University Mohamed Boudiaf of M'sila
Faculty of Mathematics and Computer Science

Introduction to Artificial Intelligence

Chapter II: Knowledge representation formalisms

Dr. SAID KADRI

Associate Professor

Department of Computer Science, Faculty of Mathematics and Informatics,

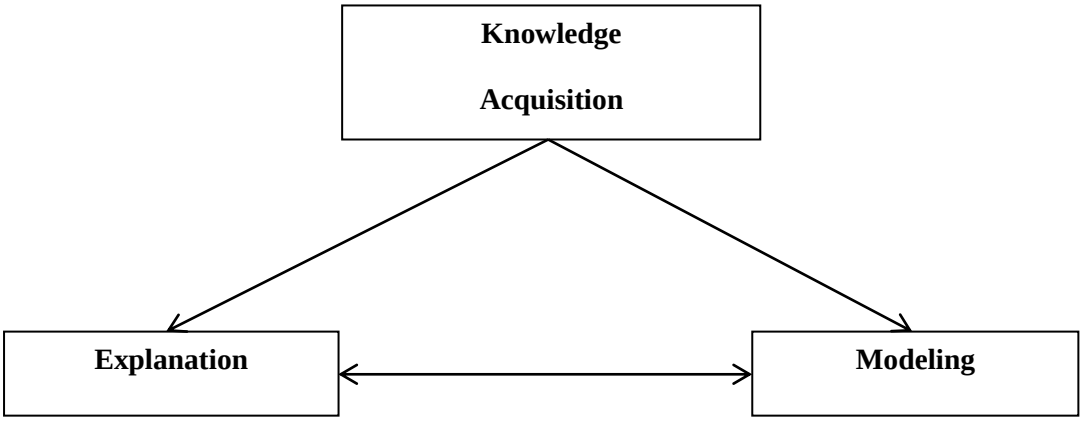
University Mohamed Boudiaf of M'sila

E-mail:kadri.said28@gmail.com

Website:https://kadrisaid28.wixsite.com/sgadri

2017 – 2018

Knowledge Acquisition Process



Problems of knowledge acquisition

- Expert knowledge is subjective (≠Objective) and often difficult to formalize.
- Formalisms used for knowledge representation are sometimes linked to the implementation.
- The used formalisms do not allow a good level of abstraction.

Types of knowledge

1.Declarative knowledge (knowledge)

- The population of Algeria was 38,000,000 inhabitants in 2011.

2.Procedural knowledge (the know-how)

- A cooking recipe => Ex : telling steps to make a chocolate cake.

3. Conceptual knowledge (combines both declarative and procedural knowledge)

- The concept of a referendum will call upon procedural knowledge:
  - Installation of polling stations and constitution of electoral lists and committees.

⇨ The declarative knowledge is: a void ballot is not counted.

Evolution of Knowledge Acquisition Processes

Knowledge Representation (≠Types)

- Digital information processing (old generations of computers)
- Alphanumeric processing (databases)
- Symbolic processing (in artificial intelligence we process facts, statements of equations, methods represented by systems of symbols, etc.)

Knowledge Representation Formalisms

1. Propositional logic (no quantifiers and no variables)
2. First-Order Predicate Logic (introduction of variables and quantifiers)
3. Production rules

Structured representation of knowledge

1. Semantic networks
2. Structured objects
3. Conceptual graphs
4. Petri nets
5. Neural networks
6. Others, …

1. Propositional logic

- A kind of formal language used to represent information and deducing conclusions from them.
- Propositional logic (propositional) is a sequence of symbols separated by conjunctions (and), disjunctions (or), or negations (not)
- Propositional logic allows us to express:

– **Real world facts:**

Ali lives in M'sila

– **Negations:**

Omar doesn't live in Msila

– **Conjunctions and disjunctions:**

Mohamed is a student, and he lives in M'sila

– **Sentences with logical "consequence":**

If the son does not meet his mother, the mother does not meet him either.

- **A proposition** : is an expression (sentence) about the world that is either (True) or (False).

Example: Represent the following expressions in propositional logic

1. *Socrates is a man* ➔ *ManSocrates/ Man(Socrates)*
2. *Plato and Socrates are men* ➔ *ManPlato∧ManSocrates*

   Man(Plato)∧Man(Socrates)
3. *Zola wrote Germinal* ➔ *(AUTHOR, ZOLA) => (BOOK, GERMINAL)*

   Author(Zola) => Book(Germinal)

Basic Elements of Propositional Logic:

– Proposition symbols: P, Q, … (sentences)

– Special sentences (values): True, False

– Operators:∧(And),∨(or), ¬ (not),⇒(implied),⇔(equivalent)

Formation of composed sentences (propositions):

1. Negation: ¬ P
2. Conjunction: P∧Q
3. Disjunction: P∨Q
4. Implication: P⇒Q (if P then Q)
5. Equivalence: P⇔Q (P and only if Q)

Examples:

1. Consider the following composed propositions:

$$\neg\,(P \wedge Q) \vee (P \Rightarrow (Q \Rightarrow R))\quad P \wedge Q \wedge (Q \Rightarrow R)$$

$$\neg\,(P \wedge Q) \vee (P \Rightarrow (Q \Rightarrow R))$$

$$((P \wedge Q) \Rightarrow R)$$

$$(HAS \Rightarrow B) \vee (\neg C)$$

$$\neg\,(P \wedge Q) \vee (P \Rightarrow (Q \Rightarrow R))$$

OBS: Order of precedence (priority) is as follows: ¬, ∧, ∨, ⇒

Examples:

- ¬A∨B⇒C is equivalent to ((¬A)∨B)⇒C
- A⇒B⇒C (incorrect/the same priority)

2. We suppose:

P = *"the son meets his mother"* ;

Q = *"The mother meets her son"*

How to represent the sentence?

