Introduction to Reinforcement Learning

Jay Gupta
NTUOSS TGIFHacks #114

Agenda

Introduction

RL Examples and Applications

RL Agents

Markov Decision Process

RL Algorithms

Code - Game 1: Pole Balancing

Code - Game 2: Mountain Car

Beyond this workshop

Overview

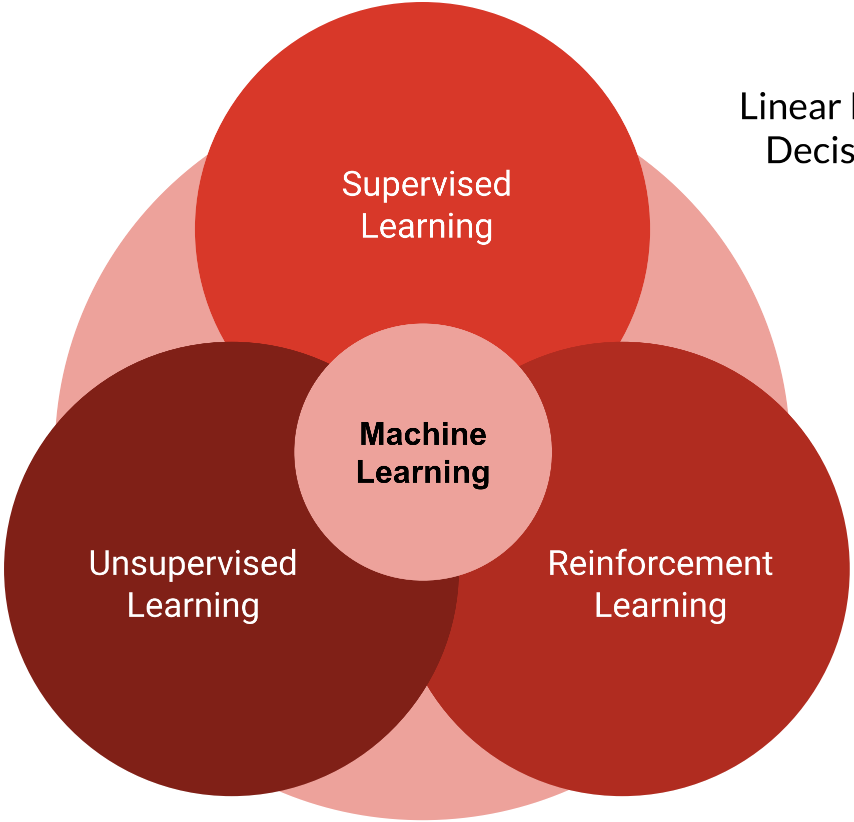
1. Reinforcement learning is the training of machine learning models to make a **sequence of decisions.**

2. The agent learns to achieve a goal in a potentially complex environment. In RL, an artificial intelligence faces a game-like situation.

3. The computer employs trial and error to come up with a solution to the problem.

4. To train the machine, the algorithm gets either **rewards or penalties** for the actions it performs. Its goal is to **maximize the total reward**.

