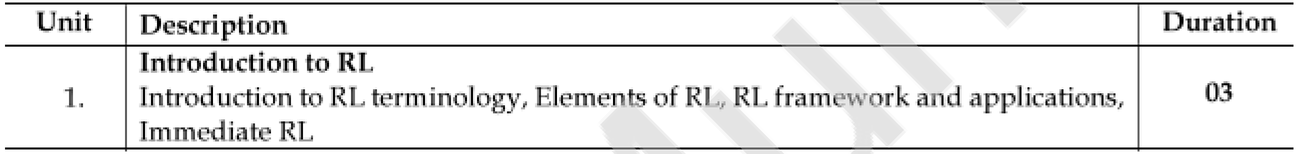
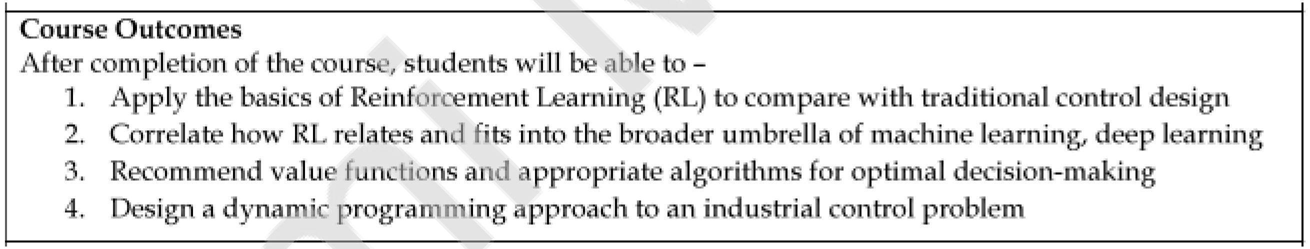
UNIT 1 INTRODUCTION TO REINFORCEMENT LEARNING

Ami Munshi

Syllabus



Unit	Description	Duration
1.	**Introduction to RL** Introduction to RL terminology, Elements of RL, RL framework and applications, Immediate RL	03

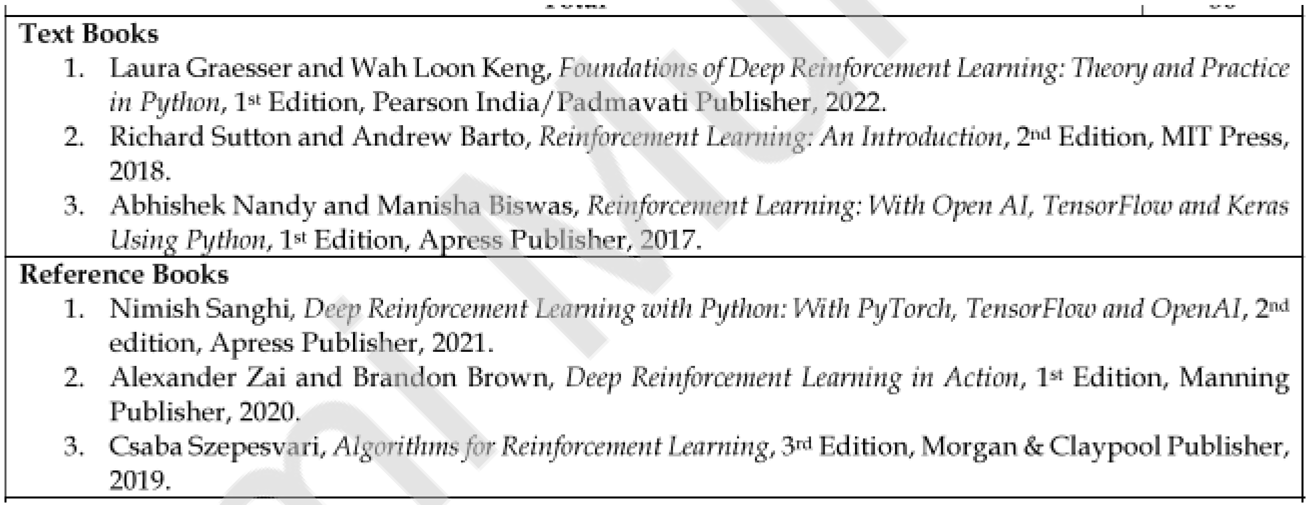


Course Outcomes

After completion of the course, students will be able to –

1. Apply the basics of Reinforcement Learning (RL) to compare with traditional control design
2. Correlate how RL relates and fits into the broader umbrella of machine learning, deep learning
3. Recommend value functions and appropriate algorithms for optimal decision-making
4. Design a dynamic programming approach to an industrial control problem

References



Text Books

1. Laura Graesser and Wah Loon Keng, *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*, 1st Edition, Pearson India/Padmavati Publisher, 2022.
2. Richard Sutton and Andrew Barto, *Reinforcement Learning: An Introduction*, 2nd Edition, MIT Press, 2018.
3. Abhishek Nandy and Manisha Biswas, *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python*, 1st Edition, Apress Publisher, 2017.

Reference Books

1. Nimish Sanghi, *Deep Reinforcement Learning with Python: With PyTorch, TensorFlow and OpenAI*, 2nd edition, Apress Publisher, 2021.
2. Alexander Zai and Brandon Brown, *Deep Reinforcement Learning in Action*, 1st Edition, Manning Publisher, 2020.
3. Csaba Szepesvari, *Algorithms for Reinforcement Learning*, 3rd Edition, Morgan & Claypool Publisher, 2019.

References

- Richard Sutton, Andrew Barto, “Reinforcement Learning: An Introduction”, 2nd edition, MIT Press, 2018
- Abhishek Nandy, Manisha Biswas, “Reinforcement Learning with OpenAI, TensorFlow and Keras Using Python
- Csaba Szepesvari , “Algorithms for RL”, Morgan and Claypool Publisher, 2010
- http://ndl.iitkgp.ac.in/he_document/nptel/courses_106_106_106106143_video_lec5
- https://www.youtube.com/watch?v=lRm6ma9NlhQ&list=PLZ_sl4f41TGvthD8dA7daahlbLV0yDW0w&index=1

Overall concepts in RL

- Introduction, Applications
- Multi-arm Bandit problem
- Markov Decision Process
- Value Iteration
- Policy Iteration
- Monte Carlo Algorithms
- Temporal Difference
- Q Learning
- SARSA
- Policy Gradient methods
- Actor-Critic method

How do we (the humans) actually learn?

- Is it
  - Unsupervised
  - Supervised
  - Reinforcement
  - All of the above

Can you identify type of learning

- Scenario
  - Child points to an object, say apple
  - Parent calls it an apple

- **Supervised**

Can you identify type of learning

- Scenario
  - Child listens to conversation happening around it
  - Child listens and tries to analyze the recurring pattern, sentence flow, context, tone etc
  - Child then picks up the language and attempts to make its own sentences

- **Unsupervised**

Can you identify type of learning

- Scenario
  - Toddler tries to crawl, tries to stand, then take few steps
  - If it manages to stand for few moments, it feels good about itself and gets applaud from parents
  - If it falls, it might feel discouraged
  - Eventually it learns to walk without any explicit instructions

- **Reinforcement**

Human learning supervised way

- **Here's how it works:**
  - **Input:** The child sees an object, let's say an apple. (This is the data for the learning algorithm, the child's brain).
  - **Supervisor:** The parent (or caregiver) labels the object by saying "apple". (This is the supervision part, providing the desired output).
  - **Learning:** The child hears the word "apple" associated with the specific object they are looking at. They begin to form connections between the visual features of the apple and the sound of the word.
  - **Practice:** The parent repeats the labelling process with the same apple or other apples, and the child strengthens the connection between the sight and sound.
  - **Output:** Eventually, the child can recognize an apple on their own and say "apple" or identify it when prompted.
- **Learning from labelled examples**
- **Through repeated exposure and correction by a supervisor (the parent), the child's brain builds a model for identifying apples and expands their vocabulary**

Human learning unsupervised way

- **Child Learning Language Structure:**
  - **Input:** A child hears adults talking around them.
  - **No Labels:** There's no one explicitly explaining grammatical rules or sentence structures.
  - **Unsupervised Learning:** The child listens and picks up on recurring patterns in speech - word order, sentence flow, verb conjugations.
  - **Identifying Patterns:** Over time, the child unconsciously starts to identify patterns in how sentences are formed. They begin to understand the relationships between words and how they function together.
  - **Language Production:** Based on these patterns, the child gradually attempts to form their own sentences, mimicking the structure they've observed.
- Children to acquire the basic rules of their language without needing formal grammar lessons.

