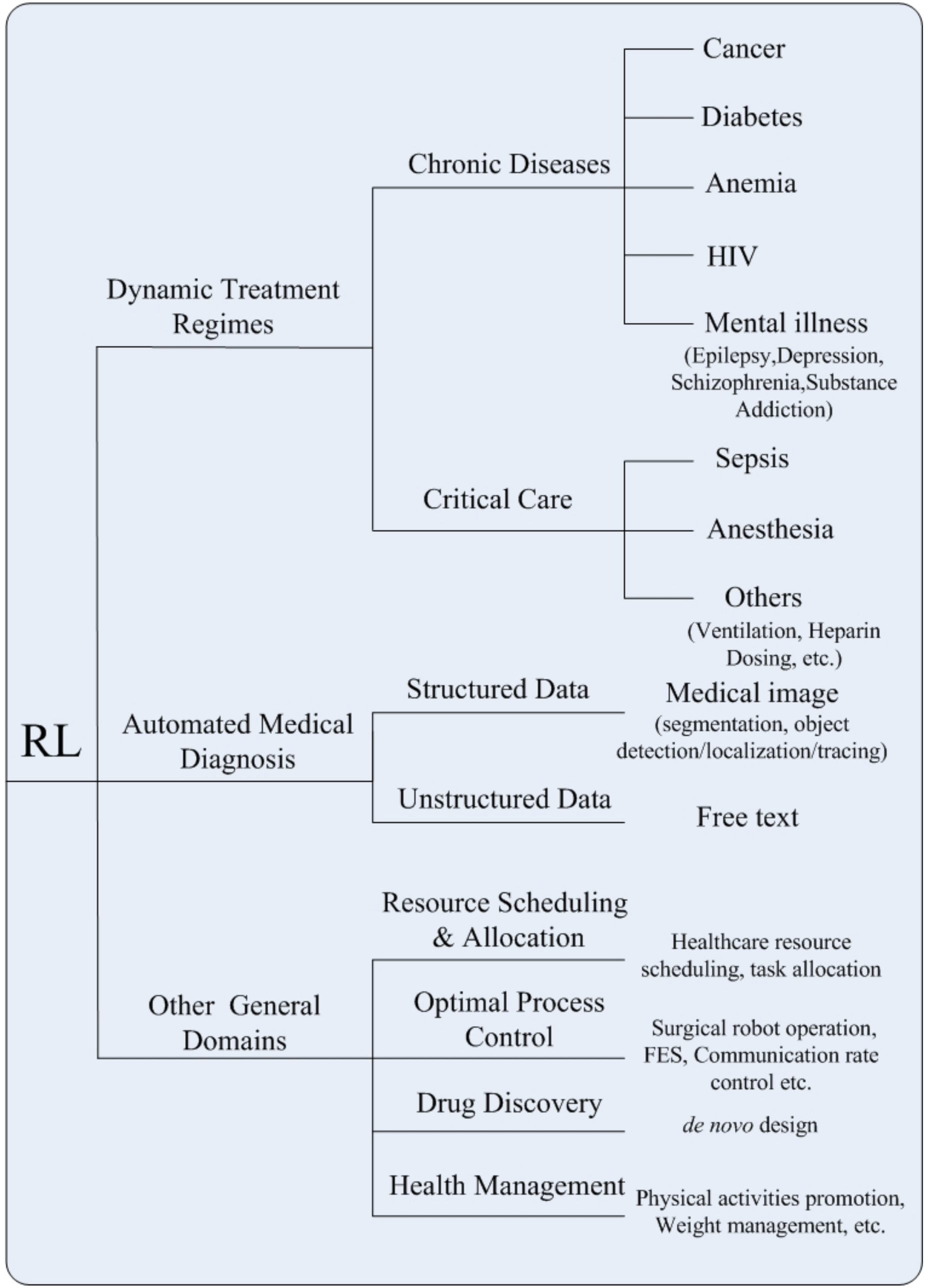
https://cs.stanford.edu/people/karpathy/reinforcejs/waterworld.html

RL examples

- Robotics
- Traffic (e.g. light control)
- Science (e.g., optimizing chemical reactions)
- Games (e.g. Atari, backgammon)
- NLP (e.g. text summarization, question answering, machine translation)
- Healthcare

