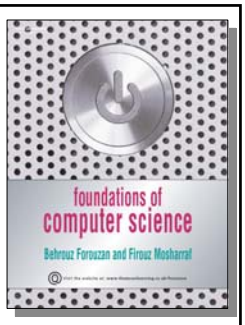
18 Artificial Intelligence



Objectives

After studying this chapter, the student should be able to:

- Define and give a brief history of artificial intelligence.
- Describe how knowledge is represented in an intelligent agent.
- Show how expert systems can be used when a human expert is not available.
- Show how an artificial agent can be used to simulate mundane tasks performed by human beings.
- Show how expert systems and mundane systems can use different search techniques to solve problems.
- Show how the learning process in humans can be simulated, to some extent, using neural networks that create the electronic version of a neuron called a perceptron.

18-1 INTRODUCTION

In this section we first try to define the term artificial intelligence (AI) informally and give a brief history of it. We also define an intelligent agent and its two broad categories. Finally, we mention two programming languages that are commonly used in artificial intelligence.

What is artificial intelligence?

Although there is no universally-agreed definition of artificial intelligence, we accept the following definition that matches the topics covered in this chapter:

Artificial intelligence is the study of programmed systems that can simulate, to some extent, human activities such as perceiving, thinking, learning and acting.

A brief history of artificial intelligence

Although artificial intelligence as an independent field of study is relatively new, it has some roots in the past. We can say that it started 2,400 years ago when the Greek philosopher **Aristotle invented the concept of logical reasoning**. The effort to finalize the language of logic continued with **Leibniz and Newton**. **George Boole** developed Boolean algebra in the nineteenth century (Appendix E) that laid the foundation of computer circuits. However, the main idea of a thinking machine came from Alan Turing, who proposed the Turing test. The term "**artificial intelligence**" was first coined by John McCarthy in **1956**.

18.5

The Turing test

In 1950, Alan Turing proposed the Turing test, which provides a definition of intelligence in a machine. The test simply compares the intelligent behavior of a human being with that of a computer. An interrogator asks a set of questions that are forwarded to both a **computer** and a **human** being. The interrogator receives two sets of responses, but does not know which set comes from the human and which set from the computer. After careful examination of the two sets, if the interrogator cannot definitely tell which set has come from the computer and which from the human, the computer has passed the Turing test for intelligent behavior.

18.6

Intelligent agents

An **intelligent agent** is a system that perceives its environment, learns from it, and interacts with it intelligently. Intelligent agents can be divided into two broad categories: **software agents** and **physical agents**.

Software agents

A software agent is a set of programs that are designed to do particular tasks. For example, a software agent can check the contents of received e-mails and classify them into different categories (junk, less important, important, very important and so on). Another example of a software agent is a search engine used to search the World Wide Web and find sites that can provide information about a requested subject.

18.7

Physical agents

A physical agent (robot) is a programmable system that can be used to perform a variety of tasks. Simple robots can be used in manufacturing to do routine jobs such as assembling, welding, or painting. Some organizations use **mobile robots** that do routine delivery jobs such as distributing mail or correspondence to different rooms. Mobile robots are used underwater to prospect for oil. A **humanoid robot** is an autonomous mobile robot that is supposed to behave like a human.

18.8

Programming languages

Although some all-purpose languages such as C, C++ and Java are used to create intelligent software, two languages are specifically designed for AI: LISP and PROLOG.

LISP

LISP (LISt Programming) was invented by John McCarthy in 1958. As the name implies, LISP is a programming language that manipulates lists.

PROLOG

PROLOG (PROgramming in LOGic) is a language that can build a database of facts and a knowledge base of rules. A program in PROLOG can use logical reasoning to answer questions that can be inferred from the knowledge base.

18.9

18-2 KNOWLEDGE REPRESENTATION

If an artificial agent is supposed to solve some problems related to the real world, it somehow needs to be able to represent knowledge. Facts are represented as data structures that can be manipulated by programs stored inside the computer. In this section, we describe four common methods for representing knowledge: **semantic networks**, **frames**, **predicate logic** and **rule-based systems**.

18.10

Semantic networks

A semantic network uses directed graphs to represent knowledge. A directed graph as discussed in Chapter 12 is made of vertices (nodes) and edges (arcs).

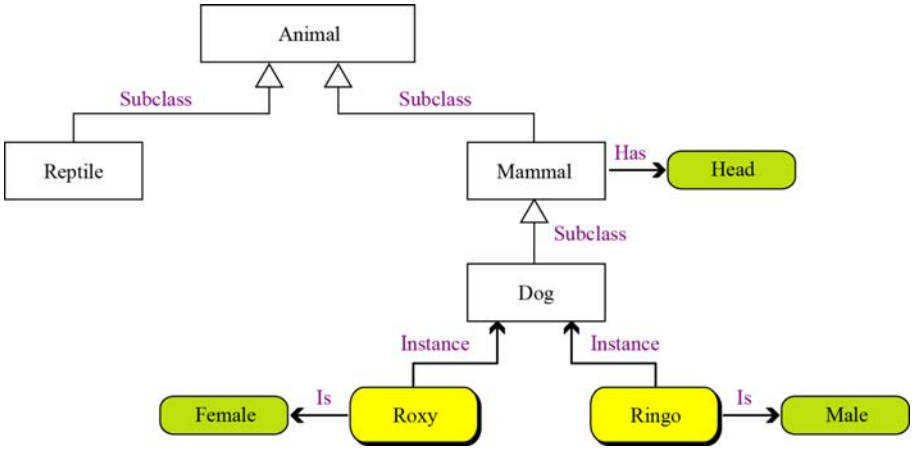


Figure 18.1 **A simple semantic network**

18.11

Concepts

A concept can be thought of as a set or a subset. For example, animal defines the set of all animals, horse defines the set of all horses and is a subset of the set animal.

Relations

In a semantic network, **relations are shown by edges**. An edge can define a subclass relation, an instance relation attribute, an object (color, size, …), or a property of an object.

18.12

Frames

Frames are closely related to semantic networks. In frames, data structures (records) are used to represent the same knowledge. One advantage of frames over semantic networks is that programs can handle frames more easily than semantic networks.

