Module -7

Module:7	Reinforcement Learning	7 hours

Single State Case - K-Armed Bandit - Elements of Reinforcement Learning - Model Based Learning - Temporal Difference Learning - Generalization - Partially Observable States

Dr. A. Anitha,
Professor
Department of Software and Systems Engineering,
School of Computer Science Engineering and Information Systems,
Vellore Institute of Technology, 632014

Artificial Intelligence

Let us know…..

- 1.What is artificial intelligence??

  -Artificial Intelligence (AI) is when a computer algorithm does intelligent work.

2. What is machine learning?

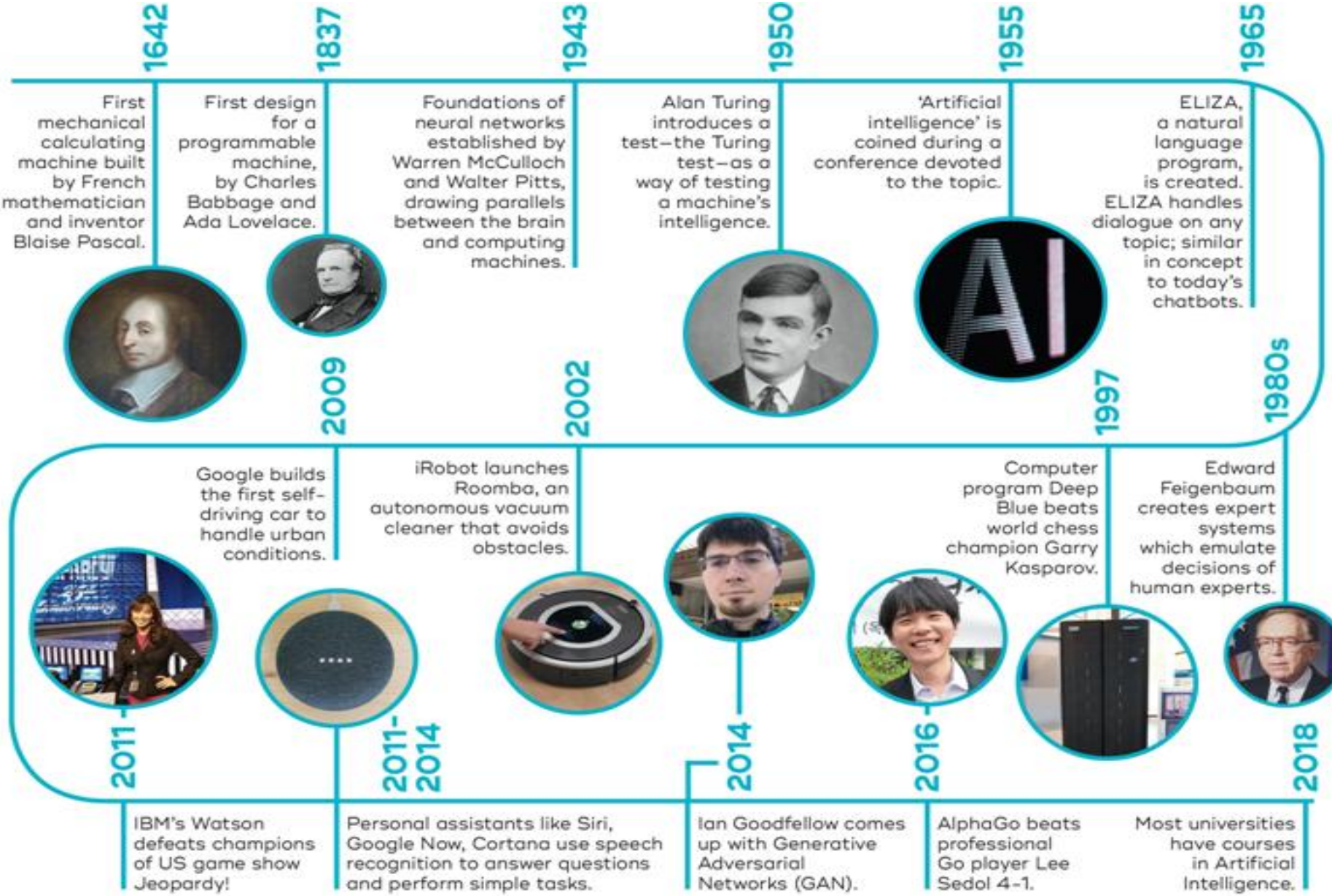
   - Machine Learning is a part of AI that learns from the data that also involves the information gathered from the previous experiences and allows the computer program to change its behavior accordingly

ARTIFICIAL INTELLIGENCE	MACHINE LEARNING
AI manages more comprehensive issues of automating a system. This computerization should be possible by utilizing any field such as image processing, cognitive science, neural systems, machine learning etc.	Machine Learning (ML) manages influencing user's machine to gain from the external environment. This external environment can be sensors, electronic segments, external storage gadgets and numerous other devices.
AI manages the making of machines, frameworks and different gadgets savvy by enabling them to think and do errands as all people generally do.	What ML does, depends on the user input or a query requested by the client, the framework checks whether it is available in the knowledge base or not. If it is available, it will restore the outcome to the user related with that query, however if it isn't stored initially, the machine will take in the user input and will enhance its knowledge base, to give a better value to the end user

- Artificial Intelligence is the superset of Machine Learning i.e. all the Machine Learning is Artificial Intelligence but not all the AI is Machine Learning.
- https://www.youtube.com/watch?v=6PL6Yr9RNjQ – Tamil
- https://www.youtube.com/watch?v=1PJsZwpwZ7k- Telugu