Source: Chao et al 2019

http://arxiv.org/abs/1908.08796



Learning Atari Breakout

Learning Atari Breakout

States: Screen–shots of the game

Rewards: Points increase

Actions: Game Controls (Left, Right)

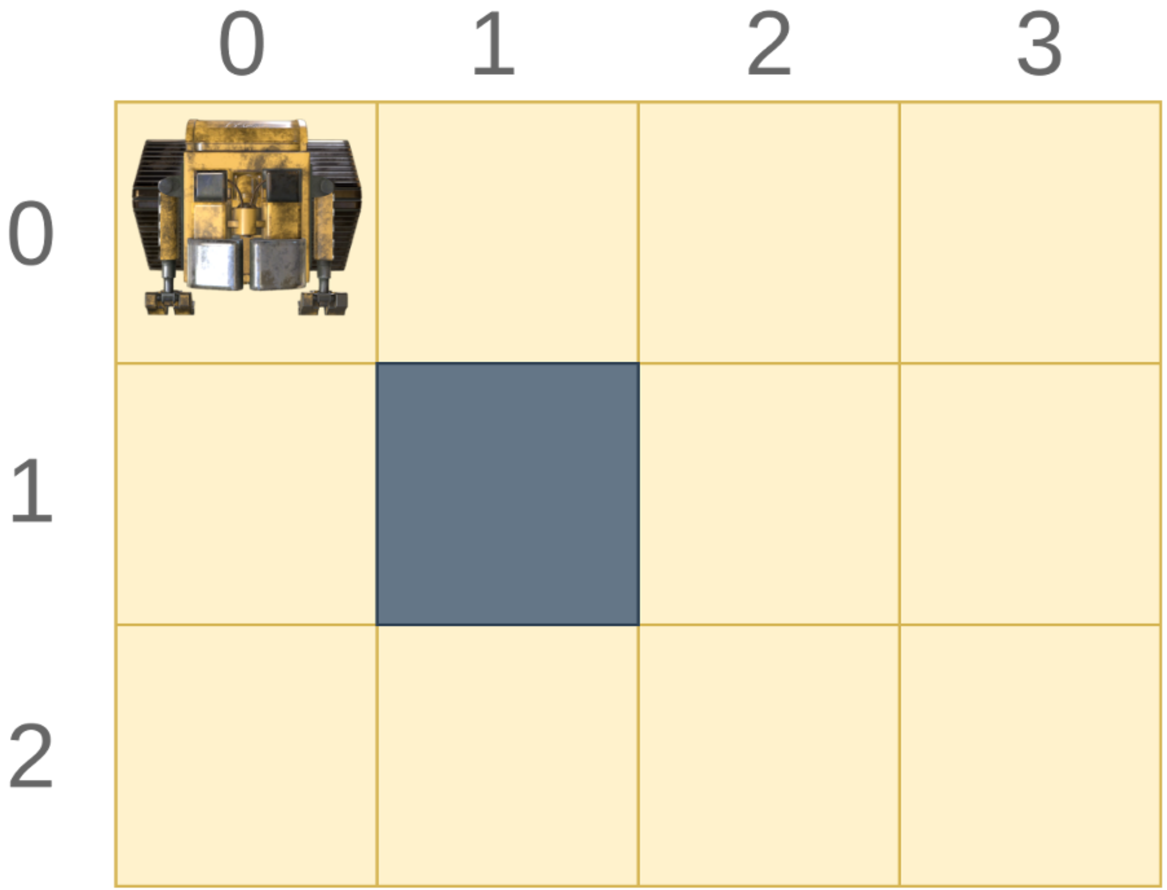
Objective: Maximize the points in a single game

