


Artificial Intelligence
&
Machine Learning
19CS5DCAML

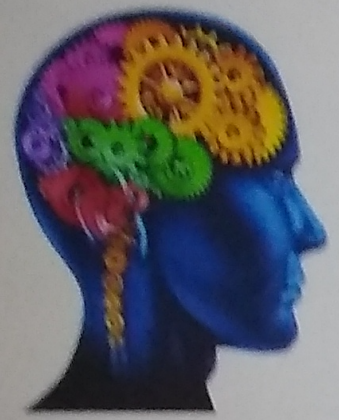

CONTENTS

- Module1: Intro to AI and Intelligent Agents
- Module2: First order logic and Inference in First-Order Logic and Reasoning
- Module3: Supervised Learning: Linear Regression & Classification
- Module4: Unsupervised Learning : Clustering
- Module5: Deep learning and Neural Networks

Text Book

1. Stuart Russel, Peter Norvig: Artificial Intelligence A Modern Approach, 3rd Edition, Pearson Education, 2003.
2. https://towardsdatascience.com/notes-on-artificial-intelligence-ai-machine-learning-ml-and-deep-learning-dl-for-56e51a2071c2
3. “Data Mining Concepts and Techniques”, Jiawei Han, Micheline Kamber, Jian Pei, Elsevier (MK) 3rd Edition, 2012.
4. Deep Learning with Python: A Hands-on Introduction Nikhil Ketkar
5. An Introduction to Statistical Learning, with Applications in R (2013), by G.James, D. Witten, T. Hastie, and R. Tibshirani.

- **AI -> machines that can mimick humans**
- **AI uses ML**



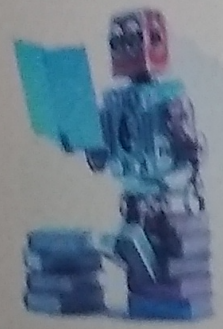
Intelligent Behaviour - Navigation, Chat, predictive analytics, Game playing, Debate ...

Symbolic reasoning
Statistical Reasoning
Case-based reasoning, fuzzy reasoning,...

Classifying, predicting, Regression,...

Reasoning, planning, perceiving, communicating , acting, ...

ML



Model

Decision trees, Rules, Neural nets, deep nets,..

Knowledge Representation

Rules, Ontologies, frames, scripts,

Supervised learning,
unsupervised learning,
Reinforced learning

Data

Knowledge Engineering

Experience

Module-1 Overview [Text book1]

- Introduction and Intelligent Agents (1,1.1)
- Agents and Environments, The Nature of Environments, Structure of Agents (2.1,2.2,2.3,2.4)
- Logical Agents(7.1,7.2,7.3,7.4)
- Difference between Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL), Supervised , Unsupervised Learning semi-supervised learning. Re-inforcement Learning, Regression vs Classification problems, How to become a practitioner with machine learning?, AI & ethics

1. https://towardsdatascience.com/notes-on-artificial-intelligence-ai-machine-learning-ml-and-deep-learning-dl-for-56e51a2071c2

❑ 1.1 What is AI?

Rationality : A system is rational if it does the "right thing," given what it knows.

- Top ones are concerned with thought processes and reasoning
- Bottom ones address behavior

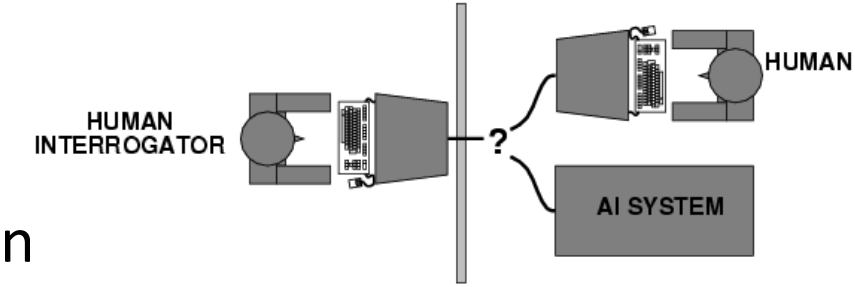
Thinking humanly	Thinking rationally
Acting humanly	**Acting rationally**

- **Left measure** is in terms of fidelity to human performance
- **Right Measure** against an ideal concept of Intelligence

❑ Acting humanly: Turing Test

"The art of creating machines that per- form functions that require intelligence when performed by people." (Kurzweil, 1990)

"The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)



- Turing (1950) "Computing machinery and intelligence":
- "Can machines think?" → "Can machines behave intelligen
- Operational test for intelligent behavior: the Imitation Game

- Predicted that by 2000, a machine might have a 30% chance of fooling a lay person for 5 minutes

- The computer would need to posses the following capabilities

1. NLP
2. Knowledge Representation
3. Automated Reasoning
4. Machine Learning
5. Computer vision
6. Robotics

NOTE:TOTAL TURING TEST

❑ Thinking humanly: cognitive modeling

"The exciting new effort to make comput- ers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985)

"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solv- ing, learning . . ." (Bellman, 1978)

- 1960s "cognitive revolution": information-processing psychology
- Requires scientific theories of internal activities of the brain
- -- How to validate? Requires
  - 1) Predicting and testing behavior of human subjects (top-down)
  - or 2) Direct identification from neurological data (bottom-up)

- Both approaches (roughly, Cognitive Science and Cognitive Neuroscience) are now distinct from AI

❑ Thinking humanly: “cognitive modelling "

- Determining the way humans think there are two ways

1. Introspection
2. Psychological Experiments

Interdisciplinary field of cognitive science brings together computer models from AI and experimental techniques fro psychology to try to construct precise and testable theories of the workings of the human mind

❑ Thinking rationally: "laws of thought"

"The study of mental faculties through the use of computational models."
(Charniak and McDermott, 1985)

"The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)

- Aristotle: what are correct arguments/thought processes?
- Syllogisms

A+B→C

- Several Greek schools developed various forms of *logic*: *notation* and *rules of derivation* for thoughts
- Direct line through mathematics and philosophy to modern AI
- Problems can be represented in logical notation can also be solvable
- Two obstacles to this approach:

1. Not easy to take informal knowledge and state it in the formal terms when it's certainty is less than 100%

2. There is a big difference between being able to solve a problem and doing so in practice

❑ Acting rationally: rational agent

"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al*., 1998)

"AI ...is concerned with intelligent be- havior in artifacts." (Nilsson, 1998)

- Rational behavior: doing the right thing
- Agent is something that acts
- Difference between programs and agents
- Rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty ,the best expected outcome
- The right thing: that which is expected to maximize goal achievement, given the available information
- Laws of thought approach to AI, the emphasis was on correct inferences. Making Correct inferences is being part of the rational agent.
- In addition it can use the scientific developments as well ; for example DeepLearning

❑ Acting rationally: rational agent

- Ways of acting rationally that cannot involve inference

Example: Recoiling from hot stove is a reflex action

For these reasons the study of AI as rational agent as 2 advantages:

1. Laws of thought approach involving correct inference is just one of the several mechanisms to achieve rationality

2. It is more amenable to scientific development

Limited rationality

❑ 1.3 History of AI

1. The Gestation of artificial intelligence(1943-1955)

➢AI first work by Warren Maculloch and walter pitts

➢ 3 resources: Knowledge of basic Psychology, Function of neurons n brain and Turing's theory of computation

➢Donald Hebb :modifying the connection strength of neurons: Hebbian Learning

➢Marvin Minsky and Dean Edmonds built a first neural network in 1951

❑ 1.3 History of AI

2. The birth of artificial Intelligence(1956)

➢Workshop by Macarthy's : artificial intelligence

➢Computational rationality

➢Why AI?

Idea of duplicating human ideas

➢AI is the only one branch of computer science and AI is the only field to attempt to build machines that will function autonomously in complex,changing environments

❑ 1.3 History of AI

3. Early Enthusiasm, great expectations(1952-1969)

➢ GPS : first program to embody the “thinking humanly” approach

➢Models of cognition by Newell and Simon to formulate the physical symbol system hypothesis, which states that “a physical symbol system has the necessary and sufficient means for general intelligent action”

➢McCarthy defined the high-level language Lisp

➢Minsky supervised a series of students who chose limited problems that appeared to require intelligence to solve. These limited domains became known as microworlds

➢Example:Tom’s Evans Analogy program solved geometric analogy problems that appear in IQ tests

➢Hebb’s Learning work enhanced by Bernie Widrow Who called his networks as adalines and Frank Rosenblatt with his perceptrons

➢Rosenblatt proved the perceptron convergence theorem, showing tha this learning algorithm could adjust the connection strengths of a perceptron to match any input data, provided such match exists

❑ 1.3 History of AI

4. A dose of reality(1966-1973)

➢Herbert Simon: “It is not my aim to surprise or shock you”

➢Early systems fail when tried on wider selections of problems because of 3 difficulties

➢1.Most early programs contained little or no knowledge of their subject matter

➢2.Intractability of many problems that AI was attempting to solve

➢Machine Evolution

➢3. Fundamental limitations on the basic structures being used to generate intelligent behavior

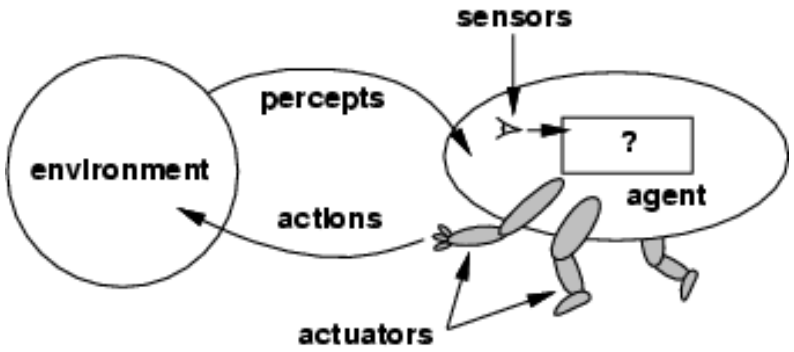
❑ 1.3 History of AI

5. Knowledge-based systems: The key to power?(1969-1979)
6. AI becomes an Industry(1980-present)
7. The return of neural networks(1986-present)
8. AI becomes a science(1987-present)
9. The emergence of Intelligent agents(1995-present)

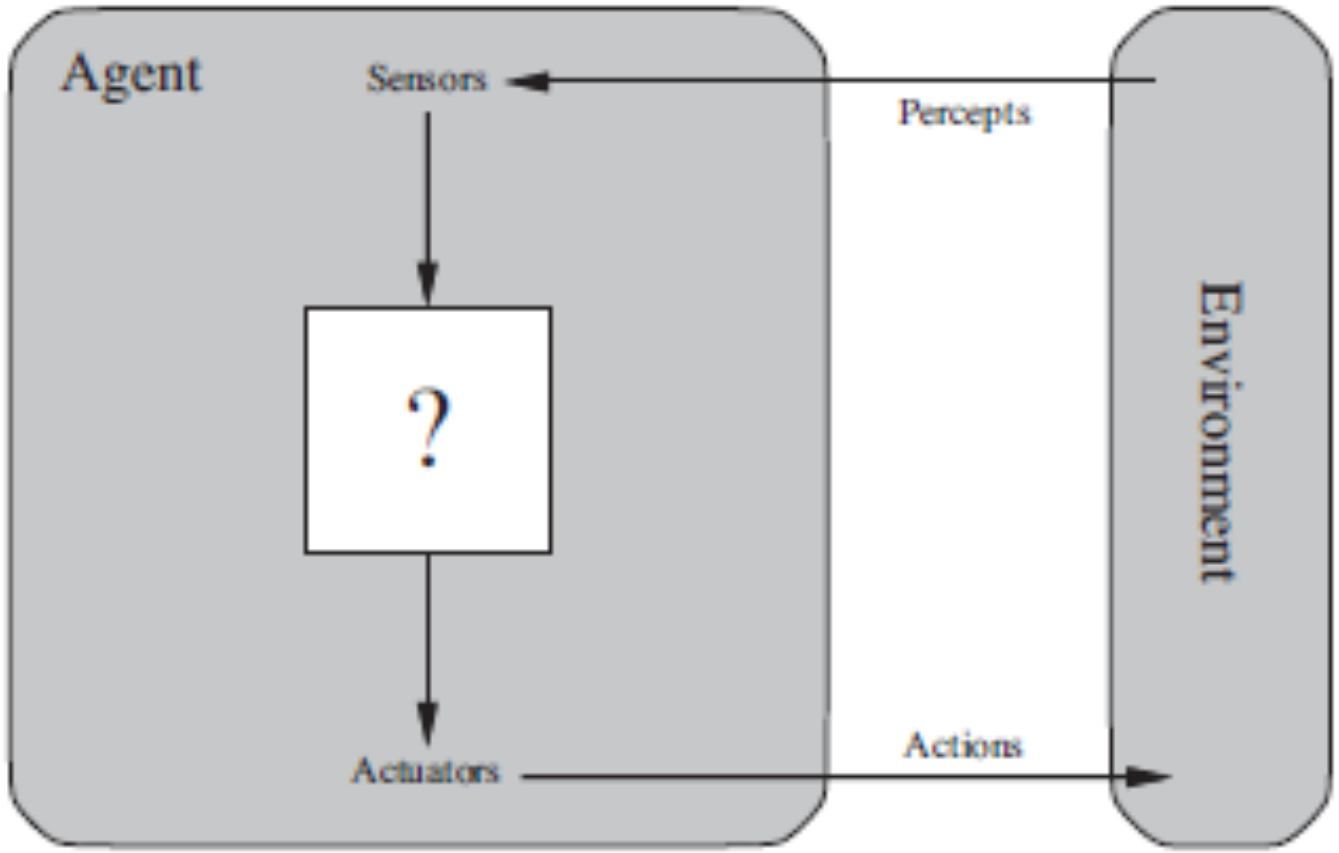
❑ 2.1 Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
- Human agent:
  - eyes, ears, and other organs for sensors;
  - hands, legs, mouth, and other body parts for actuators
- Robotic agent:
  - cameras and infrared range finders for sensors;
  - various motors for actuators