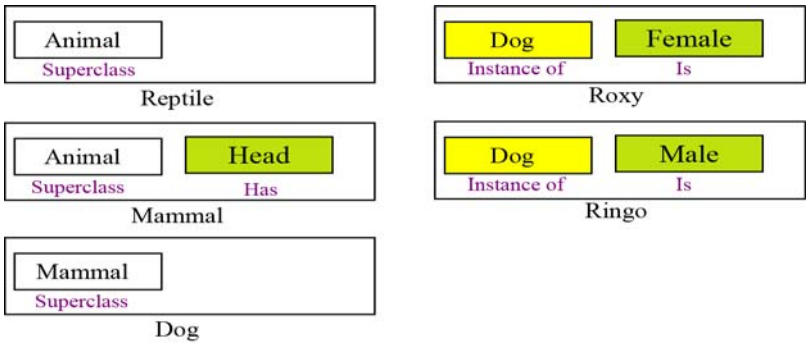


Figure 18.2 **A set of frames representing the semantic network in Figure 18.1**

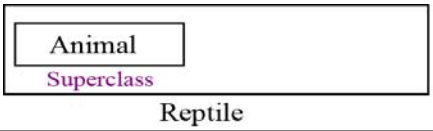
18.13

Objects

A node in a semantic network becomes an object in a set of frames, so an object can define a class, a subclass or an instance of a class. In Figure 18.2, reptile, mammal, dog, Roxy and Ringo are objects.

Slots

Edges in semantic networks are translated into **slots—fields in the data structure**. The name of the slot defines the type of the relationship and the value of the slot completes the relationship. In Figure 18.2, for example, animal is a slot in the reptile object.



18.14

Predicate logic

The most common knowledge representation is predicate logic. Predicate logic can be used to represent complex facts. It is a well-defined language developed via a long history of theoretical logic. Although this section defines predicate logic, we first introduce **propositional logic**, a simpler language. We then discuss **predicate logic**, which employs propositional logic.

18.15

Propositional logic

Propositional logic is a language made up from a set of sentences that can be used to carry out logical reasoning about the world. Propositional logic uses **five operators**.

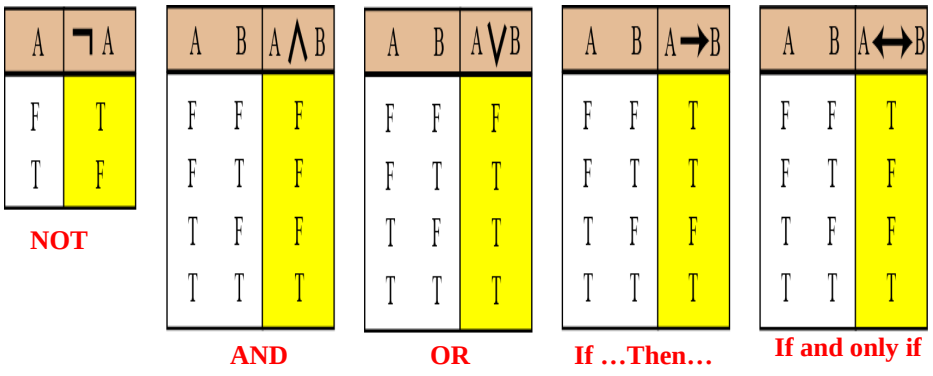


Figure 18.3 **Truth table for five operators in propositional logic**

18.16

A **sentence** in this language is defined recursively as shown below:

1. An uppercase letter, such as A, B, S or T, that represents a statement in a natural languages, is a sentence.
2. Any of the two constant values (true and false) is a sentence.
3. If P is a sentence, then ¬P is a sentence.
4. If P and Q are sentences, then P∨Q, P∧Q, P→Q and P↔Q are sentences.

18.17

Example 18.1

The following are sentences in propositional language:

a. Today is Sunday (S).
b. It is raining (R).
c. Today is Sunday or Monday (S ∨ M).
d. It is not raining (¬ R).
e. If a dog is a mammal then a cat is a mammal (D → C).

18.18

Deduction

In AI we need to create new facts from the existing facts. In propositional logic, the process is called deduction. Given two presumably *true* sentences, we can deduce a new *true* sentence. For example:

Either he is at home or at the office	Premise 1:
He is not at home	Premise 2:
Therefore, he is at the office	Conclusion

If we use H for "he is at home", O for "he is at office" and the symbol |– for the "therefore", then we can show the above argument as:

$$\{H \vee O,\ \neg H\} \vdash O$$

18.19

One way to find if an argument is **valid** is to create a truth table for the premisses and the conclusion. A conclusion is **invalid** if we can find a counterexample case: a case in which both premisses are true, but the conclusion is false.

An argument is valid if no counterexample can be found.

18.20

Example 18.2

The validity of the argument {H ∨ O, ¬H} |− O can be proved using the following truth table:

H	*O*	*H* ∨ *O*	¬*H*	*O*	
F	F	F	T	**F**	
F	T	T	T	**T**	OK
T	F	T	F	**F**	
T	T	T	F	**T**	
		Premise	Premise	Conclusion	

The only row to be checked is the second row. This row does not show a counterexample, so the argument is valid.

18.21

Example 18.3

The argument {R → C, C} |− R is not valid because a counter example can be found:

R	*C*	*R* → *C*	*C*	*R*
F	F	T	F	**F**
F	T	T	T	**F**
T	F	F	F	**T**
T	T	T	T	**T**
		Premise	Premise	Conclusion

Here row 2 and row 4 need to be checked. Although row 4 is ok, row 2 shows a counterexample (two true premisses result in a false conclusion). The argument is therefore invalid.

18.22

Predicate logic

In propositional logic, a symbol that represents a sentence is atomic: it cannot be broken up to find information about its components. For example, consider the sentences:

P_1: "Linda is Mary's mother" P_2: "Mary is Anne's mother"

We can combine these two sentences in many ways to create other sentences, but we cannot extract any relation between Linda and Anne. For example, we cannot infer from the above two sentences that Linda is the grandmother of Anne. To do so, we need predicate logic: the logic that defines the relation between the parts in a proposition.

18.23

In predicate logic, a sentence is divided into a ***predicate*** and ***arguments***. For example, each of the following propositions can be written as predicates with two arguments:

P_1: "Linda is Mary's mother"	becomes	mother (Linda, Mary)
P_2: "Mary is Anne's mother"	becomes	mother (Mary, Anne)

The relationship of motherhood in each of the above sentences is defined by the predicate *mother*. If the object Mary in both sentences refers to the same person, we can infer a new relation between Linda and Anne:

grandmother **(**Linda**,** Anne**)**

18.24

A sentence in predicate language is defined as follows:

1. A predicate with *n* arguments is a sentence.
2. Any of the two constant values (true and false) is a sentence.
3. If P is a sentence, then ¬P is a sentence.
4. If P and Q are sentences, then P ∧ Q, P ∨ Q, P → Q, and P ↔ Q are sentences.

