chapter-2

Intelligent Agents

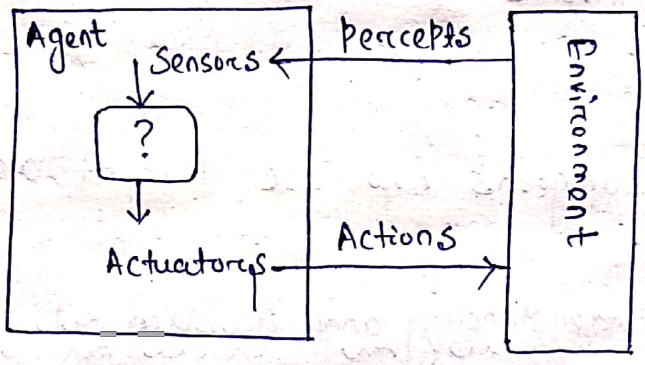
Agents

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuaters. for example:

(1). A human agent has eyes, ears and other organs for sensors and hands, legs, mouth and other body parts for actuators.

(2). A roboctic agent might have cameras and infrared range finders for sensors and various motors for actuators.

(3). A software agent receives keystrokes, file contents and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files and sending network packets.



We use the term percept to refer to the agents perceptual inputs at any given instant. An agents percept sequence is the complete history of everything the agent has ever perceived.

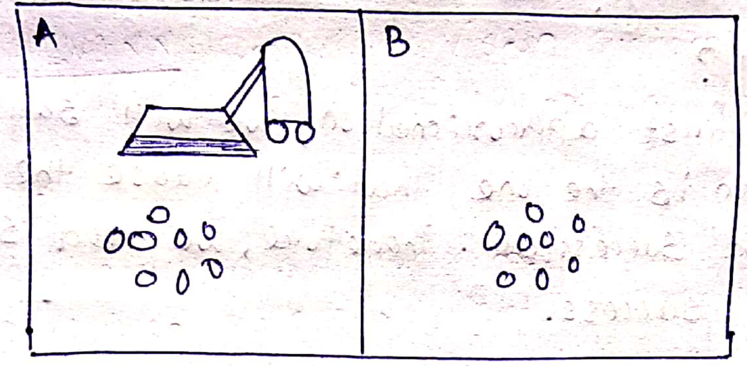
In general, an agents choice of action at any given instant can depend on entire percept sequence observed till date. If we can spacify agent's choice of action for every possible percept sequence then we have said more or less everything there is to say about the agent.

Scanned with CamScanner

Mathematically speaking, we say that an agents behaviour is described by the agent function that maps any given percept sequence to an action.

Illustration

let us take an example of vacuum cleaner as shown in the below figure.



Percept sequence	Action
[A, clean]	Right
[A, dirty]	Suck
[B, clean]	left
[B, dirty]	suck
[A, clean], [A, clean]	Right
[A, clean], [A, dirty]	Suck
⋮	⋮
[A, clean], [A, clean], [A, clean]	Right
[A, clean], [A, clean], [A, dirty]	Suck
⋮	⋮

The above tabulation depicts the partial tabulation of a simple agent function for the vacuum cleaner world.

Scanned with CamScanner

Rational Agent

A rational agent is one that does the right thing - conceptually speaking, every entry in the table for the agent function is filled out correctly.

Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?

As a first approximation, we will say that the right action is the one that will cause the agent to be most successfull. Therefore, we need some way to measure success.

Performance measures

A performance measure embodies the criterion for success of an agent's behaviour. when an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, the agent has performed well.

obviously, there is not one fixed measure suitable for all agents. As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.

Rationality

Rationality depends on four things.

(1). The performance measure that defines the criterion of success.

(2). Agent's Prior knowledge about the environment.
(3). The actions that the agent can perform.
(4). The agent's percept sequence till date.

This leads to a definition of rational agent:
For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever buildtin knowledge the agent has.

Omniscience, learning and autonomy

An omnicient agent knows the actual outcome of its action and can act accordingly. But omniscience is impossible in reality.

Our definition of rationality does not require omniscience. because the rational choice depends only on the percept sequence till date.

A rational agent not only gather information but also learn as much as possible from what it perceives. The agent's initial configuration could reflect some prior knowledge of the environment but as the agent gains experience this may be modified and augmented. A rational agent should be autonomous — it should learn what it can to compensate for partial or incorrect prior knowledge. We say that agent lacks autonomy if it relies on the prior knowledge of its designer, Rather than on its own percepts.

