REINFORCEMENT LEARNING

A playful machine learning

Avneet Kaur
PhD Applied Mathematics
Email: a93kaur@uwaterloo.ca

GUESS THE ANIMAL

QUOLL

WHAT IS MACHINE LEARNING?

- machines learn to do a given task without being explicitly programmed.

Supervised learning	Unsupervised learning	Reinforcement learning
• Labelled dataset • Learn f to map y=f(x) • Classification, Regression	• Unlabelled dataset • Learn underlying structure • Clustering, Dimensionality reduction	• Generate dataset • Maximize utility by learning to interact • Robot navigation, learning games

TRANSLATE THESE WORDS

- ਕੰਨ (Punjabi)

- Nez (French)

KEY TAKEAWAYS

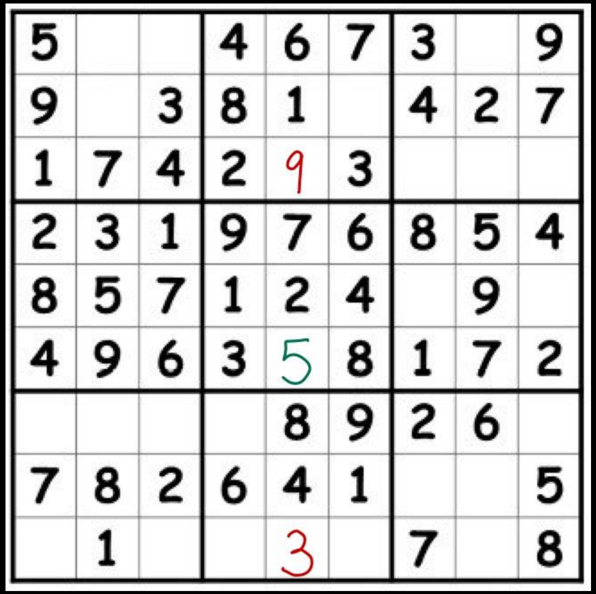
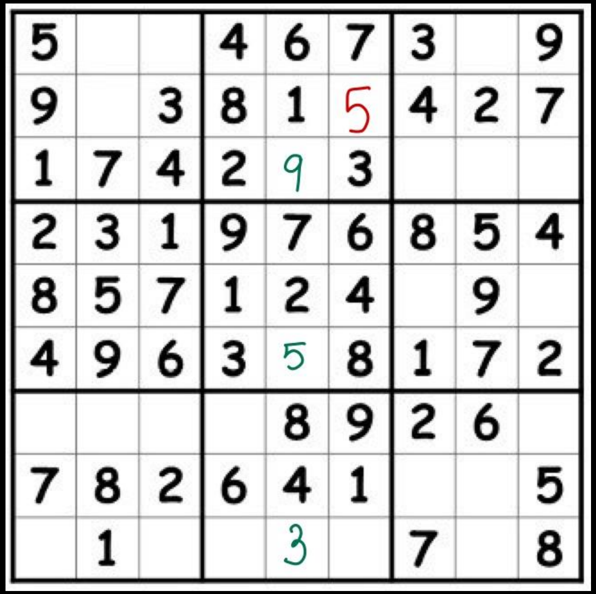
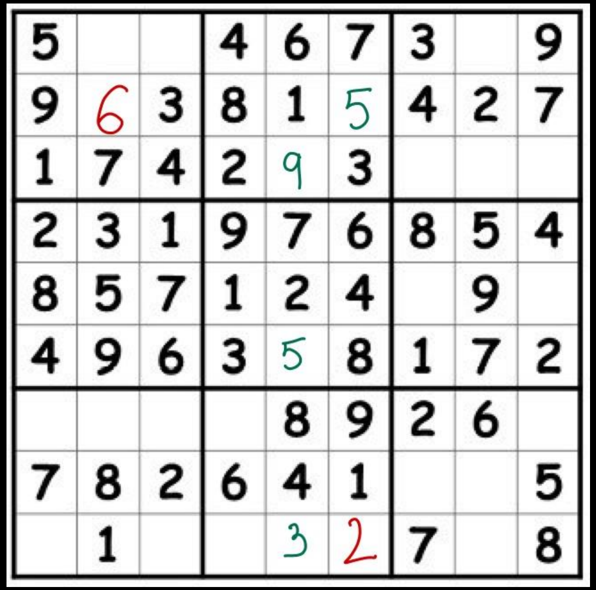
- You were rewarded for each type of answer.
- You as an agent interacted with the environment to translate better.
- Environment gave feedback in the form of rewards.