18.25

Example 18.4

1. The sentence "John works for Ann's sister" can be written as: Works [John, sister(Ann)] in which the function sister(Ann) is used as an argument.
2. The sentence "John's father loves Ann's sister" can be written as: loves[father(John), sister(Ann)]

18.26

Predicate logic allows us to use **quantifiers**. Two quantifiers are common in predicate logic:

$$\forall \text{ and } \exists$$

1. The first, which is read as "**for all**", is called the **universal quantifier**: it states that something is true for every object that its variable represents.
2. The second, which is read as "**there exists**", is called the **existential quantifier**: it states that something is true for one or more objects that its variable represents.

18.27

Example 18.5

1. The sentence "All men are mortals" can be written as:

$$\forall x[\text{man}(x) \rightarrow \text{mortal}(x)]$$

2. The sentence "Frogs are green" can be written as:

$$\forall x[\text{frog}(x) \rightarrow \text{green}(x)]$$

3. The sentence "Some flowers are red" can be written as:

$$\exists x[\text{flower}(x) \wedge \text{red}(x)]$$

18.28

Example 18.5 Continued

4. The sentence "John has a book" can be written as:

$$\exists x[\text{book}(x) \wedge \text{has}(\text{John}, x)]$$

5. The sentence "No frog is yellow" can be written as:

$$\forall x[\text{frog}(x) \rightarrow \neg \text{yellow}(x)]$$

or

$$\neg \exists x[\text{frog}(x) \wedge \text{yellow}(x)]$$

18.29

Deduction

In predicate logic, if there is no quantifier, the verification of an argument is the same as that which we discussed in propositional logic. However, the verification becomes more complicated if there are quantifiers. For example, the following argument is completely valid.

All men are mortals	Premise 1:
Socrates is a man	Premise 2:
Therefore, Socrates is mortal	Conclusion

Verification of this simple argument is not difficult. We can write this argument as shown next:

18.30

$$\forall x\,[\text{man}(x) \rightarrow \text{mortal}(x)],\ \text{man}(\text{Socrates}) \vdash \text{mortal}(\text{Socrates})$$

Since the first premise talks about all men, we can replace one instance of the class man (Socrates) in that premise to get the following argument:

$$\text{man}(\text{Socrates}) \rightarrow \text{mortal}(\text{Socrates}),\ \text{man}(\text{Socrates}) \vdash \text{mortal}(\text{Socrates})$$

Which is reduced to M1 → M2, M1 |− M2, in which M1 is man(Socrates) and M2 is mortal(Socrates). The result is an argument in propositional logic and can be easily validated.

18.31

Beyond predicate logic

There have been further developments in logic to include the need for logical reasoning. Some examples of these include **high-order logic**, **default logic**, **modal logic** and **temporal logic**.

18.32

Rule-based systems

A rule-based system represents knowledge using a set of rules that can be used to deduce new facts from known facts. The rules express what is true if specific conditions are met. **A rule-based database is a set of *if… then…* statements** in the form

If A then B **or** A → B

in which A is called the ***antecedent*** and B is called the ***consequent***. Note that in a rule-based system, each rule is handled independently without any connection to other rules.

18.33

Components

A rule-based system is made up of three components: an **interpreter** (or **inference engine**), a **knowledge base** and a **fact database**, as shown in Figure 18.4.

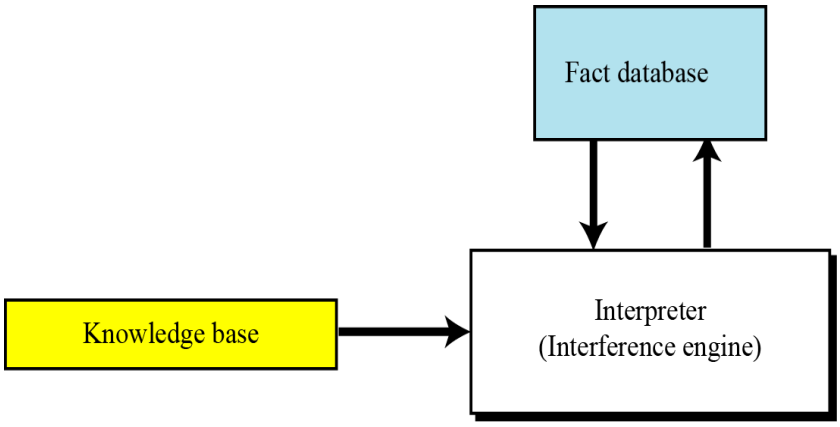


Figure 18.4 **The components of a rule-based system**

18.34

Forward chaining

Forward chaining is a process in which an interpreter uses a set of rules and a set of facts to perform an action.

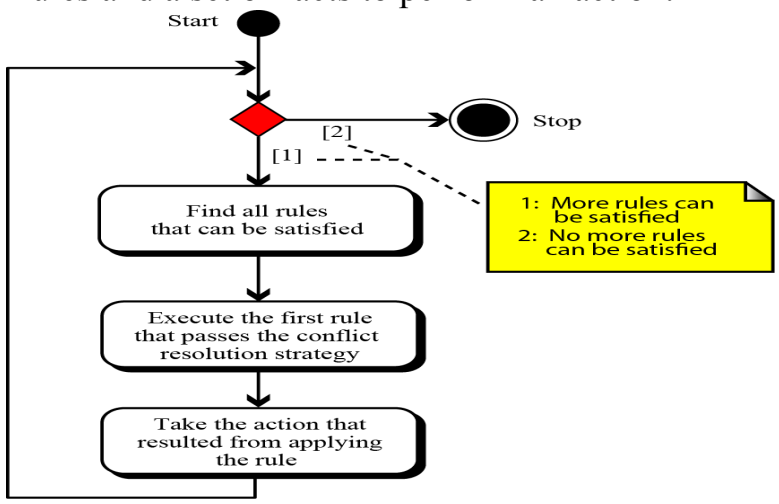


Figure 18.5 Flow diagram for forward chaining

18.35

Backward chaining

Forward chaining is not very efficient if the system tries to prove a conclusion. In this case, it may be more efficient if backward chaining is used.