"If the son does not meet his mother, then she does not meet him either"

➔ ¬ P⇒¬Q

Truth Table

P	¬ P	Q	P∧Q	P∨Q	P⇒Q	P⇔Q
V	F	V	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	V	F	F	F	V	V

OBS: The truth table helps to calculate the logical value of any sentence (a composed proposition).

Propositional Logic and Inference

Definition : Inference is a (mechanism/process) that establishes the semantics of a sentence (the truth value) by subjecting it to a series of syntactic transformations (without using TV) by reducing the original problem to a problem that depends only on known facts.

Example:

Premises:

1. "If the son does not meet his mother, she does not meet him either"
2. “Mother meets son”

Conclusion:

⇨ So "The son meets his mother" is true

Correspondence in propositional logic

Let: P ≡ "The son meets his mother"

Q ≡ "Mother meets her son"

Inference

Premises in propositional logic

1. ¬ P⇒¬ Q
2. Q

Conclusion

➔ P

Basic components of inference:

- ***Axioms:*** known facts of the world.
- ***Rules:*** syntactic transformations that propagate (generate) semantic value

Why is inference interesting?

• It provides a formalism for modeling reasoning.

• It reduces **semantic analysis** to **syntactic manipulations**

• Allows the automation of these syntactic manipulations.

Example of a knowledge base (formulated in propositional logic)

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-OK (R1)

2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Tank ⇒ Engine-Start (R2)

3. Engine-Start ∧ ¬Flat-Tire ⇒ Car-OK (R3)

4. Headlights-OK (f1)
5. Battery-OK (f2)
6. Starter-OK (f3)
7. ¬Empty-Tank (f4)
8. ¬Car-OK (f5)

9. Battery-OK ∧ Starter-OK <= (5)+(6) **(f6)**

10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Tank <= (9)+(7) **(f7)**

11. Engine-Start <= (10)+(2) **(f8)**

12. ¬Engine-Start ∨ Flat-Tire <=(**¬3)+(8)+(11)** ≡ **Engine-Start ⇒ Flat-Tire**

13. Flat-Tire <= (11+12) **(f9)**

Proof systems (Systèmes de preuve)

- A proof system is a collection of axioms and rules of inference.
- There are many proof systems for propositional logic.
- Logicians prefer "small" proof systems (one rule and few axioms/facts).
- AI specialists prefer systems with strong (and numerous) rules and few axioms.

Validity&Adequacy of sentences

1. ***Validity:*** some sentences are always true (valid) and can therefore always be assumed as valid.
2. ***Suitability:*** a rule of inference is said "adequate" if it produces from premises a conclusion that is always true.

Incomplete Proof System Vs Complete Proof System

1. ***Incomplete system:*** If we remove any axiom or rule of inference, we can not prove anything.

   ⇒We then obtain an incomplete proof system.

2. ***Complete system:*** a system with enough axioms and rules to prove any sentence (even if we remove axioms and rules we can prove any thing).

Knowledge Base

Knowledge base = set of sentences expressed in a formal language (e.g., logic).

2.First-order predicate logic

Motivation

- Propositional logic has limited expressive power (unlike natural language for example)

Example :

For instance, how to express the following sentences in propositional logic?

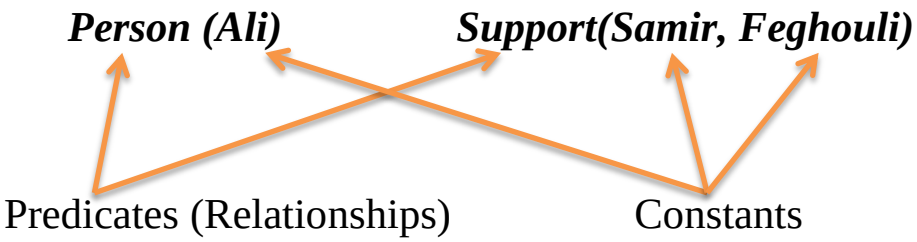
- All master's students are Algerians.
- All students were either satisfied with their teacher's course or they were not.
- Every student has a friendship relationship with another.

Definition of predicate logic

- A sequence of symbols, variables and relationships with universal and existential quantifiers (∀, ∃).
- Instead of the propositional symbols P, Q, …, use predicates and terms:

Examples

1) Ali is a person Samir supports Feghouli



2) Mourad gives a book to Salim

Give (Mourad, a book, Salim)

3) 2 + 2 = 4

Equal(Plus(2,2),4)

Predicate Function

Basic Elements of Predicate Logic

- Constants: Ali, 2, Unix, ...
- Predicates: Father, Brother, >, Equal, supports, ...
- Functions: Sqrt(.), LeftLegOf(.), Plus, ...
- Variables: x, y, a, b, ...
- Connectors: ∧, ∨, ¬, ⇒, ⇔
- Equality/Inequality: =, ≠
- Quantifiers: ∀, ∃

Form of an expression in predicate logic

Atomic sentences = Predicate(term$_1$, … , terme$_n$)

OR (term$_1$ = term$_2$ =…..=term$_n$)

With:

Term = Function(term$_1$, …, terme$_n$)

OR Constant or Variable

Examples

1) Brother (Mourad, Nassim)

2) > (Length(LeftLegOf(Omar)), 1.2m)

Complex sentences = atomic sentences joined by connectors

Such as: ¬ S, S1∧S2, S1∨S2, S1⇒S2, S1⇔S2

Examples (Complex sentences)

1) Brother (Mourad, Nassim) ⇒ Brother (Nassim, Mourad)

2) > (2, 1)∨≤ (1, 2)

3) > (2, 1)∧¬ > (1, 2)

Quantification

Quantification allows us to express properties about a set of objects without having to designate each one by name.