SUDOKU

5			4	6	7	3		9
9		3	8	1		4	2	7
1	7	4	2		3			
2	3	1	9	7	6	8	5	4
8	5	7	1	2	4		9	
4	9	6	3		8	1	7	2
				8	9	2	6	
7	8	2	6	4	1			5
	1					7		8

Task: Fill the missing squares in as less time as possible.

- Agent makes a sequence of moves(actions)
- Each move by the agent decides which subsequent squares can be filled next

- Reaching the goal state will have a reward
- Intermediate squares may or may not have reward

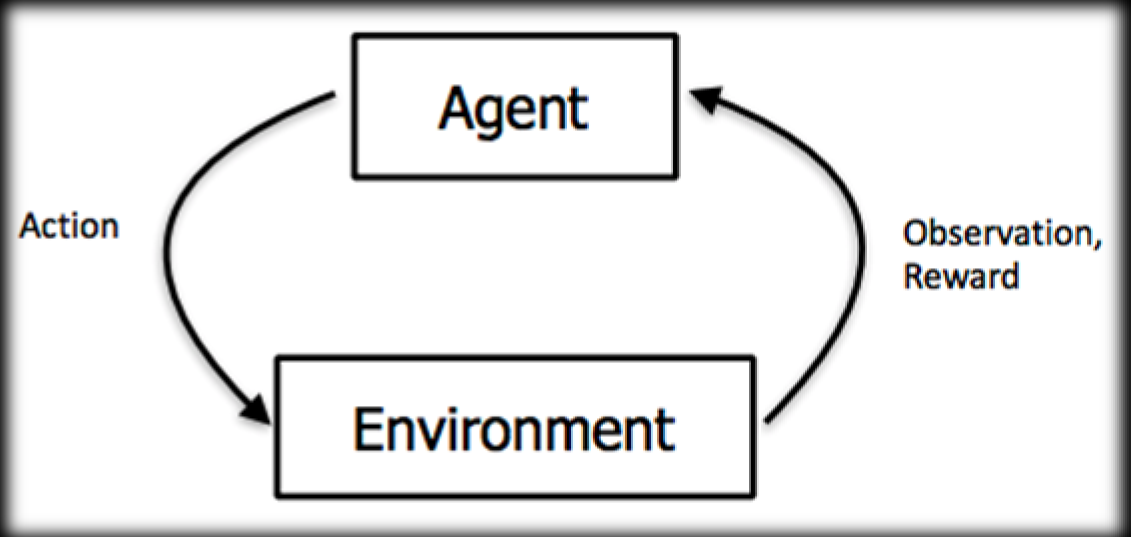
5			4	6	7	3		9
9	6	3	8	1	5	4	2	7
1	7	4	2	9	3			
2	3	1	9	7	6	8	5	4
8	5	7	1	2	4		9	
4	9	6	3	5	8	1	7	2
			7	8	9	2	6	
7	8	2	6	4	1			5
	1			3	2	7		8

An intermediate state

5	2	8	4	6	7	3	1	9
9	6	3	8	1	5	4	2	7
1	7	4	2	9	3	5	8	6
2	3	1	9	7	6	8	5	4
8	5	7	1	2	4	6	9	3
4	9	6	3	5	8	1	7	2
3	4	5	7	8	9	2	6	1
7	8	2	6	4	1	9	3	5
6	1	9	5	3	2	7	4	8

Goal state

RL FRAMEWORK



INVENTORY CONTROL EXAMPLE

- **Observation:** Stock level
- **Action:** What to purchase
- **Reward:** Profit

ENVIRONMENT

- An external system that an agent can perceive and act on
- Receives action from agent and in response emits appropriate reward and (next) observation

AGENT

- A system that takes actions to change the state of the environment (Decision maker)
- Executes action upon receiving observation
- For taking an action the agent receives an appropriate reward

STATE

- State can be viewed as a summary or an abstraction of the history of the system
- For example, in Sudoku, the state could be raw image or vector representation of the board

REWARD

- Reward is a scalar feedback signal
- Indicates how well agent acted at a certain time
- The agent’s aim is to maximise cumulative reward