K-Means Clustering
Anomaly Detection



Linear Regression
Decision Trees

Key Characteristics

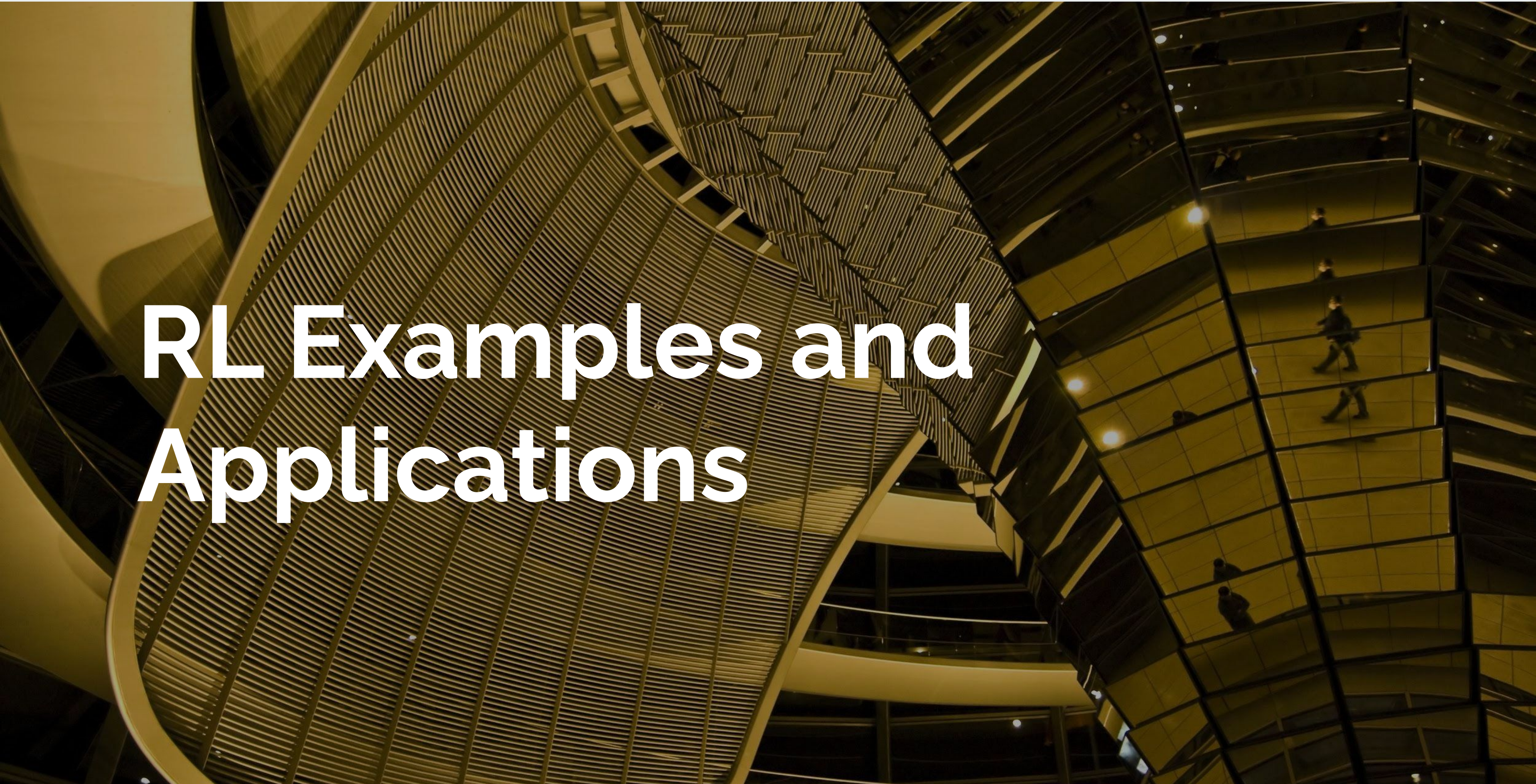
01 | There is no training data or any form of supervision.

02 | Feedback is not instantaneous, it is often delayed.

03 | The agent is trained to select a sequence of actions.

04 | Each action taken alters the course of the agent, like real-life.

RL Examples and Applications



Reinforcement Learning Examples & Applications

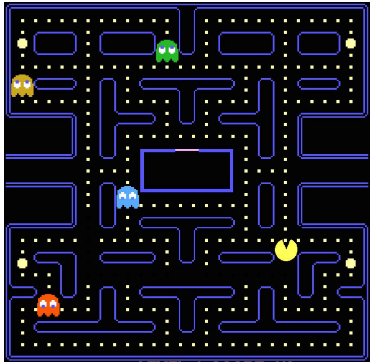
Games

01 | **Board Games** - Go, Chess

02 | **Gambling Games** - Poker, Roulette

03 | **Atari Games** - Breakout, Pac Man, Space Invaders

04 | **Graphic Games** - StarCraft, Subway Surfers



Real Life Applications

Chemistry

RL Algorithms are used to optimize chemical reactions and perform molecular optimisations.