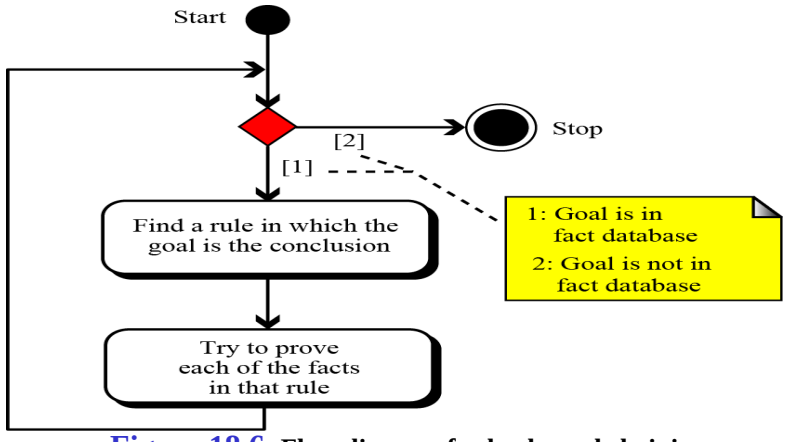


Figure 18.6 Flow diagram for backward chaining

18.36

18-3 EXPERT SYSTEMS

Expert systems use the knowledge representation languages discussed in the previous section to perform tasks that normally need human expertise. They can be used in situations in which that expertise is in short supply, expensive or unavailable when required. For example, in medicine, an expert system can be used to narrow down a set of symptoms to a likely subset of causes, a task normally carried out by a doctor.

18.37

Extracting knowledge

An expert system is built on predefined knowledge about its field of expertise. An expert system in medicine, for example, is built on the knowledge of a doctor specialized in the field for which the system is built: an expert system is supposed to do the same job as the human expert. The first step in building an expert system is, therefore, to extract the knowledge from a human expert. This extracted knowledge becomes the knowledge base we discussed in the previous section.

To be able to infer new facts or perform actions, a fact database is needed in addition to the knowledge base for a knowledge representation language. The fact database in an expert system is case-based, in which facts collected or measured are entered into the system to be used by the inference engine.

Architecture

Figure 18.7 shows the general idea behind the architecture of an expert system.

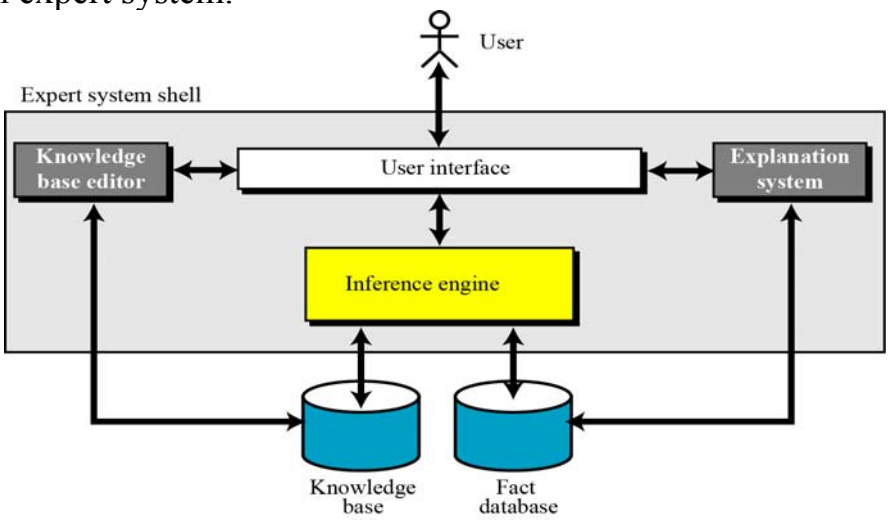


Figure 18.7 The architecture of an expert system

18.39

18-4 PERCEPTION

Another goal of AI is to create a machine that behaves like an ordinary human. One of the meanings of the word "**perception" is understanding** what is received through the senses—sight, hearing, touch, smell, taste. A human being sees a scene through the eyes, and the brain interprets it to extract the type of objects in the scene. A human being hears a set of voice signals through the ears, and the brain interprets it as a meaningful sentence, and so on.

18.40

Image processing

Image processing or **computer vision** is an area of AI that deals with the perception of objects through the artificial eyes of an agent, such as a camera. An image processor takes a two-dimensional image from the outside world and tries to create a description of the three-dimensional objects present in the scene. Although, this is an easy tasks for a human being, it turns out to be a difficult task for an artificial agent. The input presented to an image process is one or more images from the scene, while the output is a description of the objects in the scene. The processor uses a database containing the characteristics of objects for comparison.

18.41

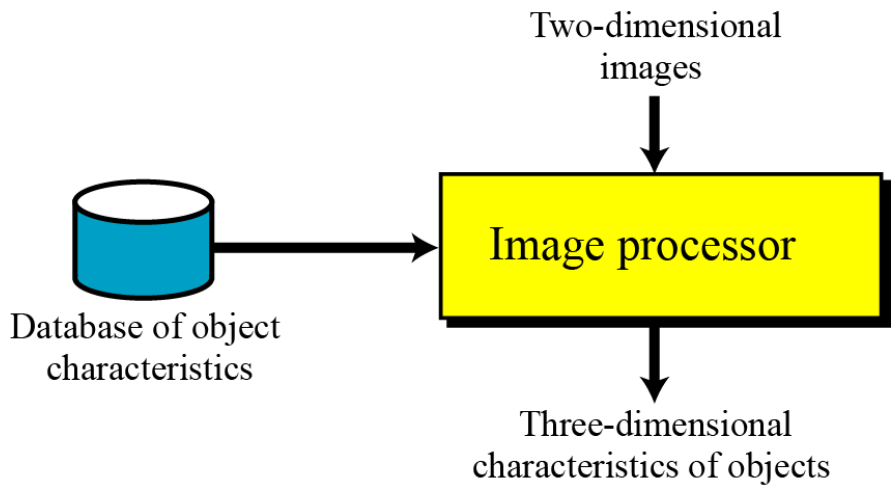
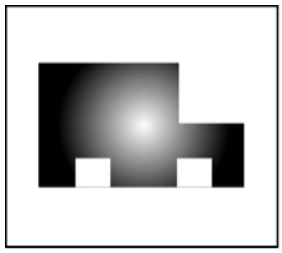


Figure 18.8 **The components of an image processor**

18.42

Edge detection

The first stage in image processing is edge detection: **finding where the edges in the image are**. Edges can define the boundaries between an object and its background in the image. Normally there is a sharp contrast between the surfaces belonging to an object and the environment, assuming that there is no camouflage.



