Cognixia

A Collabera LEARNING SOLUTIONS COMPANY

Reinforcement Learning

Agenda

1. Understanding Reinforcement Learning
2. Algorithms associated with RL
3. Q-Learning Model
4. Introduction to Artificial Intelligence

Reinforcement Learning

- A type of Machine Learning, and also a branch of Artificial Intelligence
- Learning of what to do and how to map situations to actions within a specific context
- End result is to maximize the performance

Why Reinforcement Learning ?

- Allows the machine or software agent to learn its behaviour based on feedback from the environment
- This behaviour can be learnt once and for all, or keep on adapting as time goes on
- If modelled with attention, some reinforcement learning algorithms can converge to the global optimum: ideal behaviour that maximises the reward

Elements of Reinforcement Learning

Agent: Intelligent programs

Environment: An external condition

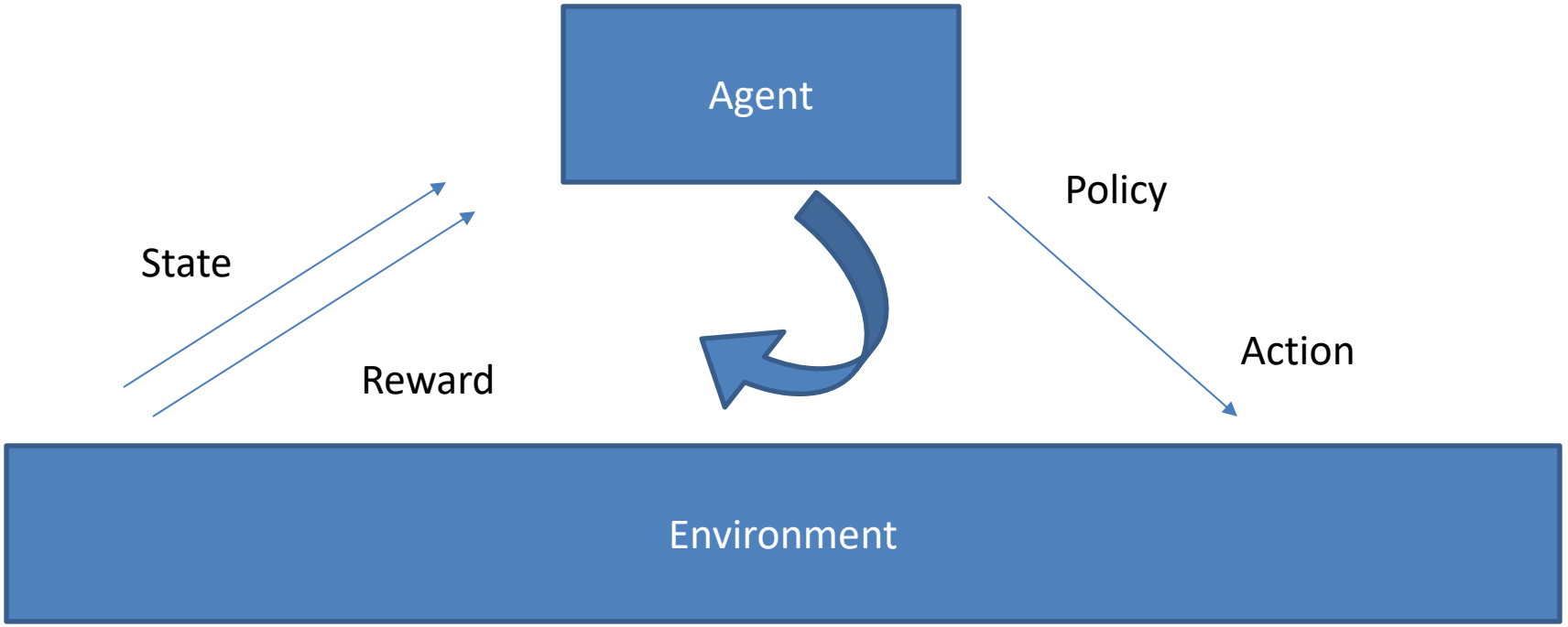
Policy: A mapping from state to actions defining agent's behaviour at a particular time

Reward Function: Could be +1 or any other value, indicating, what's good in an immediate sense

Value Function: Value of a state is the total amount of reward an agent can expect to accumulate over the future

Model of the environment: Used for planning and if know the current state and action then predict the resultant next state and next reward

Elements of Reinforcement Learning



Reinforcement Learning: Model Based

Action (A): All the possible moves that the agent can take.