Specifying the task environment

A task environment specification includes the performance measure, the external environment, the actuators and the sensors (PEAS).

Scanned with CamScanner

In designing an agent, the first step must always be to specify the task environment as fully as possible. For example, PEAS description of the task environment for an automated taxi is as follows.

1. Agent Type — Taxi Driver.
2. Performance measure — safe, fast, legal, comfortable trip, maximize profit.
3. Environment — Roads, other traffic, padestrians, customers.
4. Actuators — steering, accelerator, Break, signal, Horn, Display, ~~sensors~~.
5. Sensors — Cameras, Sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard.

Properties of task environments

The range of task environments that might arise in AI is obviously vast. However, we can identify a fairly small number of dimensions along which task environments can be categorized.

1. Fully observable vs Partially observable

If an agent's sensors give it access to the complete state of the environment at each point of time, then we say that the task environment is fully observable.

An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data.

Scanned with CamScanner

2. Deterministic vs ~~stochasti~~ Stochastic

If the next state of the environment is completely determined by the current state and action executed by the agent, then we say the environment is deterministic: otherwise, it is stochastic.

3. Episodic vs Sequential

In an episodic task environment, the agent's experience is divided into atomic episodes. Each episode consists of agent perceiving and then performing a single action. Here, the choice of action in each episode depends only on the episode itself. Many classification tasks are episodic.

In sequential environment, the current decision could affect all future decisions. For example playing chess, taxi driving etc.

Episodic environment are ~~more~~ much simpler than sequential environment because the agent does not need to think ahead.

4. Static vs Dynamic

If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent: otherwise, it is static.

Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action.

5. Discrete vs Continuous

The discrete/continuous distinction can be applied to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.

6. Single agent vs Multi agent

In a single agent environment, only one agent interact with the environment whereas in a multiagent environment more than one agent act upon the environment.

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agent
Crossword puzzle	Fully	Deterministic	sequential	Static	Discrete	Single
chess with a clock	fully	strategic	sequential	Semi	Discrete	Multi
Poker	Partially	strategic	sequential	Static	Discrete	Multi
Backgammon	Fully	stochastic	sequential	Static	Discrete	Multi
Taxi driving	Partially	stochastic	sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partially	stochastic	sequential	Dynamic	continuous	Single
Image analysis	Fully	Deterministic	Episodic	semi	Continuous	Single
Part-picking robot	Partially	Stochastic	episodic	Dynamic	Continuous	Single
Refinary controllers	Partially	Stochastic	sequential	Dynamic	Continuous	Single
Interactive English Tutors	Partially	Stochastic	sequential	Dynamic	Discrete	Multi

The structure of agents

The job of AI is to design the agent Program that implements the agent function mapping percepts to actions. We assume this Program will run on some sort of computing device with physical sensors and actuators — we call this the architecture.

Agent = Architecture + Program

obviously, the program we choose has to be one that is appropriate for the architecture. If the program is going to recommend actions like walk, the architecture had better have legs.

Agent Programs

An agent Program take the current Percept as the input from the sensors and return an action to the actuators. The agent program takes just the current percept as the input because nothing more is available from the environment. If the agent's actions depend on the entire Percept sequence, the agent will have to remember the percepts.

The table driven agent Program

```
Function TABLE-DRIVEN-AGENT (Percept) returns an
                                         action
    Static: Percepts, a sequence, initially empty
            table, a table of actions, indexed by percept
            Sequences, initially fully specified

    append percept to the end of percepts
    action ← LOOKUP( percepts, table)
    return action
```