COMPONENTS OF AN RL AGENT

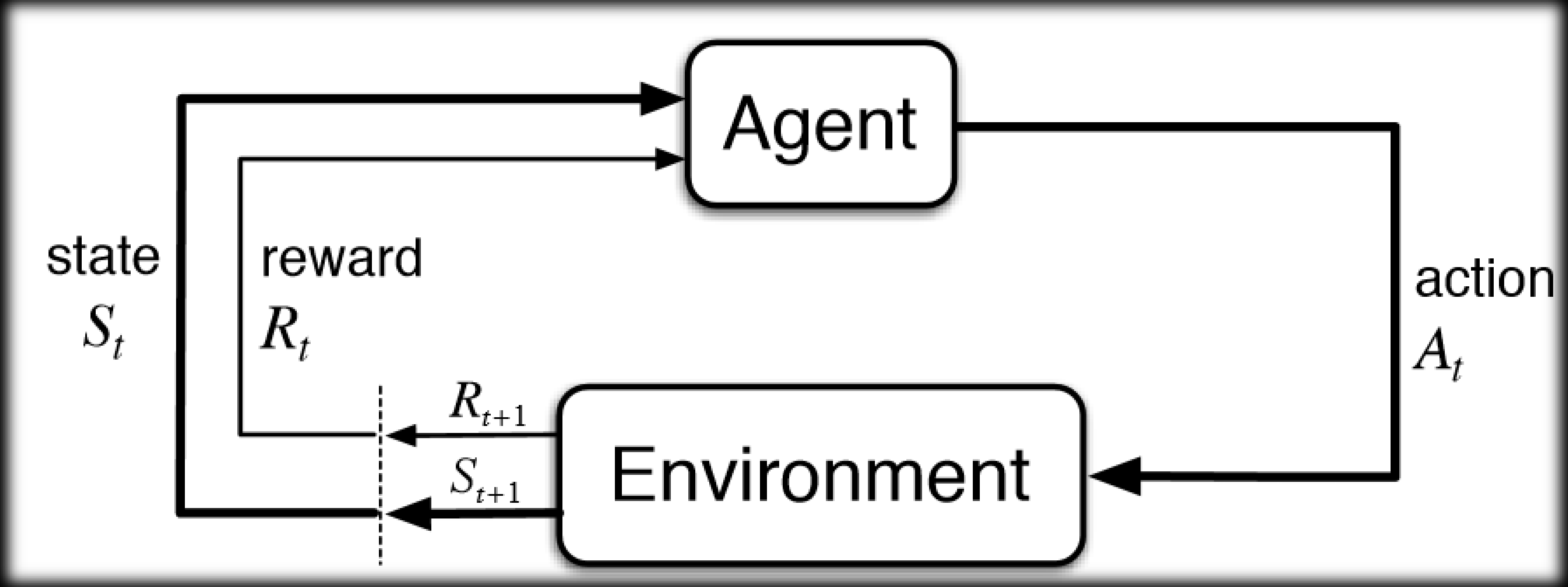
- **Policy:** agent's behaviour function; $\pi: S \rightarrow A$
- **Value function:** evaluates how good is each state and/or action. Therefore, it is used to choose appropriate action among the available options.
- **Model:** agent's representation of the environment; Mainly contains state transition information and reward function.

TIC TAC TOE

- **Observation:** Board position
- **Action:** Moves
- **Reward:** Win or loss
- **Policy:** Agent has multiple empty squares to choose
  - Random policy is to place 'X' in any one of empty squares randomly
  - Better policy is to place 'X' in square 5
- **Value Function:** Agent may have an estimate about the value of being in a certain board configuration
- **Model:** Model of transition probabilities between states

