MACHINE LEARNING TOKYO

Reinforcement Learning

Session - 1

What is MLT-RL Series?

- Series of study sessions to understand and learn RL.
- Along with application and code for problem statements
- Starting with basic RL understanding
- All the way to Deep RL algorithm understanding and applications
- We try to address them as

`["WHY", "WHAT", "HOW"]`

Alright!! [“What”] is RL?

MACHINE LEARNING TOKYO

- Proposed originally by Edward L. Thorndike, in principle, Reinforced Learning means, the ability of a system to learn from experiences by taking actions. If actions are linked with rewards, those actions are stamped in memory. He called this “*Law of Effects*”.

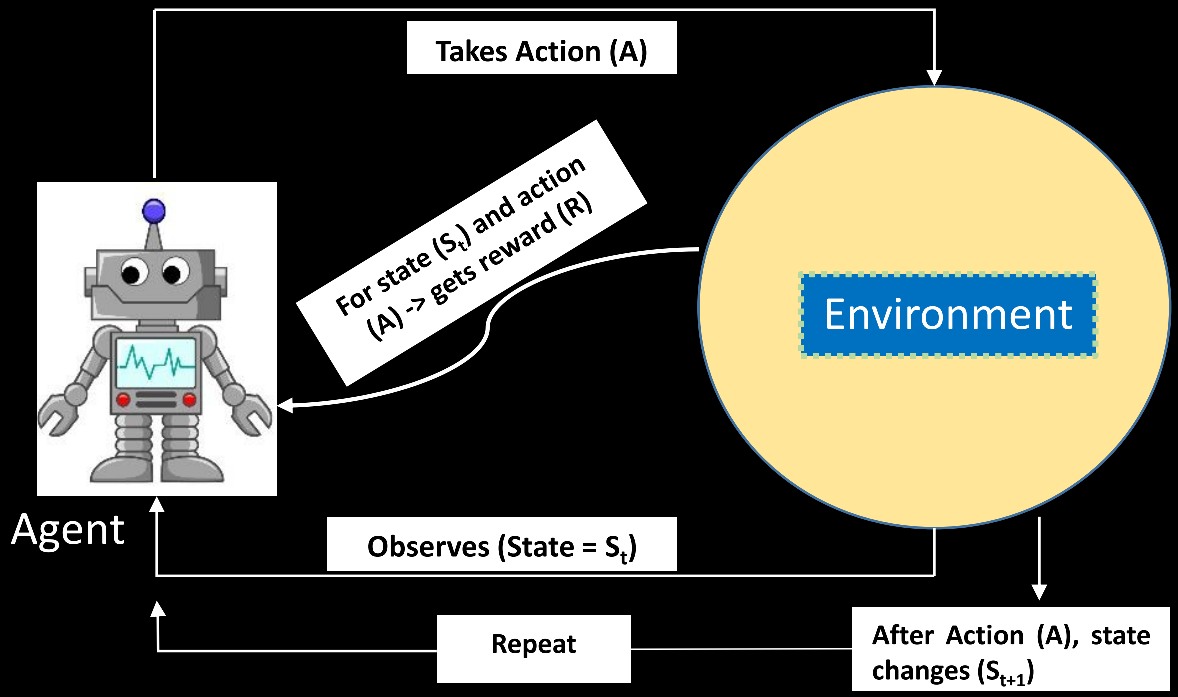
Note:
Important terminologies specific to a RL problem or mathematical formulation will be marked in “THIS COLOR”.

Edward L. Thorndike Experiment – Video : https://www.youtube.com/watch?time_continue=3&v=fanm--WyQJo

["What"] Explained

- The objective of the model is
  - Given a "State (S_t)" of the environment
  - Train an "Agent" to take 'good' "Actions (A)"
  - In order to maximize the "Rewards (R)" received.

- Conditions
  - "State (S_t)" is not in control of the agent. Agent can only observe the state.
  - "Reward (R)" is also a property of the environment. Observed by the Agent, may be with a time-lag
  - Agent can only control the "Action (A)" that it can take



["What"] is the Agent trying to achieve?

- Definition of Task
  - An RL agent tries to optimize actions to achieve a task given. For example
    - Playing a game
    - Deciding when to "Sell/Buy/Hold" stocks for trading in stock exchange
    - Making a robot walk.
    - Making a baby sleep?? **Really** !! (Duh.. Whatever!!)
- Types of Tasks
  - **EPISODIC** : Tasks which end after certain time, for example playing a game. When a game ends (Win/Loss/Draw), the **episode** gets over.
  - **CONTINUOUS** : Tasks which (theoretically) do not have an end, for example getting a robot to walk. For such tasks, we divide the continuous tasks in smaller tasks, each being referred to as **episodes**.