State (S): Current situation returned by the environment.

Reward (R): An immediate return send back from the environment to evaluate the last action.

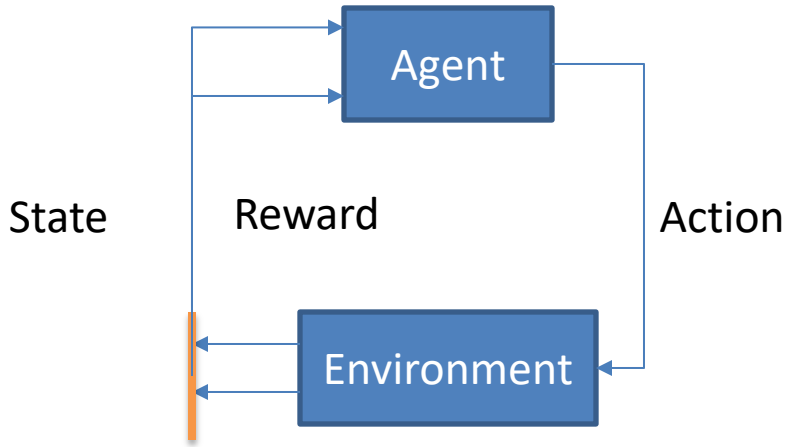
Policy (π): The strategy that the agent employs to determine next action based on the current state.

Value (V): The expected long-term return with discount, as opposed to the short-term reward R. Vπ(s) is defined as the expected long-term return of the current state sunder policy π.

Q-value or action-value (Q): Q-value is similar to Value, except that it takes an extra parameter, the current action a. Qπ(s, a) refers to the long-term return of the current state s, taking action a under policy π.

Agent – Environment Interface

- Agent and Environment interact at discrete time steps: t=0,1,2,...
- Agent observes state at step
- Produces action at step t
- Gets resulting reward
- Resulting next state

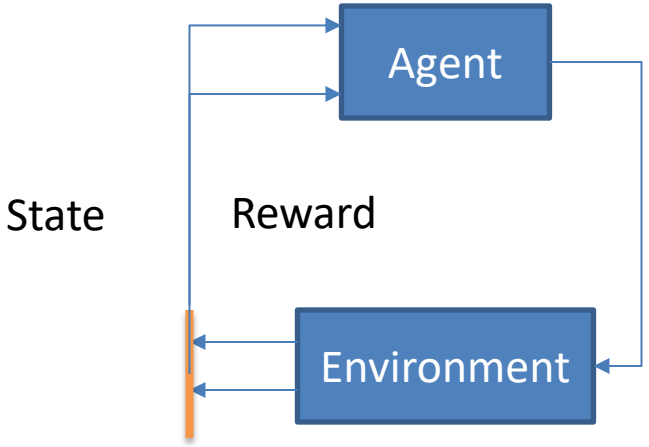


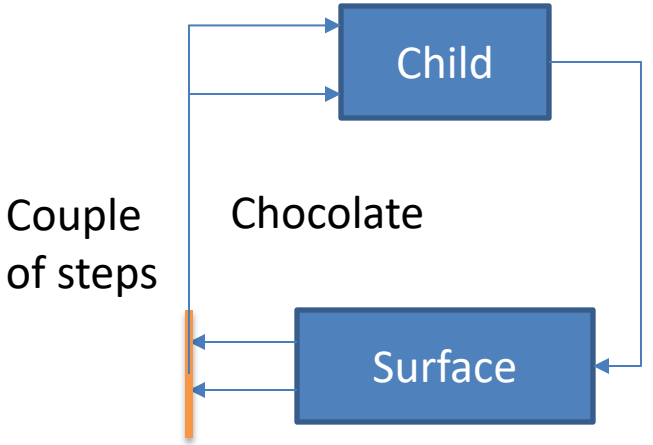
Real Life Scenario

- Consider a scenario of a child learning to walk
    - **Step 1:** He notices how you are walking and try to replicate the same
    - **Step 2:** He will understand the to start walking he needs to first stand up and he attempts to get up
    - **Step 3:** Standing up is easy but to remain still is another challenging task
    - **Step 4:** Now the real task to walk where he needs to take care of various factors i.e. balancing, which foot to put next etc