Deepmind's Deep RL – Atari Breakout

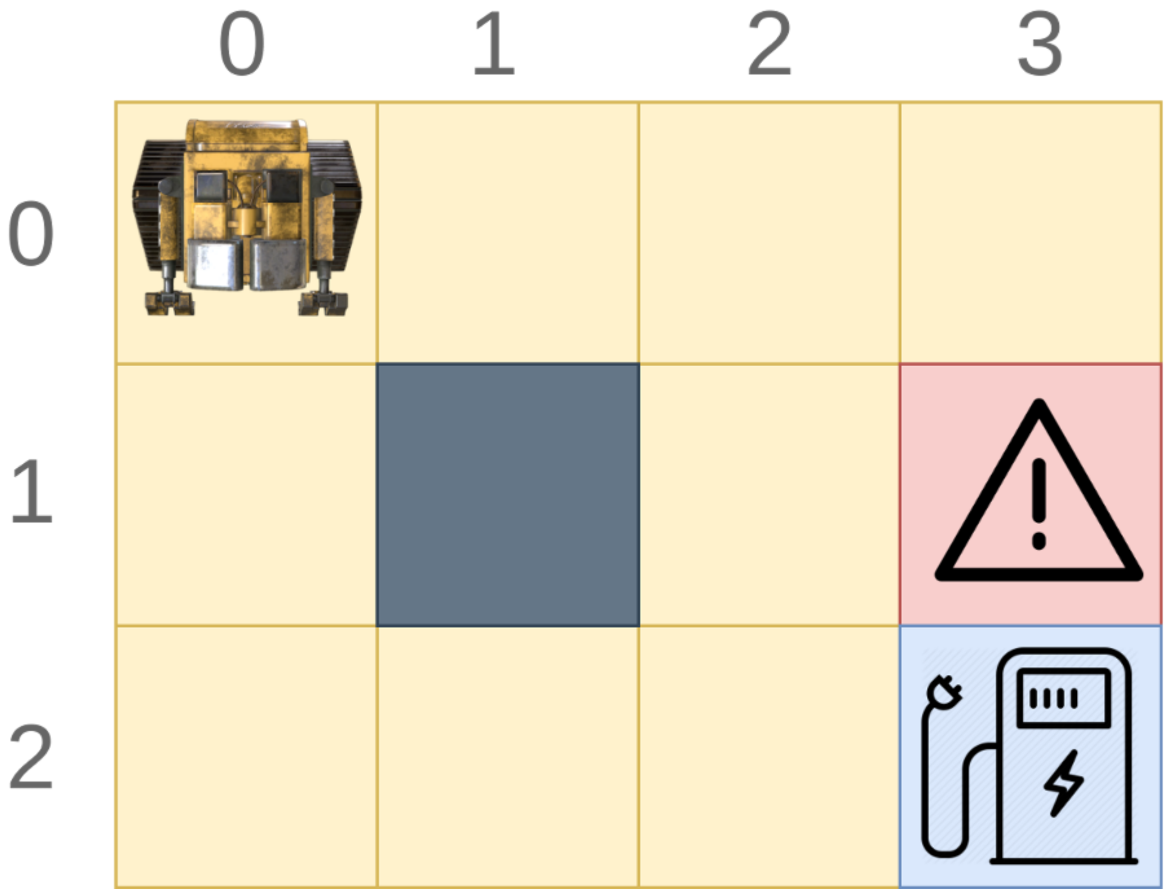
https://www.youtube.com/embed/V1eYniJ0Rnk

Example: Wall–E

4x3 World



4x3 World



States and Rewards

	0	1	2	3
0	 -0.04	-0.04	-0.04	-0.04
1	-0.04		-0.04	-1.0
2	-0.04	-0.04	-0.04	+1.0

States: (0,0), (0,1), (0,2), …, (2,3)