Human learning reinforcement way

- **A toddler learning to walk:**
  - **Trial and Error (Actions):** The toddler takes tentative steps, tries to stand, and might even crawl at first. These are all actions they take in the environment.
  - **Rewards:** When the toddler takes a successful step or manages to stand for a moment, it feels good (positive reinforcement). This could be a sense of accomplishment, praise from caregivers, or simply the joy of movement.
  - **Penalties:** Conversely, falls or stumbles can be frustrating (negative reinforcement). They might cry, feel discouraged, or experience a slight physical discomfort.
  - **Learning and Improvement:** Through repeated attempts (trials), the toddler gradually learns which actions lead to successful walking (rewards) and which ones lead to falls (penalties). They strengthen the connections between successful movements and the positive feelings associated with them.
  - **Optimized Behavior:** Over time, the toddler refines their walking technique, taking more confident steps and eventually achieving smooth, balanced walking.
- Toddler doesn't receive explicit instructions on how to walk, but they learn through the consequences of their actions and the feedback they receive (positive or negative) from their environment

Supervised

- Labeled dataset available
- Model is trained to make prediction
- Model is tested
- Example-image recognition, email classification, language translation
- Regression, classification

Unsupervised

- Unlabeled dataset available
- Identifies pattern or structure of data without any guidance
- Example- Anomaly detection, clustering customer preferences
- Clustering

Reinforcement

- Learns from interaction with the environment
- Learns to make sequence of decisions to maximize the awards
- Autonomous driving, game playing, developing trading strategies
- Exploration, Exploitation

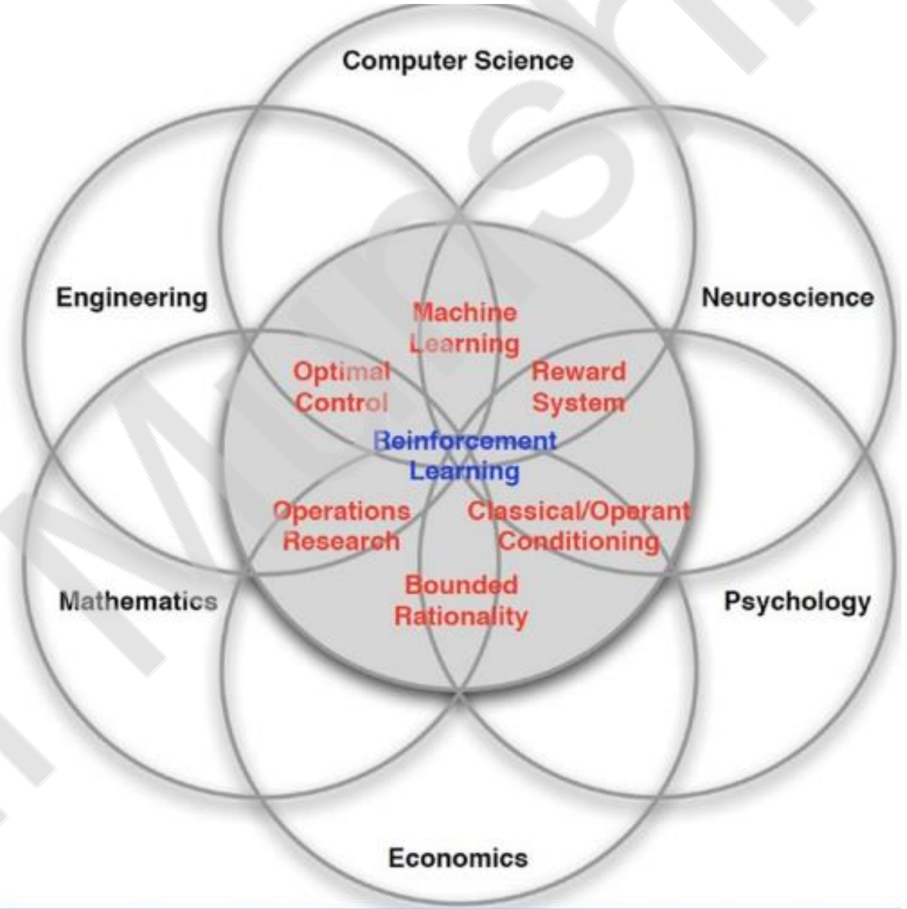
Ref: https://www.linkedin.com/pulse/demystifying-machine-learning-supervised-unsupervised-bushra-akram/
https://medium.com/@bensalemh300/supervised-vs-unsupervised-vs-reinforcement-learning-a3e7bcf1dd23

Example Video

- https://youtu.be/W_gxLKSsSIE?feature=shared

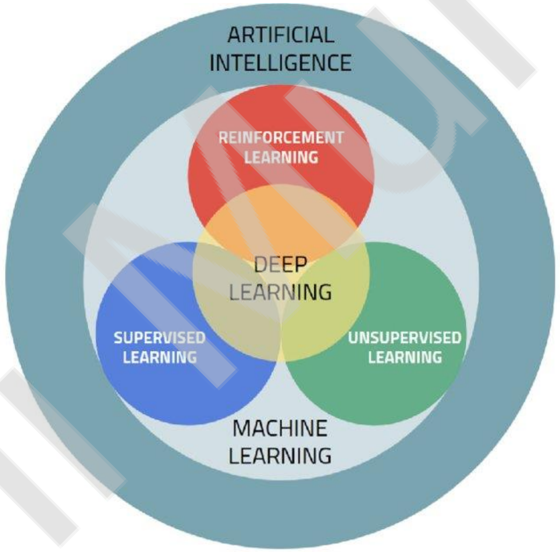
History of RL

Faces of RL



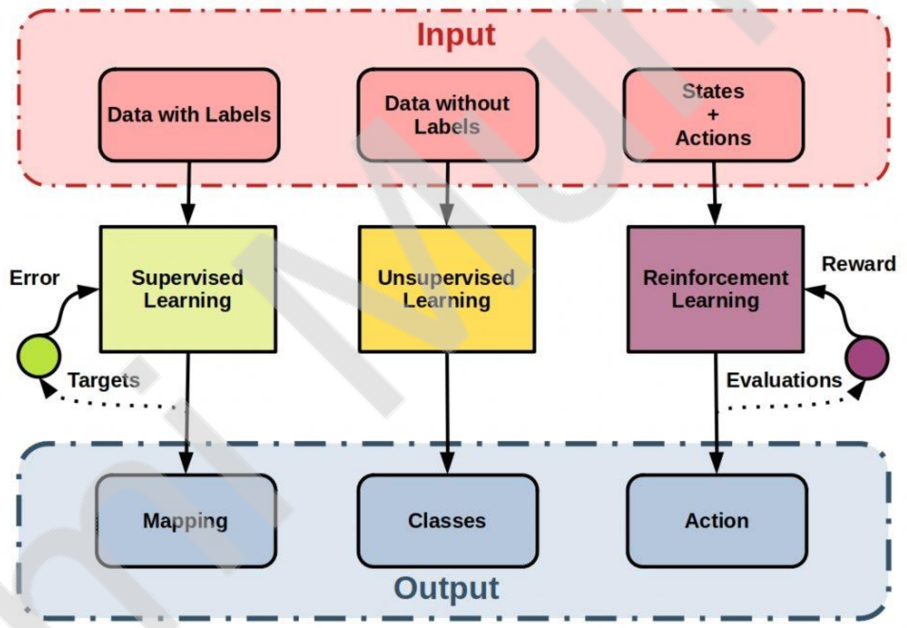
Ref: Abhishek Nandy, Manisha Biswas, Reinforcement Learning With OpenAI, TensorFlow and Keras using Python, Apress, 2018