Formulating a RL Problem

- The child is an agent trying to manipulate the environment (which is the surface on which it walks)
- Taking actions (walking) and he/she tries to go from one state (each step he/she takes) to another
- The child gets a reward (may be a chocolate) when he accomplishes a sub module of the task (taking couple of steps)
- Will not receive any chocolate (negative reward) when he is not able to walk



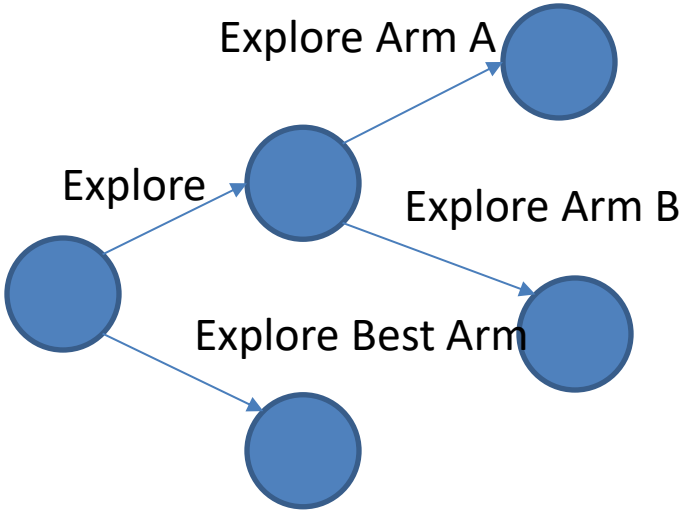


The Epsilon Greedy Algorithm

Takes whatever action seems best at the present moment

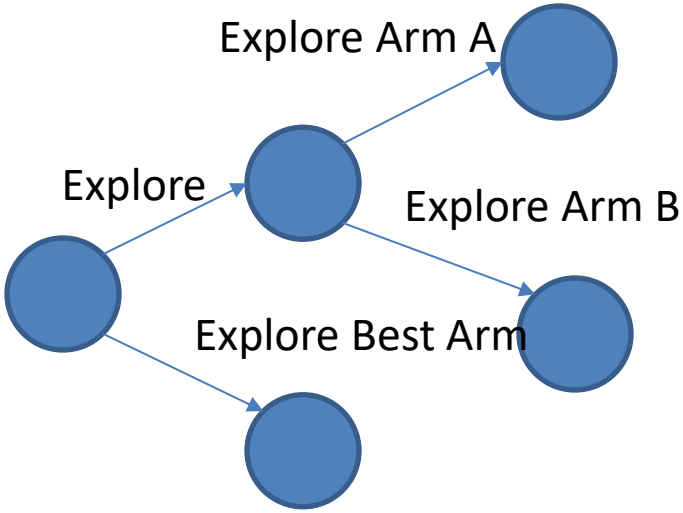
- Tries to be fair to the two opposite goals of exploration and exploitation by using a mechanism of flipping a coin

- Just like, if you flip a coin and it comes up tail you should explore for a moment but if it comes up head, you should exploit



The Epsilon Greedy Algorithm: Working

- With probability 1 - epsilon, the Epsilon-Greedy algorithm exploits the best known option
- With probability epsilon / 2, the Epsilon-Greedy algorithm explores the best known option
- With probability epsilon / 2, the Epsilon-Greedy algorithm explores the worst known option



Markov Decision Process (MDP)

- A mathematical framework for defining a solution in reinforcement learning
- It's a discrete time state transition system
- MDP's are non deterministic search problem

Components of MDP

- A terminal state (Optional)
- A start state
- A reward function R
- A set of states
- A set of actions
- A transition function T: probability that a from s leads to s' i.e. P (s'| s,a)

Markov in MDP's

- Markov generally means that given the present state, the future and past are independent
- For Markov decision processes, "Markov" means action outcomes depend only on the current state

MDP: Scenario

A Maze like problem:

- The agent lives in a grid
- Walls block the agent's path

Noisy movement: actions do not always go as planned

- 80% of the time, the action north takes the agent north
- 10% of the time, north takes the agent west, 10% East
- If there is a wall in the direction, 80% of the time, the agent would stay where it is