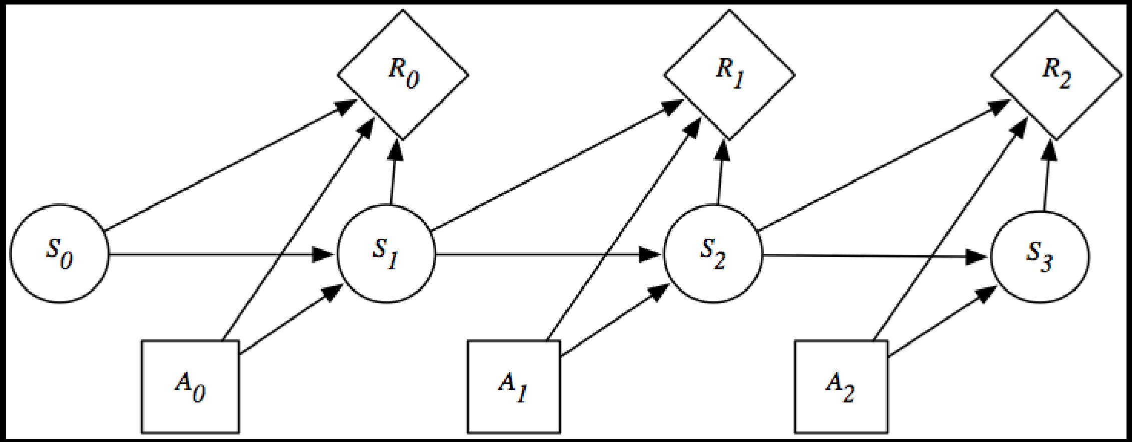
X_1	O_2	3
X_4	5	6
O_7	O_8	X_9

FRAMEWORK



MARKOV DECISION PROCESS

- Provides a mathematical framework for modelling decision making process
- Can formally describe the working of the environment and the agent
- Core problem in solving an MDP is to find an ’optimal’ policy (or behaviour) for the decision maker (agent) to maximize the total future reward



RANDOM VARIABLE

- A random variable X denotes the outcome of a random phenomenon
- Examples include outcome of a coin toss and the roll of a dice.

STOCHASTIC PROCESS

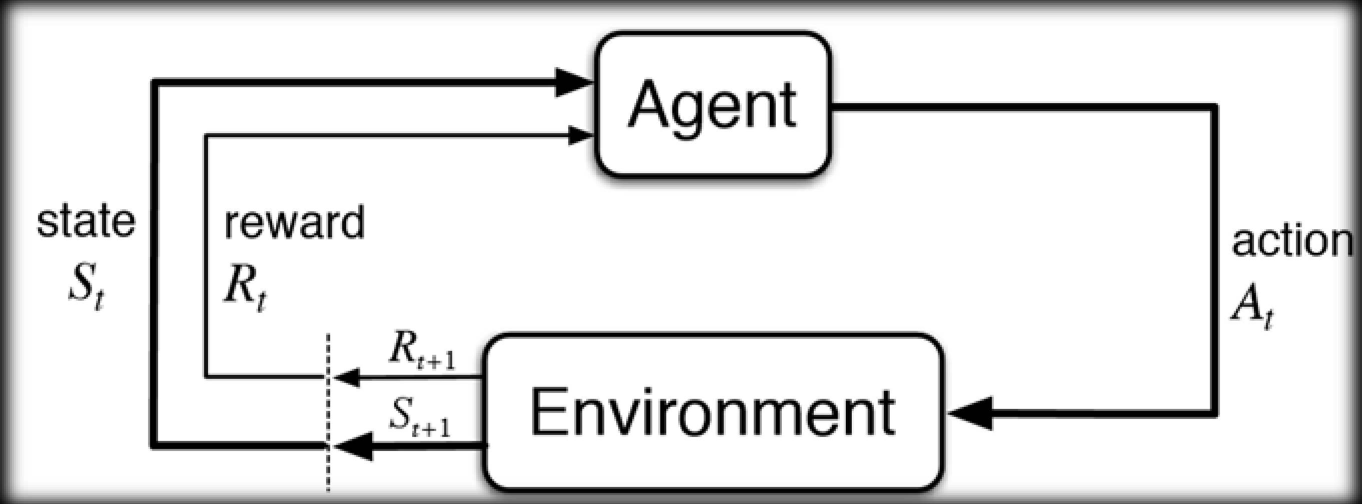
- It is a collection of random variables indexed by some mathematical set T.
- T has the interpretation of time and is typically, $\mathbb{N}$ or $\mathbb{R}$. Assume T=$\mathbb{N}$ for our sessions.
- Notation: $\{X_t\}_{t \in T}$

MARKOV PROPERTY

- A stochastic process $\{S_t\}_{t\in T}$ is said to have Markov property if for any state s_t,

$$P(S_{t+1} \mid S_t)=P(S_{t+1} \mid S_1,S_2,\ldots,S_t).$$

- S_t captures all relevant information from history and is a sufficient statistic of the future.
- Memoryless property



STATE TRANSITION PROBABILITY

- For a stochastic process $\{S_t\}_{t \in T}$, the state transition probability for successive states s and s′ is denoted by

$$\mathcal{P}_{SS'} = P(S_{t+1} = S' \mid S_t = S).$$

- State transition matrix $\mathcal{P}$ then denotes the transition probabilities from all states s to all successor states s′ (with each row summing to 1.