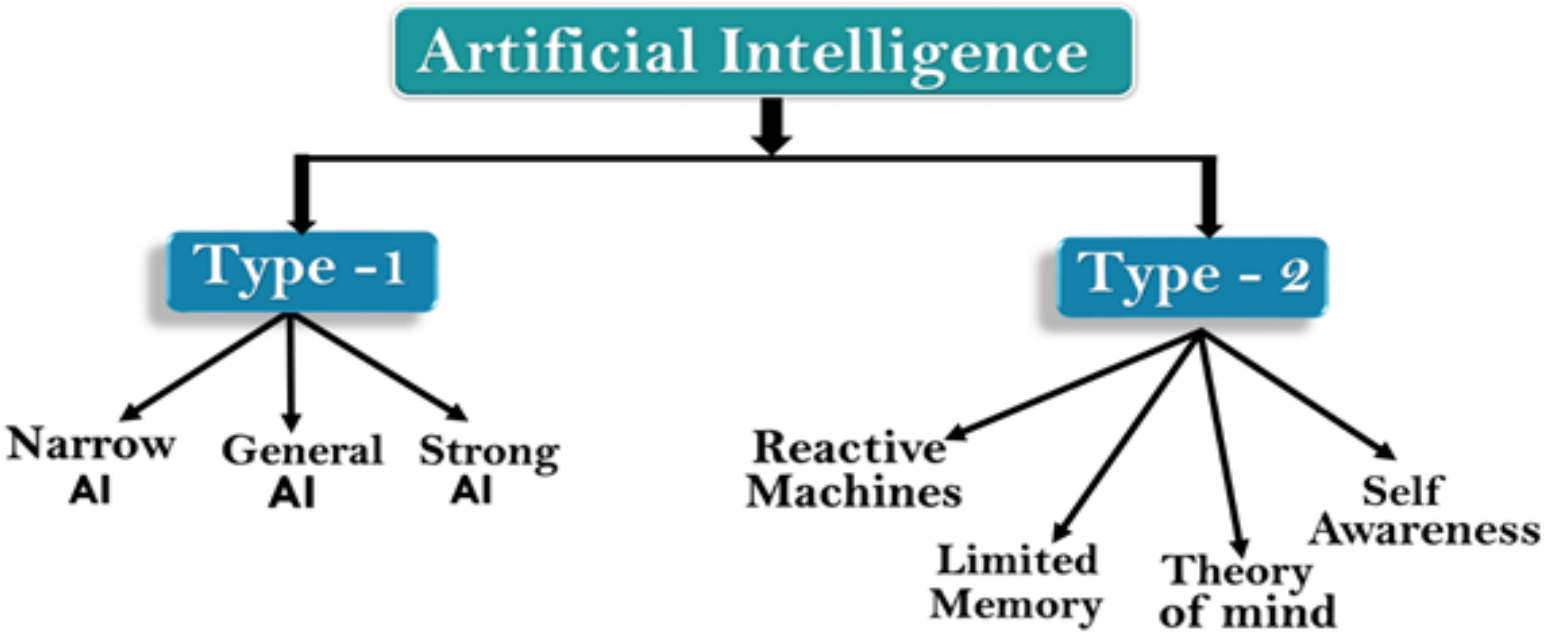
History of AI

- https://www.youtube.com/watch?v=xgypa5g0_js


1642
First mechanical calculating machine built by French mathematician and inventor Blaise Pascal.
1837
First design for a programmable machine, by Charles Babbage and Ada Lovelace.
1943
Foundations of neural networks established by Warren McCulloch and Walter Pitts, drawing parallels between the brain and computing machines.
1950
Alan Turing introduces a test—the Turing test—as a way of testing a machine's intelligence.
1955
'Artificial intelligence' is coined during a conference devoted to the topic.
1965
ELIZA, a natural language program, is created. ELIZA handles dialogue on any topic; similar in concept to today's chatbots.
AI
2009
2002
1997
1980s
Google builds the first self-driving car to handle urban conditions.
iRobot launches Roomba, an autonomous vacuum cleaner that avoids obstacles.
Computer program Deep Blue beats world chess champion Garry Kasparov.
Edward Feigenbaum creates expert systems which emulate decisions of human experts.
2011
2011-2014
2014
2016
2018
IBM's Watson defeats champions of US game show Jeopardy!
Personal assistants like Siri, Google Now, Cortana use speech recognition to answer questions and perform simple tasks.
Ian Goodfellow comes up with Generative Adversarial Networks (GAN).
AlphaGo beats professional Go player Lee Sedol 4-1.
Most universities have courses in Artificial Intelligence.

Types of AI

Artificial Intelligence can be divided in various types, there are mainly two types of main categorization which are based on capabilities and based on functionally of AI. Following is flow diagram which explain the types of AI.



AI type-1: Based on Capabilities

- **1. Weak AI or Narrow AI:**
- Narrow AI is a type of AI which is able to perform a dedicated task with intelligence. The most common and currently available AI is Narrow AI in the world of Artificial Intelligence.
- Narrow AI cannot perform beyond its field or limitations, as it is only trained for one specific task. Hence it is also termed as weak AI. Narrow AI can fail in unpredictable ways if it goes beyond its limits.
- Apple Siriis a good example of Narrow AI, but it operates with a limited pre-defined range of functions.
- IBM's Watson supercomputer also comes under Narrow AI, as it uses an Expert system approach combined with Machine learning and natural language processing.
- Some Examples of Narrow AI are playing chess, purchasing suggestions on e-commerce site, self-driving cars, speech recognition, and image recognition.

General AI:Based on Capabilities

- General AI is a type of intelligence which could perform any intellectual task with efficiency like a human.
- The idea behind the general AI to make such a system which could be smarter and think like a human by its own.
- Currently, there is no such system exist which could come under general AI and can perform any task as perfect as a human.
- The worldwide researchers are now focused on developing machines with General AI.
- As systems with general AI are still under research, and it will take lots of efforts and time to develop such systems.

Super AI:

- Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task better than human with cognitive properties. It is an outcome of general AI (Gen AI)
- Some key characteristics of strong AI include capability include the ability to think, to resolve the puzzle, make judgments, plan, learn, and communicate by its own.
- Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task.

Type-2: Based on functionality

Reactive Machines:

Purely reactive machines are the most basic types of Artificial Intelligence.

Such AI systems do not store memories or past experiences for future actions.

These machines only focus on current scenarios and react on it as per possible best action.

IBM's Deep Blue system is an example of reactive machines. https://www.youtube.com/watch?v=5l7-Rt7lVPQ

Google's AlphaGo is also an example of reactive machines. https://www.youtube.com/watch?v=8dMFJpEGNLQ

Theory of Mind & Self-Awareness

Theory of Mind:

- Theory of Mind AI should understand the human emotions, people, beliefs, and be able to interact socially like humans.
- This type of AI machines are still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.