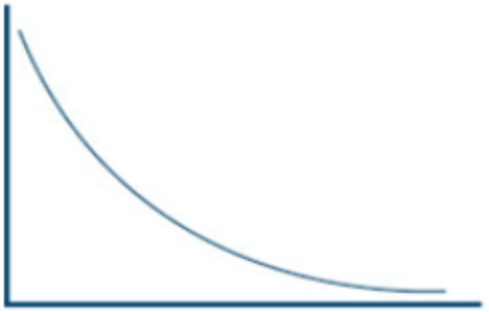
The agent receives reward each time step

- Small living rewards each step
- Big rewards come at the end

Goal: Maximize the sum of rewards

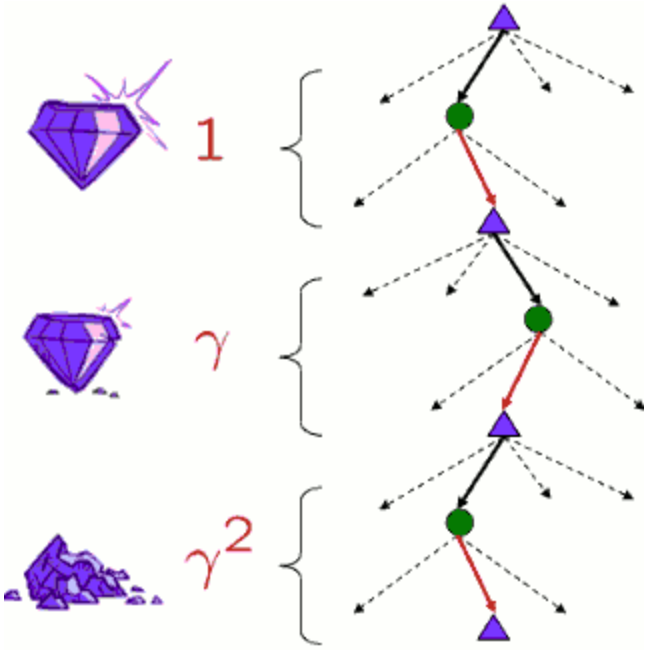
Discounting

- It's reasonable to maximize the sum of rewards
- It's also reasonable to prefer rewards now to rewards later
- Values of rewards decay exponentially



Discounting

- Each time we descend a level, we multiply in the discount once



Discounted Reward

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... + \gamma^{n-t} r_n$$

Why Discount?

- Sooner rewards probably do have higher utility that later rewards
- Also, helps our algorithms converge
- For example: Discount of 0.5
  - U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3
  - U([1,2,3]) < U([3,2,1])

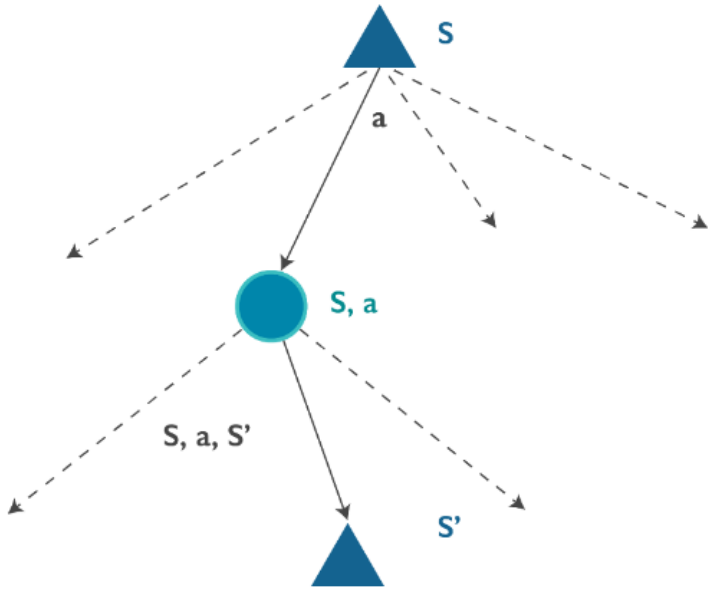
Redefining MDP's

Markov Decision Process:

Set of states S
Start state s0
Set of actions A
Transitions P(s' | s,a) or T(s,a,s')
Rewards R(s,a,s') and discount

MDP quantifies so far:

Policy: choice of action for each state
Utility: sum of rewards (discounted)



Reinforcement Learning: Model Based

• Each percept(e) is enough to determine the State(the state is accessible).