$$\mathcal{P} = \begin{pmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \cdots & \mathcal{P}_{1n} \\ . & & & \\ . & & & \\ . & & & \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \cdots & \mathcal{P}_{nn} \end{pmatrix}$$

MARKOV CHAIN

- A stochastic process $\{s_t\}_{t \in T}$ is a Markov Chain if it satisfies Markov property.
- It is represented by the tuple $<\mathcal{S}, \mathcal{P}>$ where $\mathcal{S}$ denotes the set of states.
- It is also called Markov process.

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \ldots, S_t]$$

MARKOV REWARD PROCESS

A Markov reward process is a tuple $<\mathcal{S},\mathcal{P},\mathcal{R},\gamma>$ is a Markov chain with values

- $\mathcal{S}$: Finite set of states
- $\mathcal{P}$: State transition probability
- $\mathcal{R}$: Reward for being in state s_t is given by a deterministic function $\mathcal{R}$

$$r_{t+1}=\mathcal{R}(s_t)$$

- γ: Discount factor such that $\gamma \in [0,1]$

WHY DISCOUNTING?

- Offers trade off between 'myopic' and 'far sighted' rewards
- Avoids infinite returns in cyclic and infinite horizon Markov processes
- Undiscounted Markov reward process are mostly used when sequences terminate.

TOTAL DISCOUNTED REWARD

Total discounted reward from time step t is, $\sum_{k=0}^{\infty}(\gamma^k r_{t+k+1})$

- $\gamma \rightarrow 0$ (myopic); $\gamma \rightarrow 1$ (far-sighted)
- Value of reward r after k+1 timesteps is γ^kr.

STATE-VALUE FUNCTION

Value function V(s) denotes the long-term value of state s,

$$V(s) = \mathbb{E}(G_t | s_t = s) = \mathbb{E}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s)$$

and is independent of time, t.

RECURSIVE FORMULATION OF VALUE FUNCTION

$$V(s) = \mathbb{E}(G_t | s_t = s) = \mathbb{E}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s)$$

$$= \mathbb{E}(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s)$$

$$= \mathbb{E}(r_{t+1} | s_t = s) + \gamma \mathbb{E}(G_{t+1} | s_t = s)$$

$$= \mathbb{E}(r_{t+1} | s_t = s) + \gamma \mathbb{E}(\mathbb{E}(G_{t+1} | s_t = s) | s_t = s)$$

$$= \mathbb{E}(r_{t+1} | s_t = s) + \gamma \mathbb{E}(V(s_{t+1}) | s_t = s)$$

$$= \mathbb{E}(r_{t+1} + \gamma V(s_{t+1}) | s_t = s)$$

BELLMAN EQUATION FOR MRP

- For $s' \in \mathfrak{S}$, a successor state of s with transition probability $\mathcal{P}_{ss'}$, we can rewrite $V(s)$ as

$$V(s) = \mathbb{E}(r_{t+1}) + \gamma \sum_{s' \in \mathfrak{S}} \mathcal{P}_{ss'} \, \mathrm{V}(\mathrm{s}').$$

- This is the Bellman equation for value functions

MATRIX FORM

- Let $\mathfrak{S}$={1,2,…n} and $\mathcal{P}$ be known. Then,

$$V = \mathcal{R} + \gamma \mathcal{P} V$$

where

$$\begin{bmatrix} V(1) \\ V(2) \\ . \\ . \\ . \\ V(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}(1) \\ \mathcal{R}(2) \\ . \\ . \\ . \\ \mathcal{R}(n) \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \cdots & \mathcal{P}_{1n} \\ \mathcal{P}_{21} & \cdots & \mathcal{P}_{2n} \\ \vdots & \ddots & \vdots \\ \mathcal{P}_{n1} & \cdots & \mathcal{P}_{nn} \end{bmatrix} \times \begin{bmatrix} V(1) \\ V(2) \\ . \\ . \\ . \\ V(n) \end{bmatrix}.$$