Live Bidding & Advertising

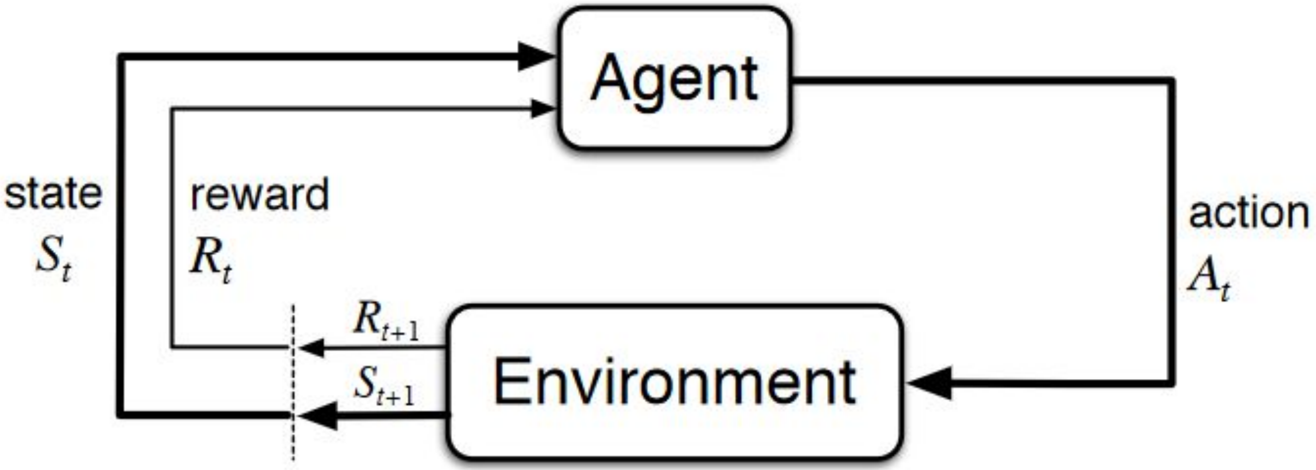
Numerous algorithms by Alibaba Group in online display bidding with a constrained budget.

Robotics

Tremendous amount of research to make robots learn to do specific tasks such as assembling an equipment or walking.

Fundamentals

Agent & the Environment



Reward

01 | Reward is like a feedback for an agent.

02 | It indicates how well a agent in performing at any step.

03 | The goal is to maximise the cumulative reward.

Example | **Playing a game of Poker**

+1 reward when the agent wins.
-1 reward when the agent loses.

Learning to make good sequential decisions

01 | Our goal is to maximise the total reward.

02 | Actions may have long term consequences and rewards may be delayed.

03 | Sometimes, it is better to sacrifice immediate reward for future gains.

Example | **A Financial Investment -** May take months/years to mature.
Chess - Blocking opponent moves may help to win in the long-run.

Agent & the Environment

At each step t, the agent

01 | Execute an action A(t)

02 | Receives observation O(t)

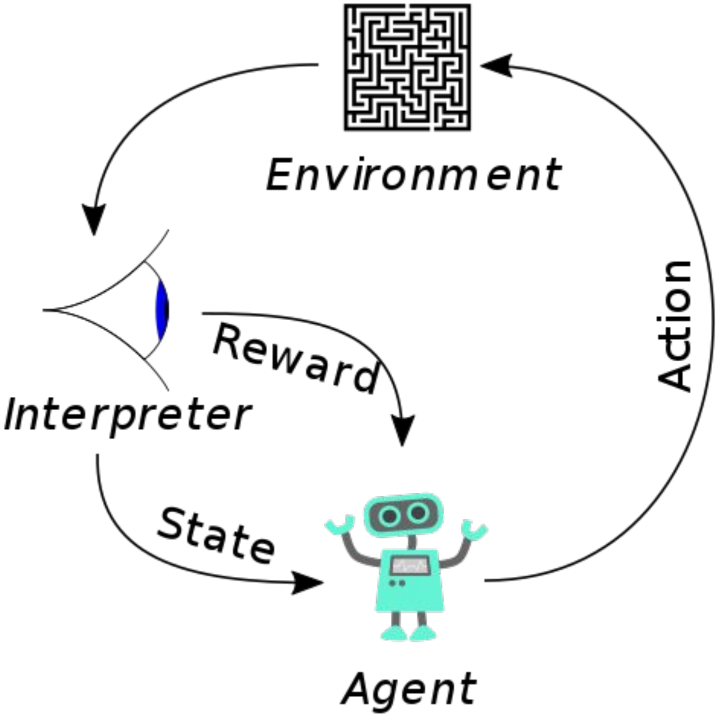
03 | Receives reward R(t)

The Environment

01 | Receives action A(t)

02 | Emits observation O(t+1)

03 | Emits reward R(t+1)



Inside a RL Agent

01 | **Policy** - How does an agent behave?