Relation of RL with AI, ML, DL



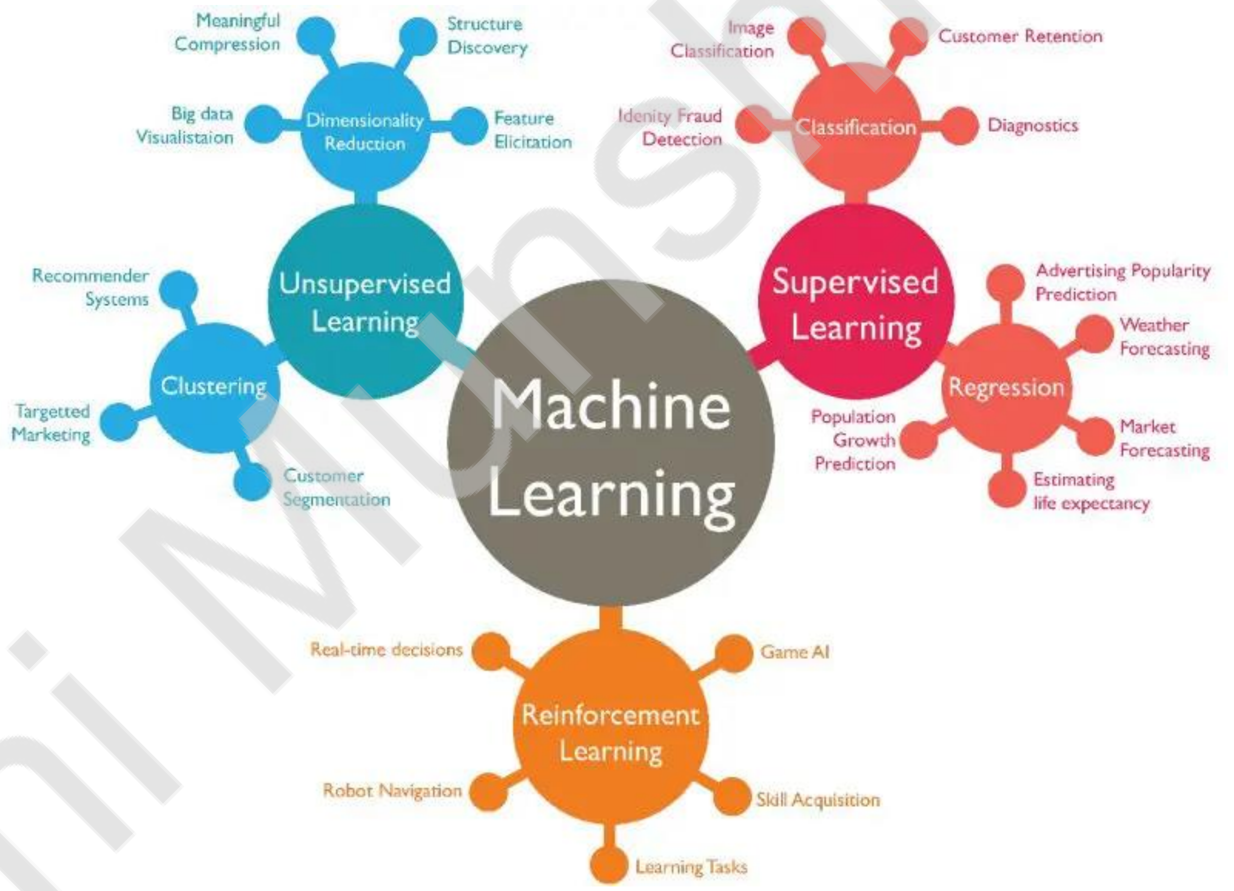
Ref: https://www.researchgate.net/publication/357234810_Artificial_intelligence_machine_learning_for_chemical_sciences/figures?lo=1

Supervised, Unsupervised and Reinforcement Machine Learning



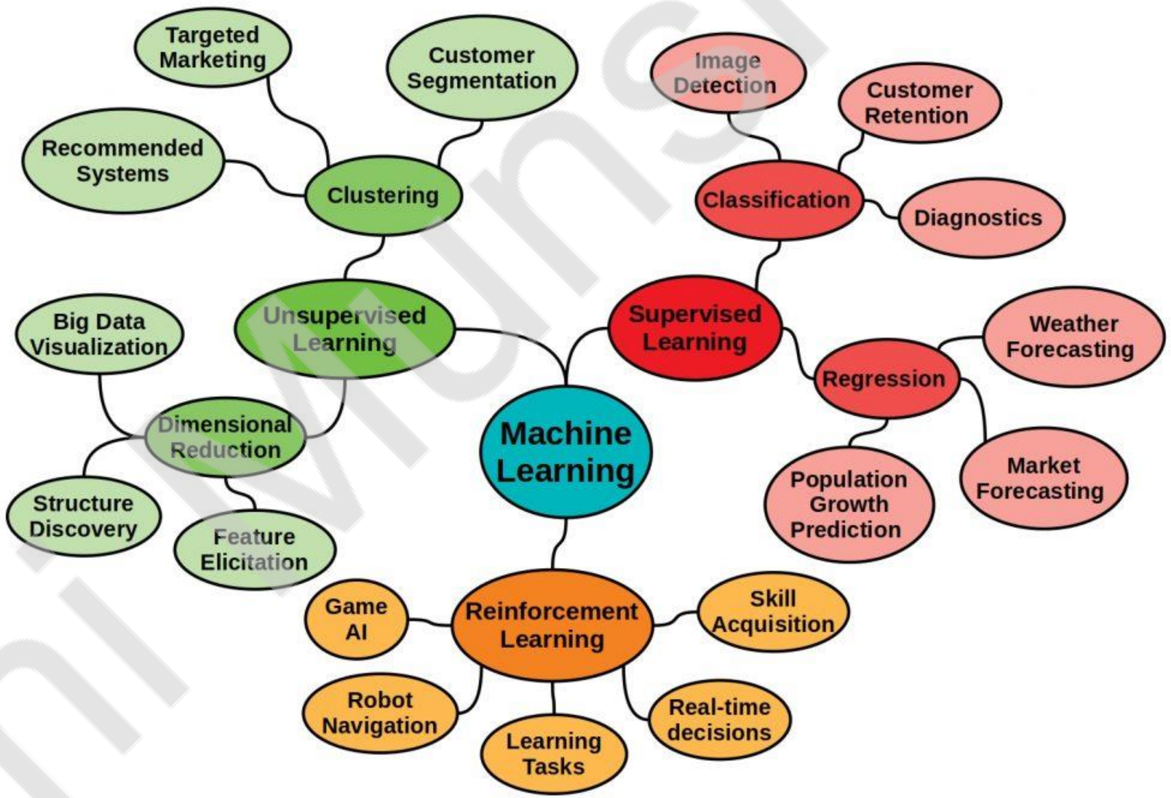
Ref: https://starship-knowledge.com/supervised-vs-unsupervised-vs-reinforcement

Applications of RL, Supervised ML and Unsupervised ML



Ref: https://datasciencedojo.com/blog/machine-learning-101/

Applications of RL, Supervised Ml and Unsupervised ML



Ref: https://starship-knowledge.com/supervised-vs-unsupervised-vs-reinforcement

Reinforcement Learning(RL) Problem

- It is learning
  - what to do
  - how to map situations to actions
  - so as to maximize a numerical reward signal
- Learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them
- Actions may affect not only the immediate reward but also the next situation and all subsequent rewards
- Distinguishing features
  - **Trial and Error**
  - **Immediate or Delayed reward**
- RL is an art and science of decision making
- It works for all the problems where sequence of decisions are important
  - Example-Chess, Maze, Industrial Robot Arm, Path Planning, Sweeper Robot

Main Characteristics of RL

- No supervisor while training
- Environment generally stochastic for real world applications
- Model of the environment can be incomplete
- Feedback (Negative/Positive Reward) can be delayed or partial
- Agent uses experience form past to improve its performance over time
- Actions which have fetched more rewards are preferred
- Agent tries various actions and prefers those actions that are best or have fetched more rewards
- RL uses Markov Decision Process Framework to define interaction between a learning agent and its environment

Reinforcement Learning(RL) Problem

- Challenges in RL
  - **Trade off between Exploration and Exploitation**
- To obtain a lot of reward, a RL agent must prefer actions that it has tried in the past and found to be effective in producing reward- **Exploit**
- But to discover such actions, it has to try actions that it has not selected before- **Explore**
- Note: **Neither exploration nor exploitation can be pursued exclusively without failing at the task**

Simulated Environment

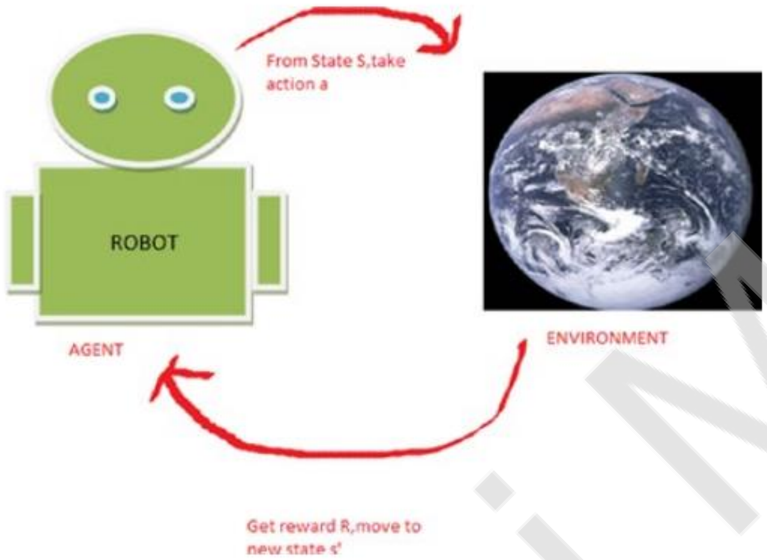


***Figure 1-1.** Reinforcement Learning cycle*

Agent gains knowledge from the environment through sensors

Agent performs action in the environment through actuators

Ref: Abhishek Nandy, Manisha Biswas, Reinforcement Learning With OpenAI, TensorFlow and Keras using Python, Apress, 2018