Image

9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9
9	2	3	2	2	9	9	9
9	2	4	7	4	9	9	9
9	2	3	4	5	3	2	9
9	2	9	3	2	9	2	9
9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9

Intensity of pixels

Figure 18.9 **The edge-detection process**

18.43

Segmentation

Segmentation is the next stage in image analysis. **Segmentation divides the image into homogenous segments or areas**. The definition of homogeneity differs in different methods, but in general, a homogenous area is an area in which the intensity of pixels varies smoothly. Segmentation is very similar to edge detection. In edge detection, the boundaries of the object and the background are found: in segmentation, the boundaries between different areas inside the object are found. After segmentation, the object is divided into different areas.

18.44

Finding depth

The next step in image analysis is to find the depth of the object or objects in the image. Two general methods have been used for this purpose: **stereo vision and motion**.

Finding orientation

Orientation of the object in the scene can be found using two techniques: **shading** and **texture**.

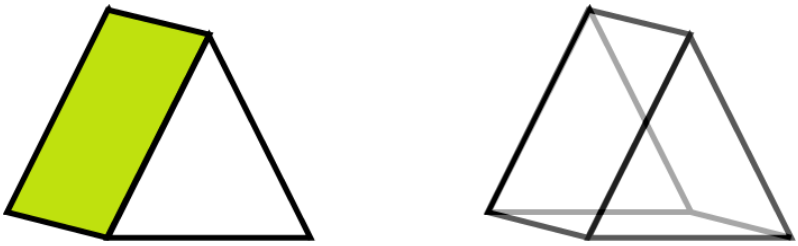


Figure 18.10 The effect of shading on orientation finding

18.45

Object recognition

The last step in image processing is object recognition. To recognize an object, the agent needs to have a model of the object in memory for comparison. However, creating and storing a model for each object in the view is an impossible task. One solution is to assume that the objects to be recognized are compound objects made of a set of simple geometric shapes.

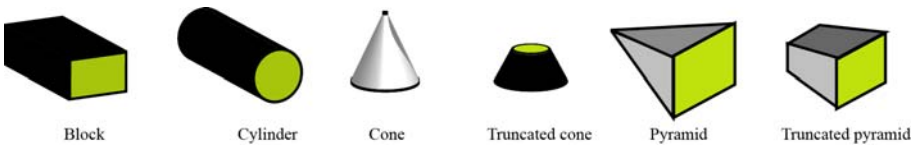


Figure 18.11 Primitive geometric shapes

18.46

Language understanding

One of the inherent capabilities of a human being is to understand—that is, interpret—the audio signal that they perceive. A machine that can understand natural language can be very useful in daily life. For example, it can replace a telephone operator—most of the time. It can also be used on occasions when a system needs a predefined format of queries. We can divide the task of a machine that understands natural language into four consecutive steps: **speech recognition**, **syntactic analysis**, **semantic analysis** and **pragmatic analysis**.

18.47

Speech recognition

The first step in natural language processing is speech recognition. In this step, a speech signal is analyzed and the sequence of words it contains are extracted. The input to the speech recognition subsystem is a continuous (analog) signal: the output is a sequence of words. The signal needs to be divided into different sounds, sometimes called *phonemes*. The sounds then need to be combined into words. The detailed process, however, is beyond the scope of this book: we leave the task to specialized books in speech recognition.

18.48

Syntactic analysis

The syntactic analysis step is used to define how words are to be grouped in a sentence. This is a difficult task in a language like English, in which the function of a word in a sentence is not determined by its position in the sentence. For example, in the following two sentences:

Mary rewarded John.
John was rewarded by Mary.

it is always John who is rewarded, but in the first sentence John is in the last position and Mary is in the first position. A machine that hears any of the above sentences needs to interpret them correctly and come to the same conclusion no matter which sentence is heard.

18.49

Grammar

The first tool to correctly analyze a sentence is a well-defined grammar. We use a simple version of BNF (Backus-Naur Form) that is used in computer science to define the syntax of a programming language (Table 18.1).

Table 18.1 A simple grammar

	Rule		
1	**Sentence**	→	NounPhrase VerbPhrase
2	**NounPhrase**	→	Noun | Article Noun | Article Adjective Noun
3	**Verb Phrase**	→	Verb | Verb NounPhrase | Verb NounPhrase Adverb
4	**Noun**	→	[home] | [cat] | [water] | [dog] | [John] | [Mary] |
5	**Article**	→	[a] | [the]
6	**Adjective**	→	[big] | [small] | [tall] | [short] | [white] | [black] |
7	**Verb**	→	[goes] | [comes] | [eats] | [drinks] | [has]

18.50

Parser

It should be clear that even a simple grammar as defined in Table 18.1 uses different options. A machine that determines if a sentence is grammatically (syntactically) correct does not need to check all possible choices before rejecting a sentence as an invalid one. This is done by a **parser**.

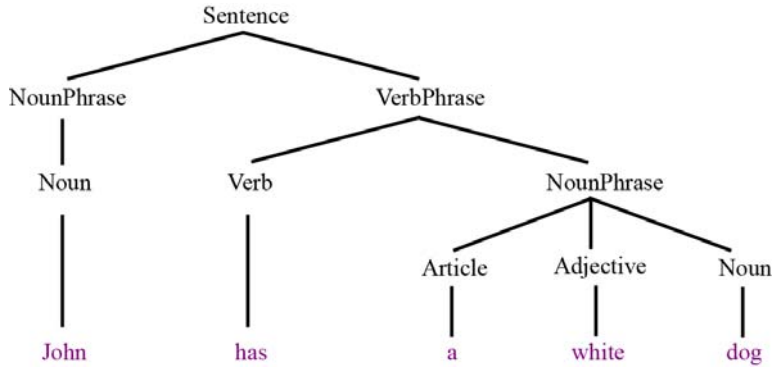


Figure 18.12 **Parsing a sentence**

18.51

Semantic analysis

The semantic analysis extracts the meaning of a sentence after it has been syntactically analyzed. This analysis creates a representation of the objects involved in the sentence, their relations and their attributes. The analysis can use any of the knowledge representation schemes we discussed before. For example, the sentence "John has a dog" can be represented using predicate logic as:

$$\exists\, x \; \text{dog}(x) \; \text{has} \; (\text{John}, x)$$

18.52

Pragmatic analysis

The three previous steps—speech recognition, syntax analysis and semantic analysis—can create a knowledge representation of a spoken sentence. In most cases, another step, pragmatic analysis, is needed to further clarify the purpose of the sentence and to remove ambiguities.