02 | **Value Function** - How good is each state/action?

03 | **Model** - How does the agent view the environment?

Inside a RL Agent

01 | **Policy** - How does an agent behave?

Mapping from State to Action

Denoted by π

02 | **Value Function** - How good is each state/action?

Used to evaluate the goodness or badness of States

$v_\pi(s) = E_\pi[\ R(t+1) + \gamma R(t+2) + \gamma^2 R(t+3) + \ldots \mid S(t) = s\ ]$

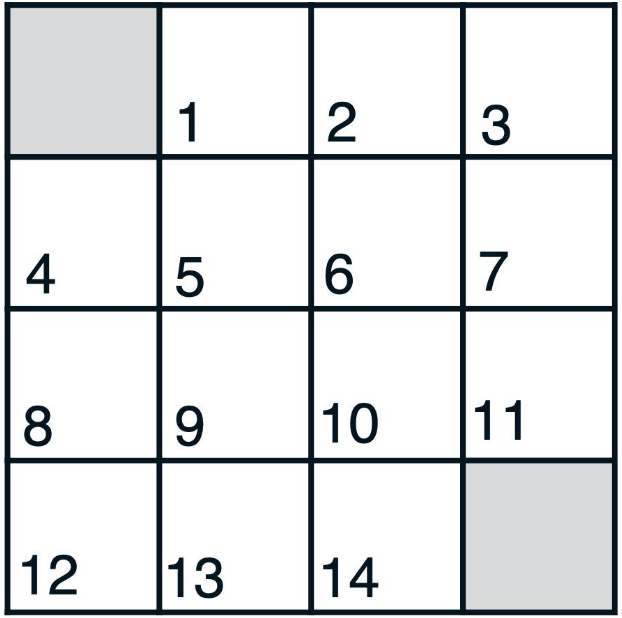
03 | **Model** - How does the agent view the environment?

Predict the next state

Predict the next and immediate reward

GridWorld Example

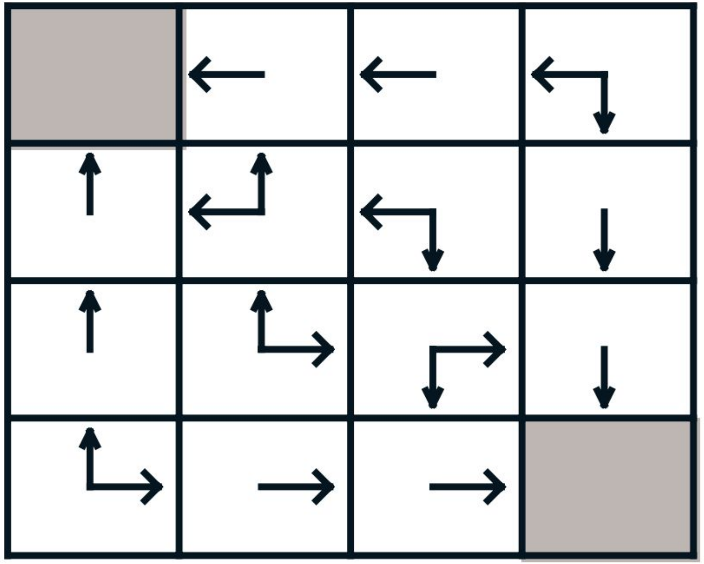
- **Rewards:** -1 per step
- **Actions:** (1) Up, (2) Down, (3) Left, (4) Right
- **States:** Agent's Location



GridWorld Example

Policy π

Take the indicated steps in each state to reach the terminal state.



GridWorld Example

Value Function

The values for each state $v_\pi(s)$ is represented as follows.

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

GridWorld Example

Model

01 | Agent will have a internal model of the environment.

02 | The model may or may not be perfect.