[“HOW”] : Parametrization

MACHINE LEARNING TOKYO

- Any ML/AI task needs numeric data. Taking this one step ahead, every element of an AI/ML problem needs to be “parametrized” so that it can be represented in the form of numbers
- Therefore, we need to parametrize [“State”, “Reward”, “Action”] in an RL problem

[“HOW”] : Understanding “STATE/ACTION/REWARD”

MACHINE LEARNING TOKYO

- State Vector
  - It is the parametrized version of the environment. For eg: While playing Tic-Tac-Toe game, the environment would be an `np.array` of shape of the board (say 3X3).
- Action
  - Generally, looks like a One-Hot-Encoded-Vector, as the model will suggest one of the actions possible. However, it is important know what actions are possible.
- Reward
  - **EPISODIC** Tasks : Find seq of tasks that result in majority of episodes successful.
  - **CONTINOUS** Tasks : Avg reward of a time period for small episodes to be maximized.

["HOW"] : State/Action/Reward Examples

MACHINE LEARNING TOKYO

- Investment Portfolio (CONTINUOUS Task)

State : $f(s_t) = (Cash - in - hand, \%\ investment, liabilities, assets, etc.)$

Action : Sell/Buy/Hold stock

Reward : Avg monthly Profit/Loss (%)

- Cab Driver Ride Selection (CONTINUOUS Task)

State : $f(s_t) = (Source\ Location, Target\ location, Fuel\ available, time, day,)$

Action : Accept Ride / Reject Ride

Reward : Avg Monthly earning

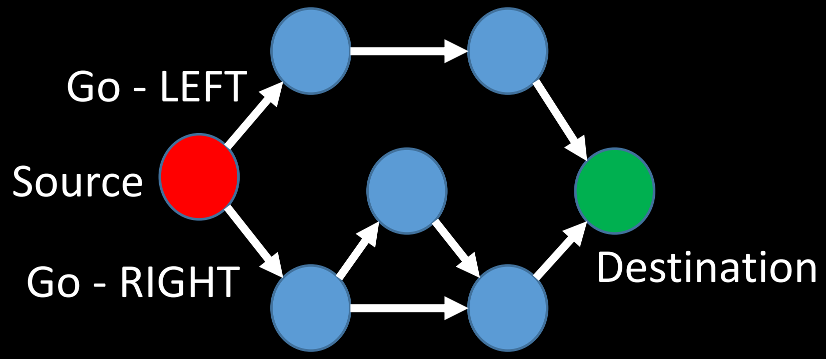
- Tic-Tac-Toe Game (EPISODIC Task)

State : $f(s_t) = \mathrm{np.array}$ -> [1,0,np.nan] based on ["X", "O", free space] on the board (respectively)

Action : Location to play the move. (Play "X" at [0,2] location)

Reward : Game Won/Loss/Draw.

[“HOW”] : Markovian Assumption

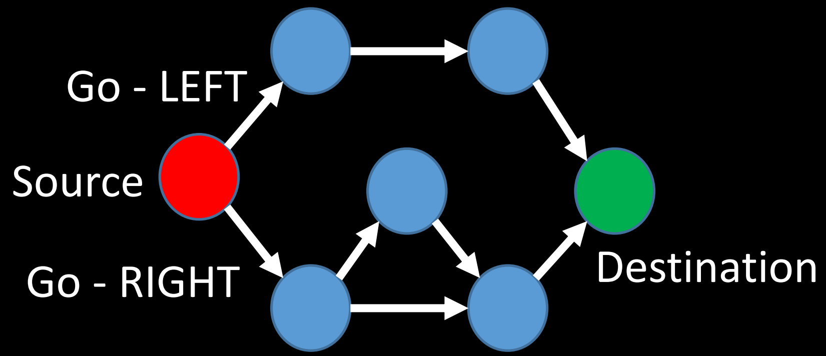


SCENARIO

- Say you are at the SOURCE and want to reach the DESTINATION. And there are some paths that you can follow
- To know, if taking LEFT is better or RIGHT, when at SOURCE, you would really want to know, what lies ahead at each node and see in totality which action would be better.
- In real life, such information may not be available.

[“HOW”] : Markovian Assumption

MACHINE LEARNING TOKYO



MARKOV ASSUMPTION

- DEFINITION
  - Given the current STATE and ACTION TAKEN, the FUTURE can be predicted. It is independent of what was in past.

- EXPLANATION
  - ACTION to be taken in current STATE is dependent only on the current STATE.
  - To predict a good ACTION, the current STATE vector has all required information
  - A good ACTION is not dependent on how we reached the current state and from what path.