Some examples of RL-Maze

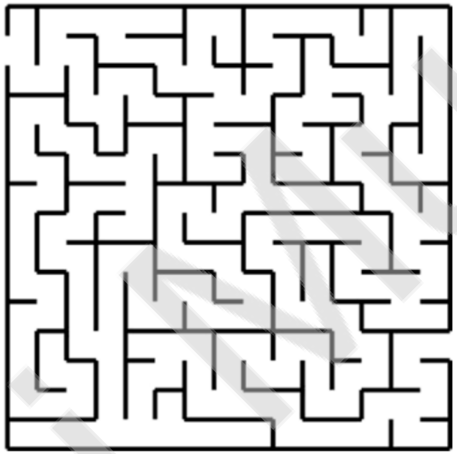


Figure 1-2. Reinforcement Learning can be applied to mazes

In Figure 1-3, we are applying Reinforcement Learning and we call it the Reinforcement Learning box because within its vicinity the process of RL works. RL starts with an intelligent program, known as agents, and when they interact with environments, there are rewards and punishments associated. An environment can be either known or unknown to the agents. The agents take actions to move to the next state in order to maximize rewards.

Ref: Abhishek Nandy, Manisha Biswas, Reinforcement Learning With OpenAI, TensorFlow and Keras using Python, Apress, 2018

Maze Example continued

- In the maze, the centralized concept is to keep moving
- Goal is to clear the maze and reach the end as quickly as possible
- Agent is the intelligent program
- Environment is the maze
- State is the place in the maze where the agent is
- Action is the move we take to move to the next state
- Reward is the points associated with reaching a particular state
  - it can be positive, negative, or zero

Some applications of RL

- Game AI
  - Anything that gives the illusion of intelligence to an appropriate level, thus making the game more immersive, challenging, and, most importantly, fun, can be considered game AI
  - Game AI should be about one thing and one thing only: enabling the developers to create a compelling experience for the player
  - Every technique we use, every trick we play, every algorithm we encode, all should be in support of that single goal
  - Alpha Go and Alpha Go zero

Ref: https://www.oreilly.com/library/view/ai-for-game/0596005555/ch01.html
https://www.gameaipro.com/GameAIPro/GameAIPro_Chapter01_What_is_Game_AI.pdf

Some applications of RL

- Game AI
  - AlphaGo and AlphaGo Zero are RL based algorithms that play Go game
  - Environment?
  - Agent?
  - State?
  - Action?
  - Reward?

Ref: https://www.oreilly.com/library/view/ai-for-game/0596005555/ch01.html
https://www.gameaipro.com/GameAIPro/GameAIPro_Chapter01_What_is_Game_AI.pdf

Some applications of RL

- Game AI
  - AlphaGo and AlphaGo Zero are RL based algorithms that play Go game
  - Environment- Board and player
  - Agent?-Player
  - State?- Current Board Configuration
  - Action?- Placing a stone
  - Reward?- Winning or Losing, capturing a position

Ref: https://www.oreilly.com/library/view/ai-for-game/0596005555/ch01.html
https://www.gameaipro.com/GameAIPro/GameAIPro_Chapter01_What_is_Game_AI.pdf

Some applications of RL

- Data Center Cooling
  - Data center cooling is the process of monitoring and maintaining the temperature within a data center environment. Its sole purpose is to ensure that the electronic equipment, such as servers, networking devices, and storage systems, operates at optimal conditions
  - To keep the servers running, data centers have high energy consumption
  - DeepMind AI Reduces Google Data Centre Cooling Bill by 40%

Ref: https://www.se.com/in/en/work/solutions/for-business/data-centers-and-networks/data-center-cooling/#:~:text=Data%20center%20cooling%20is%20the,systems%2C%20operates%20at%20optimal%20conditions.
https://deepmind.google/discover/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-by-40/

Some applications of RL

- Data Center Cooling
  - Environment?
  - Agent?
  - State?
  - Action?
  - Reward?

Ref: https://www.se.com/in/en/work/solutions/for-business/data-centers-and-networks/data-center-cooling/#:~:text=Data%20center%20cooling%20is%20the,systems%2C%20operates%20at%20optimal%20conditions.
https://deepmind.google/discover/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-by-40/

Some applications of RL

- Data Center Cooling
  - Environment- Datacenter
  - Agent- DC Controller
  - State- Energy Consumption time t
  - Action- Move Load, turn on, turn off
  - Reward- Reduction in energy consumption, Keeping datacenter cool

Ref: https://www.youtube.com/watch?v=lRm6ma9NlhQ&list=PLZ_sl4f41TGvthD8dA7daahlbLV0yDW0w&index=1
https://deepmind.google/discover/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-by-40/