03 | The agent knows the reward it received from each state.

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

Markov Decision Process (MDP)

MDP is used to describe an environment for reinforcement learning, where the environment is fully observable. Almost all RL problems can be formalized as MDPs.

Markov Decision Process - Markov Property

“The future is independent of the past, given the present.”

All history of information encountered so far may be thrown away, and that state is a sufficient statistic that gives us the same characterization of the future as if we have all the history.

- For all Markov states, a state transition probability is defined.

Markov Decision Process

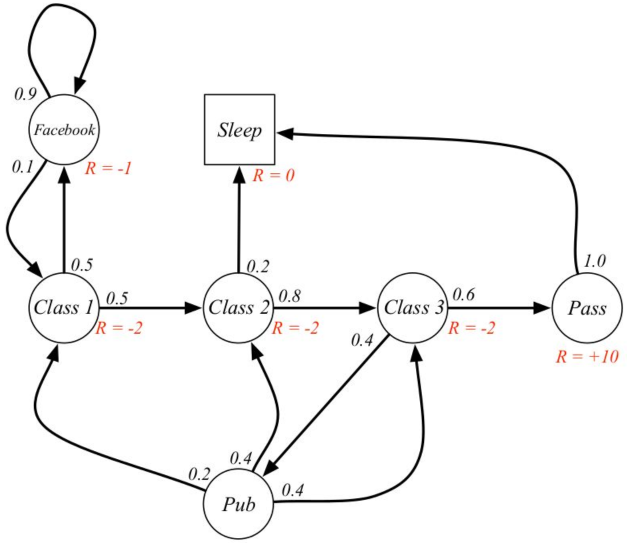
Markov Process

A Markov Process is a series of random states S1, S2, … with the Markov property.

Markov Reward Process

A Markov Reward Process is a Markov process with value judgment, saying how much reward accumulated through some particular sequence that we sampled.

$$G(t) = R(t+1) + \gamma R(t+2) + .. + \gamma^k R(t+k+1)$$



The Bellman Equation

The value function consists of two parts

01 | The immediate reward R_{t+1}

02 | Discounted value of the next state $\gamma\, v(S_{t+1})$

$$\begin{aligned} v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right] \end{aligned}$$

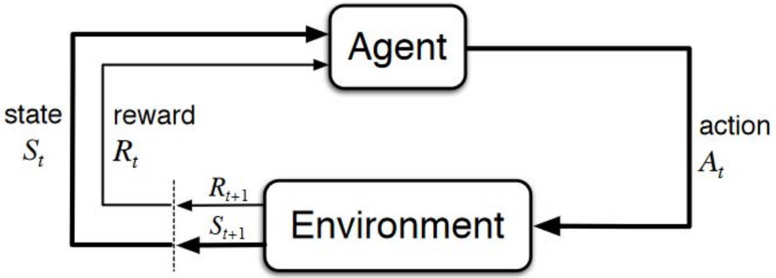
RL Algorithms - Q-Learning

Q-Learning

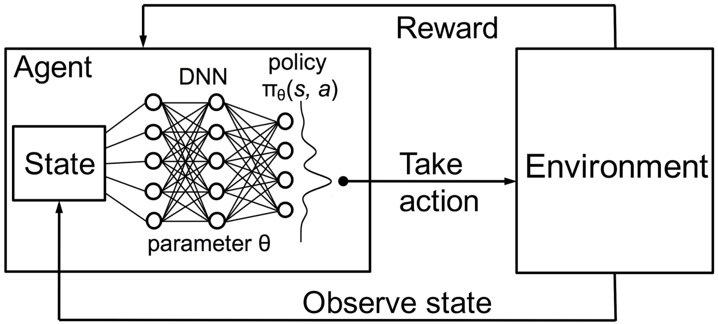
Q-learning is a **model-free** reinforcement learning algorithm to learn a policy telling an agent what action to take under what circumstances. For a finite MDP, Q-Learning always finds the optimal policy.