Also known as “One Markovian Assumption”.

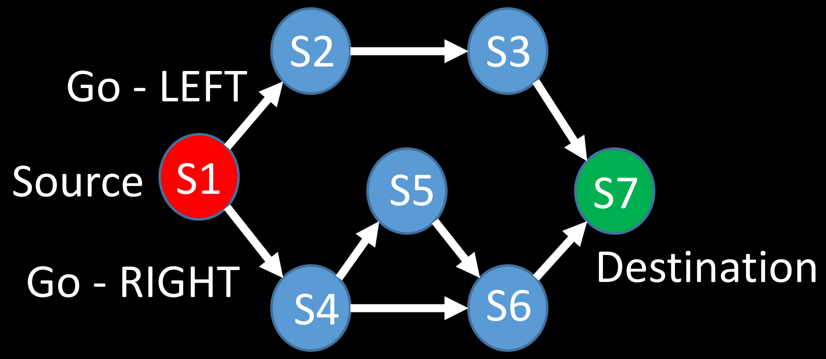
[“HOW”] : Markov Decision Process (MDP)

MACHINE LEARNING TOKYO

- MDP means, the problem statement at hand, can be described in *STATE TRANSITION MANNER,* based on some *PROBABILISTIC* conditions.
- In MDP, the environment and AGENT both follow Markovian Assumption
- Decision = Action taken by the AGENT is based on Markovian Assumption
- Markov = It is symbolic of the STATE vector, which is assumed to provide all relevant information for taking the action & predicting the future.
- MDP assumes that every STATE in that process is dependent only on the previous state
- ***Not all ML problems can be formulated in an MDP***
- If a problem can be formulated in an MDP format i.e. (STATE, ACTION, REWARD) structure only then it is good candidate for application of RL.

[“HOW”] : Model of environment & Types of RL Algorithms

MACHINE LEARNING TOKYO



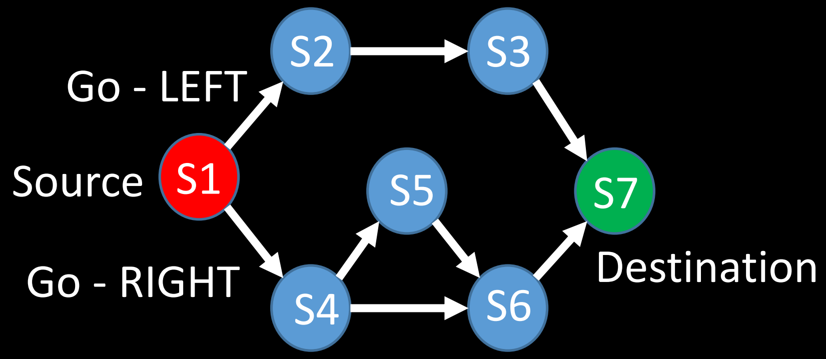
- DEFINITION :
  - The probabilistic information pertaining to the state transitions in an environment when an action is taken.

- EXPLANATION :
  - When in STATE(S1), if agent takes a LEFT, then what is the probability that it will reach STATE(S2). Based on above it is 100%

  - However, in some cases, taking the same action in a particular state may lead to different outcomes. For example : A healthy person is bit by a mosquito, he may or may not fall ill. This becomes probabilistic in nature.

["HOW"] : Model of environment & Types of RL Algorithms

MACHINE LEARNING TOKYO



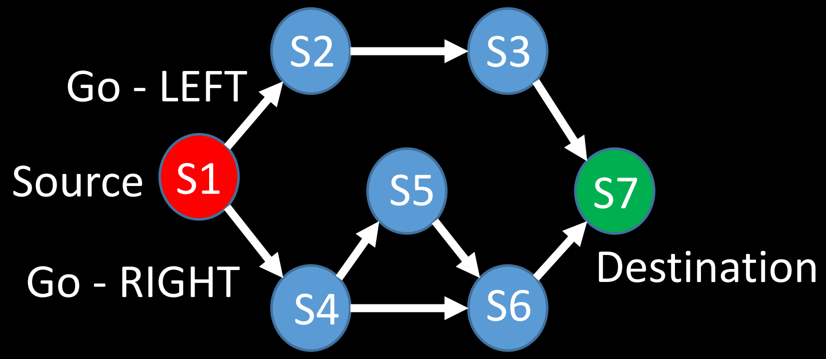
- FORMULATION

$$\boldsymbol{Model\ of\ Environment} = \boldsymbol{p(s', r \mid s, a)}$$

- Probability of reaching a STATE (S') and getting a REWARD (r) given that in current STATE (s), an action (a) was taken.