•The agent can decompose the Reward component from a percept.

•The agent task: to find a optimal policy, mapping states to actions, that maximize long-run measure of the reinforcement.

•Think of reinforcement as reward.

•Can be modeled as MDP model!

Reinforcement Learning: Model free Approach

Derive the optimal policy without explicitly learning the model

Useful when model is difficult to represent and/or learn

Reinforcement Learning: Algorithms

- Q-Learning
- Policy Iteration
- Deep Q Network (DQN)
- Deep Deterministic Policy Gradient (DDPG)

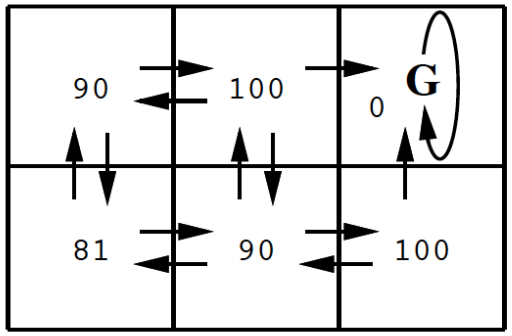
Q-Function

- One approach to RL is then to try to estimate V*(s).
- However, this approach requires you to know r(s,a) and delta(s,a).
- This is unrealistic in many real problems. What is the reward if a robot is exploring mars and decides to take a right turn?
- Fortunately we can circumvent this problem by exploring and experiencing how the world reacts to our actions. We need to *learn* r & delta.
- We want a function that directly learns good state-action pairs, i.e. what action should I take in this state. We call this Q(s,a).
- Given Q(s,a) it is now trivial to execute the optimal policy, *without knowing r(s,a) and delta(s,a).* We have:

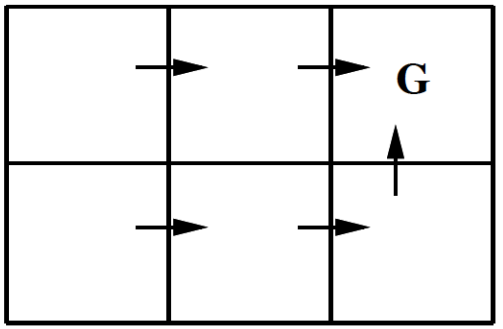
$$\pi^*(s) = \arg\max_a Q(s,a)$$

$$V^*(s) = \max_a Q(s,a)$$

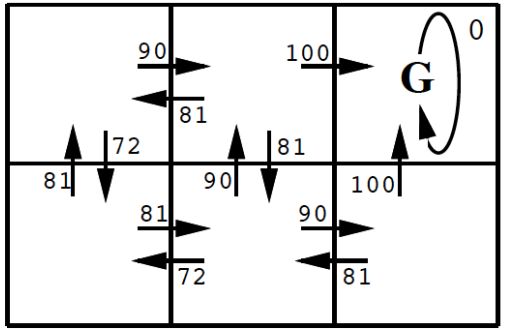
Q-Function: Example



$V^*(s)$ values



One optimal policy



$Q(s, a)$ values

Equation

$$\pi^*(s) = \arg\max_a Q(s,a)$$

$$V^*(s) = \max_a Q(s,a)$$

Q-Learning

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$
$$= r(s,a) + \gamma \max_{a'} Q(\delta(s,a),a')$$

- This still depends on r(s,a) and delta(s,a).
- However, imagine the robot is exploring its environment, trying new actions as it goes.
- At every step it receives some reward "r", and it observes the environment change into a new state s' for action a.
  How can we use these observations, (s,a,s',r) to learn a model?

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

Q-Learning

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

- This equation continually estimates Q at state s consistent with an estimate of Q at state s', one step in the future: temporal difference (TD) learning.
- Note that s' is closer to goal, and hence more "reliable", but still an estimate itself.
- Updating estimates based on other estimates is called bootstrapping.
- We do an update after each state-action pair. I.e., we are learning online!
- We are learning useful things about explored state-action pairs. These are typically most useful because they are likely to be encountered again.
- Under suitable conditions, these updates can actually be proved to converge to the real answer.