Two types of quantifiers:

- Existential quantifier: ∃
- Universal quantifier: ∀

1. Existential Quantification

General form: ***∃ <variables><phrase>***

Example :

1. There is someone smart in Unix

   ➔∃x belong-to (x, Unix)∧Intelligent(x)

∃x P: is equivalent to the disjunction of instantiations of P

belong-to (Ali, Unix)∧Intelligent(Ali)

∨belong-to (Omar, Unix)∧Intelligent(Omar)

∨belong-to (Said, Unix)∧Intelligent(Said)

∨…

- Typically ∧ is the main connector with ∃
- Common mistake: using ⇒ as main connector with ∃:

Ex :∃x belong-to (x, Unix)⇒Intelligent(s) (Wrong)

2. there is a person

   ⇔***∃x Person(x)***

   Variable linked to ∃

- Variables in an expression designate possible locations for constants:

  – x is a free variable in ***Person(x)***

– x is a linked variable in ***∃x Person(x)***

Examples:

1. ∃y (Eat(Walid, y)∧cake(y))

⇔"Walid eats a cake"

- Without the outer parentheses there is an ambiguity:

2.∃y Eat (Walid, y)∧cake(y)

Could be interpreted as:

a) (∃y Eat(Walid, y))∧cake(y)

variable linked to ∃ free variable

b) ∃y (Eat(Walid, y)∧cake(y))

variable linked to ∃

2. Universal quantization

General form: ***∀<variables><phrase>***

Example:

1. Everyone in Unix is intelligent

⇔ ***∀x belong-to (x, Unix)⇒Intelligent(x)*** **[For all x / For every x]**

∀x P : is equivalent to the conjunction of instantiations of P

belong-to (Ali, Unix)⇒Intelligent(Ali)

belong-to (Omar, Unix)⇒Intelligent(Omar)

belong-to (Said, Unix)⇒Intelligent(Said)

∧…

- Typically ⇒ is the main connector with ∀
- Common mistake: using ∧ as main connector with∀:

Ex : ***∀x belong-to (x, Unix)∧Intelligent(x)***

⇨ Means: "Everyone belongs to Unix and everyone is intelligent"

Morgan's Laws

- ¬ ***∃x φ*** is equivalent to ***∀x ¬φ***
- ¬ ***∀x φ*** is equivalent to ***∃x ¬φ***
- ¬ ***(φ ∨ψ)*** is equivalent to ¬φ∧¬ψ
- ¬ (φ***∧ψ)*** is equivalent to ¬φ***∨¬ψ***

Properties of quantifiers

- ***∀x ∀y*** is equivalent to ***∀y ∀x*** ***(permutation)***
- ***∃x∃y*** is equivalent to***∃y∃x*** ***(permutation)***
- ***∃x ∀y*** is not identical to ***∀y∃x***

Ex: ∃***x ∀y Loves (x, y)*** ⇔ "There is someone who loves everyone"

∀y∃x Loves (x, y) ⇔ "Everyone is loved by at least one person"

(Everyone is loved by some persons)

Duality of quantifiers

Each quantifier can be expressed using the other:

1. ***∀x Likes (x, Ice Cream)***⇔¬∃x ¬ Likes (x, Ice Cream)
2. ***∃x Likes (x, Kiwi)***⇔¬∀x ¬ Likes (x, Kiwi)

Power of 1st Order Predicate Logic

- Almost all sentences of natural language can be represented in first-order predicate logic.
- There is no unique correspondence between a natural language sentence and a logical expression.

Examples:

1. Ali's mother is **married** to Ali's father

⇔***Married (Father (Ali), Mother (Ali))***

2. Ali **lives** in a yellow house

⇔***a) lives(Ali, House)∧Color(House, Yellow)***

b)∃x House(x)∧Color(x, Yellow)∧Lives(Ali, x)

3. If the car **belongs** to Ali, then it is green

⇔***a) belong (Car, Ali)⇒Color(Car, Green)***

b) ∃x Car(x)∧ belong(x, Ali) ⇒Color(x, Green)

4. Some people like snakes

⇔***a) ∃x (Person(x)∧Loves(x, Snake))***

b)∃x ∀y ((Person(x)∧Snake(y)⇒Loves(x, y))

5. All students take exams

⇔ ***∀x (Student(x)⇒ Take-exam (x))***

b)∀x (Student(x)⇒∃y Exam(y)∧Takes(x,y))

Equality

Term1 = Term2 is true given an interpretation if and only if ***Term1*** and ***Term2*** refer to the same object.