Agents and environments

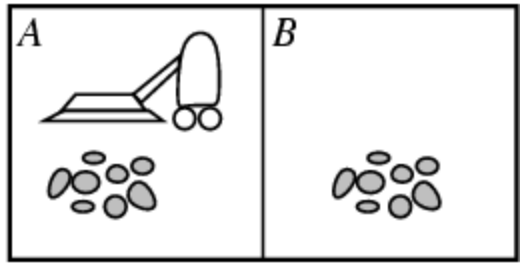


- The agent function maps from percept histories to actions:
- $[f: \mathcal{P}^* \rightarrow \mathcal{A}]$
- The agent program runs on the physical architecture to produce f
- agent = architecture + program



Agents interact with environments through sensors and actuators.

Vacuum-cleaner world



- Percepts: location and contents, e.g., [A,Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure

A vacuum-cleaner agent

- \input{tables/vacuum-agent-function-table}

❑ 2.2 Good Behavior : The concept of rationality

- A rational agent is the one that does right thing
- Some way to measure success together with the description of the environment and the sensors and actuators of the agent which provide complete specification of the task facing the agent

➢Performance Measure

➢Embodies the criterion for success of an agent's behavior(sequence that is desirable)

➢Ask the agent for a subjective opinion of how happy it is with its own performance measure

➢Insist objective performance measure

➢Example : amount of dirt cleaned up in a single eight hour shift (objective)

➢Reward the agent for having clean floor

- 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 agent should behave

❑ 2.2 Good Behavior : The concept of rationality

- Rationality :

➢The performance measure that defines the criterion of success

➢The agent's prior knowledge of the environment

➢The actions that the agent can perform

➢The agent's percept sequence to date

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 built-in knowledge the agent has

❑ 2.2 Good Behavior : The concept of rationality

➢The Performance measure awards one point in each clean square at each time step, over a lifetime of 1000 time steps

➢The Geography of the environment is known a priori but the dirt distribution and initial location of the agent or not

➢The only available actions are Left, right, suck and NoOp(do nothing)

➢The agent correctly perceives its location and whether that location contains dirt

- We can see that same agent irrational under different circumstances

❑ 2.2 Good Behavior : The concept of rationality

- **Omniscience, learning, and autonomy**

➢An **omniscient agent** knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality

➢Rationality is not same as perfection

➢Rationality maximizes expected performance

➢Perfection maximizes actual performance

➢Does not require omniscience

➢Doing actions in order to modify future percepts– called as information gathering

➢Information gathering provided by exploration

❑ 2.2 Good Behavior : The concept of rationality

➢Rational agent not only to gather information ,but also to learn as much as possible from what it perceives

➢Successful agents split the task of computing the agent function into three different periods:

1. When the agent being designed, some of the computation done by designers
2. When it is deliberating on its next action, the agent does more computation
3. As it learns from experience ,it does even more computation

❑ 2.2 Good Behavior : The concept of rationality

- A rational agent should be **autonomous** –it should learn what it can to compensate for partial or incorrect prior knowledge
- It would be reasonable to provide an artificial intelligent agent with initial knowledge as well as an ability to learn.
- After sufficient experience of its environment the behavior of a rational agent can become effectively independent of its prior knowledge.
- Hence, the incorporation of learning allows one to design a single rational agent that will succeed in a vast variety of environments

❑ 2.3 PEAS

- Task environments are the problems to which rational agents are solutions

Specifying Task Environment

- PEAS: Performance measure, Environment, Actuators, Sensors
- Must first specify the setting for intelligent agent design
- Consider, e.g., the task of designing an automated taxi driver:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

PEAS

- Must first specify the setting for intelligent agent design
- Consider, e.g., the task of designing an automated taxi driver:

  - Performance measure: Safe, fast, legal, comfortable trip, maximize profits
  - Environment: Roads, other traffic, pedestrians, customers
  - Actuators: Steering wheel, accelerator, brake, signal, horn
  - Sensors: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

PEAS

- Agent: Medical diagnosis system
- Performance measure: Healthy patient, minimize costs, lawsuits
- Environment: Patient, hospital, staff
- Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
- Sensors: Keyboard (entry of symptoms, findings, patient's answers)

PEAS

Question

- Agent: Part-picking robot
- Performance measure: ?
- Environment: ?
- Actuators: ?
- Sensors: ?

Answer

- Agent: Part-picking robot
- Performance measure: Percentage of parts in correct bins
- Environment: Conveyor belt with parts, bins
- Actuators: Jointed arm and hand
- Sensors: Camera, joint angle sensors

PEAS

Question

- Agent: Interactive English tutor
- Performance measure: ?
- Environment: ?
- Actuators: ?
- Sensors: ?

Answer

- Agent: Interactive English tutor
- Performance measure: Maximize student's score on test
- Environment: Set of students
- Actuators: Screen display (exercises, suggestions, corrections)
- Sensors: Keyboard

Environment types/Properties of Task Environments

- Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time. Example : Chess game
- Example for partially observable: Vaccum Agent, Automated taxi
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)
- Example for Deterministic : Vaccum World
- Example for Stochastic :Automated taxi
- Episodic (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.
- Example for Episodic: Agent that to detect defective parts on an assembly line bases each decision on the current part
- Example for Sequential : Chess and taxi driving

Environment types

- Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semi dynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- Dynamic - With the Passage of time agent no need to look at the environment
  - Example for Dynamic : Taxi driving
  - Example for Static : crossword puzzle
  - Example for semi dynamic : Chess when playing with clock
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
  - Example for Discrete : Chess
  - Example for continuous : taxi driving
- Single agent (vs. multiagent): An agent operating by itself in an environment.

  Example for single agent : agent solving crossword puzzle by itself

  Example for 2 agent/multi-agent environment : Playing chess,taxi(competitive, cooperative)

Environment types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

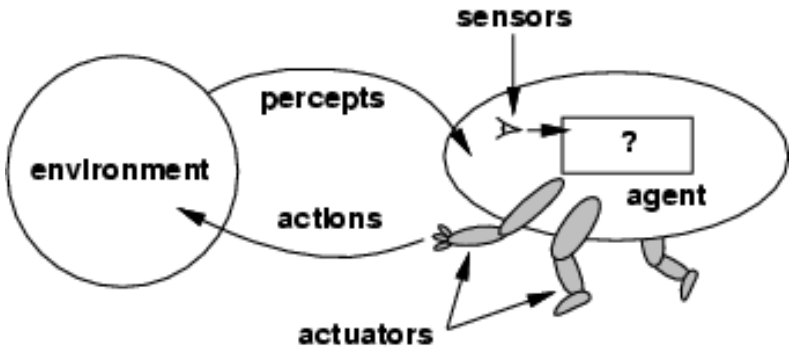
- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent
- Many environments are drawn from environment class
- If we designed of the particular case but might not identify good design for particular class in general
- For this reason code repository includes environment generator

2.4 Structure of Agents

- An agent is completely specified by the agent function mapping percept sequences to actions
- One agent function (or a small equivalence class) is rational

- Aim: find a way to implement the rational agent function concisely

Structure of Agents



- The agent function maps from percept histories to actions:

$$[f: \mathcal{P}^* \rightarrow \mathcal{A}]$$

- The agent program runs on the physical architecture to produce *f*
- agent = architecture + program

Agent Programs

- Let P be the set of possible percepts and let T be the lifetime of the agent (the total number of percepts it will receive). The lookup table will contain entries.
- Consider the automated taxi: the visual input from a single camera comes in at the rate of roughly 27 megabytes per second (30 frames per second, 640×480 pixels with 24 bits of color information). This gives a lookup table with over 10250,000,000,000 entries for an hour's driving.
- Even the lookup table for chess—a tiny, well-behaved fragment of the real world—would have at least 10150 entries.

$$\sum_{t=1}^{T} |\mathcal{P}|^t$$

Agent Programs

- Disadvantages are:

(a) no physical agent in this universe will have the space to store the table,

(b) the designer would not have time to create the table,

(c) no agent could ever learn all the right table entries from its experience, and

(d) even if the environment is simple enough to yield a feasible table size, the designer still has no guidance about how to fill in the table entries. a long time to learn the table entries

Agent Programs

```
function TABLE-DRIVEN-AGENT(percept) returns an action
    persistent: 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
```

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

Simple Reflex Agents

Agent types

Four basic types in order of increasing generality:

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

* Learning Agents

Simple Reflex Agents

Agent types

Four basic types in order of increasing generality:

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

Simple reflex agents

- The simplest kind of agent is the simple reflex agent.
- These agents select actions on the basis of the current percept , ignoring the rest of the percept history
- Example : with respect to figure2.3, decision is based only on the current location and on whether that contains dirt. An agent program is as shown in Figure 2.8

Simple reflex agents

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

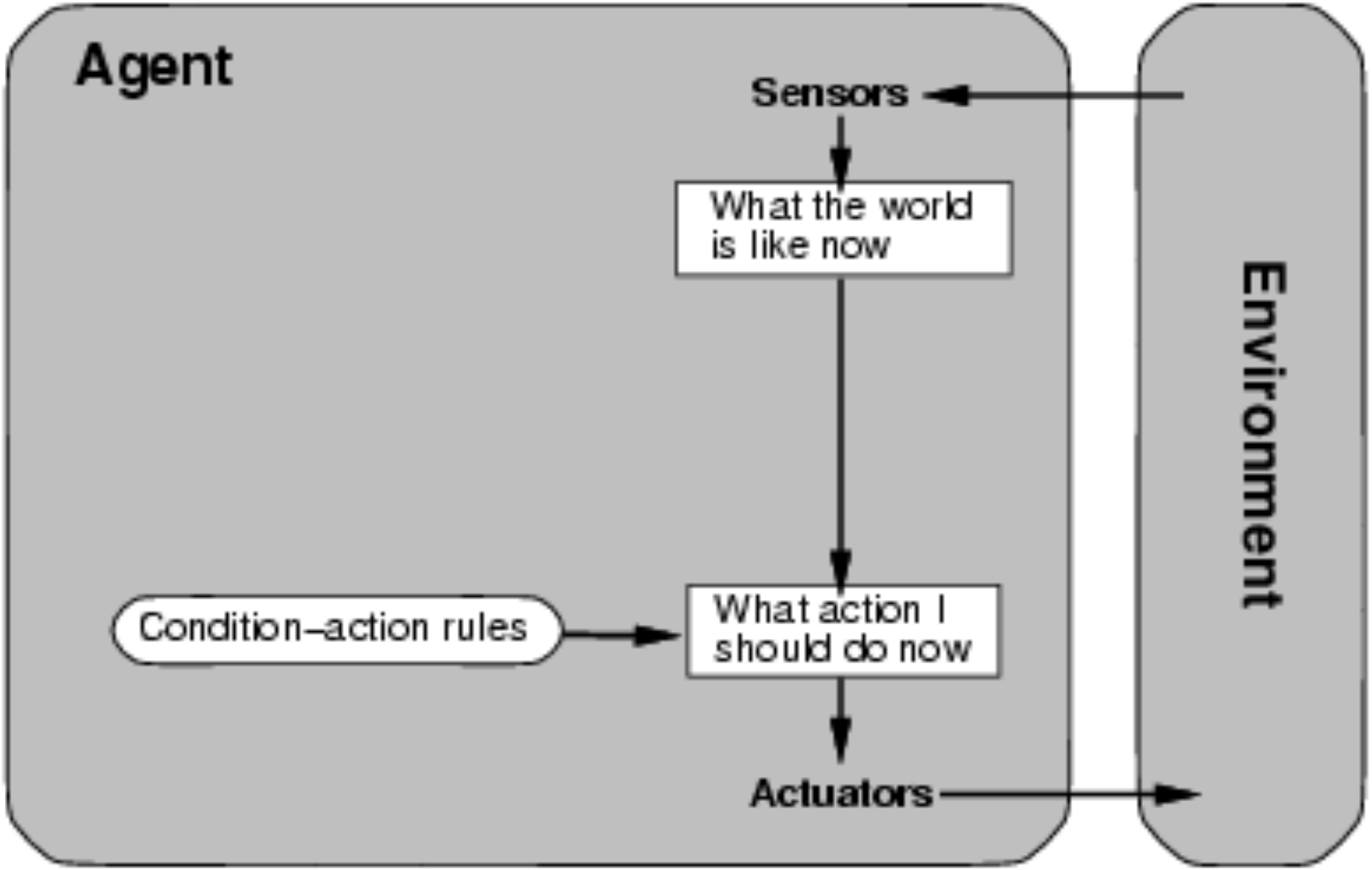
```
function SIMPLE-REFLEX-AGENT(percept) returns an action
    persistent: rules, a set of condition–action rules

    state ← INTERPRET-INPUT(percept)
    rule ← RULE-MATCH(state, rules)
    action ← rule.ACTION
    return action
```