Solving for V, we get,

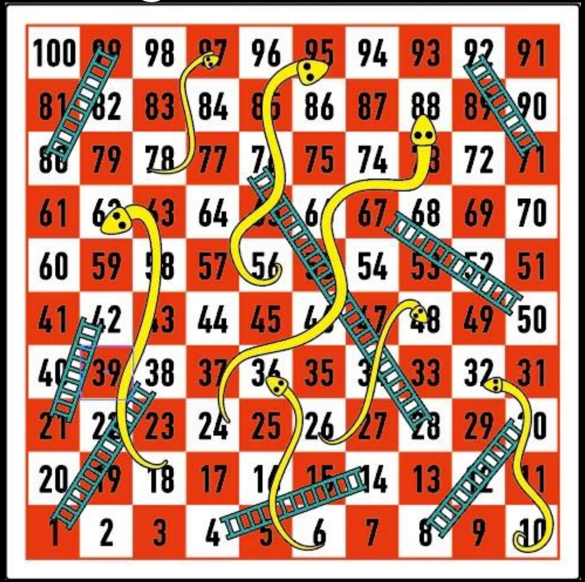
$$V = (I - \gamma \mathcal{P})^{-1} \mathcal{R}.$$

ABSORBING STATE

- A state $i \in \mathfrak{S}$ is said to be absorbing if it is impossible to leave that state, Mathematically,

$$P_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

- In a game of snake and ladders, the state '100' is an absorbing state.



MARKOV DECISION PROCESS

MDP is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ where

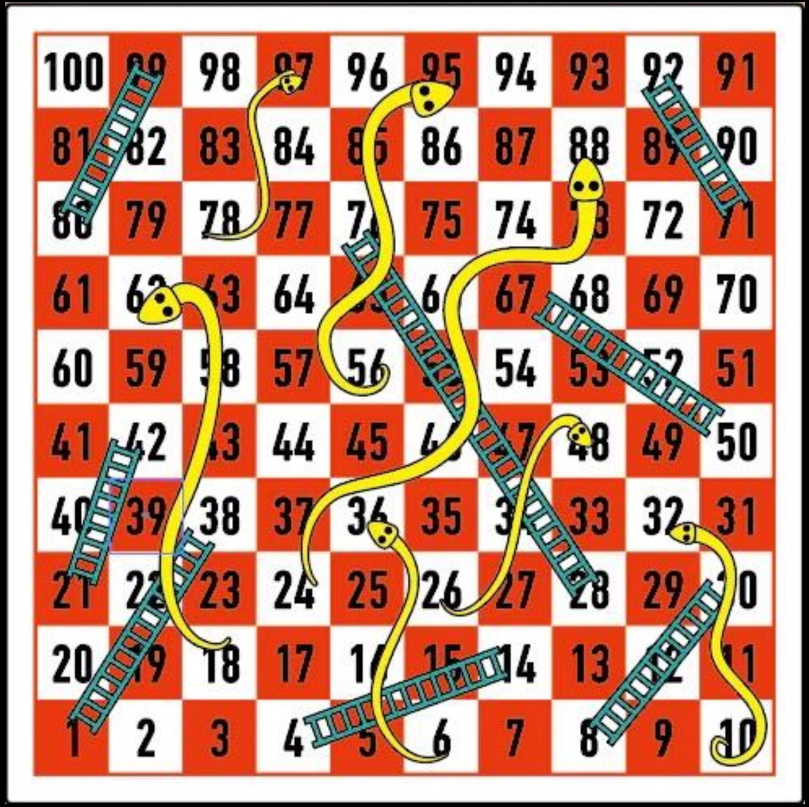
- $\mathcal{S}$: Finite set of states
- $\mathcal{A}$: Finite set of actions
- $\mathcal{P}$: State transition probability

$$\mathcal{P}^a_{ss'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a), a_t \in A$$

- $\mathcal{R}$: Reward for taking action a_t at state s_t and transitioning to state s_{t+1} is given by the deterministic function $\mathcal{R}$
- $$r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1}).$$
- γ: Discount factor such that $\gamma \in [0,1]$