Examples:

1. ***Father(Omar) = Mohamed***
2. ***Phone (Mourad) = "0770576344"***

Inference in first-order logic

Premises:

1. If x is a parent of y, then x is older than y
2. If x is the mother of y, then x is a parent of y
3. Fatma is Zahra's mother

Conclusion:

– Fatma is older than Zahra (from (2) then (1))

Correspondence in first order logic:

Premises:

1.∀x∀y Parent(x,y)⇒Older(x, y)

2.∀x∀y Mother(x,y)⇒Parent(x, y)

3. Mother (Fatma, Zahra)

Conclusion:

Older (Fatma, Zahra)

Inference is obtained by axioms and rules (i.e. by syntactic transformations) which extend those of propositional logic.

Proof by Forward Chaining

Find a proof for the conclusion ***Older(Fatma, Zahra)*** using forward chaining:

1. Mother (Fatma, Zahra) (given)

2. Alive (Fatma) (given)

3.∀x∀y Mother(x,y)⇒Parent(x, y) (given)

4.∀x∀y (Parent(x,y)∧Alive(x)⇒Older(x, y) (given)

By replacing fact (1) in rule (3) we obtain the following new fact:

5. Parent(Fatma, Zahra)

Replacing facts (5), (2) in rule (4) we obtain the new fact:

6. Older(Fatma, Zahra) (what is requested)

Principle of forward chaining: perform a series of replacements of the given facts in rules also given until arriving at the fact to be proven (this is a chaining guided by the initial data Data-Driven).

Proof by Backward chaining

Principle: Start from an objective (to be proven) and derive new sub-objectives until you only have sub-objectives known to be true.

➔Similar to problem reduction (goal-driven chaining).

Example: Let's take the same previous example

Proof by backward chaining of ***Older(Fatma, Zahra)*** from the premises:

1. Mother (Fatma, Zahra)
2. Alive (Fatma)
3. ∀x∀y Mother(x,y)⇒Parent(x, y)
4. ∀x∀y(Parent(x,y)∧Alive(s)⇒Older(x, y)

Objective: (i) ***Older(Fatma, Zahra)***

We see that objective (i) corresponds to the right part of (4)

⇨ Get new sub-objectives:

(ii) Parent (Fatma, Zahra)

(iii) Alive (Fatma)

(iii) corresponds to (2) (already proven or given)

(ii) corresponds to the right part of (3)

⇨ We obtain a new sub-objective:

(iv) Mother (Fatma, Zahra)

(iv) corresponds to (1) (which is true)

We see that all the sub-objectives are true.➔the original goal is also true.

Remarks :

- Backward chaining is goal directed.
- Backward chaining is the basis of logic programming (Prolog)

3. Other Logics

a) Logic of belief

Ex: Believe(Ali, Father(Mourad, Ibrahim))

b) Temporal logic

Allows you to reason about time

c) Non-monotonic logic

Allows the truth values associated with facts to change

d) Fuzzy logic

Accompanies the facts with likelihood values

4. Production rules

- The most used formalism by expert systems.
- Simple and effective forms.
- Based on rules of the form "***If***… ***Then***…"

Examples:

1. If there is no picture but you can hear sound, check the screen intensity setting"

If *No Image* ***AND*** *Sound* ***THEN*** *Screen Intensity*

2. A customer is insolvent if his bank account is negative and he does not have any amount.

If ((*Account_Balance < 0)****AND****(****Title_Amount=0)) Then*** *Insolvent_Client*

Benefits

- Ease to express
- Efficiency and clarity of representation
- Easy development => wide distribution of software
- Doesn't require programmer training (Knowledge Engineer).
- Easy to update
- Possibility of verification and validation of knowledge.
- User involvement in development.
- Ability to call other formalisms such as structured objects.

5. Semantic Networks

Some characteristics

- A directed graph
- Nodes represent concepts.
- Arcs represent the semantic links between concepts such as:
  - ***Is a:*** inclusion relationship (allows inheritance)
  - ***Instance of:*** membership relationship between an instance and the class to which it belongs.
- Used mainly in natural language interpretation
- An instance/subclass inherits all the semantic properties of the parent class.
- An instance shares all the semantic links of the class to which it belongs.
- Inheritance can be single or multiple. i.e., a subclass has several parent classes.