Some applications of RL

- Resource Allocation

Some applications of RL

- Trading and Finance

Some applications of RL

- Healthcare

Some applications of RL

- Robotics

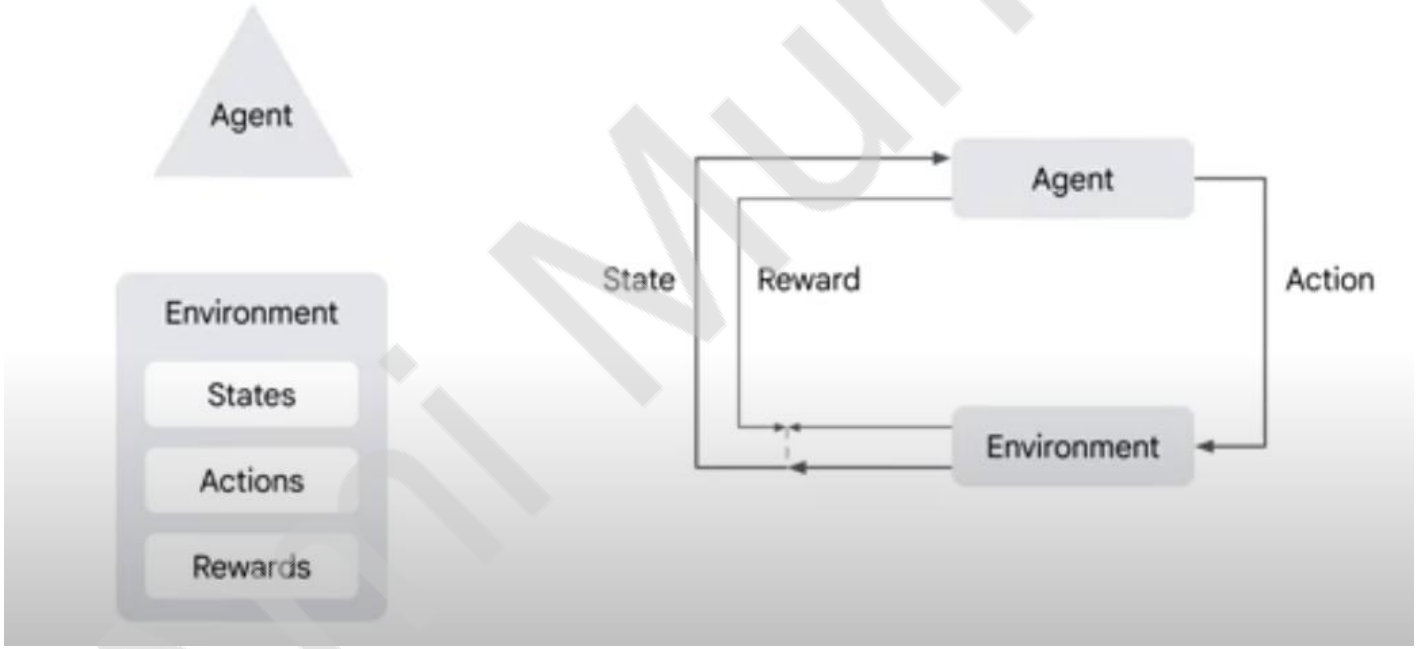
Some applications of RL

- Robotics

Some examples of RL

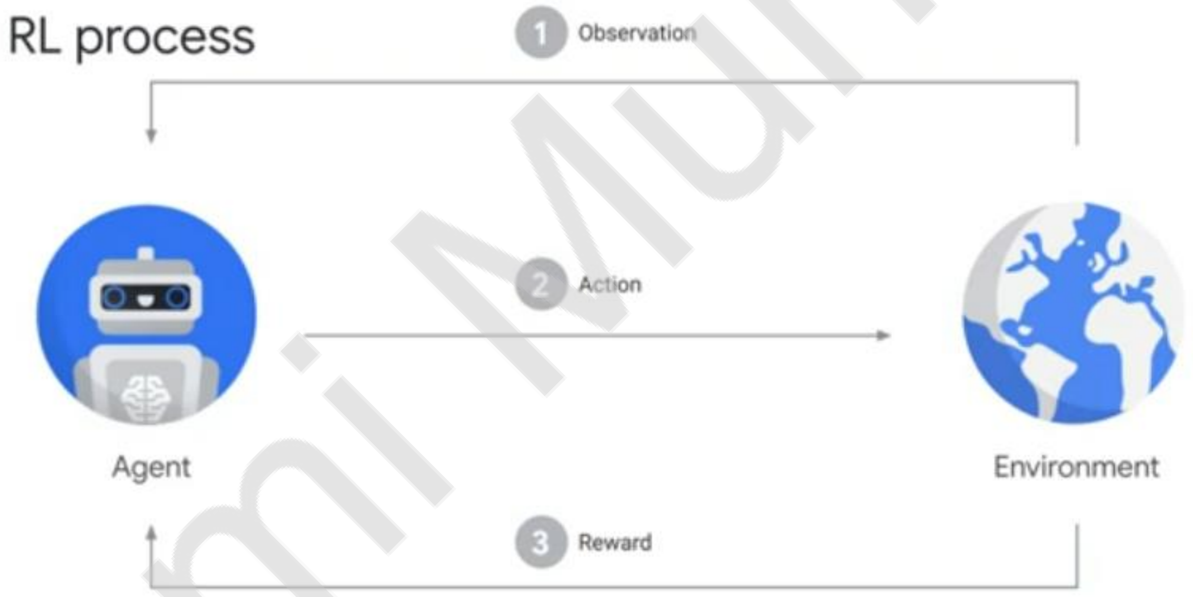
- Refer to some examples of RL in chapter 1 of
  - Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2018

RL framework



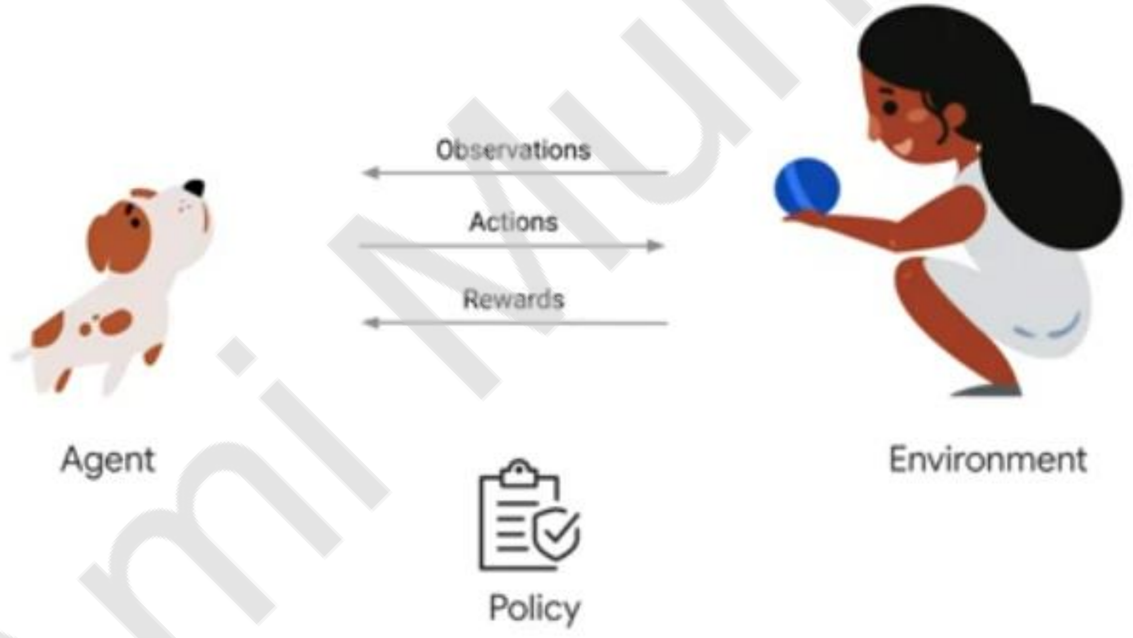
Ref: https://www.cloudskillsboost.google/course_templates/39/video/325107?locale=id

RL Process



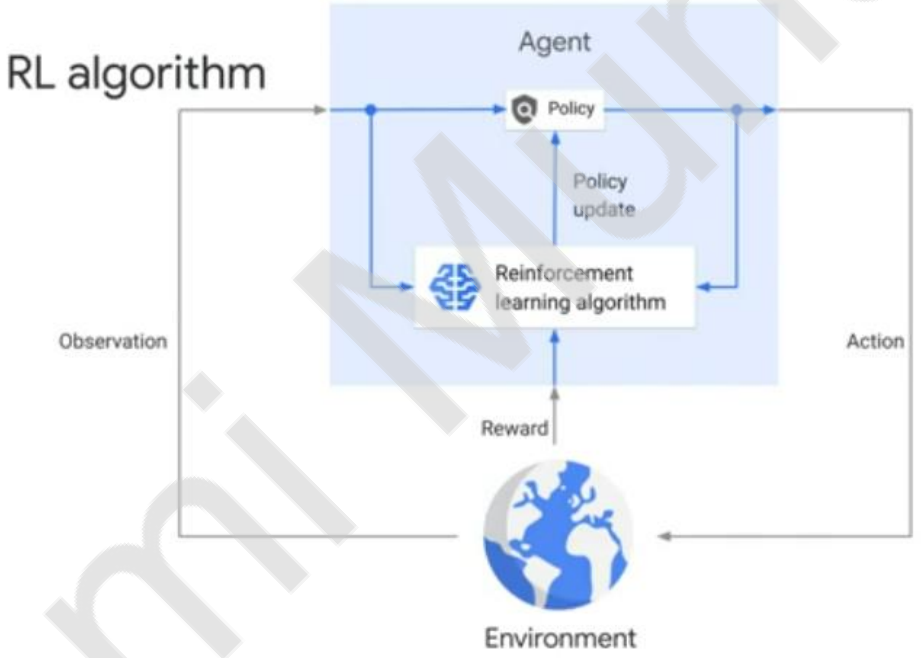
Ref: https://www.cloudskillsboost.google/course_templates/39/video/325107?locale=id

RL in training a dog



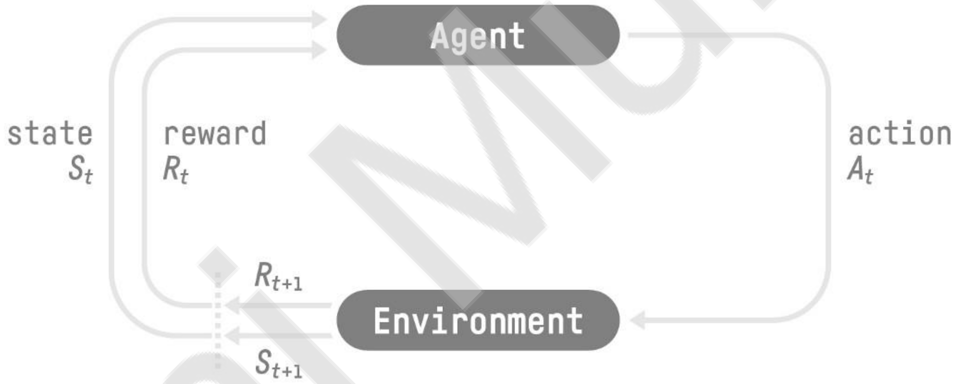
Ref: https://www.cloudskillsboost.google/course_templates/39/video/325107?locale=id

RL Algorithm



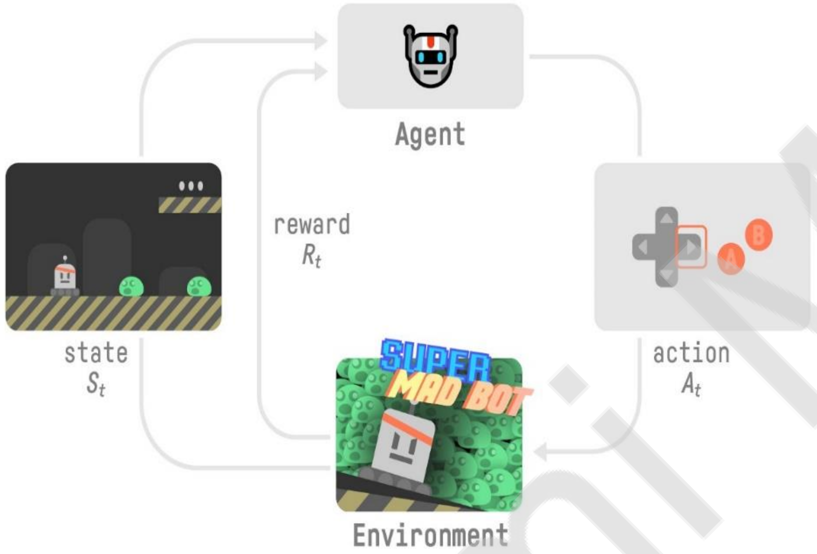
Ref: https://www.cloudskillsboost.google/course_templates/39/video/325107?locale=id