Rewards: Numbers in each box

Actions: Up, Down, Left, Right

Terminal States Red, Blue

Utility

	0	1	2	3
0	 -0.04	-0.04	-0.04	-0.04
1	-0.04		-0.04	-1.0
2	-0.04	-0.04	-0.04	+1.0

Goal of Agent: Maximum Utility

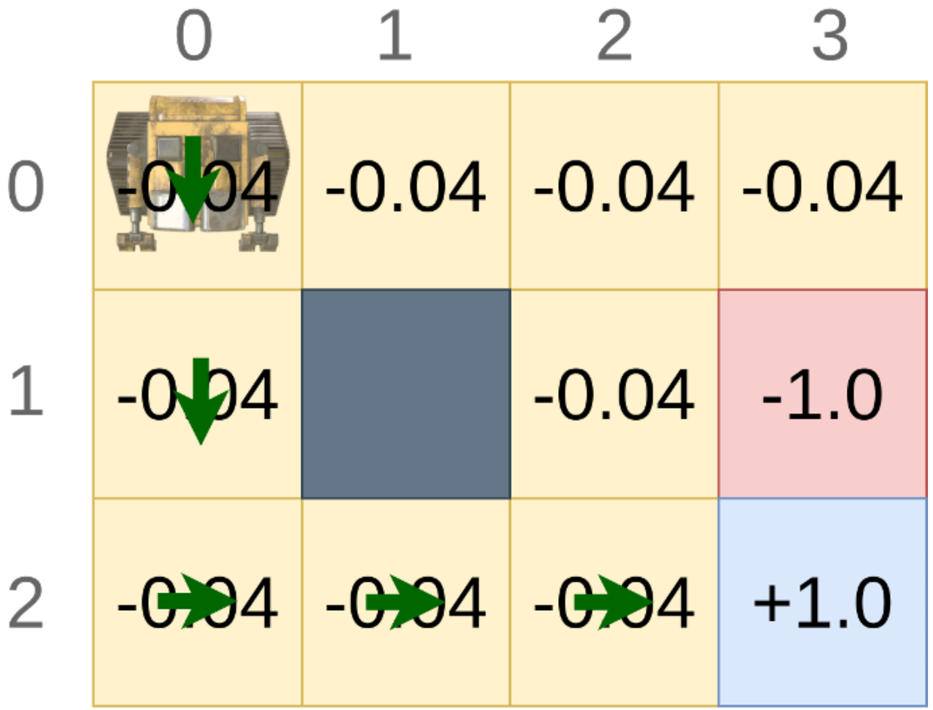
A state sequence

[(0,0), (0, 1), (0,2), (0,3), (1,3)]

$$seq = [s_0, s_1, s_2, s_3, s_4]$$

$$\begin{aligned} Useq = {} & R(s_0) + R(s_1) + R(s_2) \\ & + R(s_3) + R(s_4) \\ = {} & 4 \times (-0.04) - 1.0 = -1.16 \end{aligned}$$

Deterministic Environment

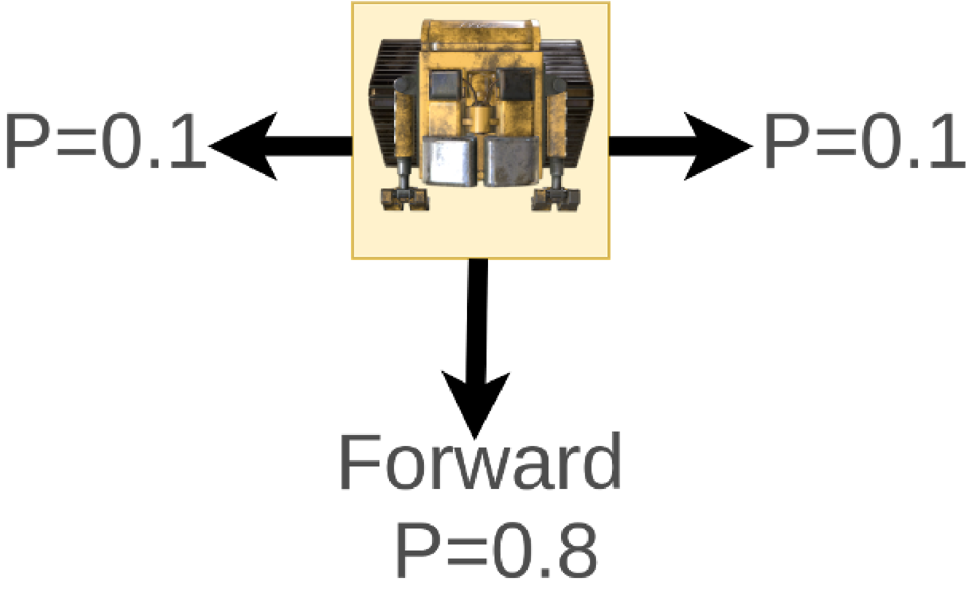


An optimal solution

$$Useq = 5 \times (-0.04) + 1.0 = 0.8$$

Stochastic Environment

	0	1	2	3
0	 -0.04	-0.04	-0.04	-0.04
1	-0.04		-0.04	-1.0
2	-0.04	-0.04	-0.04	+1.0



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

Stochastic Environment

	0	1	2	3
0	 -0.04	-0.04	-0.04	-0.04
1	-0.04		-0.04	-1.0
2	-0.04	-0.04	-0.04	+1.0

Starting state: (0,0)