Types

Q-Table



Q-Learning



Q-Table

Q-Learning

Q-Table

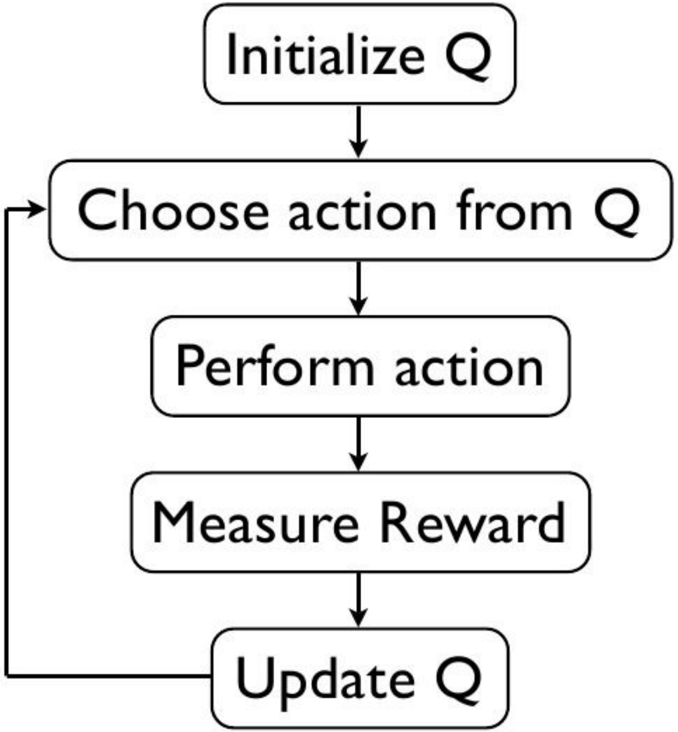
Q-Table is the table used to calculate the maximum expected future rewards for action at each state which will tell us to the best action at each state.

Q-Function

The Q-function uses the Bellman equation to compute the value of each state-action pair.

State-Action	Value
A	1
B	2
C	3
...	..

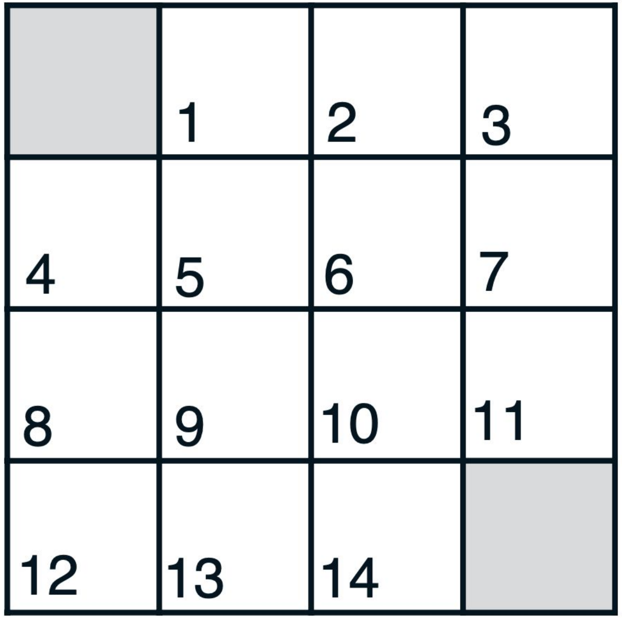
Q-Learning



Initialize Q
Choose action from Q
Perform action
Measure Reward
Update Q

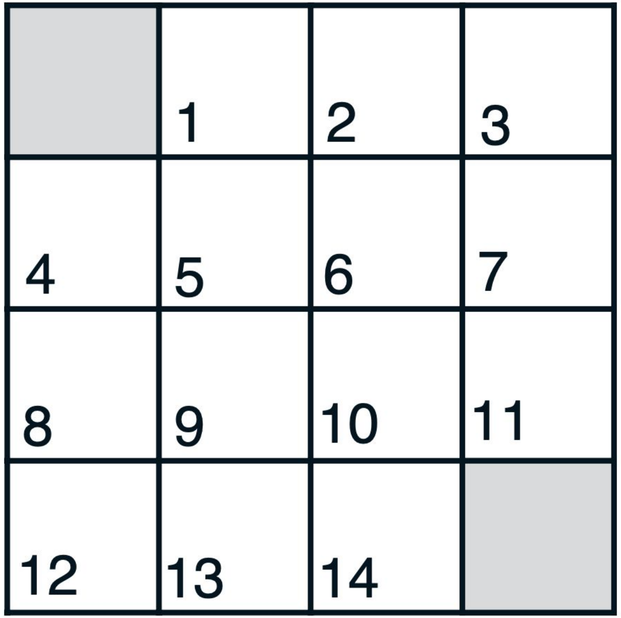
GridWorld Example

- **Rewards:** -1 per step
- **Actions:** (1) Up, (2) Down, (3) Left, (4) Right
- **States:** Agent's Location