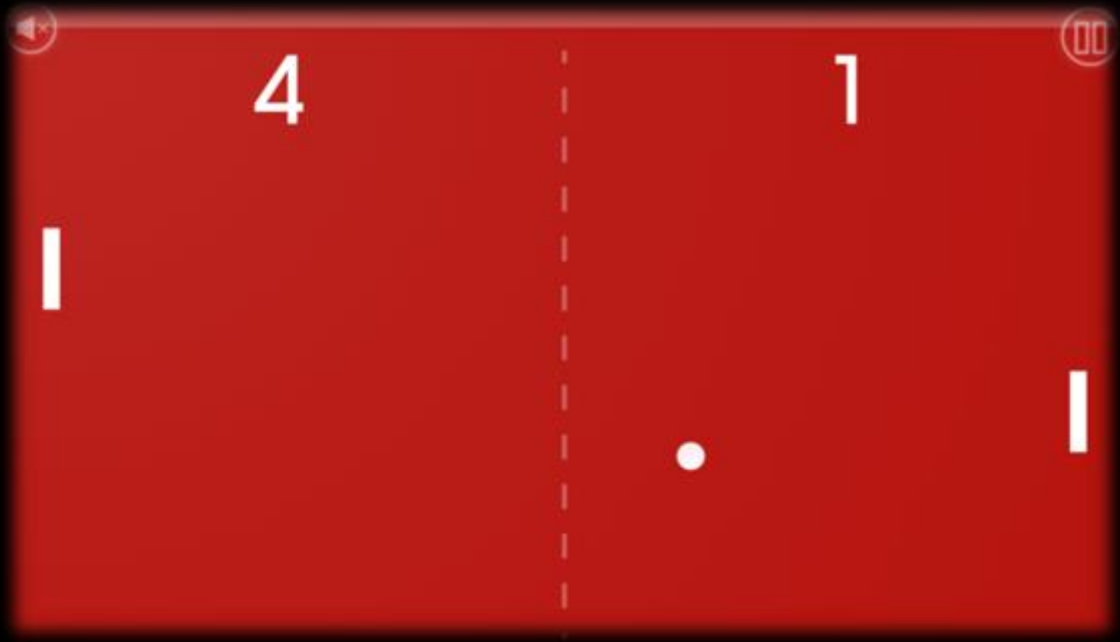
SNAKE AND LADDERS

States: Each square from 1 to 100
Actions: Move right, climb ladders, or come down snakes depending on the number on the die throw
Rewards: -1 for every move made until reaching ‘100’



ATARI-PONG GAME

States: Possible set of all images
Actions: Paddle up or down
Rewards:
+1 for making the opponent miss the ball,
-1 if the agent misses the ball,
0 otherwise.



POLICY

- Let π denote a policy that maps state space $\mathcal{S}$ to action space $\mathcal{A}$.

There are 2 types of policies:

- Deterministic policy: $a=\pi(s)$, $s\in\mathcal{S}$, $a\in\mathcal{A}$.
- Stochastic policy: $\pi(a|s)=P[a_t=a|s_t=s]$

TIC TAC TOE REVISITED

- **Deterministic Policy:** Place 'X' in square 5
- **Stochastic policy:** Place 'X' in square 5 with probability 0.8 and place 'X' in square 6 with probability 0.2

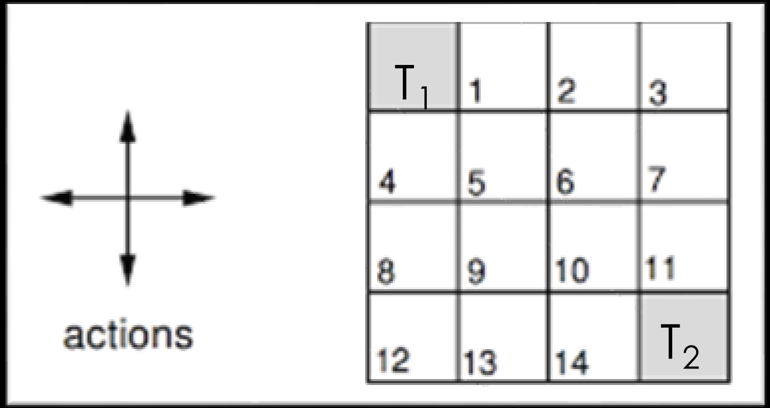
X_1	O_2	3
X_4	5	6
O_7	O_8	X_9

NAVIGATION GRID

- States: {1,2,…,14, T_1, T_2}
- Actions: {right, left, up, down}
- Deterministic Policy:

$$\pi(s) = \begin{cases} down, & s = \{3,7,11\} \\ right, & otherwise \end{cases}$$

- Example sequences: {{12,13,14,T_2}, {4,5,6,7,11,T_2}}
- Stochastic Policy: $\pi(a|s)$ could be a uniform random action between all possible actions at state s
- Example sequences: {{4,5,9,8,12,…},{1,2,6,5,1,2,3,…}}



TIME FOR PROBLEM SETS