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

Simple reflex agents

- Figure 2.9 gives structure of the general program i.e. it works as general interpreter for condition action rules and then to create rules for specific task environments. Rectangle denotes the current internal state and oval represent the background information
- The agent program is as shown in Figure 2.10 . In this interpret_input function generates an abstracted description of the current state from the percept, and the Rule-Match function returns the first rule in the set of rules that matched the given state description

Simple reflex agents



Simple reflex agents

- Randomization
- Partially observable

Model-based reflex agents

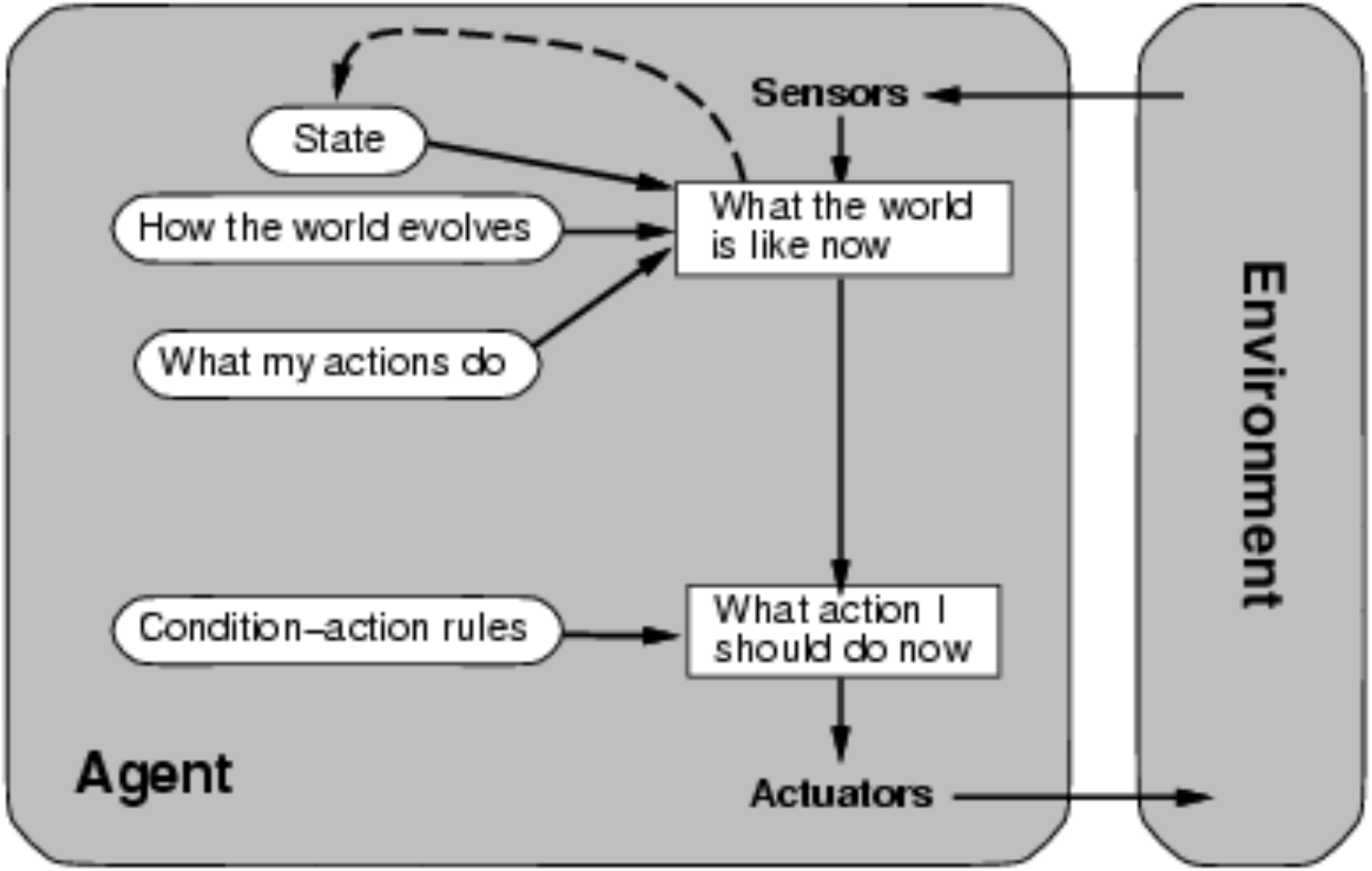
- The most effective way to handle partial observability is for the agent to *keep track of the part of the world it can't see now*. That is, the agent should maintain some sort of **internal state** that depends on the percept history and thereby reflects at least some of the unobserved aspects of the 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, that an overtaking car generally will be closer behind than it was a moment ago.
2. Second, we need some information about how the agent's own actions affect the world

Model-based reflex agents

- knowledge about "how the world works"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **model** of the world. An agent that uses such a MODEL-BASED model is called a **model-based agent**.
- Figure gives the structure of the model-based reflex agent with internal state, showing how the current percept is combined with the old internal state to generate the updated description of the current state, based on the agent's model of how the world works. The agent program is shown in Figure 2.12. The function UPDATE-STATE, which is responsible for creating the new internal state description.

Model-based reflex agents



Model-based reflex agents

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

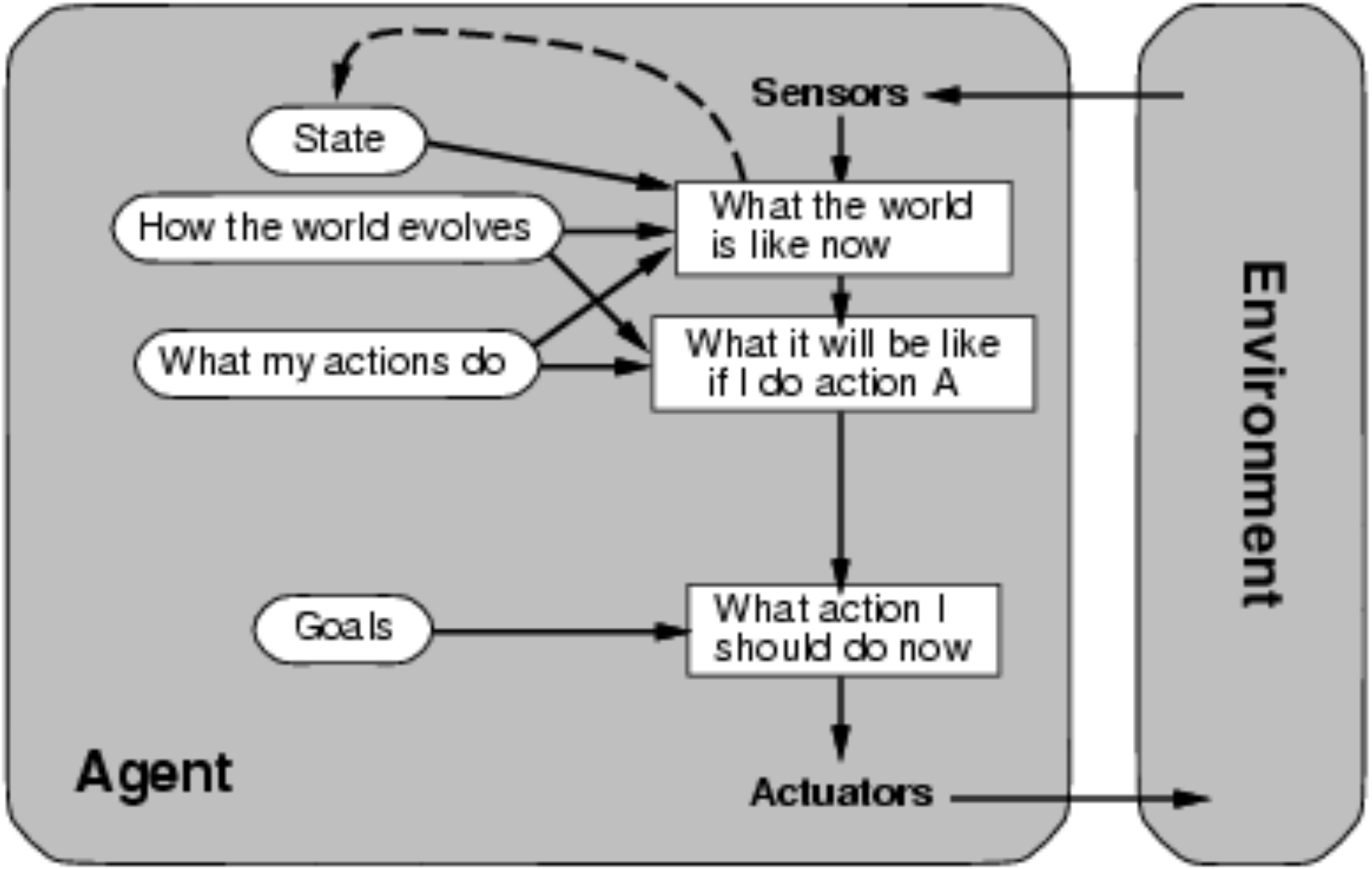
Goal-based agents

- A current state description, the GOAL agent needs some sort of **goal** information that describes situations that are desirable—for example, being at the passenger's destination. The agent program can combine this with the model (the same information as was used in the model based reflex agent) to choose actions that achieve the goal. Figure 2.13 shows the goal-based agent's structure.
- **Search** and **planning** are the subfields of AI devoted to finding action sequences that achieve the agent's goals.

Goal-based agents

- Although the goal-based agent appears less efficient, it is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified. If it starts to rain, the agent can update its knowledge of how effectively its brakes will operate; this will automatically cause all of the relevant behaviors to be altered to suit the new conditions.
- For the reflex agent, on the other hand, we would have to rewrite many condition–action rules. The goal-based agent's behavior can easily be changed to go to a different destination, simply by specifying that destination as the goal. The reflex agent's rules for when to turn and when to go straight will work only for a single destination; they must all be replaced to go somewhere new.

Goal-based agents



Utility-based agents

- ✓ A performance measure assigns a score to any given sequence of environment states, so it can easily distinguish between more and less desirable ways of getting to the UTILITY FUNCTION (taxi's destination)
- ✓ An agent's **utility function** is essentially an internalization of the performance measure. If the internal utility function and the external performance measure are in agreement, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure.