Self-Awareness

- Self-awareness AI is the future of Artificial Intelligence. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.
- These machines will be smarter than human mind.
- Self-Awareness AI does not exist in reality still and it is a hypothetical concept.
- Ex: Yn iniya endhira

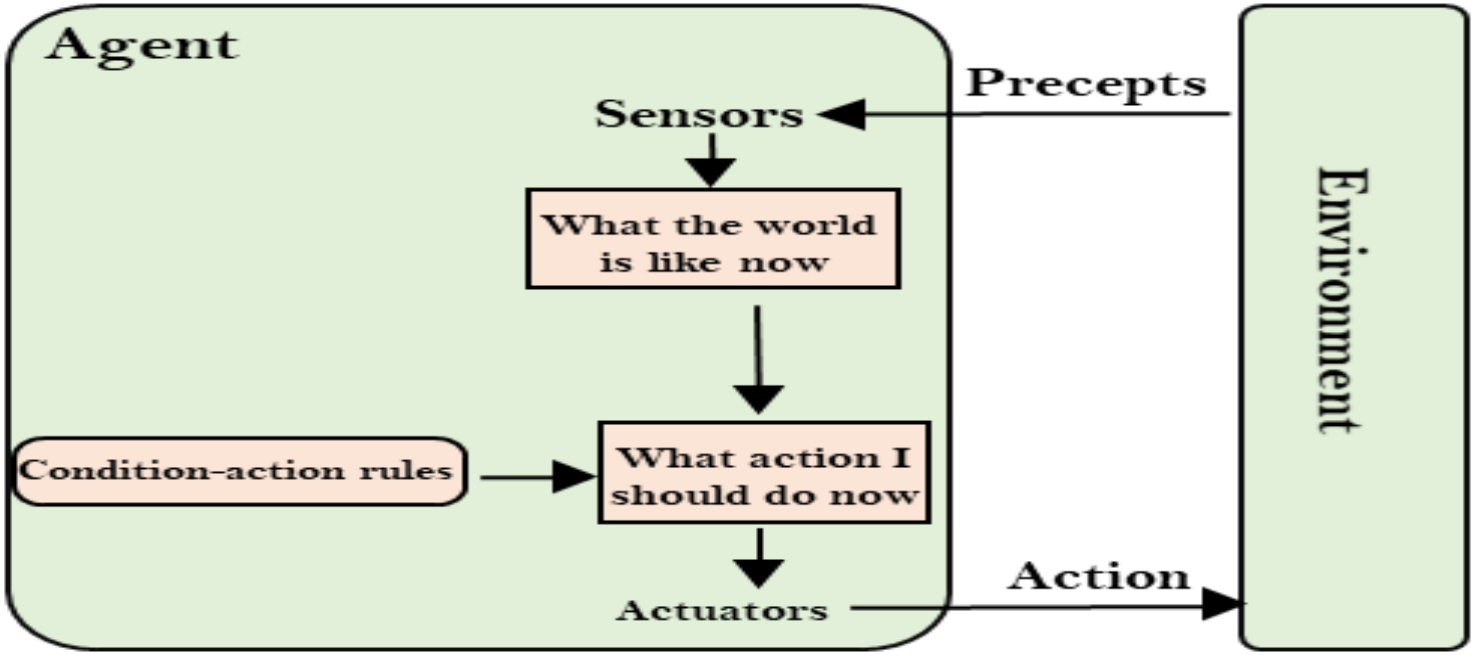
Agents

- An AI system is composed of an **agent and its environment**. The agents act in their environment. The environment may contain other agents. An agent is anything that can be viewed as :
- perceiving its environment through **sensors** and
- acting upon that environment through **actuators**
- (An **actuator** is a component of a machine that is responsible for moving and controlling a mechanism or system)

Types of AI Agents

- Agents can be grouped into five classes based on their degree of perceived intelligence and capability. All these agents can improve their performance and generate better action over the time. These are given below:
- Simple Reflex Agent
- Model-based reflex agent
- Goal-based agents
- Utility-based agent
- Learning agent