18.53

18-5 SEARCHING

One of the techniques for solving problems in artificial intelligence is searching, which is discussed briefly in this section. Searching can be describe as solving a problem using a set of states (a situation). A search procedure starts from an initial state, goes through intermediate states until finally reaching a target state. For example, in solving a puzzle, the initial state is the unsolved puzzle, the intermediate states are the steps taken to solve the puzzle and the target state is the situation in which the puzzle is solved. The set of all states used by a searching process is referred to as the **search space**.

18.54

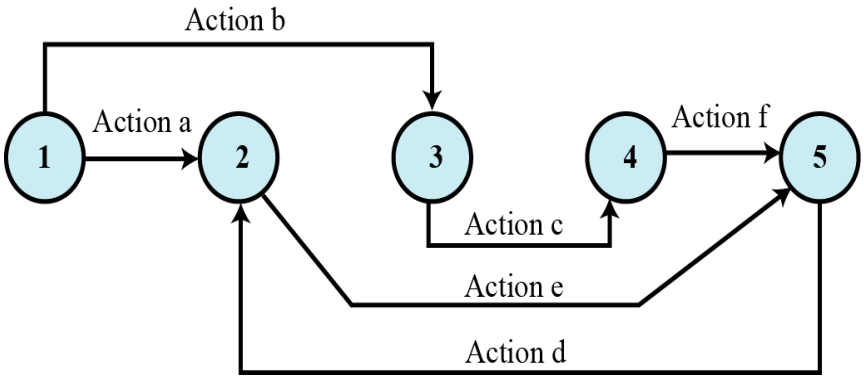


Figure 18.13 An example of a search space

18.55

Example 18.6

One example of a puzzle that shows the search space is the **famous 8-puzzle**. The tiles are numbered from 1 to 8. Given an initial random arrangement of the tiles (the initial state), the goal is to rearrange the tiles until a ordered arrangement of the tiles is reached (the target state). The rule of the game is that a tile can be slid into an empty slot.

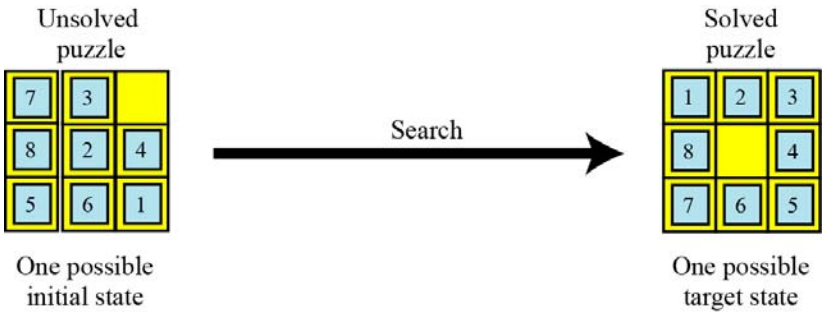


Figure 8.14 A possible initial and final state for Example 18.6

18.56

Search methods

There are two general search methods: **brute-force** and **heuristic**. The brute force method is itself either breadth-first or depth first

Brute-force search

We use brute-force search if we do not have any prior knowledge about the search. For example, consider the steps required to find our way through the maze in Figure 18.15 with points A and T as starting and finishing points respectively. The tree diagram for the maze is shown in Figure 18.16.