Utility-based agents

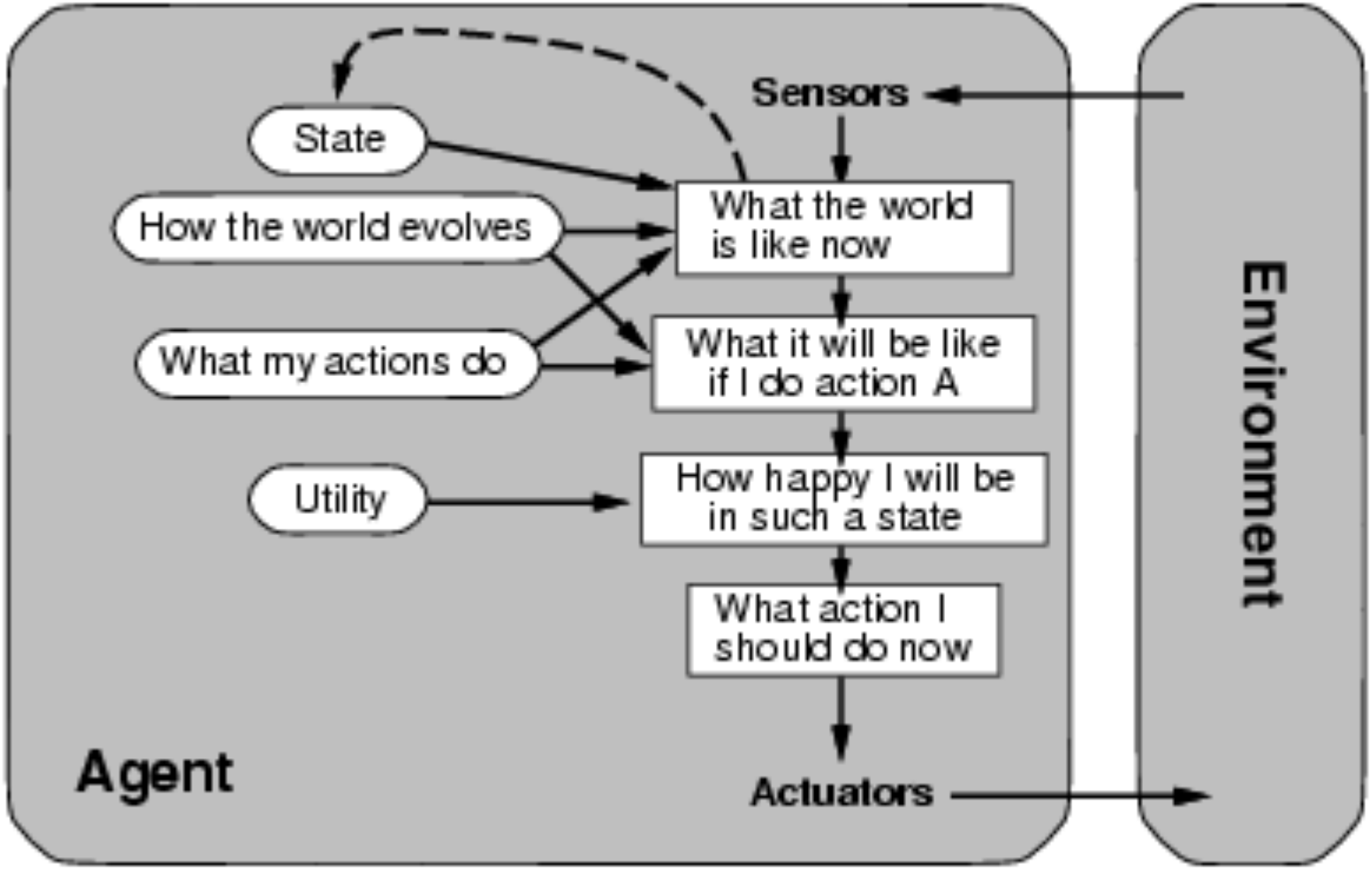
- In two kinds of cases, goals are inadequate but a utility-based agent can still make rational decisions.

1. First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff.

2. Second, 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 weighed against the importance of the goals.

❑ The utility-based agent structure appears in Figure 2.14.

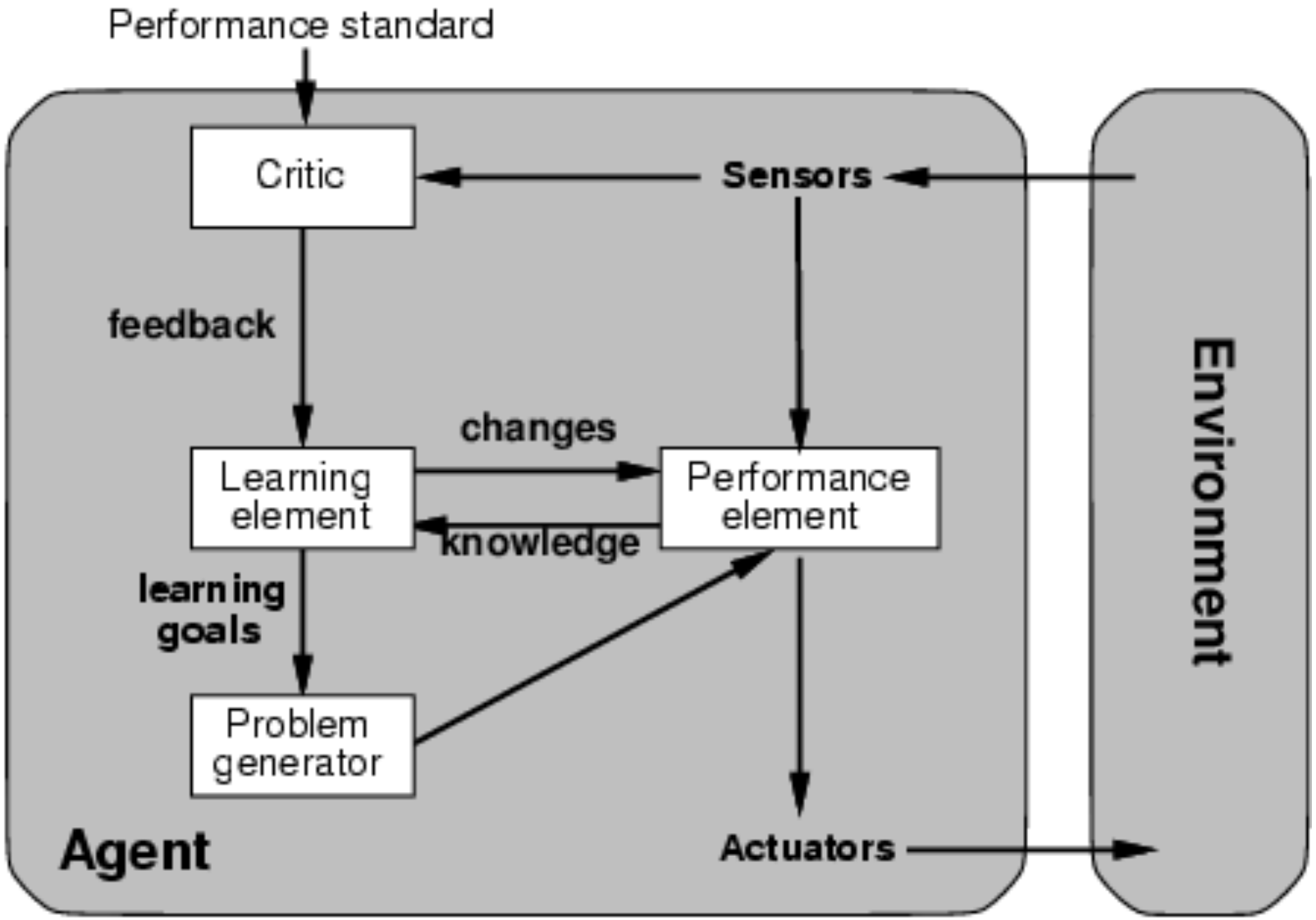
Utility-based agents



Learning agents

- A learning agent can be divided into four conceptual components, as shown in Figure 2.15. The most important distinction is between the **learning element**, which is responsible for making improvements, and the **performance element**, which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions.
- The learning element uses CRITIC feedback from the **critic** on how the agent is doing and determines how the performance element should be modified to do better in the future.
- The last component of the learning agent is the **problem generator**. It is responsible for suggesting actions that will lead to new and informative experiences.

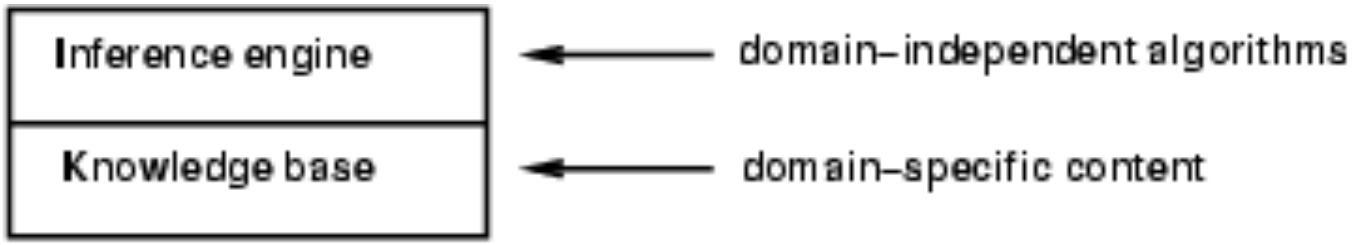
Learning agents



7.1Knowledge bases

- A knowledge-based agent can combine general knowledge with current percepts to infer hidden aspects of the current state prior to selecting actions
- Example 1: Physician diagnoses a patient
- Example 2: John saw the diamond through the window and coveted it
- Reasoning allows us to cope with the virtually infinite variety of utterences using finite store of common sense knowledge
- The knowledge of logical agents is always definite-each proposition is either true or false
- Logic has the pedagogical advantage of being simple example of a representation for knowledge based agents

7.1Knowledge bases



- Knowledge base = set of sentences in a formal language
- Each sentence is expressed in language called a Knowledge Representation Language and represents some assertion about the world
  - `Tell` it what it needs to know
  - Then it can `Ask` itself what to do - answers should follow from the KB

  - Both TELL and ASK tasks may involve inference-i.e. deriving new sentences from old
  - Logical agents ,inference must obey the fundamental requirement that when one ASKs a question of the knowledge base, the answer should follow from what has been told to the knowledge base previously

❑ 7.1Knowledge bases

- Figure 7.1 shows the outline of a knowledge-based agent program. Like all our agents, it takes a percept as input and returns an action. The agent maintains a knowledge base, KB, which may initially contain some background knowledge.
- Each time the agent program is called, it does three things.

➢ First, it TELLs the knowledge base what it perceives.

➢ Second, it ASKs the knowledge base what action it should perform. In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences, and so on.

➢ Third, the agent program TELLs the knowledge base which action was chosen, and the agent executes the action.