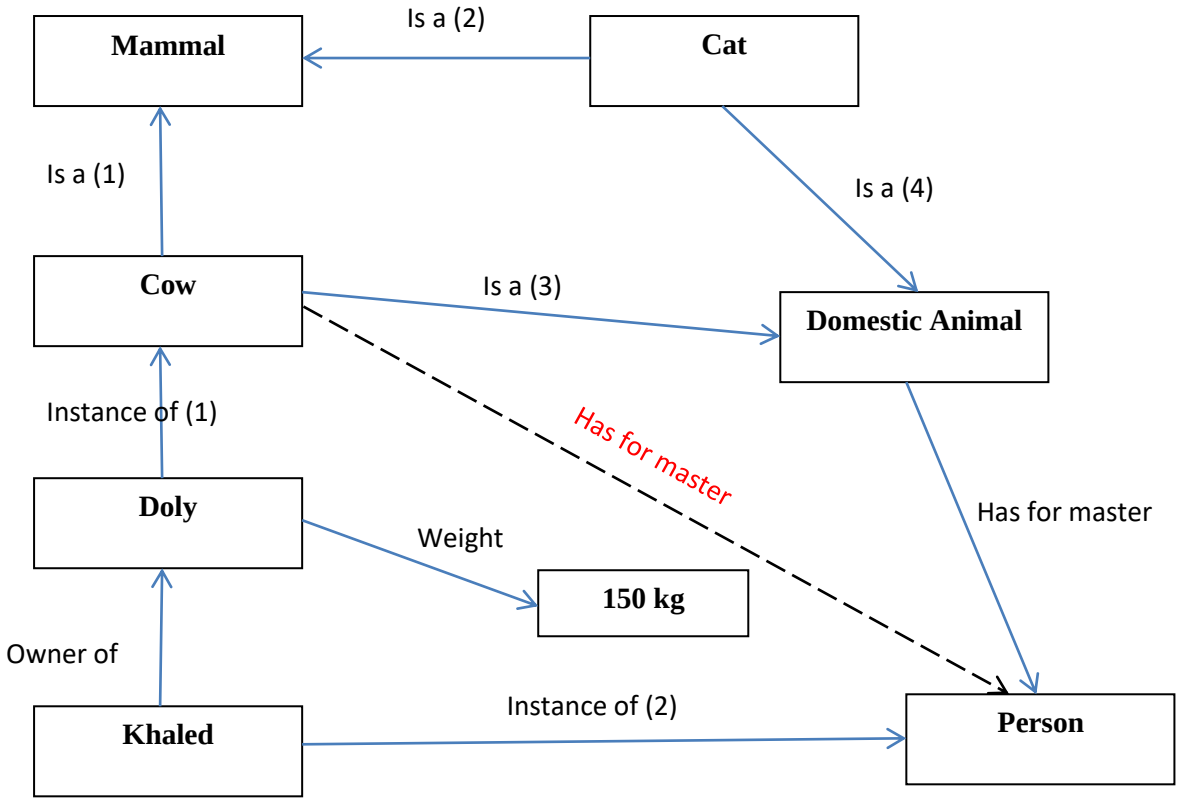


Fig II. 1. Characteristics of the semantic network

Example of a simple semantic network

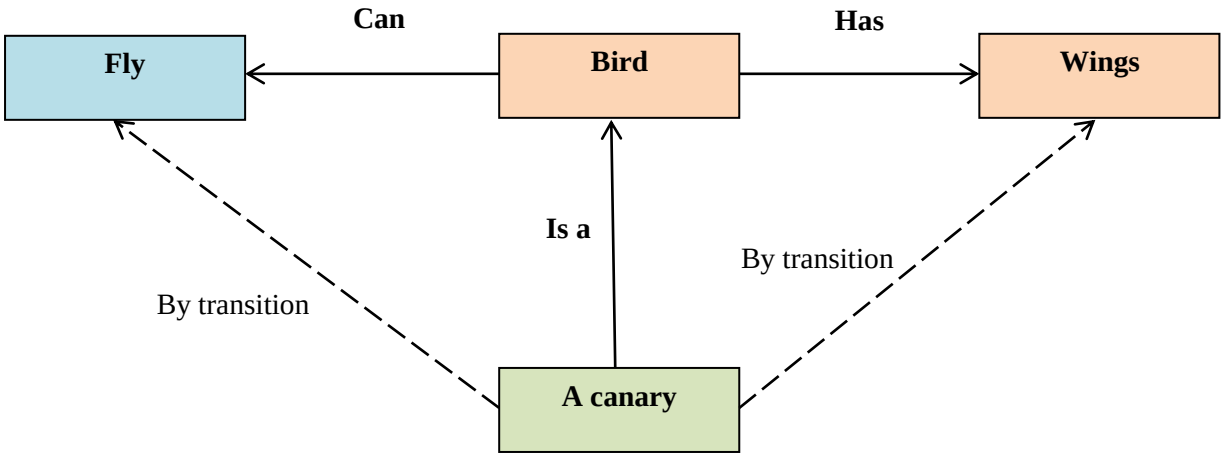


Fig II.2. Example of a simple semantic network

Network Interpretation

a) - Knowledge considered by the graph

- A canary is a bird
- A bird has wings
- A bird can fly

b)- Inferred knowledge (obtained by transition)