The RL Process: a loop of state, action, reward and next state



Ref: Richard Sutton, Andrew Barto, “Reinforcement Learning: An Introduction”, 2nd edition, MIT Press, 2020
https://huggingface.co/learn/deep-rl-course/en/unit1/rl-framework

The RL Process: Example



•Our Agent receives **state S0** from the **Environment** — we receive the first frame of our game (Environment).
•Based on that **stateS0,** the Agent takes **action*A0*** — our Agent will move to the right.
•The environment goes to a **new state*S1*** — new frame.
•The environment gives some **reward*R1*** to the Agent — we're not dead *(Positive Reward +1)*.

Ref: https://huggingface.co/learn/deep-rl-course/en/unit1/rl-framework

State Transition Process

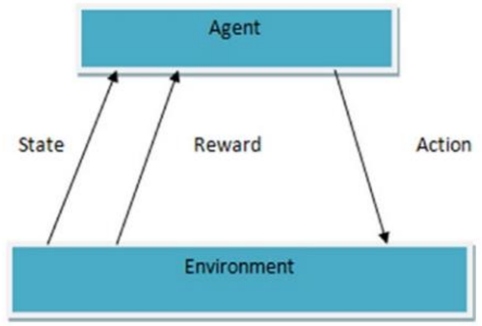
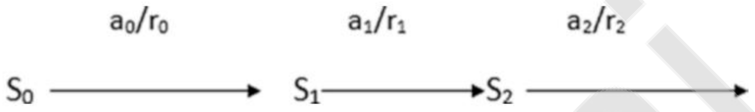


Figure 1-8. Scenario of state changes



$$\begin{tikzcd} S_0 \arrow[r, "a_0/r_0"] & S_1 \arrow[r, "a_1/r_1"] & S_2 \arrow[r, "a_2/r_2"] & {} \end{tikzcd}$$

Figure 1-9. The state transition process

- Learn to choose actions that maximize the following:
  - $r0 + \gamma r1 + \gamma 2r2 + \ldots\ldots\ldots\ldots$ where $0 < \gamma$
- At each state transition, the reward is a different value, hence we describe reward with varying values in each step, such as r0, r1, r2, etc
- Gamma (γ) is called a discount factor and it determines what future reward types we get:
  - A gamma value of 0 means the reward is associated with the current state only
  - A gamma value of 1 means that the reward is long-term

Ref: Abhishek Nandy, Manisha Biswas, Reinforcement Learning With OpenAI, TensorFlow and Keras using Python, Apress, 2018

Elements of RL

- Agent
- Environment
- Policy
- Reward Signal
- Value Function
- Model

Agent

- In terms of Reinforcement Learning, agents are the software programs that make intelligent decisions
- Agents should be able to perceive what is happening in the environment
- Here are the basic steps of the agents
  - When the agent can perceive the environment, it can make better decisions
  - Decision the agents take results in an action
  - Action that the agents perform must be best and optimal one

Environment

- Environments in the Reinforcement Learning space are comprised of certain factors that determine the impact on the Reinforcement Learning agent
- Agent must adapt accordingly to the environment
- These environments can be 2D worlds or grids or even a 3D world
- Here are some important features of environments:
  - Deterministic vs Stochastic
  - Fully observable vs partially observable environment
  - Competitive vs Collaborative
  - Discrete or continuous
  - Single-agent vs Multi-agent

Ref: Abhishek Nandy, Manisha Biswas, Reinforcement Learning With OpenAI, TensorFlow and Keras using Python, Apress, 2018
https://www.scaler.com/topics/artificial-intelligence-tutorial/types-of-environment-in-ai/

Task Environment classifications

- Full observable/Partially Observable/ Unobservable
- Single agent/Multi agent
- Competitive/Cooperative
- Deterministic/Stochastic
- Discrete/Continuous

Task Environments-classification

- Fully observable
  - An agent's sensors give it access to the complete state of the environment at each point in time
  - Sensors detect all aspects that are *relevant* to the choice of action; relevance, in turn, depends on the performance measure
- Partially observable
  - Could be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data
    - For example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares
    - An automated taxi cannot see what other drivers are thinking
- Unobservable
  - Agent has no sensors at all then the environment

Task Environments-classification

- Single agent
  - Only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment
  - Example- solving crossword puzzle by itself
- Multi agent
  - Multiple agents are operating in an environment, then such an environment is called a multi-agent environment
  - Example- Playing soccer match is a multi agent environment

Task Environments-classification

- Competitive
  - Example –chess
  - Entity B is trying to maximize the performance measure which minimizes the performance of B
- Cooperative
  - Example-
  - Taxi driving environment is partially cooperative- avoids collision and hence maximizes the performance
  - But it is partially competitive- why???
  - For instance like- only one car can occupy the parking space

Task Environments-classification

- Deterministic/Stochastic
  - If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic otherwise it is stochastic
  - Taxi driver example is stochastic because one can never predict behaviour of traffic, tyres can blow, engine can fail

Task Environments-classification

- Discrete/Continuous
  - Discrete/continuous distinction applies to the *state* of the environment, to the way *time* is handled, and to the *percepts* and *actions* of the agent
  - For example, the chess environment has a finite number of distinct states
  - Chess also has a discrete set of percepts and actions
  - Taxi driving is a continuous-state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time
  - Taxi-driving actions are also continuous (steering angles, etc.)

Elements of RL

- Policy
  - Defines the learning agent’s way of behaving at a given time
  - Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states
  - It corresponds to what in psychology would be called a set of stimulus–response rules or associations
  - In some cases the policy may be a **simple function or lookup table**, whereas in others it may involve **extensive computation** such as a search process
  - Policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior
  - Policies may be stochastic, specifying probabilities for each action

Elements of RL

- Reward signal defines the goal of a reinforcement learning problem
- On each time step, the environment sends to the reinforcement learning agent a single number called the reward
- Agent's sole objective is to maximize the total reward it receives over the long run
- Reward signal thus defines what are the good and bad events for the agent
- In a biological system, we might think of rewards as analogous to the experiences of pleasure or pain
- They are the immediate and defining features of the problem faced by the agent
- Reward signal is the primary basis for altering the policy
  - If an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in that situation in the future
- In general, reward signals may be stochastic functions of the state of the environment and the actions taken

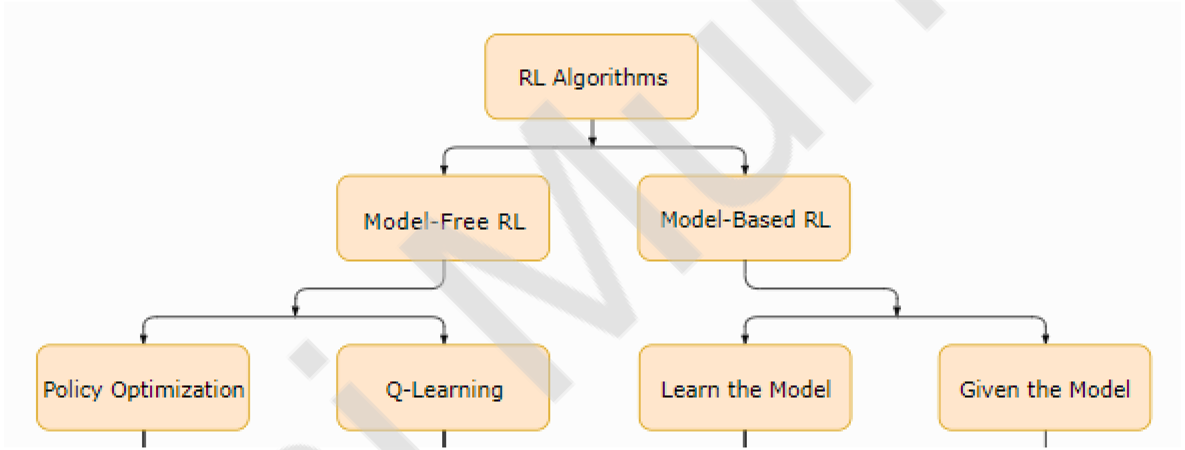
Elements of RL

- Reward signal indicates what is good in an immediate sense, a value function specifies what is good in the long run
- Roughly speaking, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state
- Rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow and the rewards available in those states
- For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards
- Or the reverse could be true
- To make a human analogy
  - Rewards are somewhat like pleasure (if high) and pain (if low)
  - Whereas values correspond to a more refined and farsighted judgment of how pleased or displeased we are that our environment is in a particular state