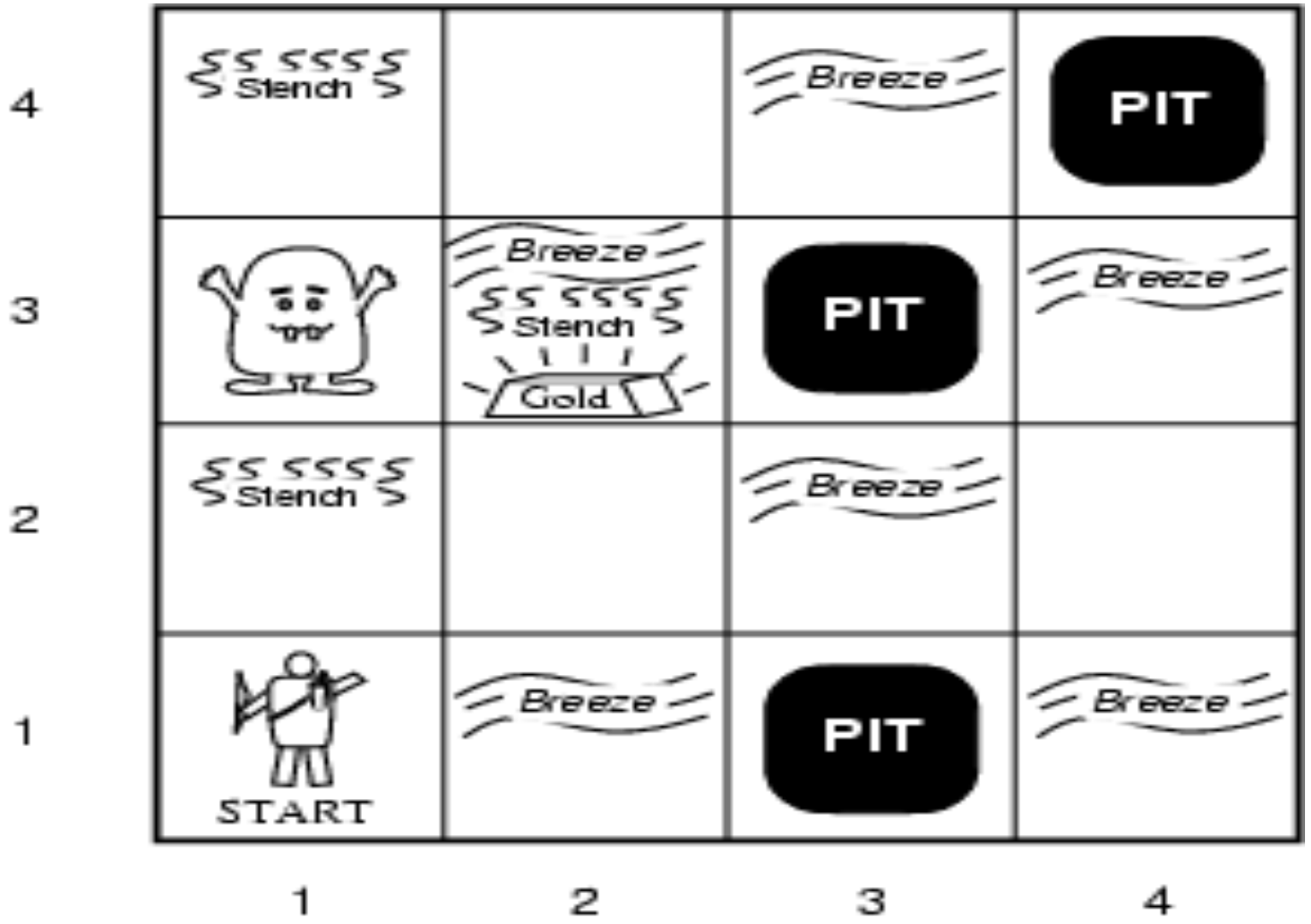
- The details of the representation language are hidden inside three functions that implement the interface between the sensors and actuators on one side and the core representation and reasoning system on the other.
- MAKE-PERCEPT-SENTENCE constructs a sentence asserting that the agent perceived the given percept at the given time. MAKE-ACTION-QUERY constructs a sentence that asks what action should be done at the current time. Finally, MAKE-ACTION-SENTENCE constructs a sentence asserting that the chosen action was executed. The details of the inference mechanisms are hidden inside TELL and ASK

A simple knowledge-based agent

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

- A knowledge-based agent can be built simply by TELLing it what it needs to know. Starting with an empty knowledge base, the agent designer can TELL sentences one by one until the agent knows how to operate in its environment. This is called the **declarative approach** to system building. In contrast, the **procedural approach** encodes desired behaviors directly as program code

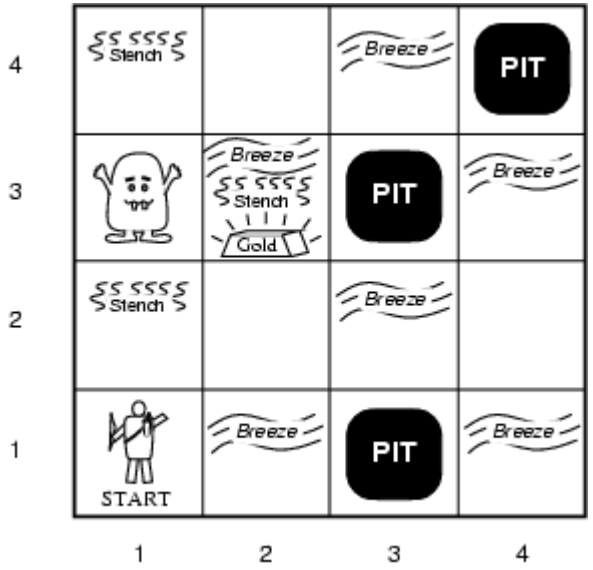
7.2 Wumpus World PEAS description



❑ 7.2 Wumpus World PEAS description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow

- Environment
  - 4x4 grid of rooms[The agent always start
    In a square labeled [1,1],facing to the right
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square



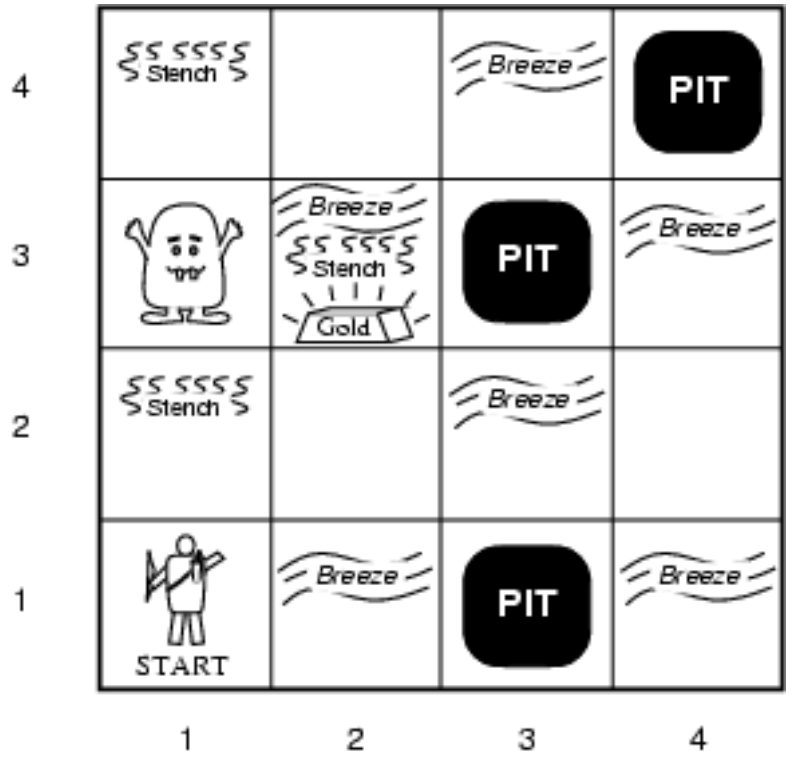
- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Wumpus world characterization

- Fully Observable No – only local perception
- Deterministic Yes – outcomes exactly specified
- Episodic No – sequential at the level of actions
- Static Yes – Wumpus and Pits do not move
- Discrete Yes
- Single-agent? Yes – Wumpus is essentially a natural feature

Exploring a wumpus world

?	?		
A[1,1]	A[2,1]	?	



❑ 7.3Logic in general

- Logics are formal languages for representing information such that conclusions can be drawn
- Syntax defines the sentences in the language
- Semantics define the "meaning" of sentences;
  - i.e., define truth of a sentence in a world
- E.g., the language of arithmetic
  - x+2 ≥ y is a sentence; x2+y > {} is not a sentence
  - x+2 ≥ y is true iff the number x+2 is no less than the number y
  - x+2 ≥ y is true in a world where x = 7, y = 1
  - x+2 ≥ y is false in a world where x = 0, y = 6

❑ 7.3Logic in general

- The semantics defines the **truth** of each sentence with respect to each **possible world**. For example, the semantics for arithmetic specifies that the sentence “x + y =4” is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1.
- In standard logics, every sentence must be either true or false in each possible world—there is no “in between.”
- When we need to be precise, we use the term **model** in place of “possible world.”
- If a sentence α is true in model m, we say that m **satisfies** α or sometimes m **is a model of** α. We use the notation M(α) to mean the set of all models of α.
- Now that we have a notion of truth, we are ready to talk about logical reasoning. This involves the relation of **logical entailment** between sentences—the idea that a sentence *follows logically* from another sentence. In mathematical notation, we write

  α |= β

Entailment

- Entailment means that one thing follows from another:

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true

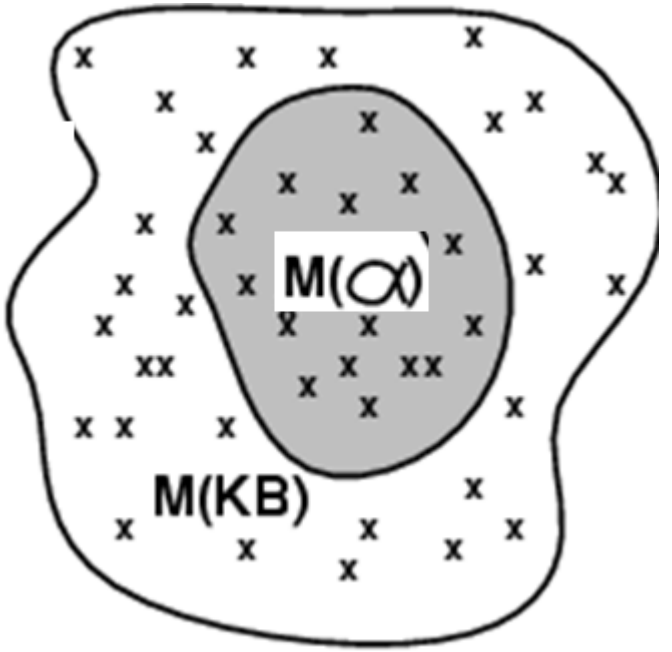
  – E.g., x+y = 4 entails 4 = x+y

  – x+y = 4 entails (x > 0 OR y > 0)
  – Entailment is a relationship between sentences (i.e., syntax) that is based on semantics
  – Informally: if KB is true *then* α must be true

Models

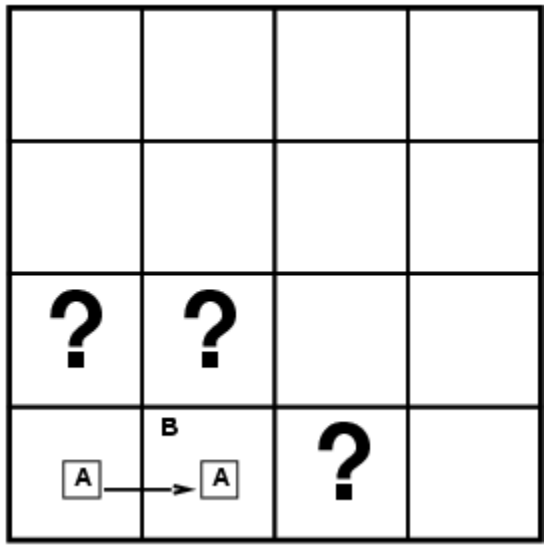
- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
- We say *m* is a model of a sentence α if α is true in *m*
- *M(α)* is the set of all models of α
- Then KB $\models$ α iff $M(KB) \subseteq M(\alpha)$
  - E.g.