Simple Reflex Agent



1. The Simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history.
2. Fully observable.

Ex: chess games, Automatic switch on/off the appliances

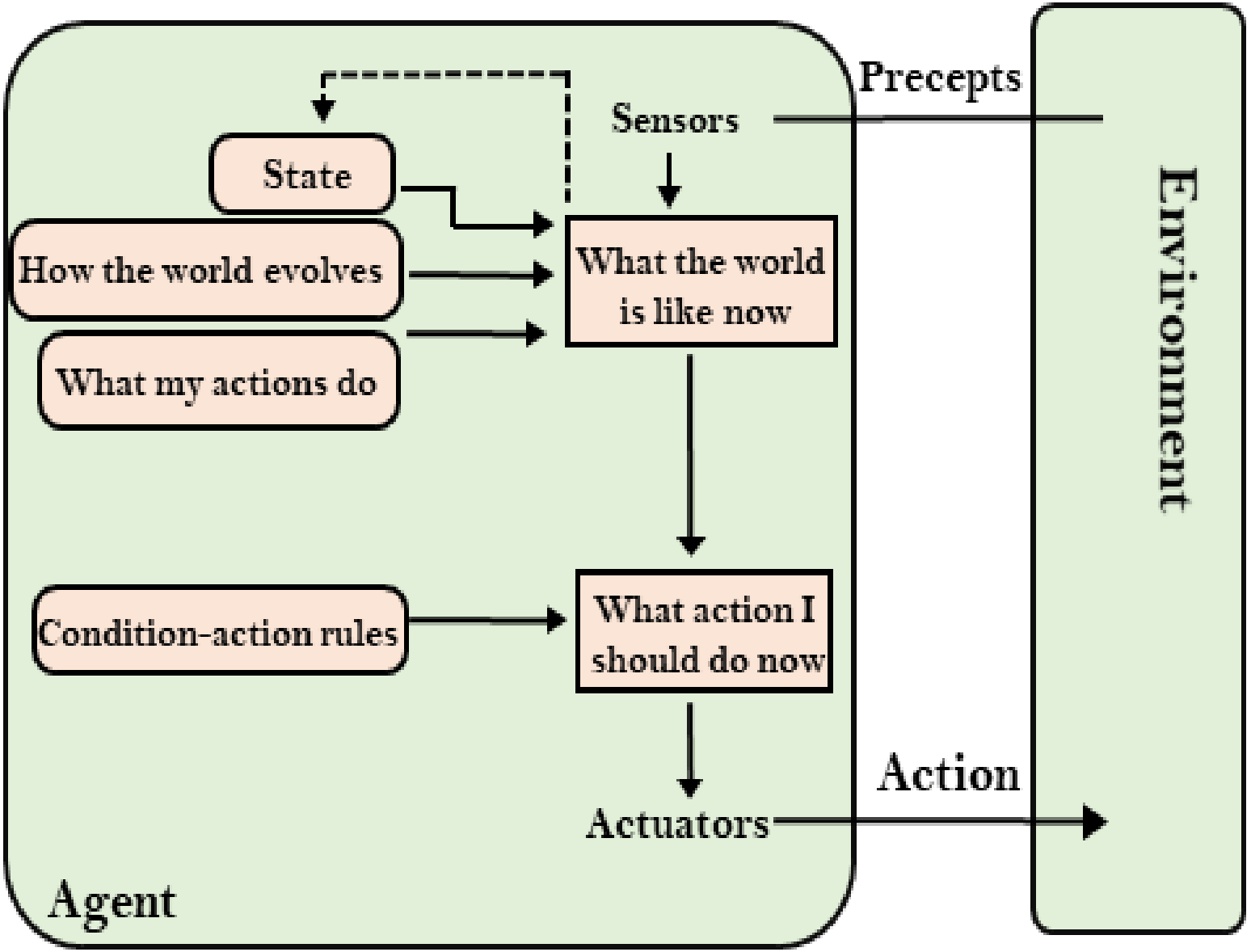
Limitations of Simple reflex agent

- They have very limited intelligence
- They do not have knowledge of non-perceptual parts of the current state
- Mostly too big to generate and to store.
- Not adaptive to changes in the environment.

Model-based reflex agent

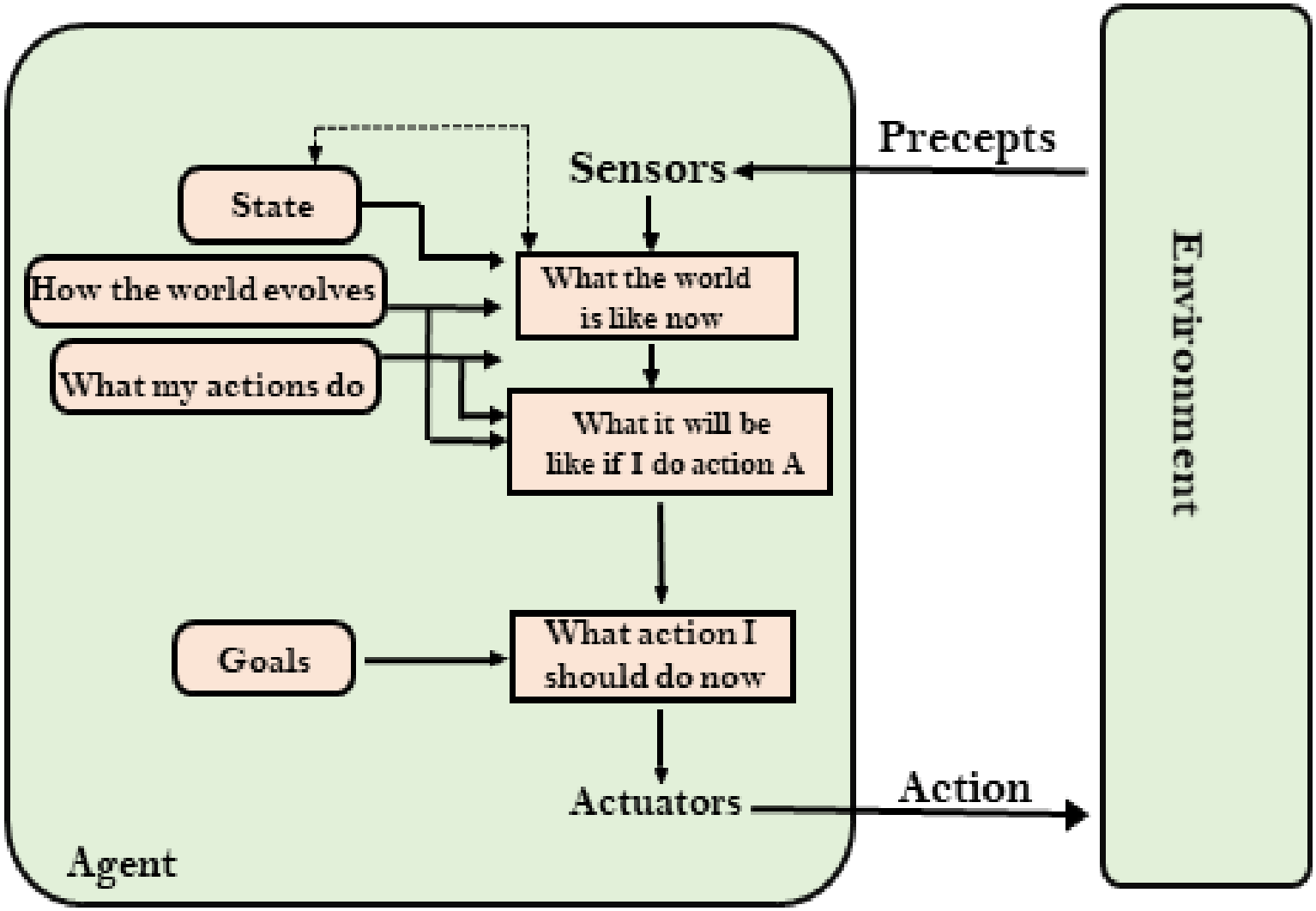
- The Model-based agent can work in a partially observable environment, and track the situation.
- A model-based agent has two important factors:
  - **Model:** It is knowledge about "how things happen in the world," so it is called a Model-based agent.
  - **Internal State:** It is a representation of the current state based on percept history.
- These agents have the model, "which is knowledge of the world" and based on the model they perform actions.
- Updating the agent state requires information about:
  - How the world evolves
  - How the agent's action affects the world.