Action: $a = DOWN$

Potential Outcomes:

- $s' = (1, 0)$ with $P = 0.8$
- $s' = (0, 1)$ with $P = 0.1$
- $s' = (0, 0)$ with $P = 0.1$

Stochastic Environment

- Here transition model: $P(s' | s, a)$
- More generally it could be: $P(s', r | s, a)$
- Even more generally: $P(s', r | s, a, r', s'', a'' \cdots)$
    - E.g. Weather forecast: depends on yesterday & days before

- $P(s', r' | s, a)$ satisfies **Markov property**

"Given the present, the future is conditionally independent of the past"

We have a **Markov Decision Process (MDP)**

Markov Decision Process (MDP)

Finite MDP

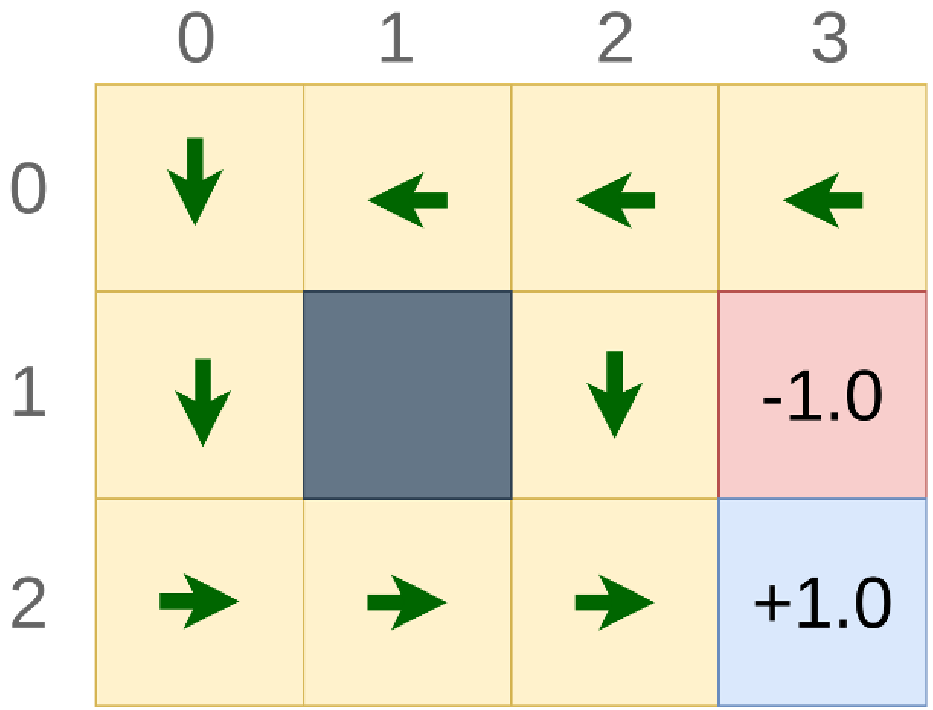
	0	1	2	3
0	 -0.04	-0.04	-0.04	-0.04
1	-0.04		-0.04	-1.0
2	-0.04	-0.04	-0.04	+1.0

Environment consisting of :

- Set of states S
- Set of actions A
- Reward function $R(s)$
- A transition model $P(s', r | s, a)$

How to navigate?