Step 1 - Initialize Q-Table

State	Up	Down	Left	Right
1	0	0	0	0
2	0	0	0	0
..	0	0	0	0

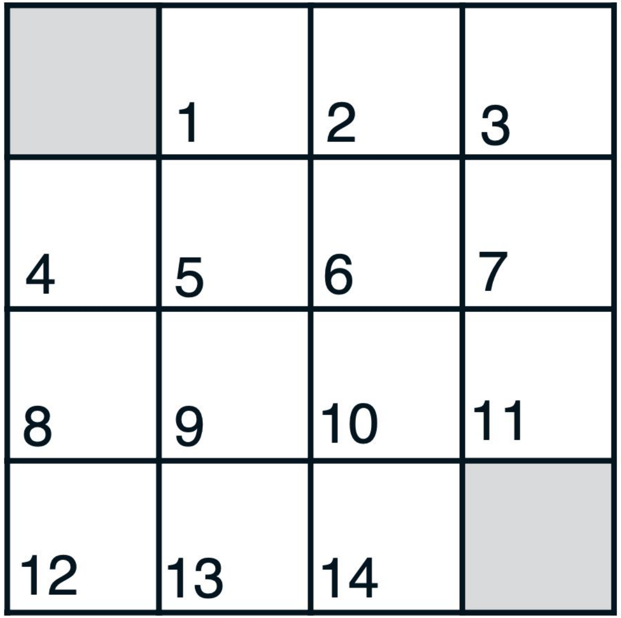


Step 2 & 3 - Choose and Perform an Action

- First, an action (a) in the state (s) is chosen based on the Q-Table. It can random or fixed.

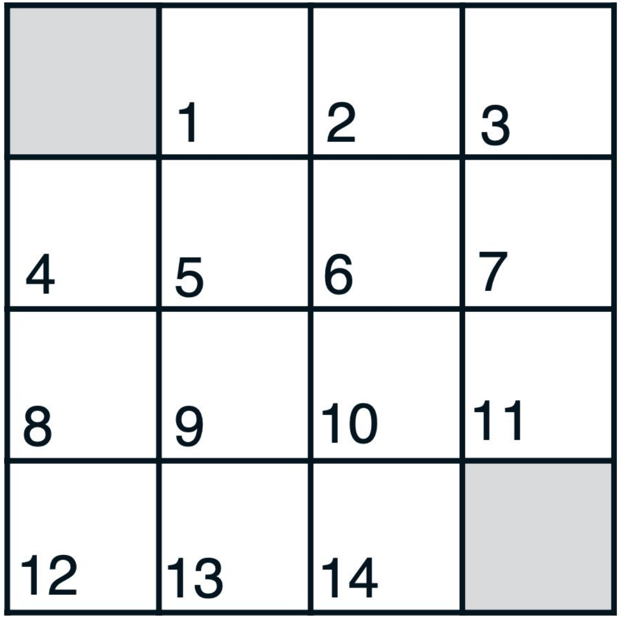
Step 4 & 5 - Observe Reward and Update Q-Table

- A reward is observed and the Q-values for the state are updated using the bellman equation.