  Ex: automatic cars


Precepts
Sensors
State
How the world evolves
What my actions do
What the world is like now
Environment
Condition-action rules
What action I should do now
Action
Actuators
Agent

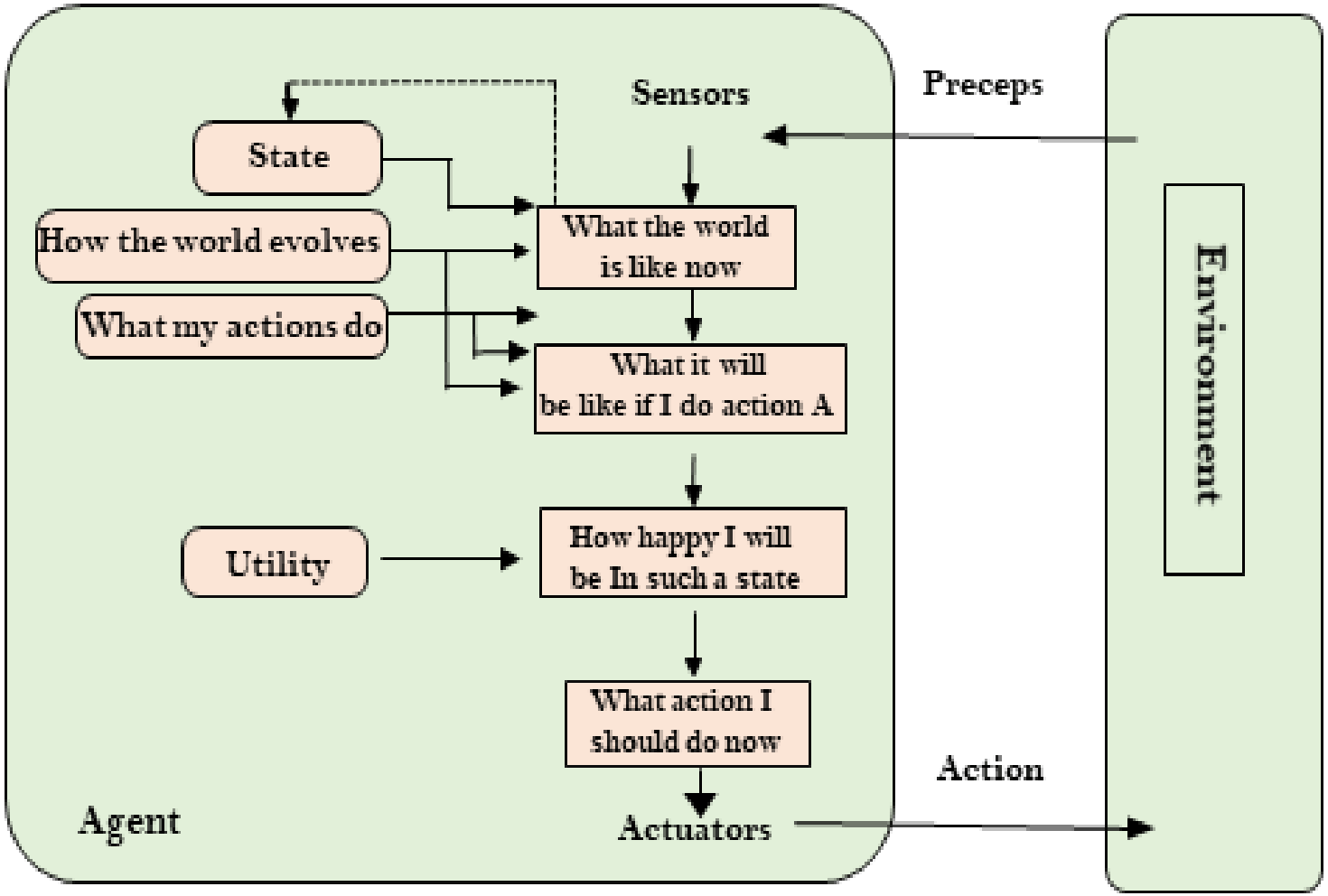
Goal-based agents

- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.
- The agent needs to know its goal which describes desirable situations.
- Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.
- They choose an action, so that they can achieve the goal.
- These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called searching and planning, which makes an agent proactive.
- Ex: amazon delivery system, Alibaba company..


Precepts
Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Actuators
Action
Agent
Environment