Elements of RL

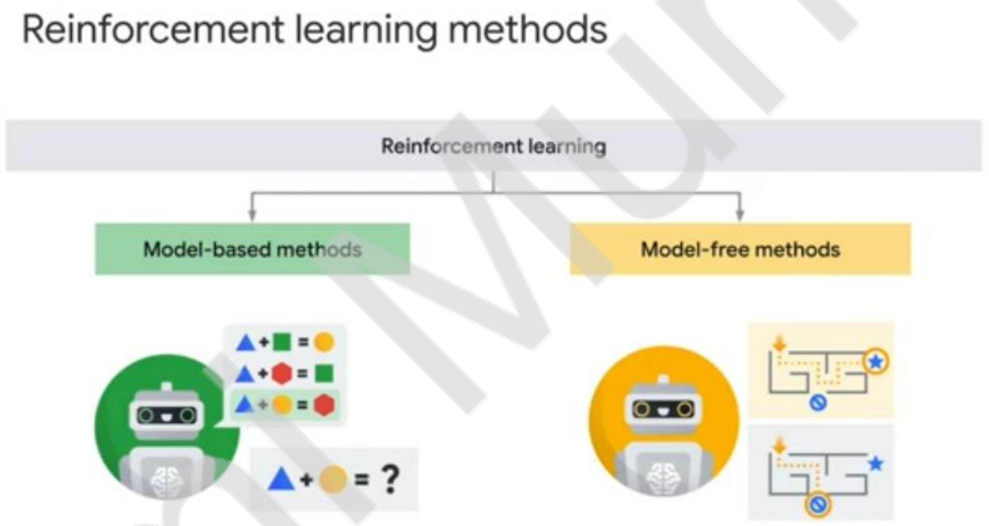
- Final element of some reinforcement learning systems is a model of the environment
- This is something that mimics the behavior of the environment, or more generally, that allows inferences to be made about how the environment will behave
- For example, given a state and action, the model might predict the resultant next state and next reward
- Models are used for planning, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced
- Methods for solving reinforcement learning problems that use models and planning are called model-based methods, as opposed to simpler model-free methods that are explicitly trial-and-error learners—viewed as almost the opposite of planning

Kinds of RL or Taxonomy of RL Algorithms



Ref:https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Kinds of RL or Taxonomy of RL Algorithms



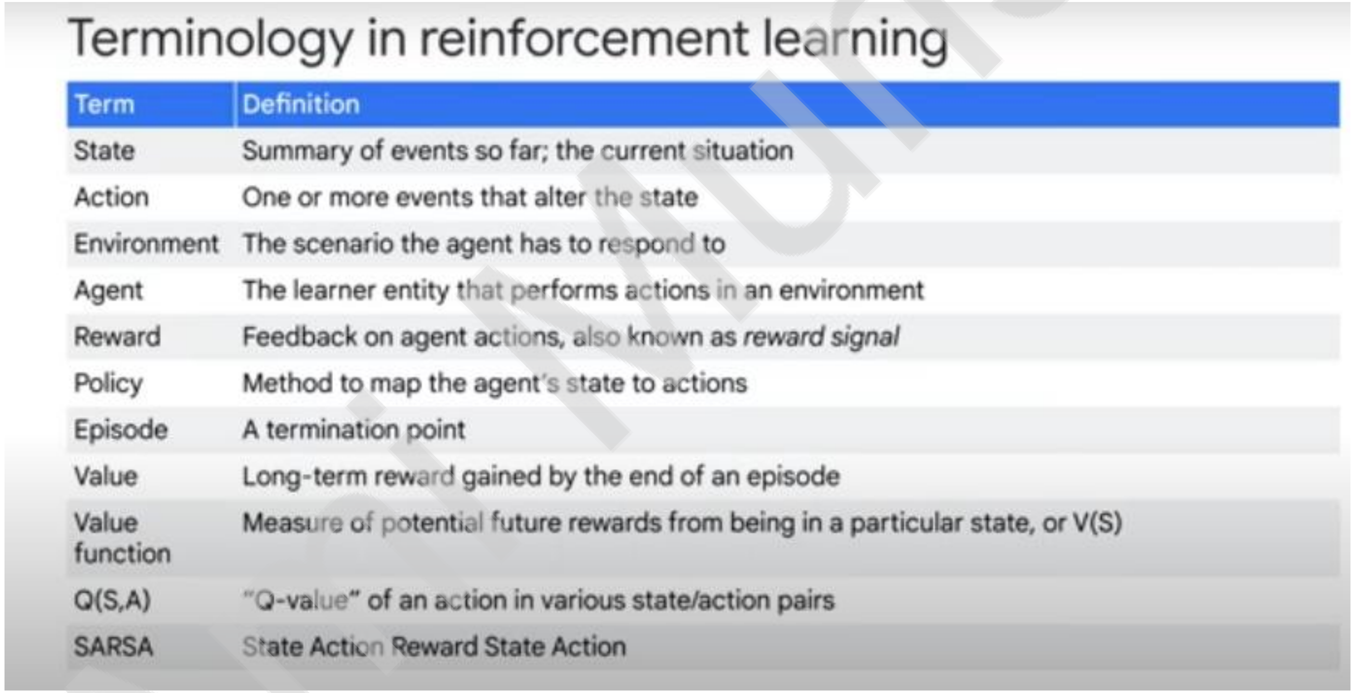
Ref: https://www.cloudskillsboost.google/course_templates/39/video/325106?locale=id

Model based and Model free RL

- Model based
  - Agent trying to understand its environment and creating a model for it based on its interactions with this environment
  - In model-based, you try to understand the rules first — creating a **mental map** of how everything works
- Model free
  - seek to learn the consequences of their actions through experience via algorithms such as Policy Gradient, Q-Learning, etc
  - model-free doesn't bother with all that planning. Instead, it learns by **trying things out** and remembering what got good results

Refhttps://medium.com/@kalra.rakshit/the-difference-between-model-based-and-model-free-reinforcement-learning-9499af3770db

Some Terminology in RL

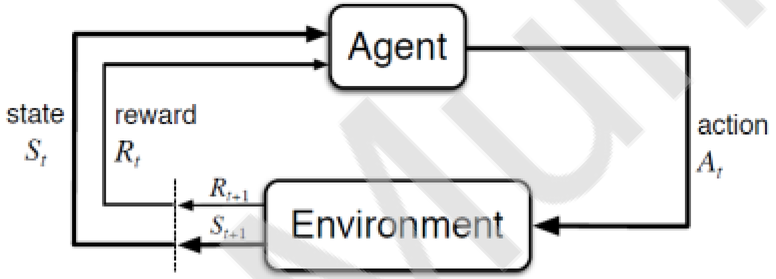


Terminology in reinforcement learning

Term	Definition
State	Summary of events so far; the current situation
Action	One or more events that alter the state
Environment	The scenario the agent has to respond to
Agent	The learner entity that performs actions in an environment
Reward	Feedback on agent actions, also known as *reward signal*
Policy	Method to map the agent's state to actions
Episode	A termination point
Value	Long-term reward gained by the end of an episode
Value function	Measure of potential future rewards from being in a particular state, or V(S)
Q(S,A)	"Q-value" of an action in various state/action pairs
SARSA	State Action Reward State Action

Ref: https://www.cloudskillsboost.google/course_templates/39/video/325107?locale=id

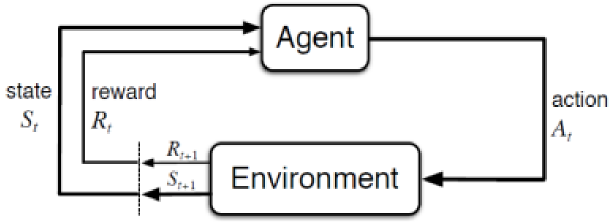
RL Settings and Notion



Agent: learner and decision maker.
Environment: everything outside the agent.
Timestep: at each timestep t, the agent and environment interacts, t =0, 1,2,3,…
At timestep t:
State: $S_t \in \mathcal{S}$
Action: $A_t \in \mathcal{A}(\mathcal{S})$
Reward: at timestep t+1, as the consequence of the action A_t, the agent receives a ***reward*** $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and the environment moves to a new state S_{t+1}.
History (or trajectory): an interaction sequence, $S_0, A_0, R_1, S_1, A_1, R_2, S_2$, ...