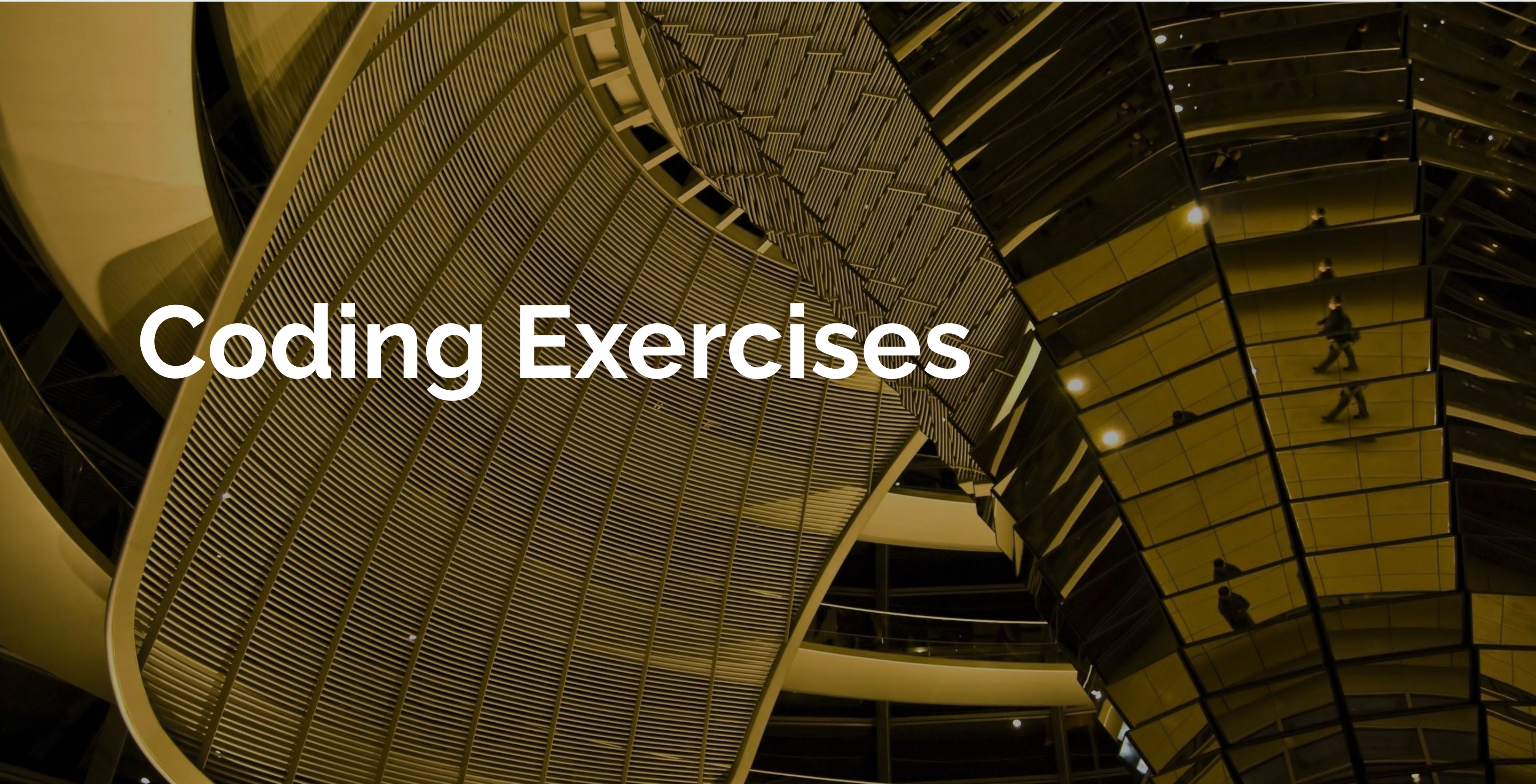


Step 1 - Final Q-Table

State	Up	Down	Left	Right
1	1.1	2.6	5	2.6
2	1.1	1.8	4	0.8
..	..	..	..	..



Coding Exercises



Beyond this Workshop

We have barely scratched the surface.

- **David Silver "Reinforcement Learning" Course at University College London -** https://www.youtube.com/playlist?list=PLacBNHqv7n9gp9cBMrA6oDbzz_8JqhSKo
- **Reinforcement Learning: An Introduction** - Book by Andrew Barto and Richard S. Sutton
- **Lectures from Stanford's Machine Learning course by Andrej Karpathy** - https://www.youtube.com/playlist?list=PLkt2uSq6rBVctENoVBg1TpCC7OQi31AlC
- **The Medium Series of Arthur Juliani, to get some coding of the RL algorithms in TensorFlow** - https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0

Thank you.