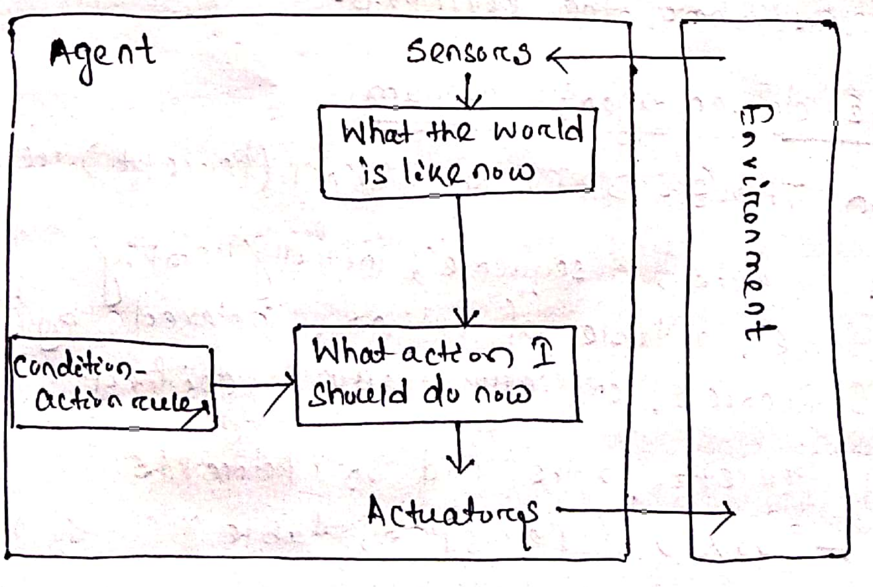
Types of agents Program

Four basic kinds of agent Program that embody the principles ~~of~~ underlying almost all intelligent System are

(i) Simple reflex agents
(ii) Model based reflex agents
(iii) goal based agents
(iv) utility based agents.

i) Simple reflex agents

This is a simplest kind of agent. These agents select actions on the basis of current percept, ignoring the rest of the perceptent history.
For example, the vacuum cleaner agent whose agent function is a simple reflex agent because its decision is based on only on the current location and on whether that contains dirt or not.



A simple reflex agent Program

```
function SIMPLE-REFLEX-AGENT(Percept) returns an action
Static : rules , a set of condition-action rules
State ← INTERPRET - INPUT (Percept)
rule ← RULE_MATCH (State, rules )
action ← RULE - ACTION [rule]
return action
```

Simple reflex agents have the admirable property of being simple but they turn out to be of very limited intelligence. The agent will work only if the correct decision can be made on the basis of the current percept only - i.e., only if the environment is fully observable. Even a little bit of unobservability can cause serious trouble.