Ref: Weidong Kuang, Fundamentals of RL

RL Settings and Notion



Dynamics of the environment: $p(s', r|s, a) \equiv Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$ is the conditional joint probability of the next state and reward, given the current state and action. In our context, we assume that the dynamics satisfy the Markov property.

Markov property:

- $p(s', r|s, a)$ completely characterizes the environment's dynamics.
- The state captures all relevant information from history.
- Once the state is known, the history may be thrown away.
- The state is a sufficient statistic of the past.

Return at t: the sum of the rewards received after timestep t (from t+1 to T, T is the final time step)

$$G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \dots + \gamma^{T-(t+1)} R_T$$

The general goal of reinforcement learning is to maximize the overall return by selecting appropriate actions during the interaction.

Discount rate: $\gamma \in [0,1]$. A small discount rate gives less weights of future rewards towards the return. There are two extreme cases: 1) only immediate reward (R_{t+1}) is counted when $\gamma = 0$; 2) all rewards are equally counted when $\gamma = 1$.

Ref: Weidong Kuang, Fundamentals of RL

Limitations of RL

Immediate RL Vs Full RL

- In Immediate RL
  - Agent receive rewards immediately after each action
  - Hence decision making becomes quicker
- In Full RL
  - Rewards are delayed
  - This requires agents to strategize for long term goals
  - Profound understanding of environment is needed in this case

Examples of Immediate Reinforcement-Based in real life scenarios

- Giving Treats for Homework Completion
- Earning Points in a Game
- Receiving Applause After a Performance
- Receiving Praise for a Task
- Getting Paid Directly After Work
- Eating immediately after Feeling Hungry
- Social Media Notifications

Ref: https://fuskal.com/immediate-reinforcement/#Immediate_Reinforcement_and_Operant_Conditioning

Examples of Delayed Reinforcement-Based in real life scenarios

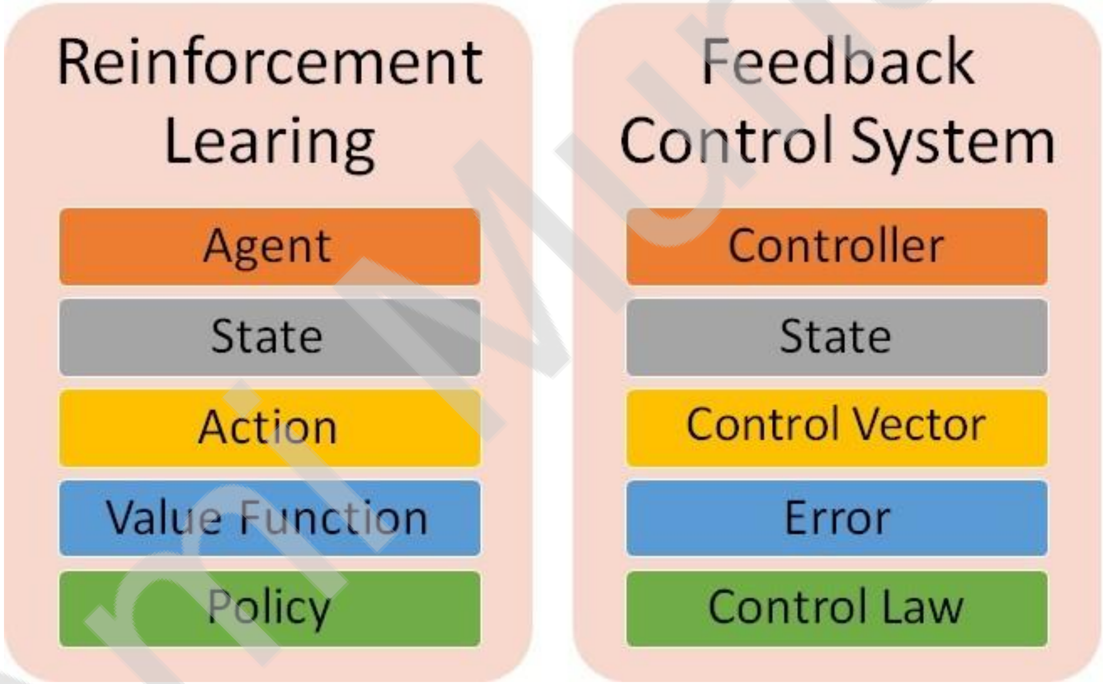
- Saving Money for Future Goals
- Completing a Degree for Career Advancement
- Physical Fitness and Exercise
- Learning a Musical Instrument
- Learning a New Language

Ref: https://fuskal.com/immediate-reinforcement/#Immediate_Reinforcement_and_Operant_Conditioning

Immediate RL

- Agent takes action A_t at time t to get an immediate reward R_t
- Agent needs to explore all the available actions to identify the near optimal action which may maximize the cumulative reward
- Once after enough exploration, near optimal action is identified, then the agent can explore that action
- Here we encounter explore/exploit dilemma or conflict
  - How much to explore and when to start exploiting
- Example of Immediate RL is Bandit Problem

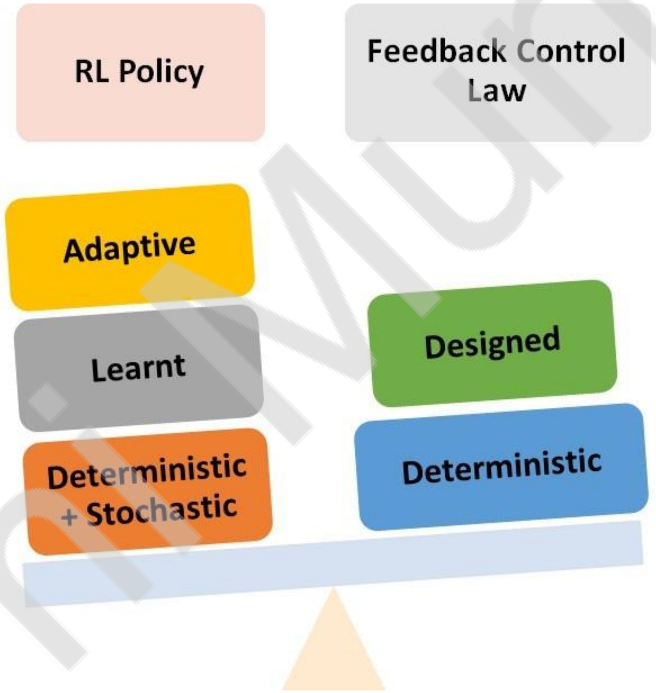
Can we compare Reinforcement Learning to traditional Control System??



Reinforcement Learing	Feedback Control System
Agent	Controller
State	State
Action	Control Vector
Value Function	Error
Policy	Control Law

Ref: https://www.linkedin.com/pulse/reinforcement-learning-analogies-from-feedback-siby-jose-plathottam/

Can we compare Reinforcement Learning to traditional Control System??



Ref: https://www.linkedin.com/pulse/reinforcement-learning-analogies-from-feedback-siby-jose-plathottam/

Can we compare Reinforcement Learning to traditional Control System??

Parameter	Traditional Control System	Reinforcement Learning
Model	Prebuilt mathematical model	No model as such Agent learns by -interacting with the environment - Receiving rewards for its actions
Adaptability	Limited	Very flexible, learns from complex and dynamic environment
Approach	Pre-programmed with rules – Specific results for the given action	Trial and Error

Can we compare Reinforcement Learning to traditional Control System??

Parameter	Traditional Control System	Reinforcement Learning
Time required	Faster implementation	More time required to train
When to use	When the system is deterministic and well understood	Complex, dynamic environments where mathematical modeling is difficult

Some terms

- Optimal Control
  - Branch of mathematical optimization
  - Designing a controller to maximize or minimize the objective function
  - Optimal control involves finding a control policy that optimizes a certain objective, typically the cumulative reward or cost over time
  - It deals with dynamical systems and aims to determine the best sequence of actions to take from any given state to achieve the optimal outcome.
- Dynamic Programming
  - Dynamic programming is a mathematical approach used to solve optimization problems by breaking them down into simpler subproblems
  - In the context of Markov Decision Processes (MDPs), DP methods are used to find optimal policies by solving the Bellman equations
  - Two primary DP methods are:
    - **Policy Iteration**: Alternates between evaluating a policy and improving it
    - **Value Iteration**: Iteratively updates the value function directly to find the optimal policy

Ref:https://www.youtube.com/watch?v=lRm6ma9NlhQ&list=PLZ_sI4f41TGvthD8dA7daahIbLV0yDW0w&index=1

Dynamic Programming