18.57

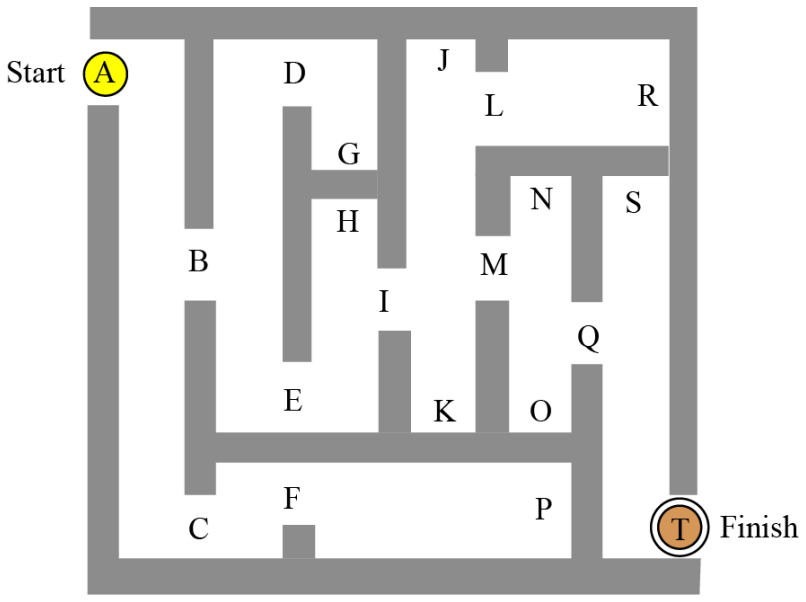


Figure 18.15 A maze used to show brute force search

18.58

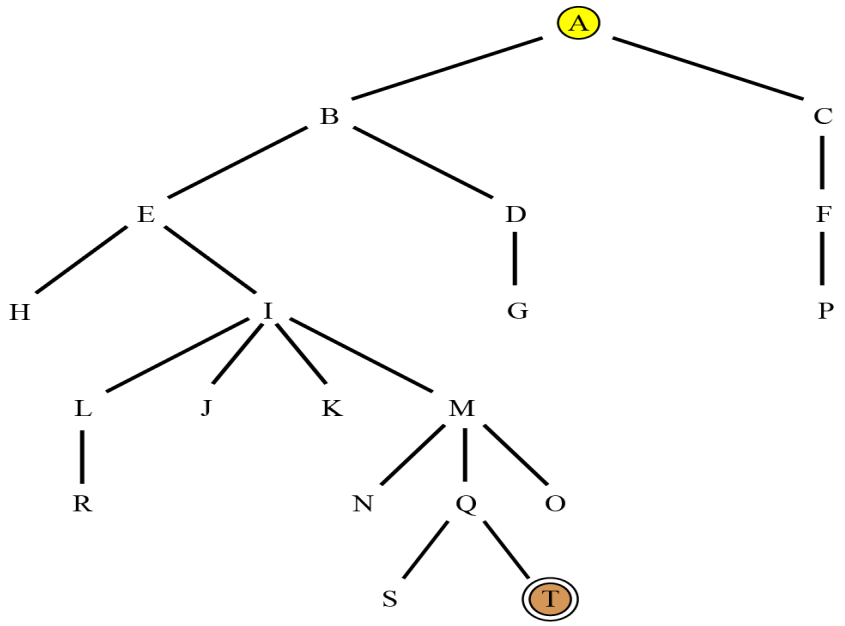


Figure 18.16 The tree for the maze in Figure 18.15

18.59

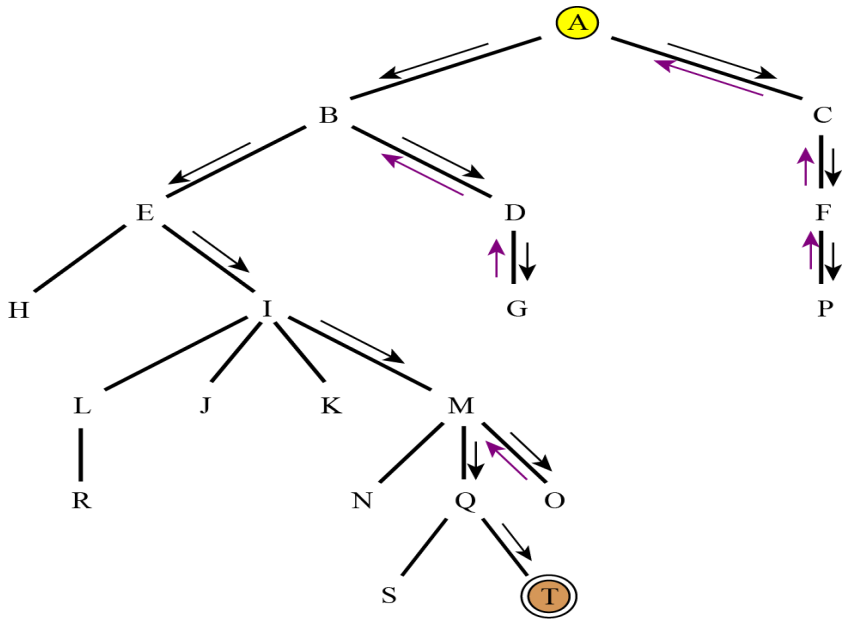


Figure 18.17 Depth-first search of the tree in Figure 18.16

18.60

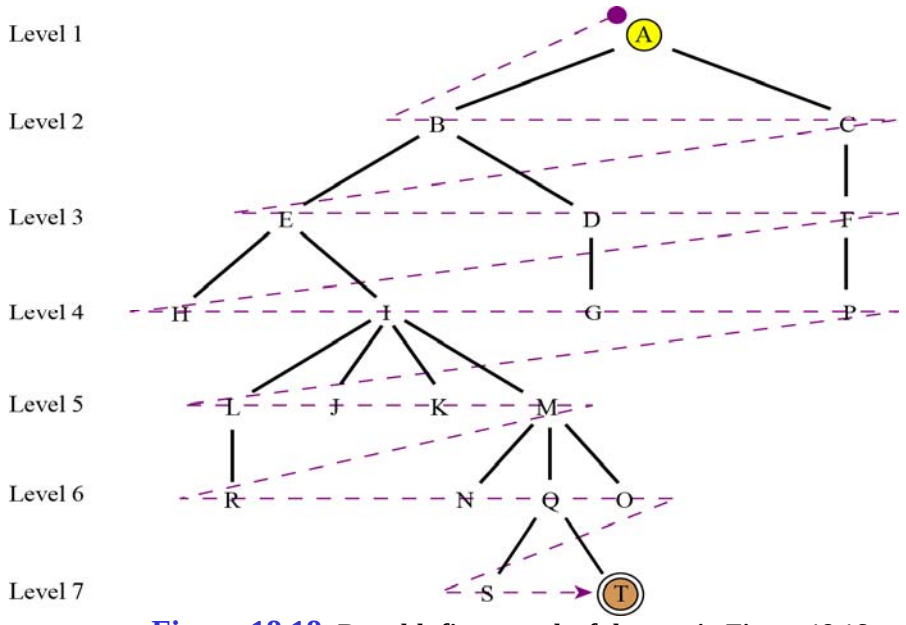


Figure 18.18 **Breadth-first search of the tree in Figure 18.16**

18.61

Heuristic search

Using heuristic search, we assign a quantitative value called a heuristic value (h value) to each node. This quantitative value shows the relative closeness of the node to the goal state. For example, consider solving the 8-puzzle of Figure 18.19.

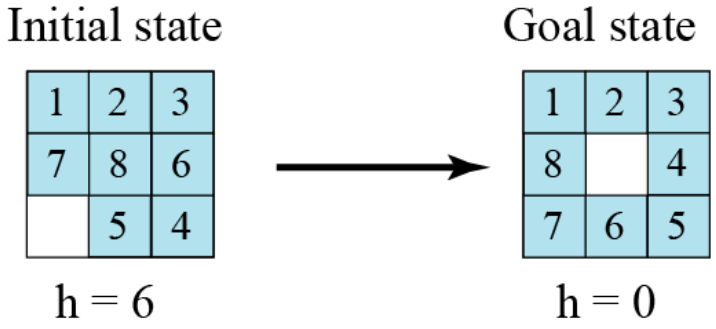
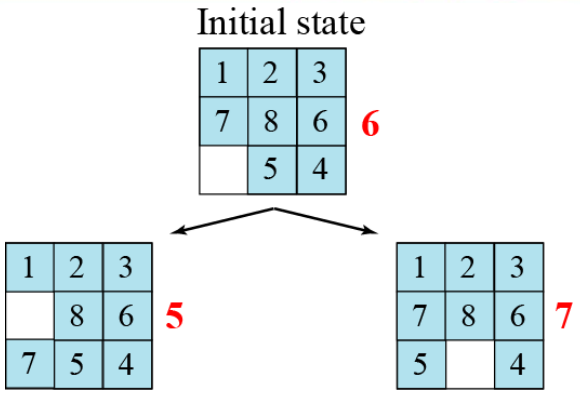