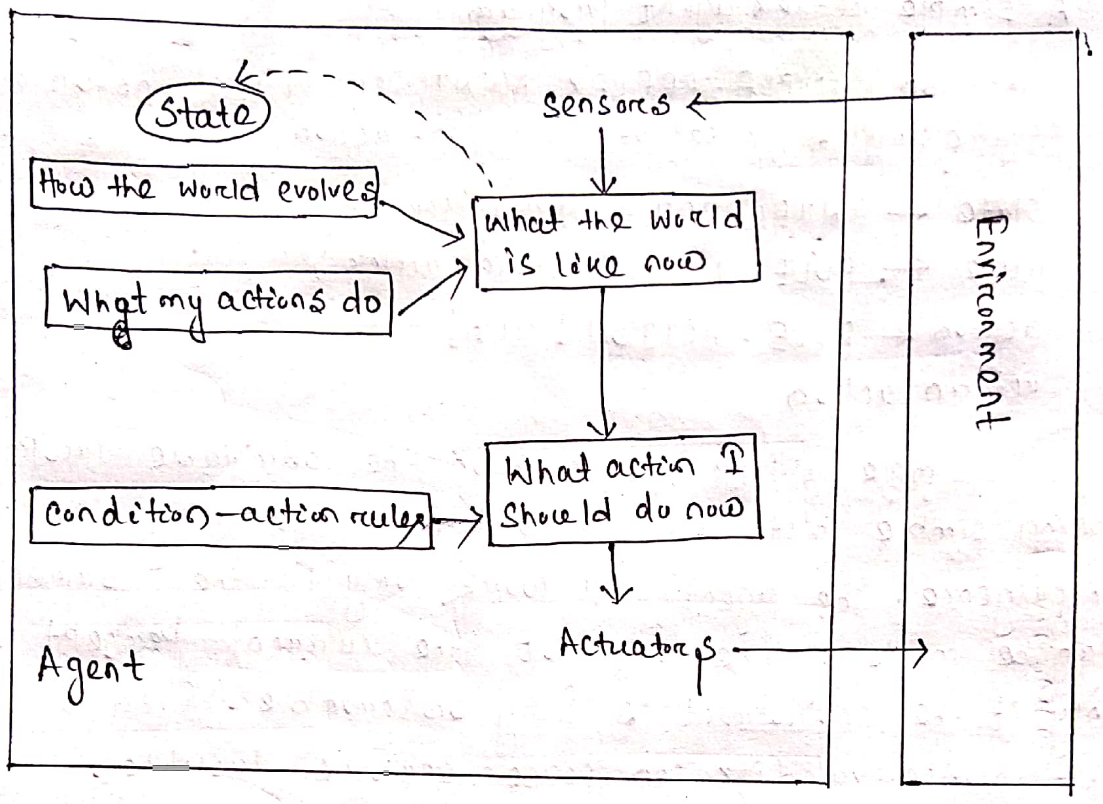
(ii) Model based reflex agents

The most effective way to handle Partial observability is for the agent to keep track of the part of the world it can not see now. i.e, the agent should maintain some sort of internal state that depends on the percept history and there by reflects at least some of the unobserved aspects of the current state.

Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent Program.

1. First we need some information about how the world evolves independently of the agent? For example overtaking
2. a car generally will be closer behind than it was a moment ago.
2. We need some information about how the agent's own action affect the world?

Scanned with CamScanner



Model based reflex agent Program

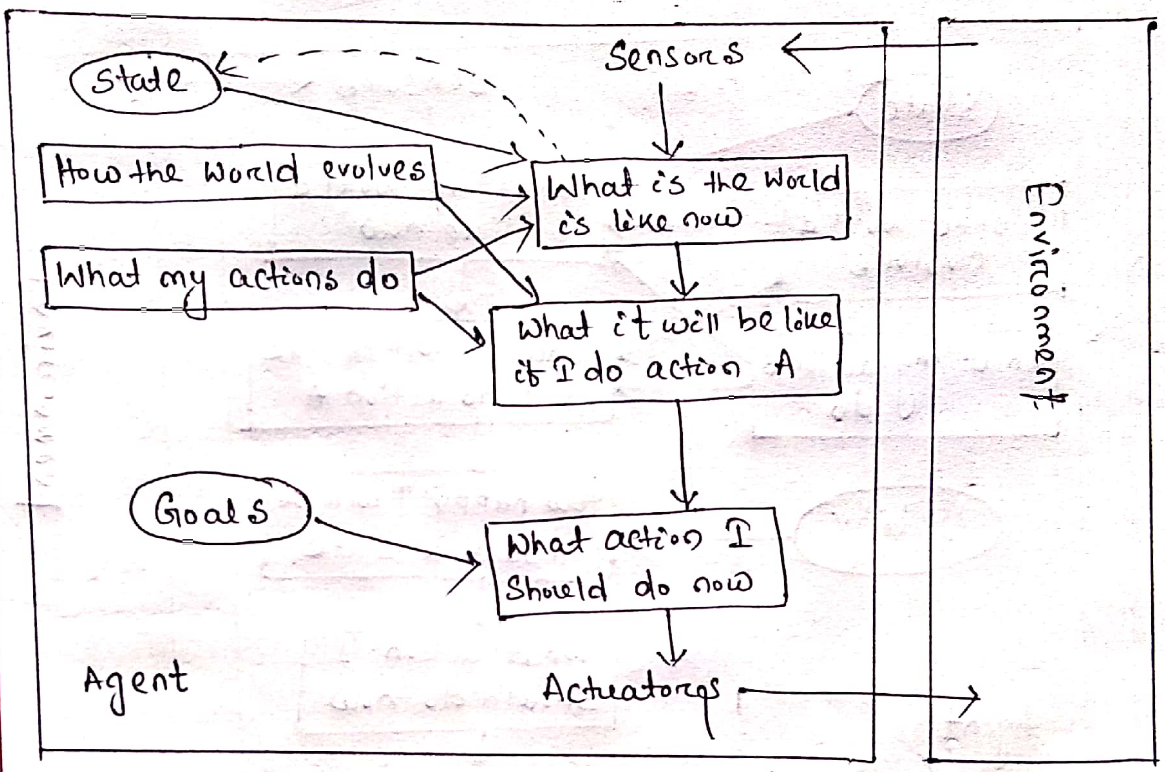
```
function REFLEX-AGENT-WITH-STATE(Percept) returns an action
Static: state, a description of the current world state
        rules, a set of Condition-action rules
        action, the most recent action, initially none

state ← UPDATE-STATE(State, action, Percept)
rule ← RULE-MATCH (State, rules)
action ← RULE-ACTION [rule]
return action
```

(iii) Goal-based agents

In goal-based agents, the agent needs some sort of goal information that describes situations that are desirable along with the current state description.