- A canary has wings
- A canary can fly

An example of a more detailed semantic network

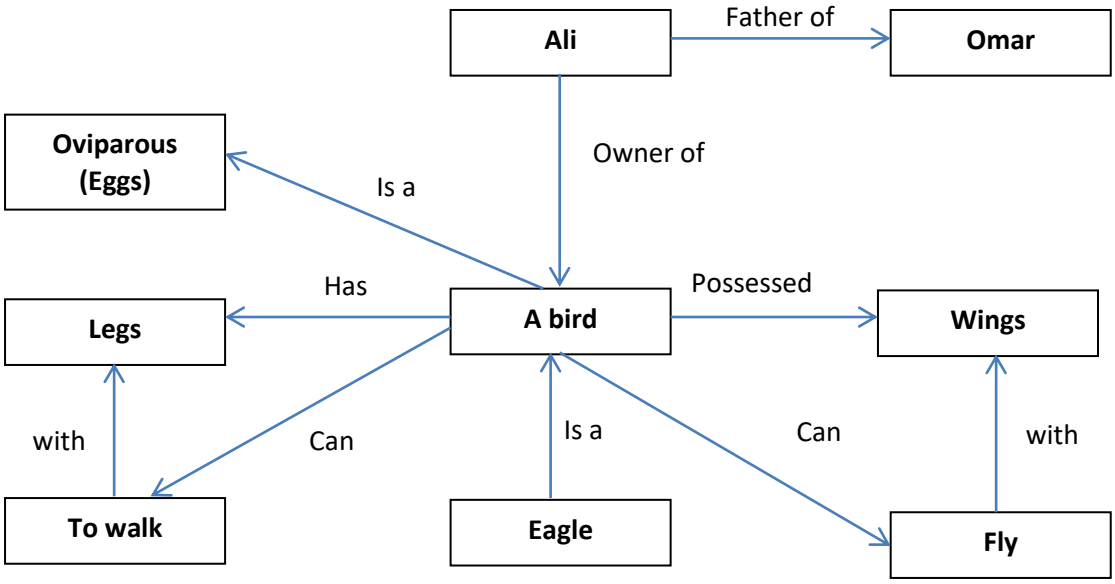


Fig II.3. A detailed semantic network

Benefits

- The power of expression
- Inheritance Properties

Disadvantages

- Lack of formalism standardization.
- Difficulty of updating the graph.
- Unable to express programs.

6. Structured Objects (OOP)

An object is a structure that contains both data (declarative knowledge) and procedures or methods (procedural knowledge)

Example: *Define an object « point »*

Point: object

X, Y: integer **(Data)**

Distance: point X point➔real **(Method 1)**

Center: point X point➔point **(Method 2)**

END

Basic Concepts

- Classes and subclasses
- Objects
- Instances
- Attributes
- Methods

Some characteristics

- Encapsulation (Data + Methods): grouped together in the same box
- Inheritance (instance, object, class, subclass)
- Polymorphism
- Instantiation