Artificial Intelligence: Introduction

Artificial intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans by having a learning mechanism. These computer systems (Robots) doesn't require constant dictation of commands to perform a task. On contrary, decision rules being taught to the machines..

Not only these machines take decision based upon those rules, also, a new data comes, they defines new rules based on the previous rules defined and the change in the pattern of the data. That's why this is known as artificial intelligence,

These systems works on the paradigm of human brains. Like neurons make the human brain network, we make artificial neural network in computers to make this as close as possible to humans.

Some of the Examples are as below:

- Speech recognition
- Learning
- Planning
- Problem solving

Artificial Intelligence: Terminologies

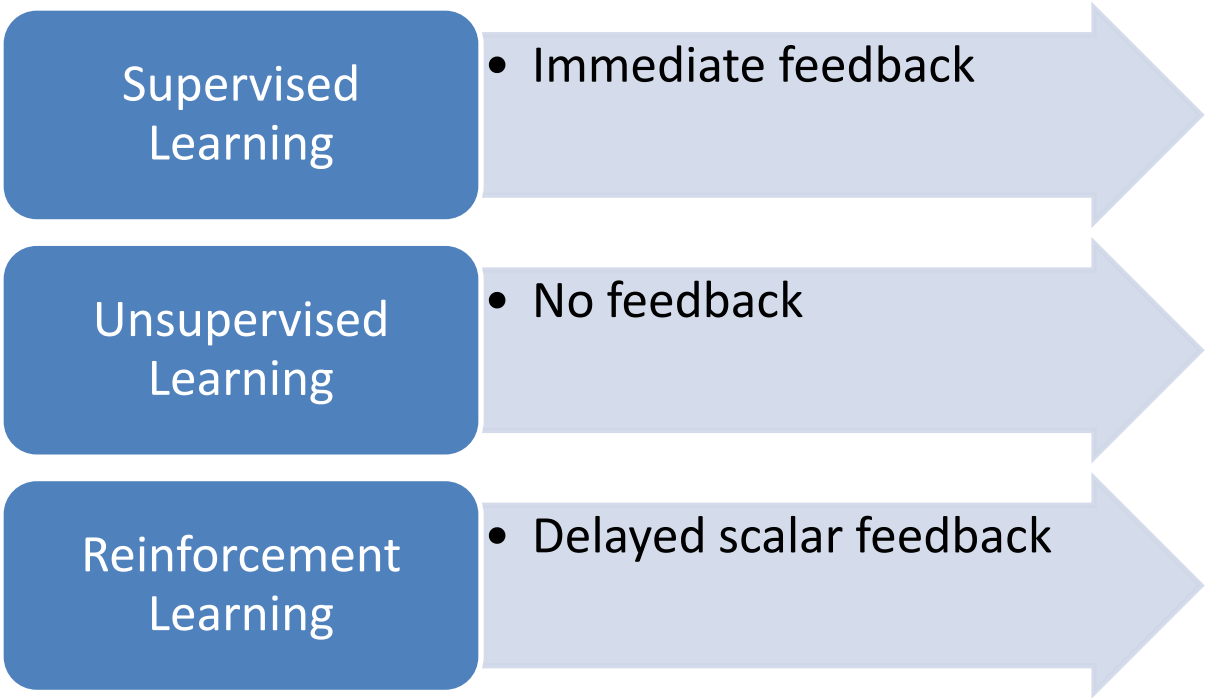
Autonomous Agent – A hardware (or software) based system that has the following properties:

Autonomy: able to operate without the direct intervention of humans or others
Social ability: able to interact with other agents and possibly humans
Reactivity: able to perceive their environment and respond to changes that occur in it

Agency – A particular system composed of intelligent agents, such as computers or robots, that cooperate in order to solve a problem.

Behavior Engineering – A methodology used to develop behavior- based autonomous agents.

Reinforcement Learning: Introduction



RL is a sense & act technique which combines Artificial intelligence and machine learning. It the most comprehensive problem setting.

Example:

- Scheduling planes to their destinations
- A traffic light system can measure the delay of cars and act accordingly (Red, Light, Yellow)


Cognixia
A Collabera LEARNING SOLUTIONS COMPANY
THANK YOU
www.cognixia.com