Scanned with CamScanner

The agent Program can combine the information about the results of Possible actions in order to choose actions that achieve the goal. Sometimes goal-based action selection is straight forward, when goal satisfaction results immediately from a single action.



State
Sensors
How the World evolves
What is the World is like now
What my actions do
What it will be like if I do action A
Goals
What action I Should do now
Agent
Actuators
Environment

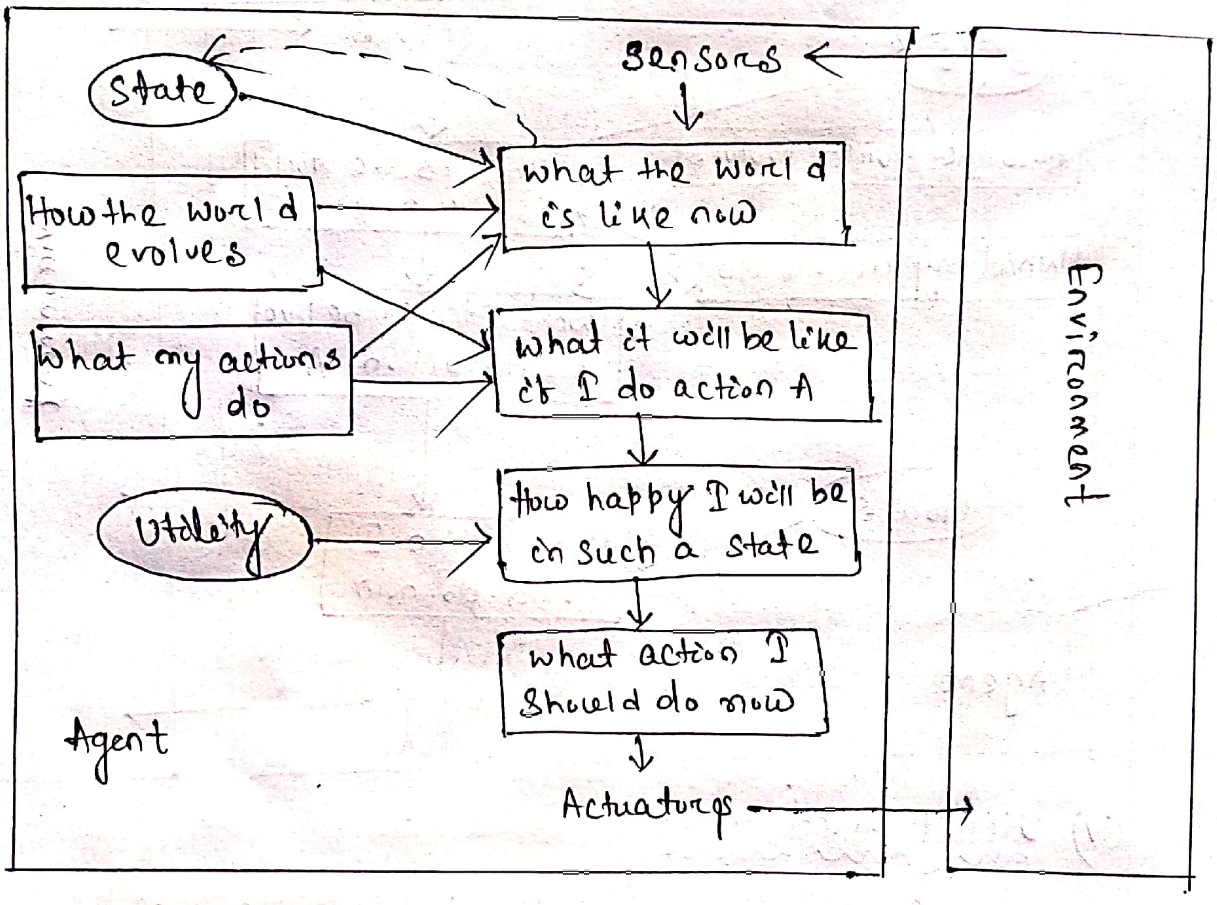
(iv) Utility based agent

Goals alonge are not really enough to generate high quality behaviour in most environments. A utility function maps a state (or a sequence of states) on-to a real number, which describes the associated degree of happiness.