  KB = Giants won and Reds won
  α = Giants won



Entailment in the wumpus world

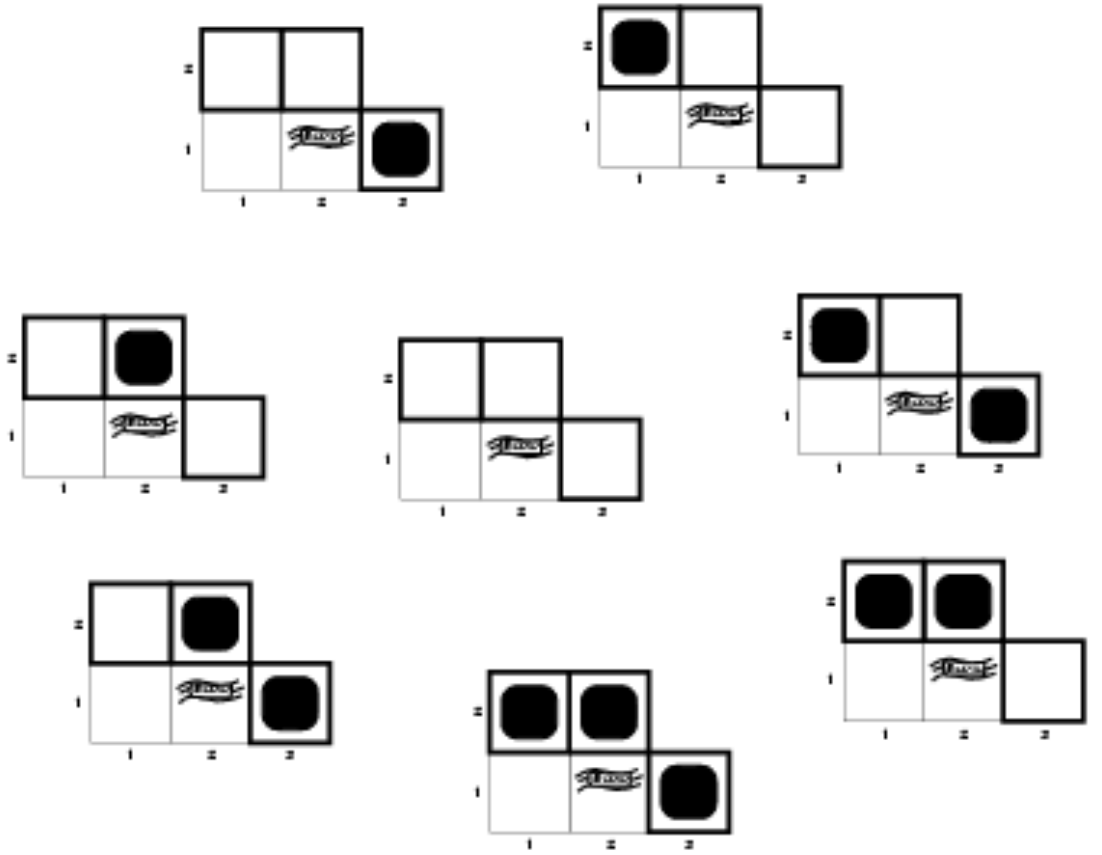
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]



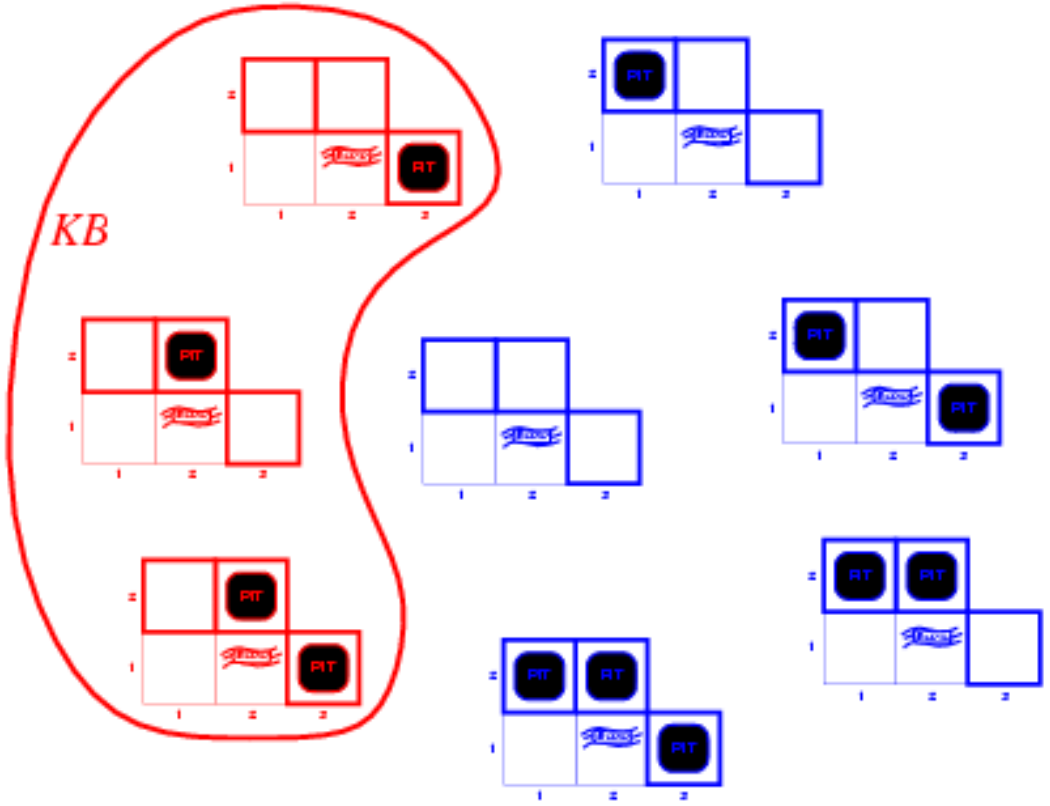
Consider possible models for *KB* assuming only pits

3 Boolean choices $\Rightarrow 2^3 = 8$ possible models

Wumpus models

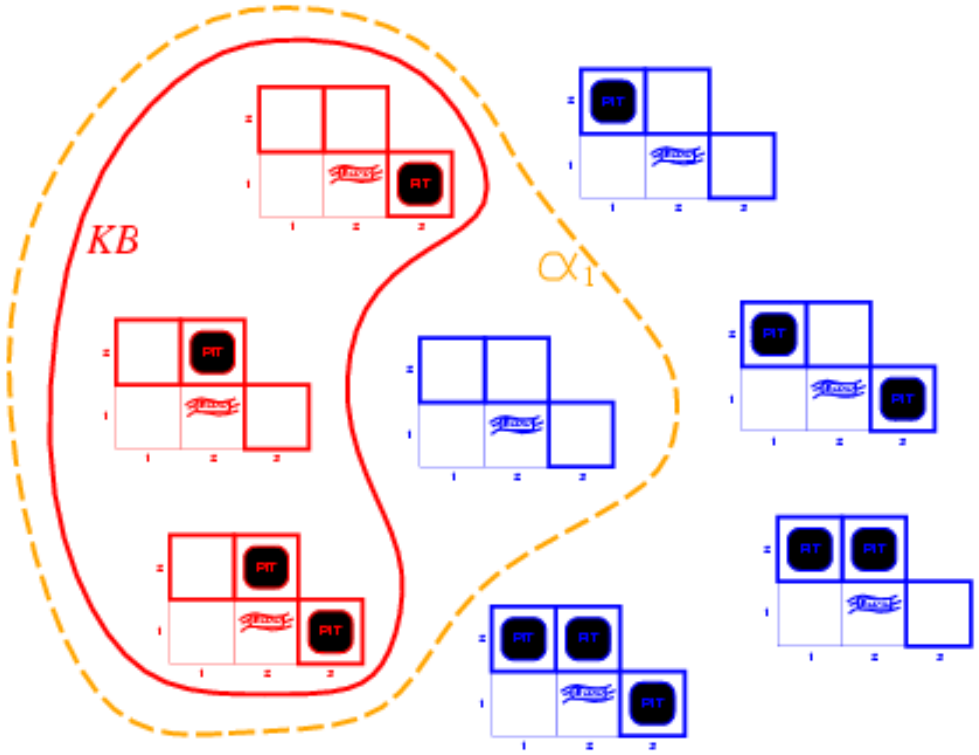


Wumpus models (2)



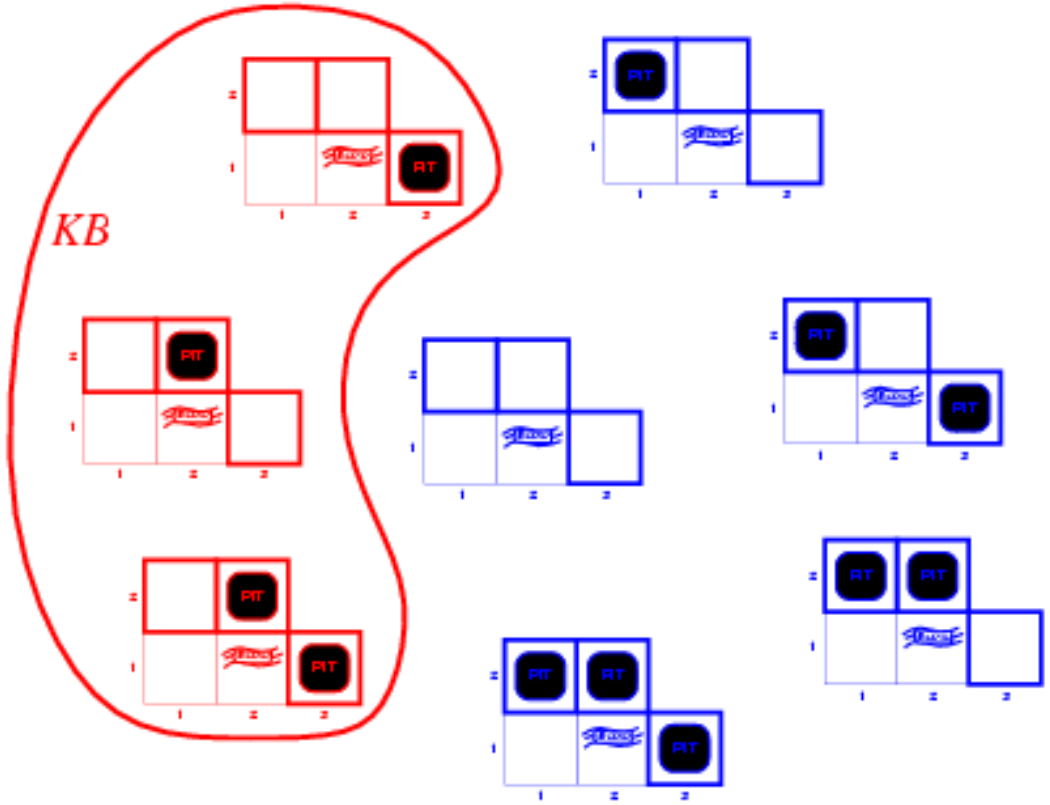
- *KB* = wumpus-world rules + observations

Wumpus models (3)



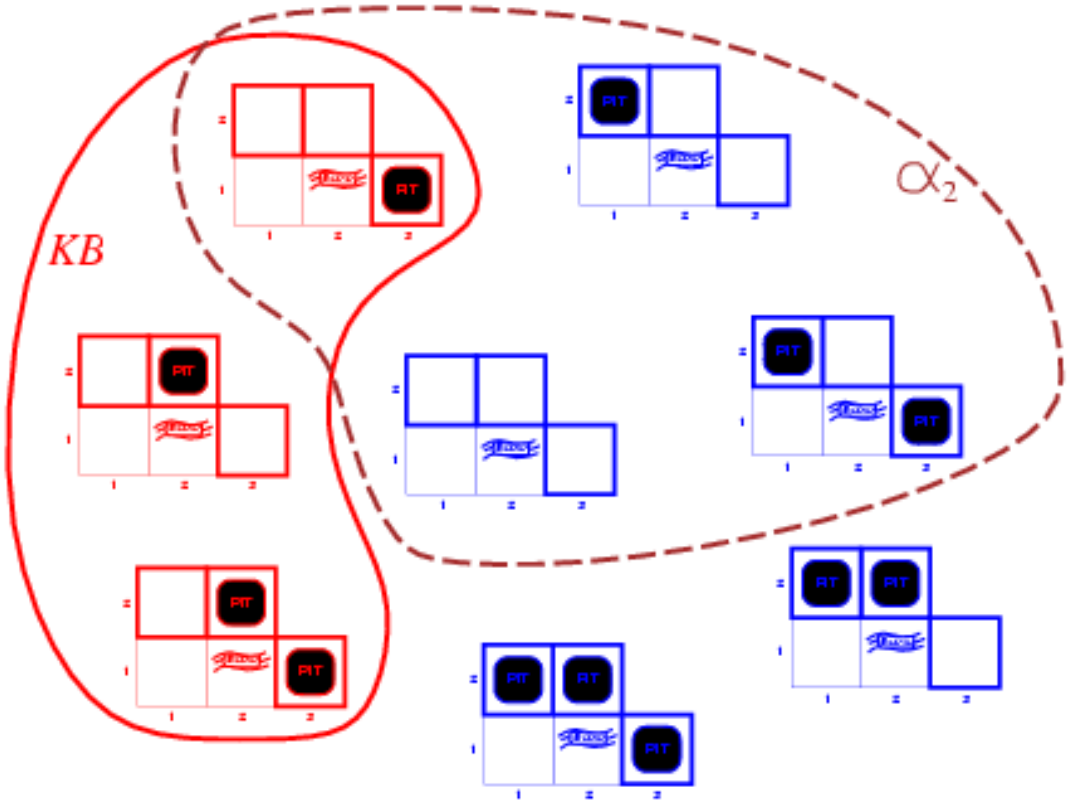
- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe", *KB* $\models \alpha_1$, proved by model checking

Wumpus models (4)



- *KB* = wumpus-world rules + observations

Wumpus models (5)



- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

Inference

- $KB \vdash_i \alpha$ = sentence α can be derived from *KB* by procedure *i*
- (*i is an algorithm that derives* α *from KB*)
- Soundness: *i* is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- Completeness: *i* is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
- Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.
- That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

Inference

- if KB is true in the real world, then any sentence α derived from KB by a sound inference procedure is also true in the real world
- The final issue to consider is **grounding**—the connection between logical reasoning processes and the real environment in which the agent exists.
- agent's sensors create the connection. For example, our wumpus-world agent has a smell sensor.