Policy



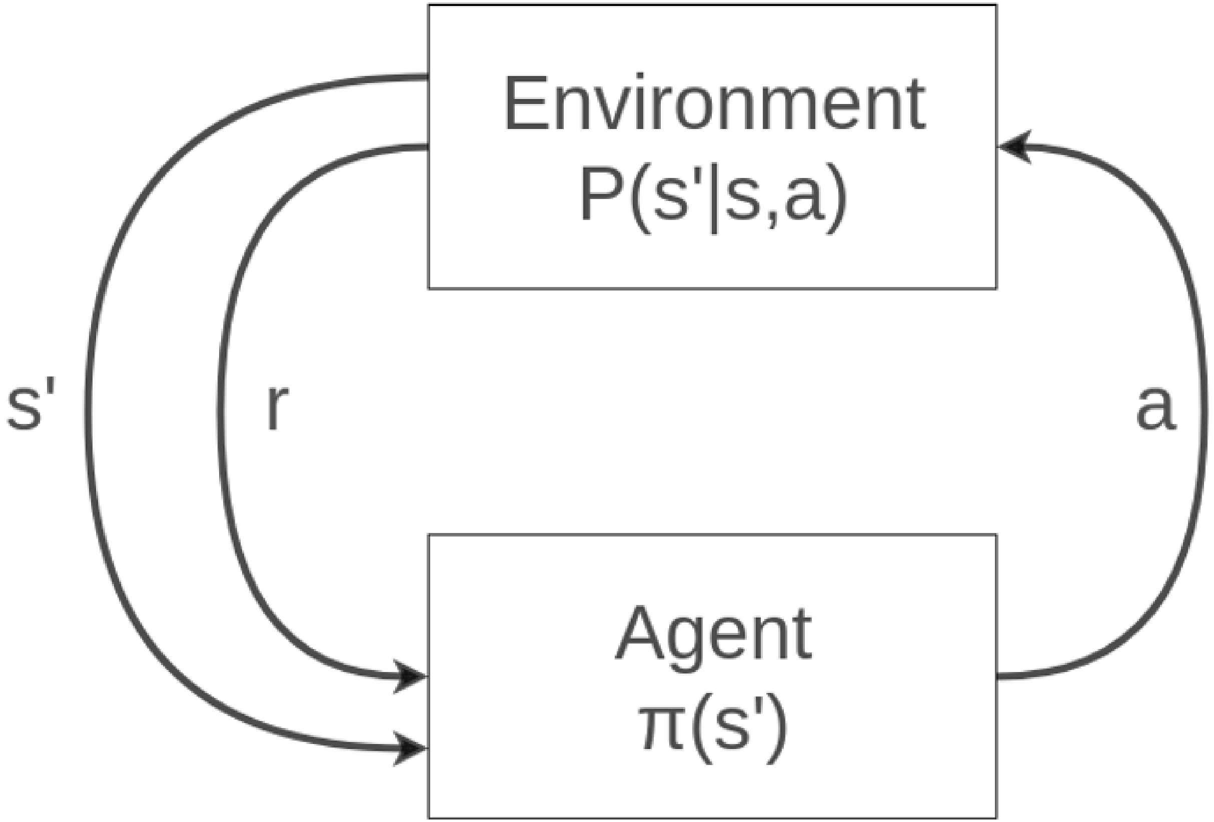
Mathematically

$$\pi(s) = a$$

$$\pi(s = (2, 1)) = \text{RIGHT}$$

$$\pi(s = (1, 2)) = \text{DOWN}$$

Agent – Environment



What policy is best?

Utility $U_{seq} = R(s_0) + R(s_1) + R(s_2) + \cdots + R(s_n)$

Discounted Utility $U_{seq} = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots + \gamma^n R(s_n), \quad \gamma \in [0, 1]$

To compare policies we define expected utility for each state

$$U^\pi(s) = E\left(R(s) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots + \gamma^n R(s_n)\right)$$

First state: $R(s)$

Optimal policy: A policy that maximizes $U^\pi(s)$

Optimal Policy

Utilities

	0	1	2	3
0	0.705	0.655	0.611	0.388
1	0.762		0.660	-1.0
2	0.812	0.868	0.918	+1.0

Optimal Policy

	0	1	2	3
0	↓	←	←	←
1	↓		↓	-1.0
2	→	→	→	+1.0

How to get the magic values?