[“HOW”] : Model of environment & Types of RL Algorithms

MACHINE LEARNING TOKYO



- RL Algorithm Types
  - **Model Based Methods Algorithms**

    These algorithms assume that such a probabilistic map `p(s',r | s,a)` is available for modelling.

  - **Model Free Methods Algorithms**

    These models do not assume the availability of probabilistic map, they are based on the theory of experimentation and learning along the way.

["HOW"] : All said, what does the agent learn

- Given a particular **STATE** of the environment, the job of the AGENT is to predict a good **ACTION**
- Making the **AGENT** learn this ability is the main objective of RL modelling.
- In the language of RL, this objective is called **POLICY** or **CONTROL OBJECTIVE.**

[“HOW”] : All said, what does the agent learn

MACHINE LEARNING TOKYO

Food for Thoughts:

- Think about what a **POLICY** should look like, what information should it capture?
- For example
  - Given a **STATE**, what is the best **ACTION** (main objective) ?
  - Given a **STATE**, differentiate empirically that some **ACTIONS** are better than other
- These 2 elements are key for taking a good decision
- So…….

[“HOW”] : All said, what does the agent learn

- The Policy looks something like this

$$\boldsymbol{Policy} = \boldsymbol{\pi}\,(\,\boldsymbol{action} \mid \boldsymbol{state}\,)$$

- Probability of taking an ACTION, given the STATE
- This probability map is what the agent aims to learn, so as to predict one of the possible actions

Quick Recap : What we know….

- We know, what are **STATE**, **ACTION** & **REWARDS** in an RL Problem.
- What are **EPISODIC** & **CONTINUOUS** tasks for an RL agent.
- What is parametrization of **STATE**, **ACTION** & **REWARDS**
- We know, what is **Markovian Assumption**.
- We know, what is **Markov Decision Process (MDP)**
- We know, what is model of environment (Model Based/Model Free Methods)
- We know, what is **POLICY** with respect to an AGENT and in terms of an RL Problem.

["HOW"] : RL Equations

MACHINE LEARNING TOKYO

- Say you build an RL agent to play the game of CounterStrike.
- In the scene below, not all places/positions in the space visible would be considered as "good" for a player.

"Duck Down" (S1)

"Stand Here" (S2)

- Human logic suggests that State (S1) – "Ducking" is a better than State (S2) – "Standing". A less visible place, with maximum attack range.
- Therefore, some states are good, some are "not so good".

[“HOW”] : RL Equations

MACHINE LEARNING TOKYO

- How to decide, which **STATE** is better
  - In the CounterStrike example, may be, one way to judge that is, how long the player is “alive” in the game by being in that **STATE**. Longer the player stays alive and also with more targets, the better the state.
  - Inturn, what is the ***immediate reward*** & ***cumulative future rewards***.
- How to decide which **ACTION** is better
  - Again in the Counter Strike example, if the player is under attack, would “ducking” be a better action, or “standing” and shooting be a better action. Again, the answer lies in measuring the rewards for it.

BIG QUESTION

- How does an AGENT possibly learn something like this??
- What Equation?

[“HOW”] : RL Equations & Algorithms

MACHINE LEARNING TOKYO

<u>OVER TO THE BOARD!!</u>

In brief, now we know…

MACHINE LEARNING TOKYO

- What are **STATE**, **ACTION** & **REWARDS** in an RL Problem.
- What are **EPISODIC** & **CONTINUOUS** tasks for an RL agent.
- What is parametrization of **STATE**, **ACTION** & **REWARDS**
- What is **Markovian Assumption** & **Markov Decision Process (MDP)**
- What is model of environment (Model Based/Model Free Methods).
- What is **POLICY** with respect to an AGENT and in terms of an RL Problem.
- What are **STATE VALUE FUNCTION** `v(s)` & **ACTION VALUE FUNCTION** `q(a | s)`
- For Model Based Methods, what is **Value Iteration algorithms** & what is **Policy Iteration algorithms**

The CRUX

MACHINE LEARNING TOKYO

(s)	– Current State
(a)	– Action
(s′)	– Next State
(r)	– Immediate Reward
(π)	– Policy
(γ)	– Discount Factor

$$Policy = \pi\,(\,action\,|\,state\,)$$

$$Model\ of\ Environment = p(s', r \mid s, a)$$

$$State\ Value\ v_\pi(s) = \sum_a \pi\,(a \mid s)\, q_\pi(s,a)$$

$$Action\ Value\ q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma v_\pi(s')]$$

__Combined equation for State Value & Action Value__

$$State\ Value\ v_\pi(s) = \sum_a \pi\,(a \mid s) \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma v(s')]$$

THANK YOU!

We’ll be back with <u>**Session -2**</u> soon!!