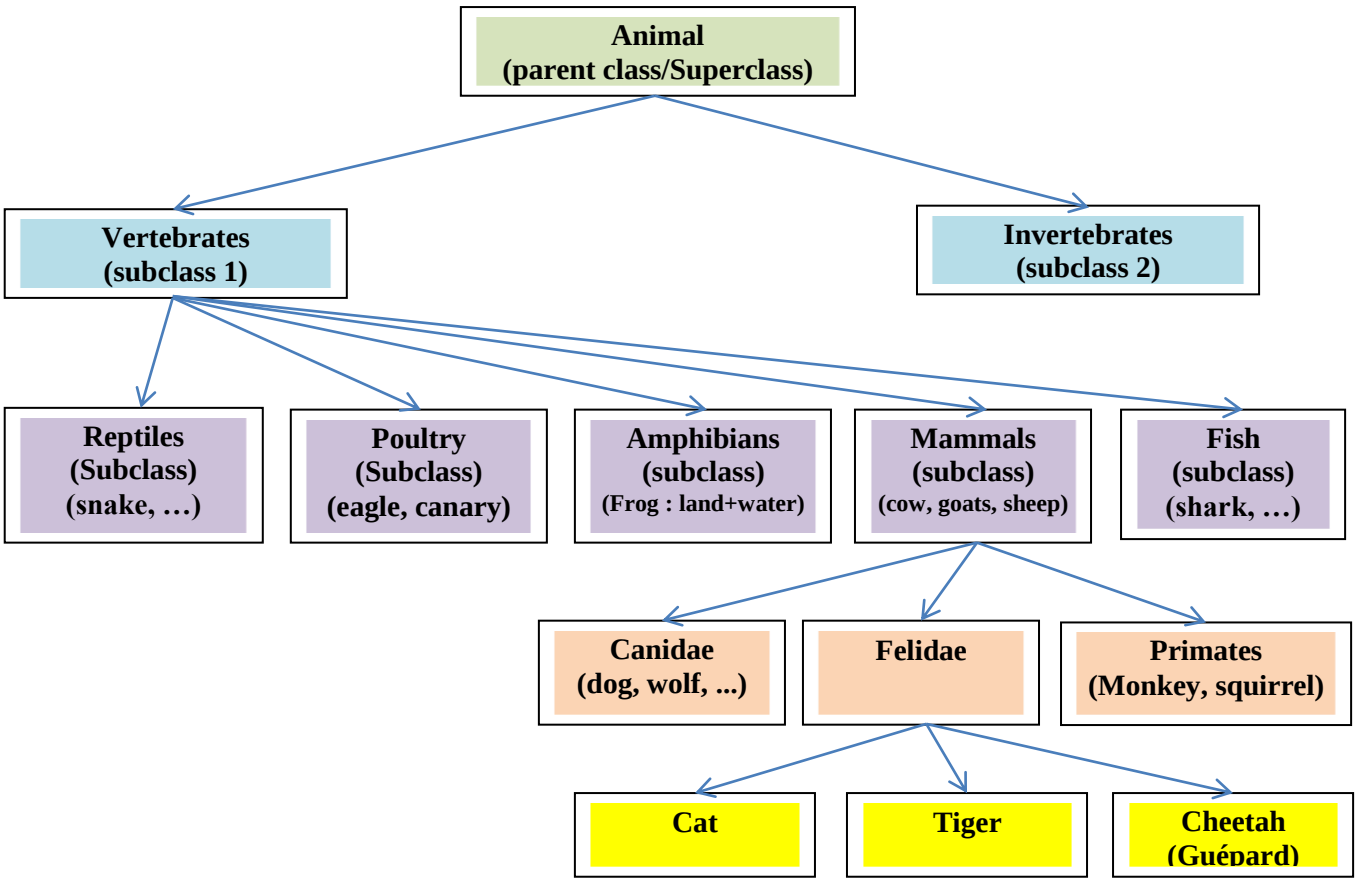


Fig II.3. Hierarchy of the "Animal" class

Objects and Classes

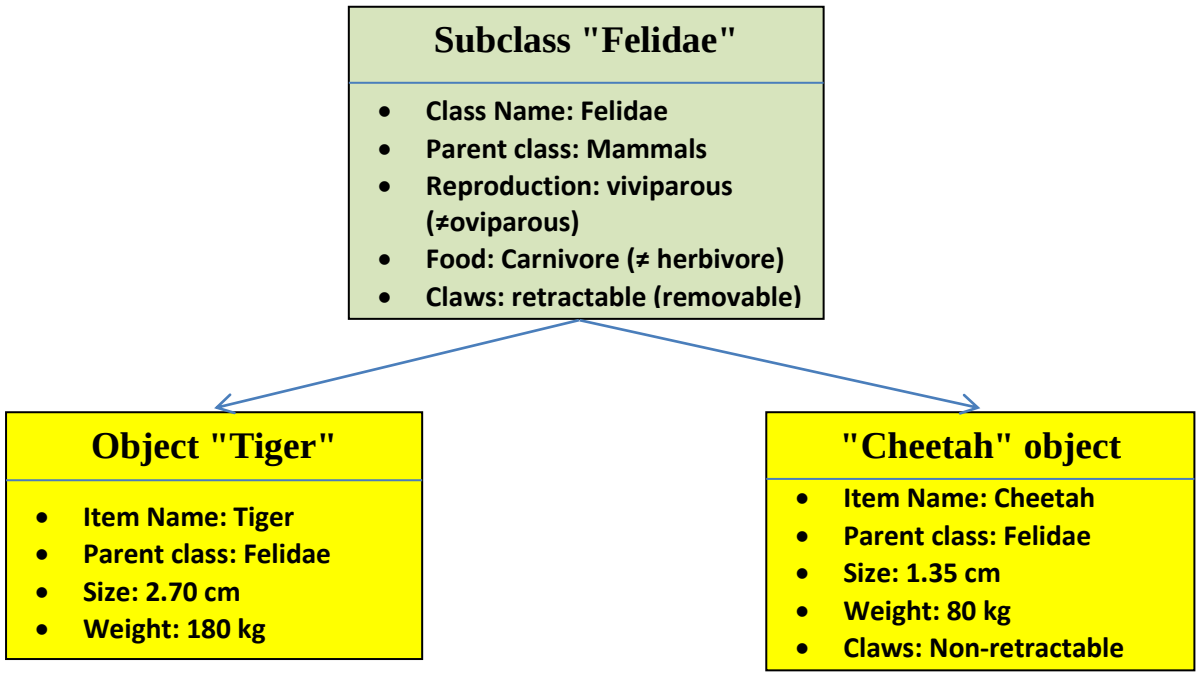


Fig II.4. Object representation and inheritance mechanism

OBS: An object inherits the properties of the parent class

7. Neural Network

Motivation

- Humans are capable of manipulating large amounts of knowledge using their brains.
- The brain is composed of neurons (nerve cells) which are the fundamental elements of the nervous system.
- A neuron is made up of: nucleus, axon, dendrites