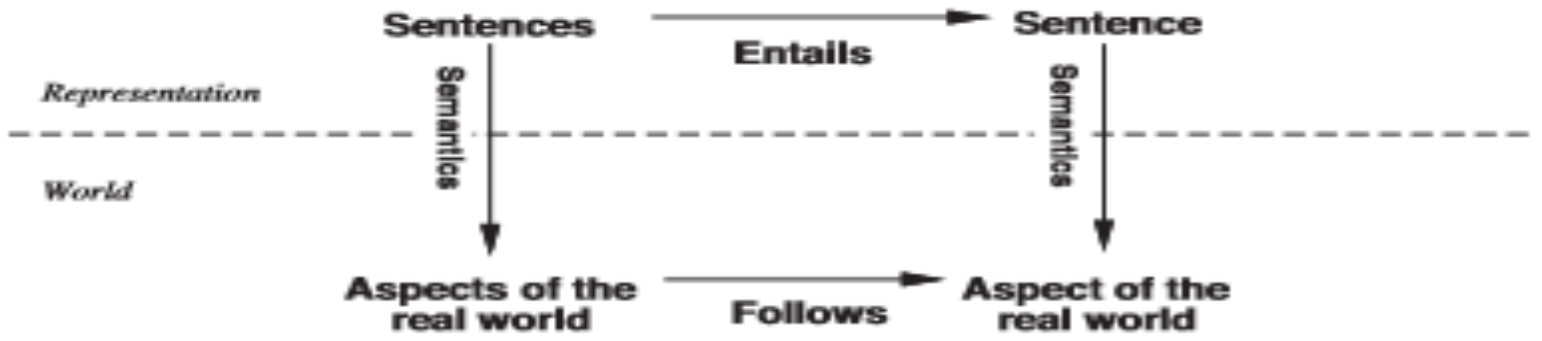


Figure 7.6 Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones. Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent.

❑ 7.4 Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols P_1, P_2 etc are sentences
  - If S is a sentence, $\neg S$ is a sentence (negation)
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

❑ 7.4 Propositional logic: Syntax

- **Complex sentences** are constructed from simpler sentences, using parentheses and **logical connectives**
- ¬ (not). A sentence such as ¬W1,3 is called the **negation** of W1,3. A **literal** is either an atomic sentence (a **positive literal**) or a negated atomic sentence (a **negative literal**).
- ∧ (and). A sentence whose main connective is ∧, such as W1,3 ∧ P3,1, is called a **conjunction**; its parts are the **conjuncts**. (The ∧ looks like an "A" for "And.")
- DISJUNCTION ∨ (or). A sentence using ∨, such as (W1,3∧P3,1)∨W2,2, is a **disjunction** of the **disjuncts** (W1,3 ∧ P3,1) and W2,2. (Historically, the ∨ comes from the Latin "vel," which means "or." For most people, it is easier to remember ∨ as an upside-down ∧.)
- IMPLICATION ⇒ (implies). A sentence such as (W1,3∧P3,1) ⇒ ¬W2,2 is called an **implication** (or conditional). Its **premise** or **antecedent** is (W1,3 ∧P3,1), and its **conclusion** or **consequent** is ¬W2,2. Implications are also known as **rules** or **if–then** statements. The implication symbol is sometimes written in other books as ⊃ or →.
- ⇔ (if and only if). The sentence W1,3 ⇔ ¬W2,2 is a **biconditional**. Some other books write this as ≡.

Propositional logic: Semantics

- In propositional logic, a model simply fixes the **truth value**—true or false—for every proposition symbol. For example, if the sentences in the knowledge base make use of the proposition symbols $P_{1,2}$, $P_{2,2}$, and $P_{3,1}$, then one possible model is
- m1 = {$P_{1,2}$ =false, $P_{2,2}$ =false, $P_{3,1}$ =true} With these symbols, 8 possible models, can be enumerated automatically.
- $P_{1,2}$ is just a symbol; it might mean "there is a pit in [1,2]" or "I'm in Paris today and tomorrow."
- The semantics for propositional logic must specify how to compute the truth value of *any* sentence, given a model. This is done recursively. **All sentences are constructed from atomic sentences and the five connectives.**
- Atomic sentences are easy:

1. True is true in every model and False is false in every model.
2. The truth value of every other proposition symbol must be specified directly in the model. For example, in the model m1 given earlier $P_{1,2}$ is false.

Propositional logic: Semantics

- For complex sentences, we have five rules, which hold for any sub sentences P and Q in any model m (here “iff” means “if and only if”):
- Rules for evaluating truth with respect to a model *m*:

1. $\neg$P is true iff P is false in m.
2. P $\wedge$ Q is true iff both P and Q are true in m.
3. P $\vee$ Q is true iff either P or Q is true in m.
4. P $\Rightarrow$ Q is true unless P is true and Q is false in m.
5. P $\Leftrightarrow$ Q is true iff P and Q are both true or both false in m

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \mathit{true} \wedge (\mathit{false} \vee \mathit{true}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$$

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

- a square is breezy *if* a neighboring square has a pit, and a square is breezy *only if* a neighboring square has a pit. So we need a biconditional,
- B1,1 ⇔ (P1,2 ∨ P2,1) ,
- where B1,1 means that there is a breeze in [1,1].

A simple knowledge base

- Px,y is true if there is a pit in [x, y].
- Wx,y is true if there is a wumpus in [x, y], dead or alive.
- Bx,y is true if the agent perceives a breeze in [x, y].
- Sx,y is true if the agent perceives a stench in [x, y].
- The sentences we write will suffice to derive ¬P1,2 (there is no pit in [1,2]),
- There is no pit in [1,1]:

  R1 : ¬P1,1 .
- A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:

  R2 : B1,1 ⇔ (P1,2 ∨ P2,1) .

  R3 : B2,1 ⇔ (P1,1 ∨ P2,2 ∨ P3,1) .
- The preceding sentences are true in all wumpus worlds. Now we include the breeze percepts for the first two squares visited in the specific world the agent is in, leading upto the situation in Figure 7.3(b).

  R4 : ¬B1,1 .

  R5 : B2,1 .

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Figure 7.9 A truth table constructed for the knowledge base given in the text. *KB* is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, $P_{1,2}$ is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