A complete specification of the utility function allows rational decisions in two kinds of cases where goals are inadequate.

1. when there are conflicting goals, only some of which can be achieved (speed or safety), the utility functions specify the appropriate tradeoff.

2. when there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be ~~waited~~ weighed up against the importance of the goals.


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

Learning agents

A learning agent can be divided into four conceptual components.

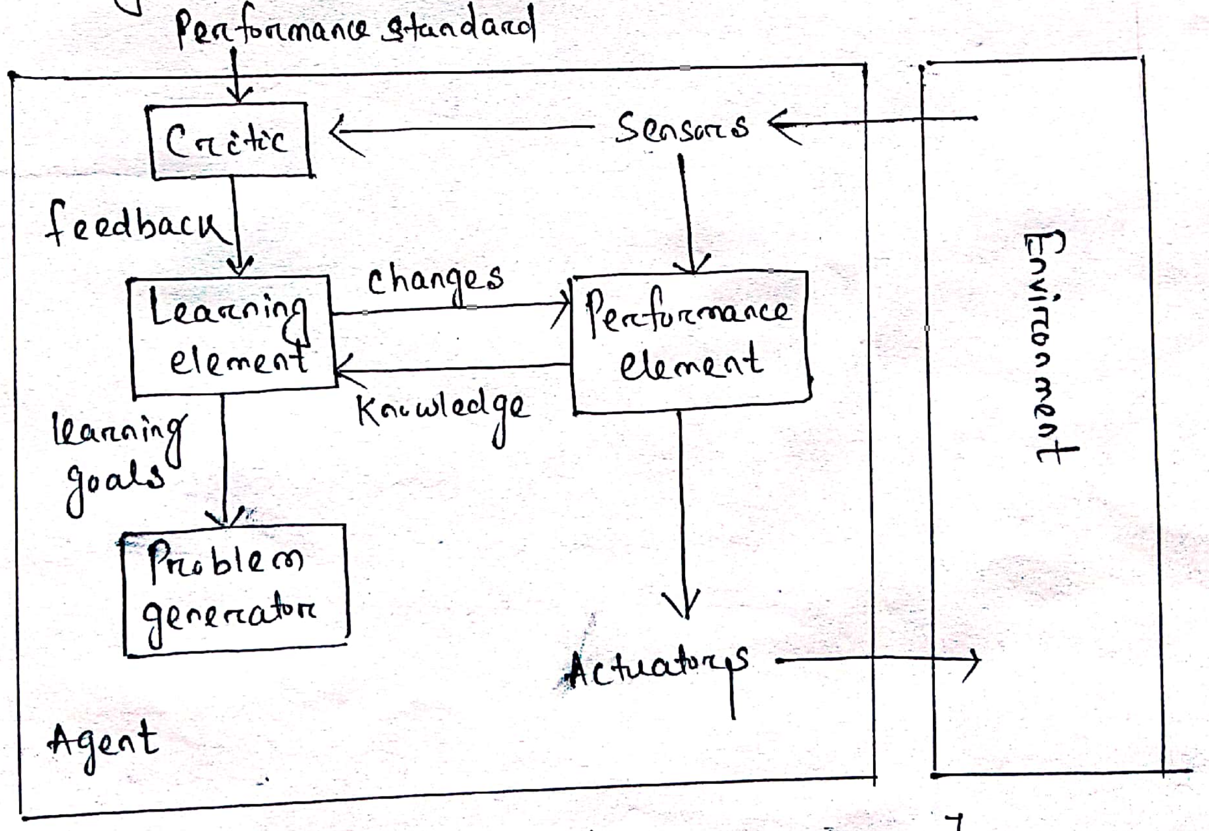
1. Critic
2. learning element
3. Performance element
4. Problem generator.

A learning element is responsible for making improvements. Performace element is responsible for selecting external actions.

The learning element uses feedback from the Critic on how the agent is doing and determines how the performance element should be modified to do better in future.

The problem generator component is responsible for suggesting actions that will lead to new and informative experiences.

All agents can improve their performance through learning.



[General model of learning agents]