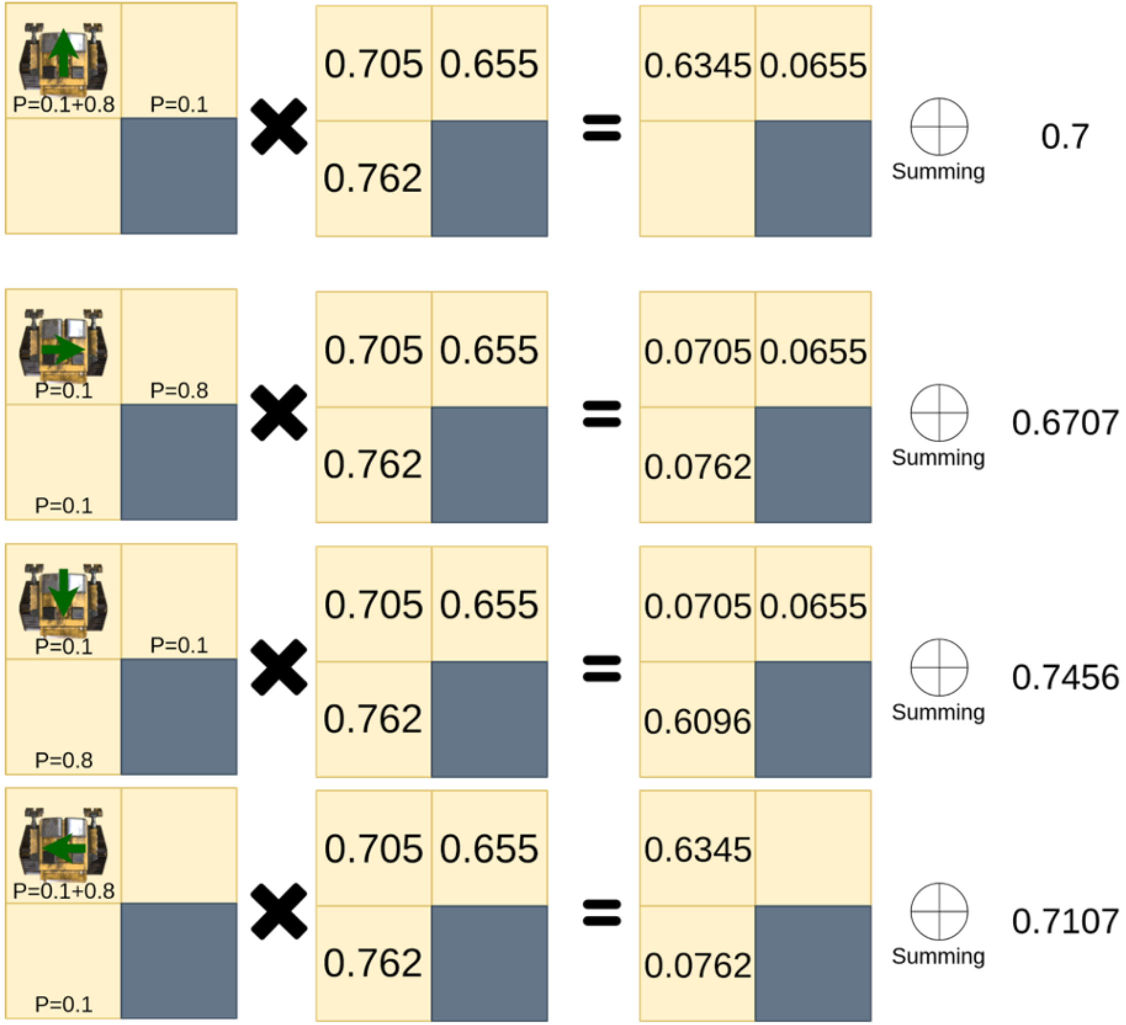
Are they related?

YES.

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s' | s, a) U(s')$$

Bellman Equations

Applying Bellman Equations



How to get the magic values?

Are they related?

YES.