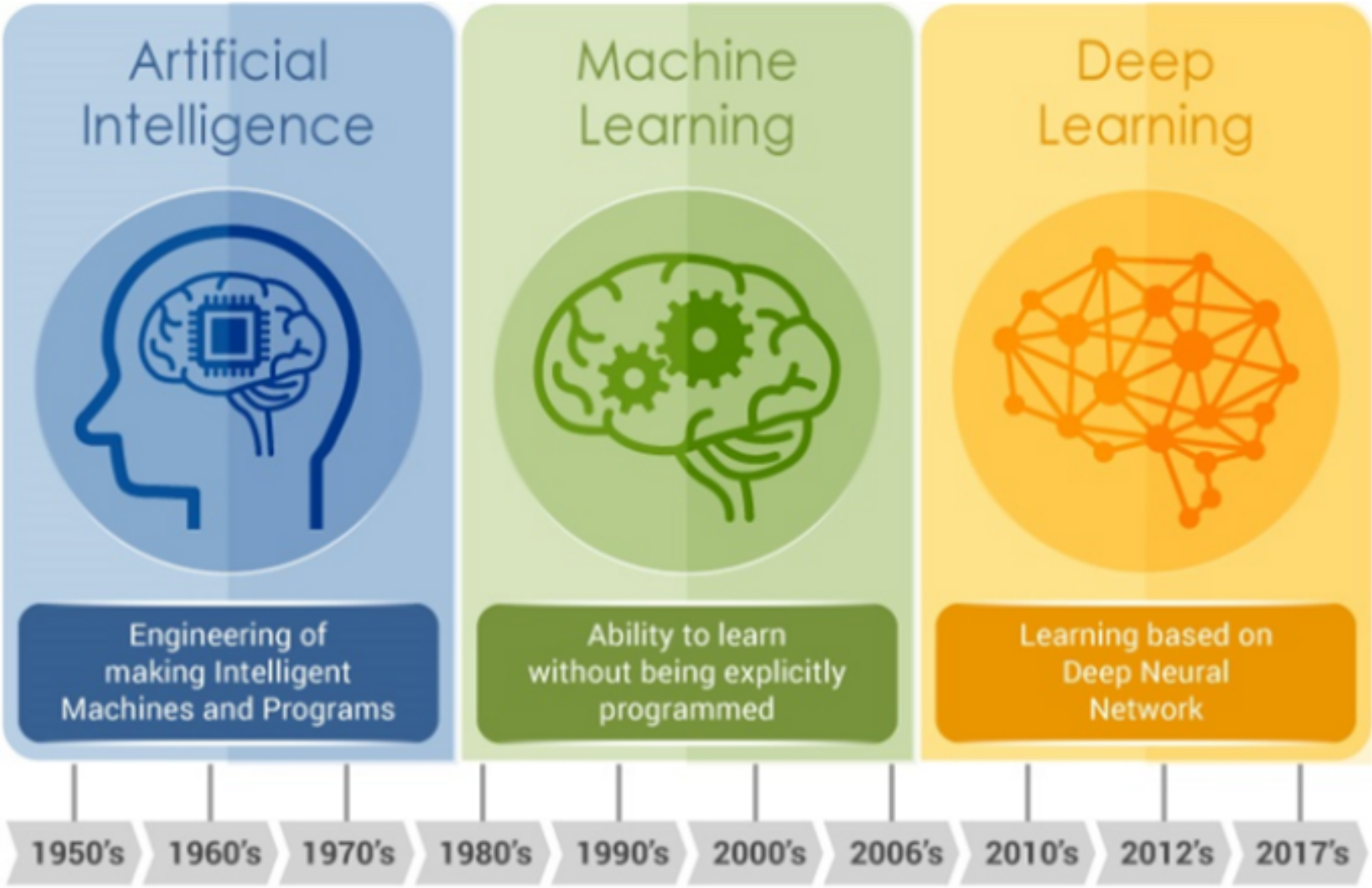
Inference by enumeration

```
function TT-ENTAILS?(KB, α) returns true or false
    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true
    else do
        P ← FIRST(symbols); rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

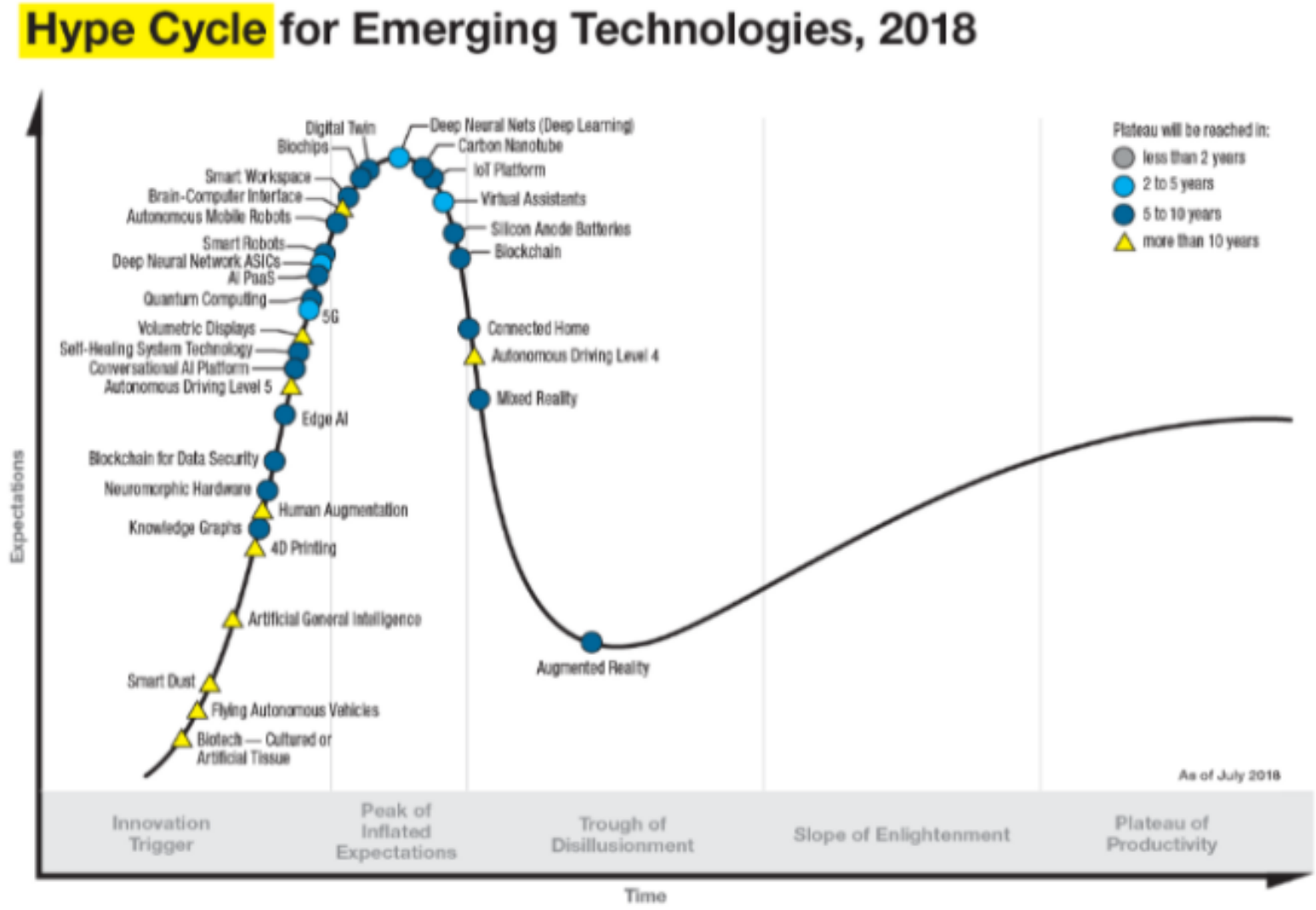
- For *n* symbols, time complexity is $O(2^n)$, space complexity is $O(n)$
- PL-True = Evaluate a propositional logical sentence in a model

  TT-Entails = Say if a statement is entailed by a KB
- Extend = Copy the s and extend it by setting var to val; return copy

Difference between AI,ML & DL



Difference between AI,ML & DL



ARTIFICIAL INTELLIGENCE

IS NOT NEW

ARTIFICIAL INTELLIGENCE

Any technique which enables computers to mimic human behavior

MACHINE LEARNING

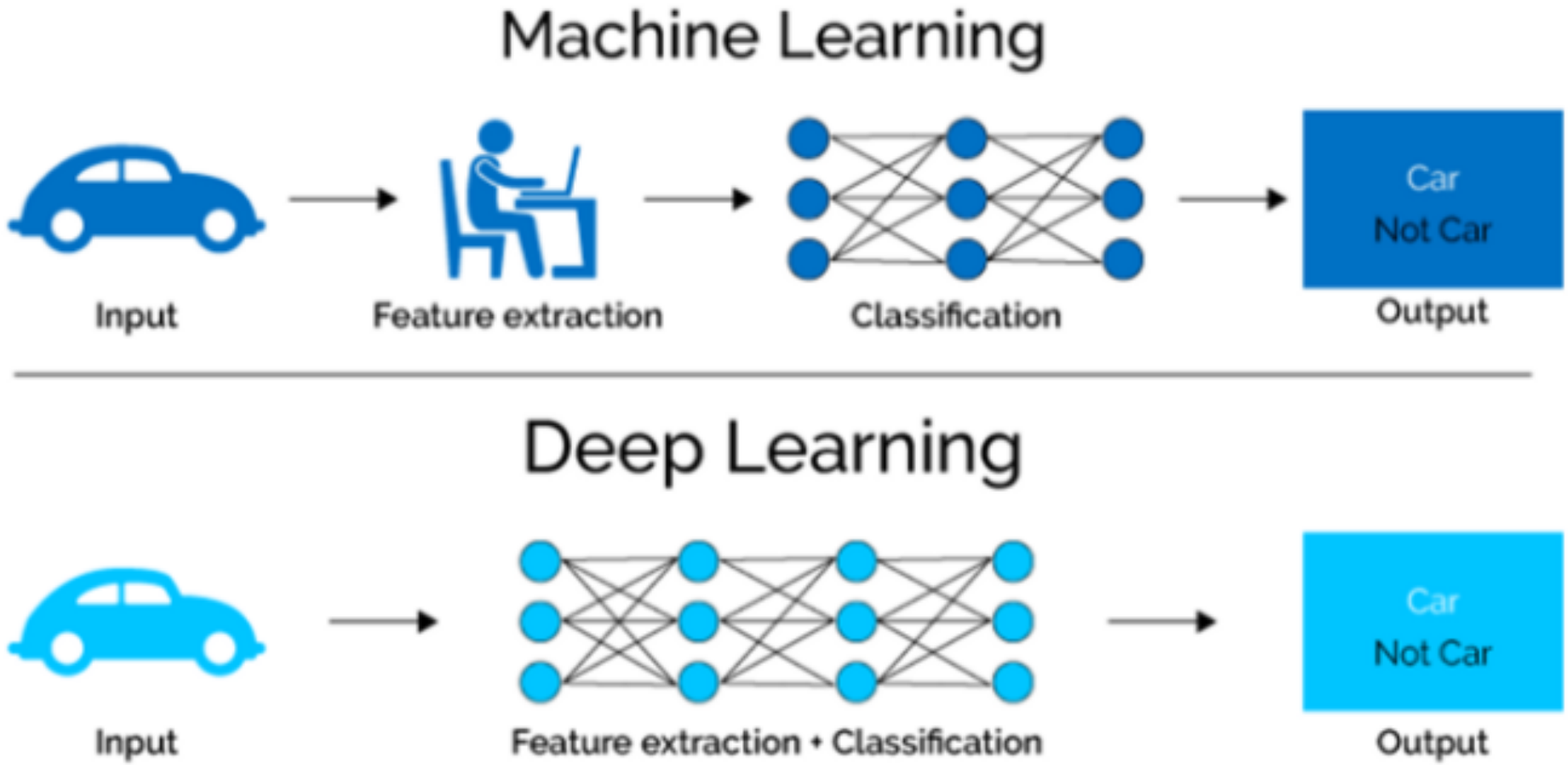
AI techniques that give computers the ability to learn without being explicitly programmed to do so

DEEP LEARNING

A subset of ML which make the computation of multi-layer neural networks feasible

1950's 1960's 1970's 1980's 1990's 2000's 2010s

ORACLE


Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output

- Labeled data
- Direct feedback
- Predict outcome/future

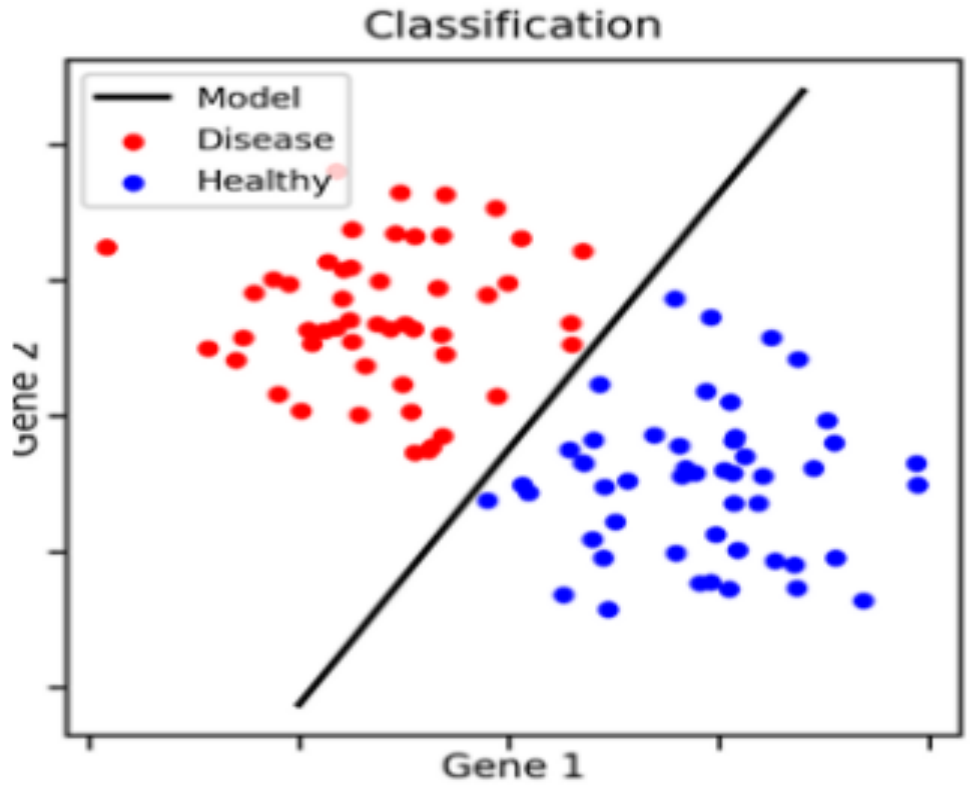
- No labels
- No feedback
- "Find hidden structure"

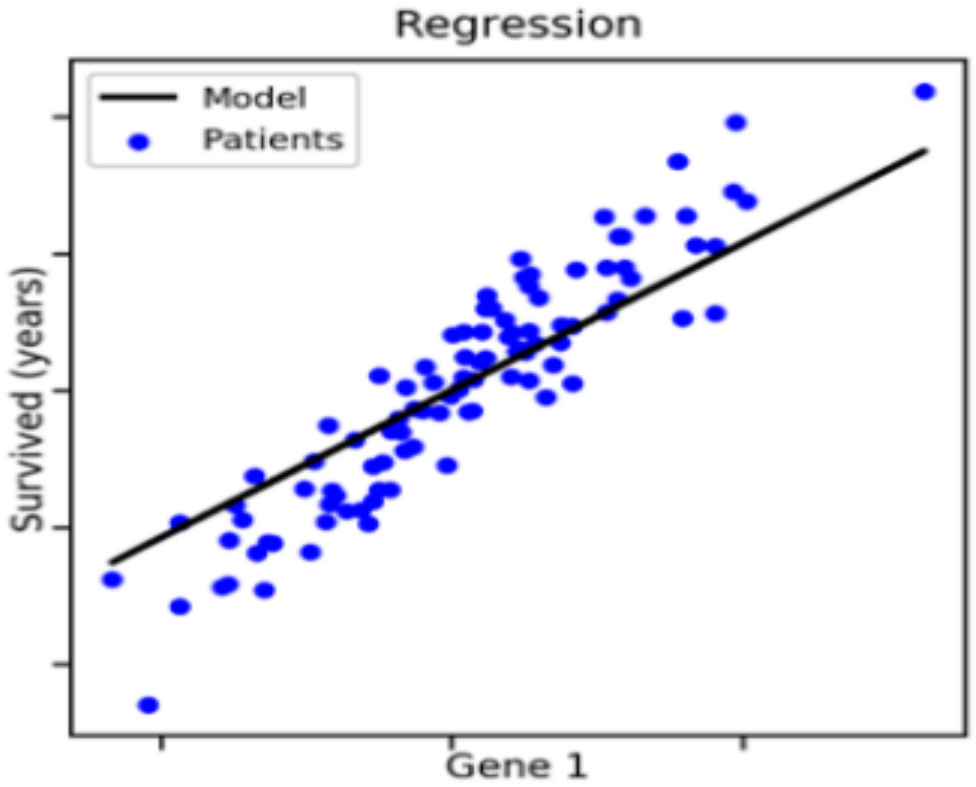
- Decision process
- Reward system
- Learn series of actions

Regression Vs Classification Problems

Regression: This is a type of problem where we need to predict and forecast for the continuous-response values.

Classification: Where you need to categorize a certain observation into a group.





How to become a practitioner with machine learning?

- (1) Learn some Python to get started and
- (2) experiment with Keras (or one of the other popular DL libraries below).
- (3) Take a practical real world problem and tackle it.

AI & ethics

- *Microsoft CEO Satya Nadella* *" We need to take accountability for the AI we create..."*
- Will AI create unemployment?
- Biased robots
- Security/Privacy
- Inequality of AI capabilities
- Artificial errors and mistakes
- Human interactions & cognitive skills
- Finally, the Singularity

THANK YOU