Figure 18.19 **Initial and goal states for heuristic search**

18.62

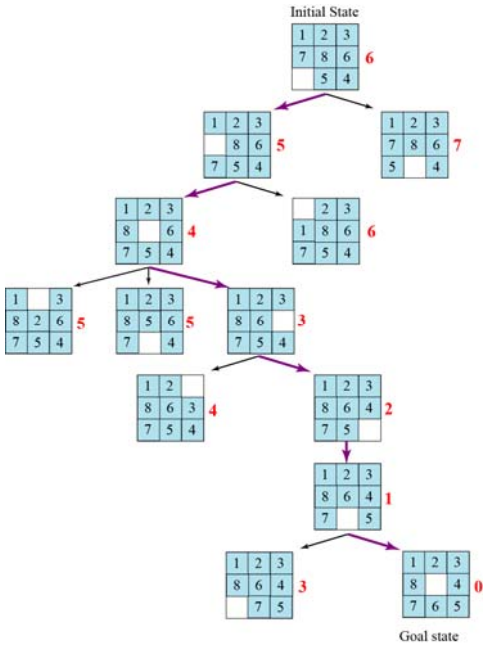
Table 18.2 Heuristic value

Tile number	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*Total*
Heuristic value of initial state	0	0	0	1	1	2	1	1	6
Heuristic value of goal state	0	0	0	0	0	0	0	0	0



18.63

Figure 18.20 The heuristic values for the first step



18.64

Figure 18.21 Heuristic search for solving the 8-puzzle

18-5 NEURAL NETWORKS

If an intelligent agent is supposed to behave like a human being, it may need to learn. Learning is a complex biological phenomenon that is not even totally understood in humans. Enabling an artificial intelligence agent to learn is definitely not an easy task. However, several methods have been used in the past that create hope for the future. Most of the methods use inductive learning or learning by example. This means that a large set of problems and their solutions is given to the machine from which to learn. In this section we discuss only neural networks.

18.65

Biological neurons

The human brain has billions of processing units, called neurons. Each neuron, on average, is connected to several thousand other neurons. A neuron is made of three parts: soma, axon and dendrites, as shown in Figure 18.22.

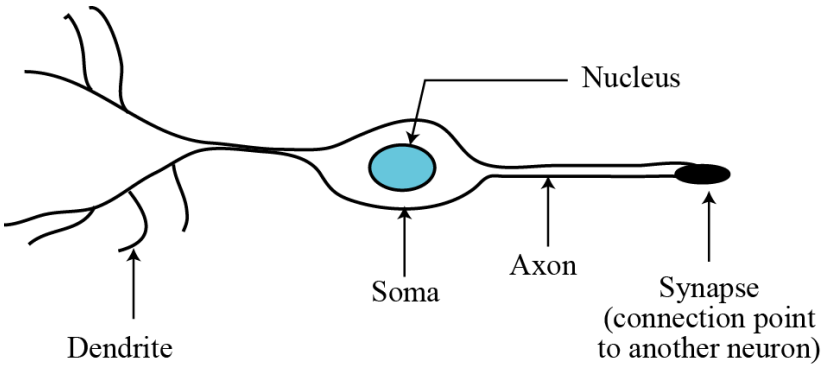


Figure 18.22 **A simplified diagram of a neuron**

18.66

Perceptrons

A perceptron is an artificial neuron similar to a single biological neuron. It takes a set of weighted inputs, sums the inputs and compares the result with a threshold value. If the result is above the threshold value, the perceptron fires, otherwise it does not. When a perceptron fires, the output is 1: when it does not fire, the output is zero.

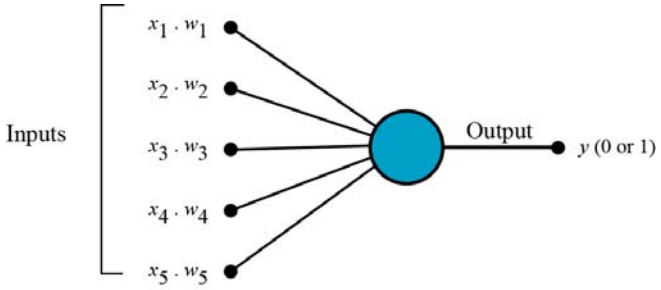


Figure 18.23 **A perceptron**

18.67

Example 18.7

Assume a case study with three inputs and one output. There are already four examples with known inputs and outputs, as shown in the following table:

Inputs			*Output*
1	0	0	0
0	0	1	0
1	0	1	0
1	1	1	1

This set of inputs is used to train a perceptron with all equal weights ($w_1 = w_2 = w_3$). The threshold is set to 0.8. The original weight for all inputs is 50%. The weights remain the same if the output produced is correct—that is, matches the actual output.

18.68

Example 18.7 Continued

The weights are increased by 10% if the output produced is less than the output data: the weights are decreased by 10% if the output produced is greater than the output data. The following table shows the process of applying the previous established examples to train the perceptron.

Inputs			Weight	Weighted sum	Output produced	Actual output	Action
1	0	0	50%	0.5	0	0	None
0	0	1	50%	1	1	0	Decrease
1	0	1	40%	8.0	0	0	None
1	1	1	40%	1.2	1	1	None

18.69

Multi-layer networks

Several layers of perceptions can be combined to create multilayer neural networks. The output from each layer becomes the input to the next layer. The first layer is called the input layer, the middle layers are called the hidden layers and the last layer is called the output layer.

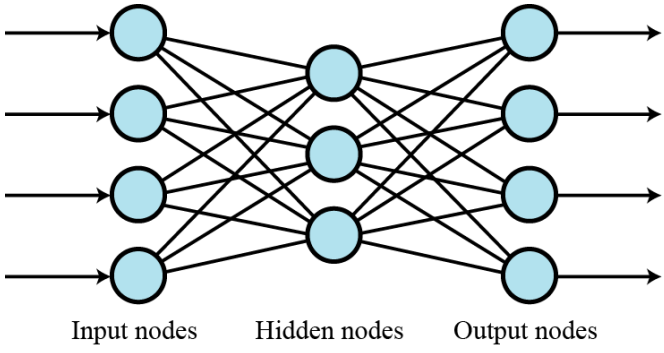


Figure 18.24 **A multi-layer neural network**

18.70

Applications

Neural networks can be used when enough pre-established inputs and outputs exist to train the network. Two areas in which neural networks have proved to be useful are **optical character recognition (OCR)**, in which the intelligent agent is supposed to read any handwriting, and credit assignment, where different factors can be weighted to establish a credit rating, for example for a loan applicant.

18.71