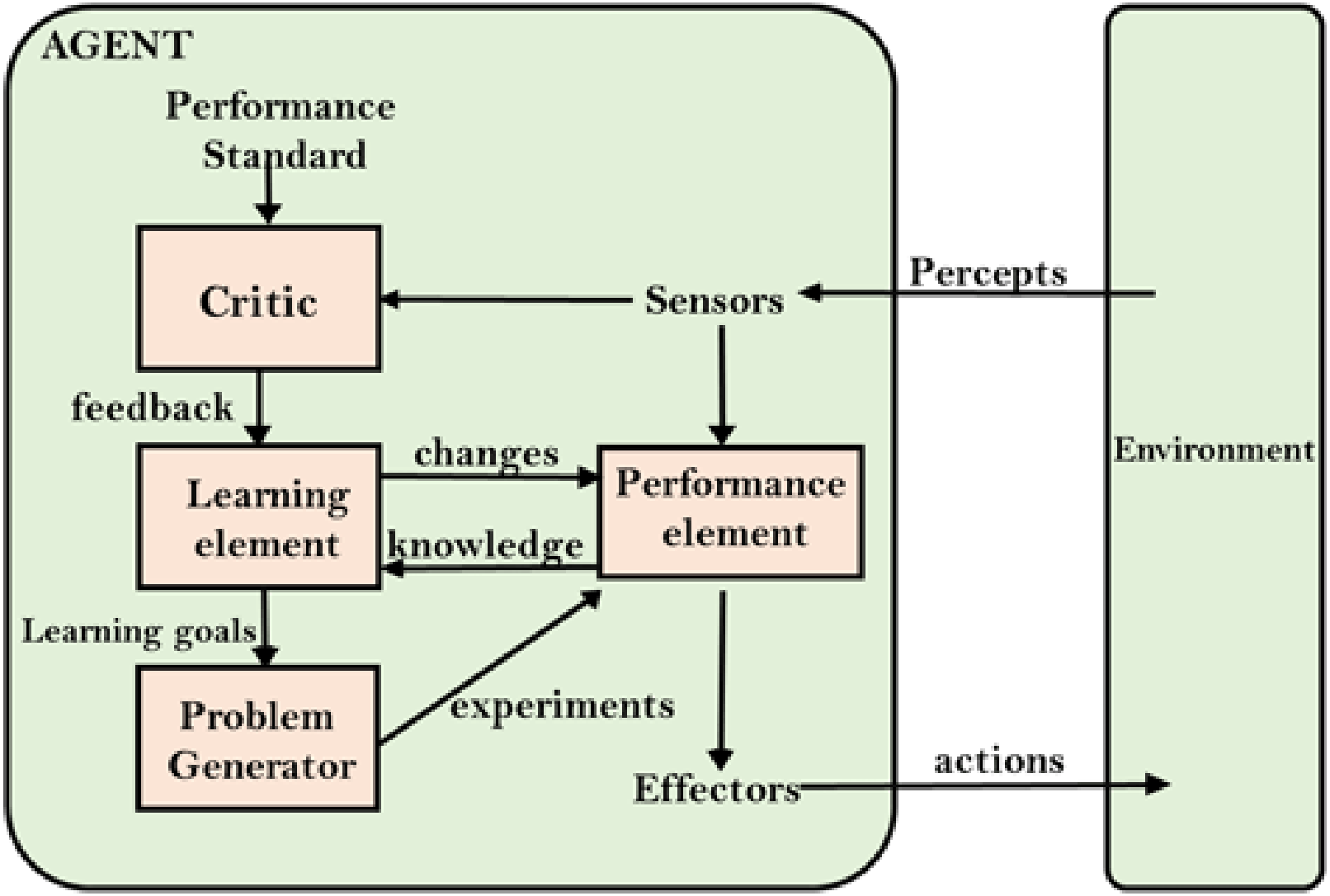
Utility-based agents

- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state.
- Utility-based agent act based not only goals but also the best way to achieve the goal.
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
- The utility function maps each state to a real number to check how efficiently each action achieves the goals.
- Ex: gps system, google maps – shortest path, traffic path, if accident then the alternate route is provided


Sensors
Preceps
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Utility
How happy I will be In such a state
What action I should do now
Agent
Actuators
Action
Environment

Learning Agents

- A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities.
- It starts to act with basic knowledge and then able to act and adapt automatically through learning.
- A learning agent has mainly four conceptual components, which are:
  - **Learning element:** It is responsible for making improvements by learning from environment
  - **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
  - **Performance element:** It is responsible for selecting external action
  - **Problem generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.


AGENT
Performance Standard
Critic
Sensors
Percepts
feedback
changes
Learning element
Performance element
knowledge
Environment
Learning goals
Problem Generator
experiments
Effectors
actions

Intelligent agent

- What is an Agent??
- An agent can be anything that perceive its environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**.

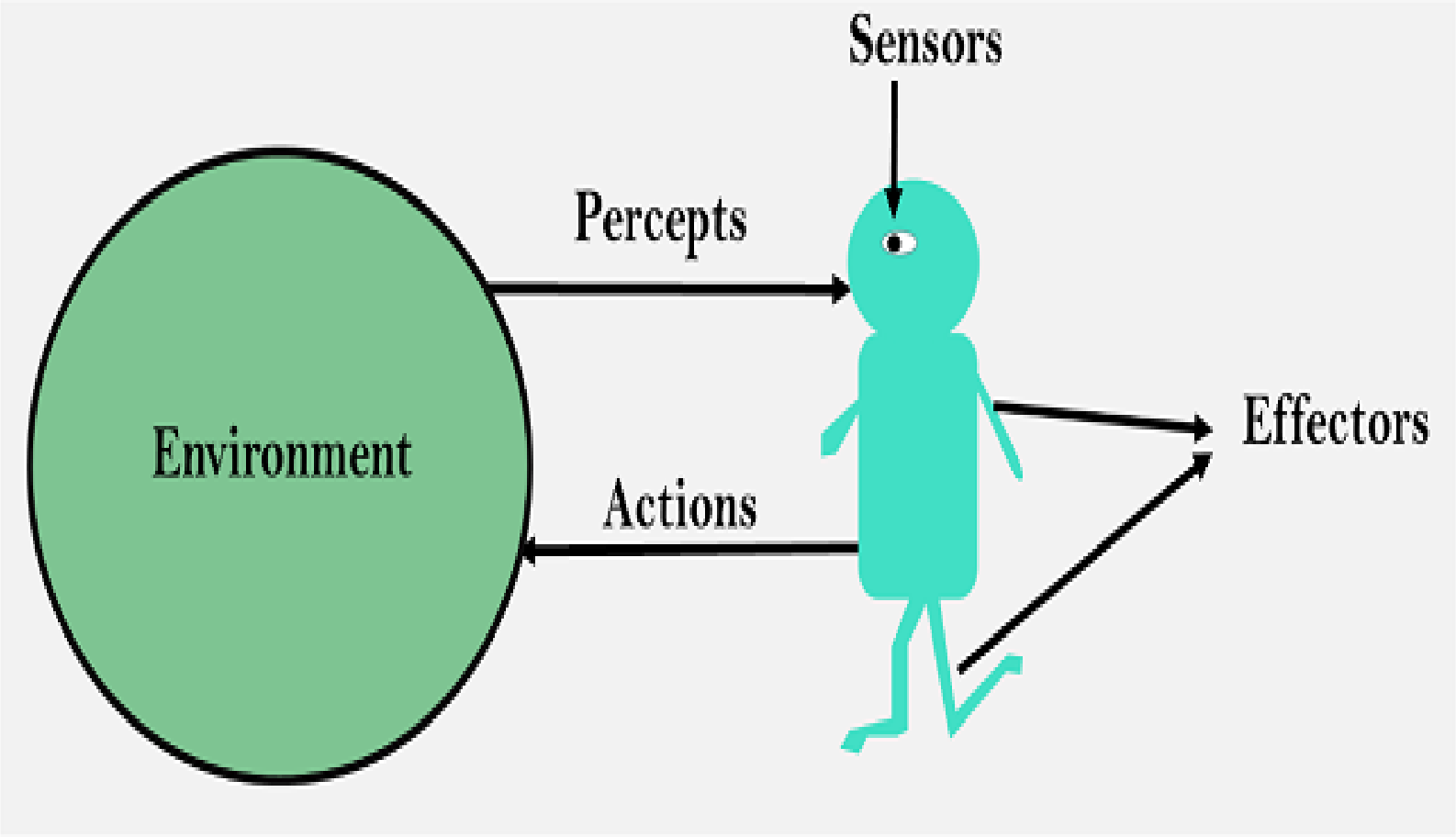
An agent can be:

- **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
- **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
- **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

Hence the world around us is full of agents such as thermostat, cellphone, camera, and even we are also agents.