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s' | s, a) U(s')$$

Bellman Equtations

$$U(s = (0, 0)) = -0.04 + 1.0 \times 0.7456 \sim 0.705$$

Bellman Equations: System of equations, but not linear

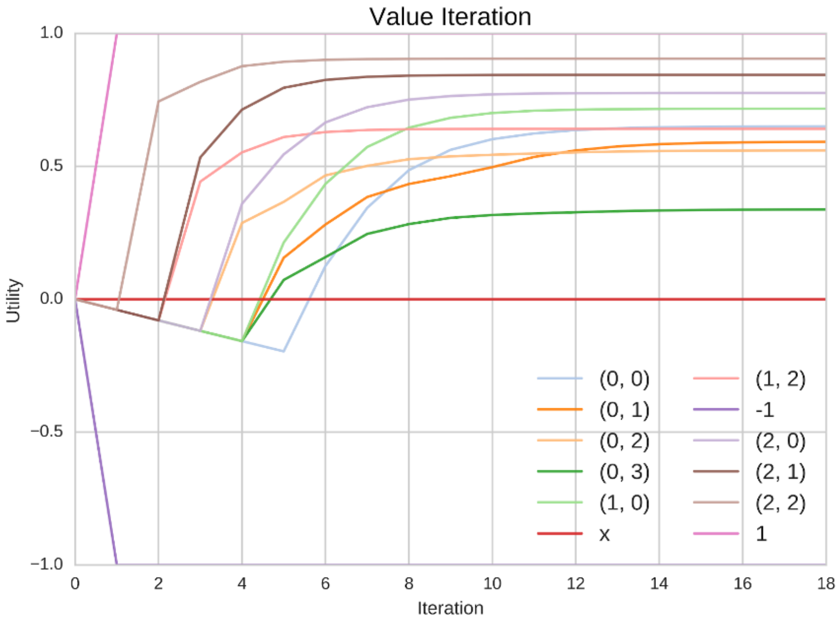
Value iteration algorithm

Iterative way to get utility Steps:

1. Initialize $U(s)$
2. For all states apply Bellman Update

   $$U(s) \leftarrow R(s) + \gamma \max_{a} \sum_{s'} P(s' \mid s, a) U(s')$$

3. Repeat step 2 until U(s) converges



Value iteration: Demo

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

How to get the policy?

- We have only the utilities for the optimal policy
- Can we get the utilities of other policies?

YES.

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' | s, \pi(s)) U^{\pi}(s')$$

Simplified Bellman Equations

Policy evaluation algorithm

Iterative way to get utility Steps:

1. Initialize $U(s)$
2. For all states apply bellman update

$$U^{\pi}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s' | s, \pi(s)) U^{\pi}(s')$$

3. Repeat step 2 until $U(s)$ converges

Policy iteration algorithm

Iterative way to get optimal policy Steps:

1. Initialize $U(s)$, and $\pi(s)$ randomly
2. Perform policy evaluation to update $U(s)$
3. Update policy $\pi(s) \leftarrow \operatorname*{argmax}_a \sum_{s'} P(s' | s, a) U^{\pi}(s')$
4. Repeat steps 2 and 3 until policy doesn't change

Policy iteration example

Initialized Random Policy

	0	1	2	3
0	→	→	↓	↓
1	↑		↑	-1.0
2	↑	↑	↓	+1.0

	0	1	2	3
0	→	→	→	↓
1	↑		↓	-1.0
2	→	→	→	+1.0

	0	1	2	3
0	→	→	↓	←
1	↓		↓	-1.0
2	→	→	→	+1.0

	0	1	2	3
0	-1.58	-1.52	-1.48	-1.09
1	-1.63		-1.47	-1.0
2	-1.62	-1.15	-0.27	+1.0

	0	1	2	3
0	-0.99	-0.93	-0.89	-1.03
1	-1.04		0.660	-1.0
2	0.610	0.868	0.918	+1.0

	0	1	2	3
0	0.508	0.526	0.576	0.357
1	0.761		0.660	-1.0
2	0.811	0.868	0.918	+1.0

Policy iteration example

Reached Optimal Policy

	0	1	2	3
0	↓	→	↓	←
1	↓		↓	-1.0
2	→	→	→	+1.0

	0	1	2	3
0	↓	←	↓	←
1	↓		↓	-1.0
2	→	→	→	+1.0

	0	1	2	3
0	↓	←	←	←
1	↓		↓	-1.0
2	→	→	→	+1.0

	0	1	2	3
0	0.690	0.526	0.576	0.357
1	0.761		0.660	-1.0
2	0.811	0.868	0.918	+1.0

	0	1	2	3
0	0.704	0.655	0.590	0.369
1	0.761		0.660	-1.0
2	0.811	0.868	0.918	+1.0

	0	1	2	3
0	0.704	0.655	0.610	0.387
1	0.761		0.660	-1.0
2	0.811	0.868	0.918	+1.0

References

- https://mpatacchiola.github.io/blog/2016/12/09/dissecting-reinforcement-learning.html
- Artificial Intelligence: A Modern Approach, Russell and Norvig
- Reinforcement Learning: An Introduction, Sutton and Barto
- https://github.com/cmarasinou/WALLE-RL
  - Wall-E grid-world implementation
- Demos: https://cs.stanford.edu/people/karpathy/reinforcejs/