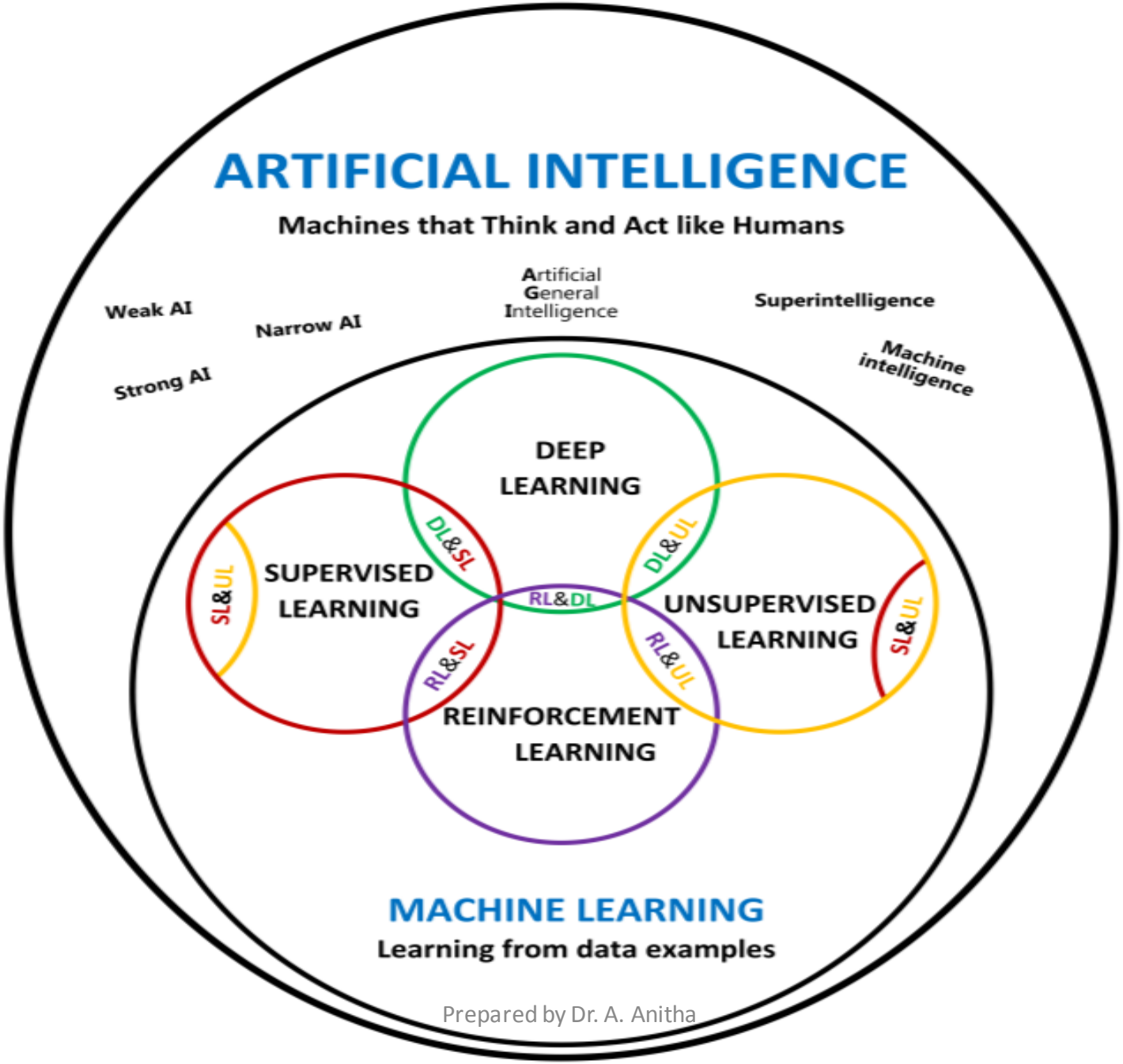
Sensor, Actuator and Effectors

- **Sensor:** Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.
- **Actuators:** Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc.
- **Effectors:** Effectors are the devices which affect the environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.


Sensors
Percepts
Environment
Actions
Effectors

Intelligent Agent

- An intelligent agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals.
- An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.
- Following are the main four rules for an AI agent:
- **Rule 1:** An AI agent must have the ability to perceive the environment.
- **Rule 2:** The observation must be used to make decisions.
- **Rule 3:** Decision should result in an action.
- **Rule 4:** The action taken by an AI agent must be a rational action.


ARTIFICIAL INTELLIGENCE
Machines that Think and Act like Humans
Weak AI
Narrow AI
Strong AI
Artificial General Intelligence
Superintelligence
Machine intelligence
DEEP LEARNING
SUPERVISED LEARNING
UNSUPERVISED LEARNING
REINFORCEMENT LEARNING
DL&SL
DL&UL
SL&UL
RL&DL
RL&SL
RL&UL
SL&UL
MACHINE LEARNING
Learning from data examples

Reinforcement Learning

Overview

- Introduction to Reinforcement Learning
- Finite Markov Decision Processes
- Temporal-Difference Learning (SARSA, Q-learning, Deep Q-Networks)
- Policy Gradient Methods (Finite Difference Policy Gradient, REINFORCE, Actor-Critic)
- Asynchronous Reinforcement Learning

Introduction to Reinforcement Learning

What is Reinforcement Learning?

- Learning from interaction with an environment to achieve some long-term goal that is related to the state of the environment
- The goal is defined by reward signal, which must be maximised
- Agent must be able to partially/fully sense the environment state and take actions to influence the environment state
- The state is typically described with a feature-vector

Exploration versus Exploitation

- We want a reinforcement learning agent to earn lots of reward
- The agent must prefer past actions that have been found to be effective at producing reward
- The agent must exploit (develop) what it already knows to obtain reward
- The agent must select untested actions to discover reward-producing actions
- The agent must **explore** actions to make better action selections in the future
- In short:
- **Exploration**: Trying random actions to discover better rewards.
- **Exploitation**: Choosing the best-known action to maximize rewards.

Reinforcement Learning Systems

- Reinforcement learning systems have 4 main elements:
  - Policy
  - Reward signal
  - Value function
  - Optional model of the environment

Policy

- A policy is a mapping from the perceived states of the environment to actions to be taken when in those states
- A reinforcement learning agent uses a policy to select actions given the current environment state

Reward Signal

- The reward signal defines the goal
- On each time step, the environment sends a single number called the reward to the reinforcement learning agent
- The agent's objective is to maximise the total reward that it receives over the long run
- The reward signal is used to alter the policy

Value Function (1)

- The reward signal indicates what is good in the short run while the value function indicates what is good in the long run
- The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting in that state
- Compute the value using the states that are likely to follow the current state and the rewards available in those states
- Future rewards may be time-discounted with a factor in the interval [0, 1]

Value Function (2)

- Use the values to make and evaluate decisions
- Action choices are made based on value judgements
- Prefer actions that bring about states of highest value instead of highest reward
- Rewards are given directly by the environment
- Values must continually be re-estimated from the sequence of observations that an agent makes over its lifetime

Model-free versus Model-based

- A model of the environment allows inferences to be made about how the environment will behave
- Example: Given a state and an action to be taken while in that state, the model could predict the next state and the next reward
- Models are used for planning, which means deciding on a course of action by considering possible future situations before they are experienced
- Model-based methods use models and planning. Think of this as modelling the dynamics p(s' | s, a)
- Model-free methods learn exclusively from trial-and-error (i.e. no modelling of the environment)
- This presentation focuses on model-free methods

1. Model-Free RL

- **Definition**: The agent does not have or use an explicit model of the environment's dynamics (i.e., the transition probabilities and reward function).
- **How It Works**: It learns directly from trial and error by interacting with the environment and updating its policies based on received rewards.
- **Pros**:
  - Simpler to implement.
  - Works well when the environment is too complex to model explicitly.
  - Often more sample-efficient when using deep learning (e.g., Deep Q-Networks).
- **Cons**:
  - Requires large amounts of data to learn effective policies.
  - Cannot plan ahead since it lacks knowledge of the environment's transitions.
- **Examples**:
  - **Q-Learning** (off-policy)
  - **Deep Q-Networks (DQN)**
  - **Policy Gradient Methods** (e.g., REINFORCE, PPO, A3C)

2. Model-Based RL

- **Definition**: The agent learns or is given a model of the environment's dynamics, which includes transition probabilities and the reward function.
- **How It Works**: It can use this model to simulate future actions and outcomes, allowing it to **plan** rather than rely solely on trial-and-error learning.
- **Pros**:
  - More sample-efficient since it can predict future outcomes instead of always interacting with the real environment.
  - Can perform **planning** using methods like Monte Carlo Tree Search (MCTS) or Dynamic Programming.
- **Cons**:
  - Requires an accurate model of the environment, which may be difficult or expensive to obtain.
  - Model inaccuracies can lead to poor decision-making.
- **Examples**:
  - **AlphaZero** (uses MCTS)
  - **Dyna-Q** (combines model-free Q-learning with a learned model)

Feature	Model-Free RL	Model-Based RL
Uses a model?	✗ No	✓ Yes
Sample efficiency?	✗ Less efficient (needs more data)	✓ More efficient (can plan ahead)
Computation time?	✓ Faster at runtime	✗ Slower due to planning
Planning ability?	✗ No (pure trial and error)	✓ Yes (can simulate outcomes)
Robustness?	✓ More robust (not affected by model errors)	✗ Dependent on model accuracy

On-policy versus Off-policy

- An on-policy agent learns only about the policy that it is executing
- An off-policy agent learns about a policy or policies different from the one that it is executing

Credit Assignment Problem

- Given a sequence of states and actions, and the final sum of time-discounted future rewards, how do we infer which actions were effective at producing lots of reward and which actions were not effective?
- How do we assign credit for the observed rewards given a sequence of actions over time?
- Every reinforcement learning algorithm must address this problem

Reward Design

- We need rewards to guide the agent to achieve its goal
- Option 1: Hand-designed reward functions
- This is a black art
- Option 2: Learn rewards from demonstrations
- Instead of having a human expert tune a system to achieve the desired behaviour, the expert can demonstrate desired behaviour and the robot can tune itself to match the demonstration

What is Deep Reinforcement Learning?

- Deep reinforcement learning is standard reinforcement learning where a deep neural network is used to approximate either a policy or a value function
- Deep neural networks require lots of real/simulated interaction with the environment to learn
- Lots of trials/interactions is possible in simulated environments
- We can easily parallelise the trials/interaction in simulated environments
- We cannot do this with robotics (no simulations) because action execution takes time, accidents/failures are expensive and there are safety concerns

Summarizing RL

1. Key Components of Reinforcement Learning

1. **Agent**: The decision-maker (e.g., a robot, game-playing AI, self-driving car).
2. **Environment**: The world in which the agent operates (e.g., a chessboard, a robotic simulation, or a city for an autonomous car).
3. **State** (s): The current situation of the agent in the environment.
4. **Action** (a): The choices the agent can make.
5. **Reward** (R): The feedback signal indicating how good or bad an action was.
6. **Policy** (π): The agent's strategy for choosing actions based on states.
7. **Value Function** ($V(s)$): The expected reward of being in a state and following the policy.
8. **Q-Function** ($Q(s, a)$): The expected reward of taking action a in state s and following the policy thereafter.

2. How Reinforcement Learning Works

The agent follows this cycle:

1. **Observe** the current state (s).
2. **Choose an action** (a) based on a policy (π).
3. **Execute the action**, receiving a **reward** (R) and moving to the next **state** (s').
4. **Update the policy** based on the reward.
5. **Repeat** until the agent learns the best way to act.

The agent learns through **trial and error**, improving its policy over time to maximize rewards.

Reinforcement Learning Algorithms

Category	Algorithm	Description
Value-Based	Q-Learning	Learns the Q-values for state-action pairs.
	Deep Q-Networks (DQN)	Uses deep learning to approximate Q-values.
Policy-Based	REINFORCE	Directly optimizes the policy.
	Proximal Policy Optimization (PPO)	A stable and widely used policy optimization method.
Actor-Critic	A3C (Asynchronous Advantage Actor-Critic)	Uses both value and policy learning.
Model-Based	AlphaGo/AlphaZero	Uses Monte Carlo Tree Search (MCTS) with a learned model.

Example: Training an AI to Play a Game

Scenario: A robot in a **grid world** must reach a goal while avoiding obstacles.

- States: Grid positions (e.g., (0,0), (1,2)).
- Actions: {Up, Down, Left, Right}.
- Rewards:
  - $+10$ for reaching the goal.
  - -1 for hitting a wall.
  - 0 otherwise.