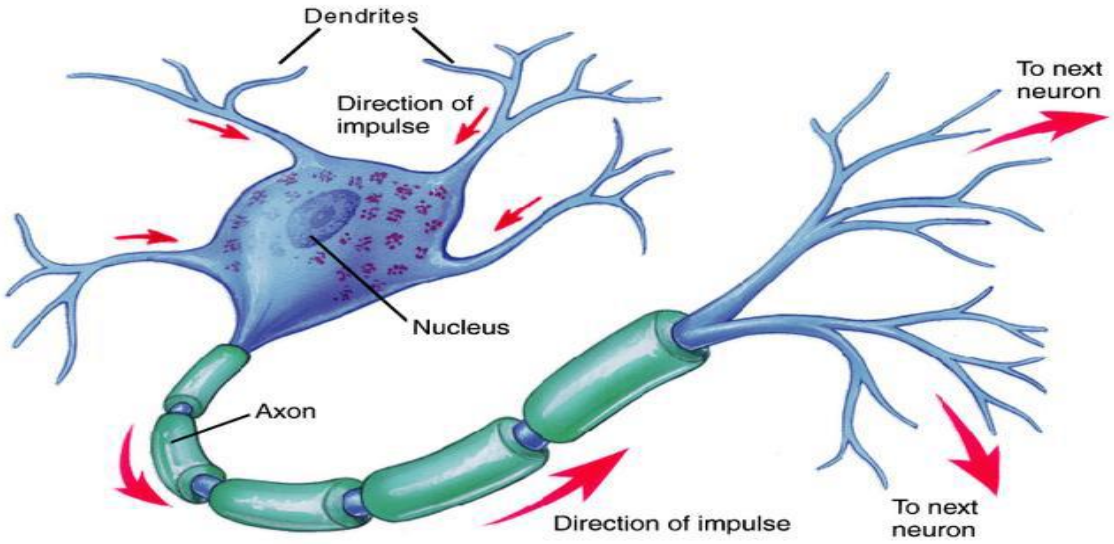


Fig II.5. Architecture of the biological neuron

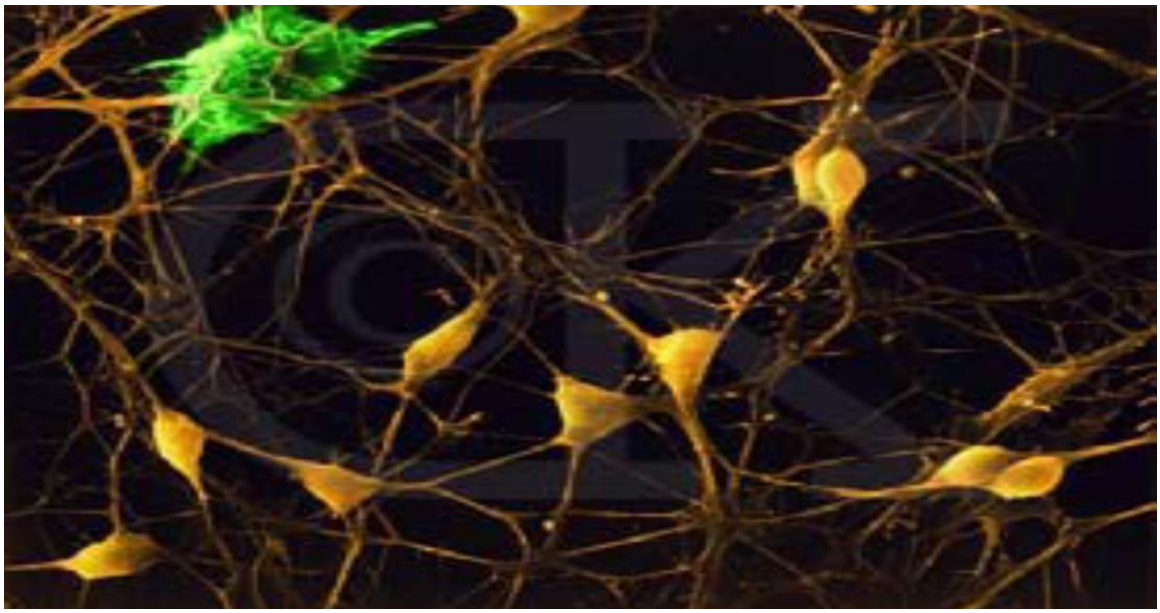
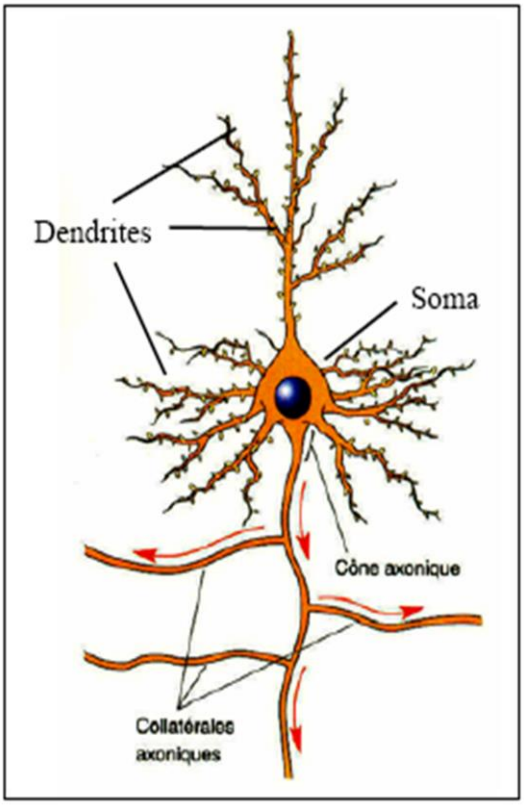


Fig II.6. Architecture of the nervous system

Characteristics of the biological neuron



- **Cell diameter:** 0.01 - 0.05 mm,
- **Soma (cell body)**
  - Variable shapes (generally spherical),
  - 20 µm in diameter,
  - Contains the core,
  - Surrounded by a 5 nm thick membrane.
- **Axon**
  - Unique to nerve cells,
  - Diameter: 1-25 µm (human),
  - Length: 0.1 mm to 1 mm. (!!!),
  - Connection to other neurons (synapses),
  - Allows the transmission of information,
- **Dendrites**
  - Receive signals from other neurons,
  - Each covered with hundreds of synapses.

Important Numbers

- Number of neurons in the brain: ~10^{11} (100 billions)
- Number of connections per neuron:~10^{4} - 10^{5} (dynamic connections)
- Switching time: ~ 10^{-3} seconds
- Average time of a cognitive activity: ~ 10^{-1} second

(e.g., face recognition)

⇨ So there is only a place for 100 treatment cycles ($10^{-1}/10^{-3}$)

which is insufficient for a complex activity!!!

⇨ The brain must therefore perform operations in parallel!!!

Characteristics of the nervous system

- Robust and tolerant of errors and "failures".
- Flexible and capable of learning, no need to be explicitly “programmed” (synaptic plasticity).
- Able to process partial (fuzzy), uncertain (probabilistic) or incomplete information.
- Very massively parallel.
- Abilities (intelligence?) reside in the connections between the billions of neurons in the human brain.

How a biological neuron works

Each neuron receives electrical impulses through its dendrites, if these impulses are sufficiently important, the axon transmits an electrical impulse to excite the different connected neurons which will in turn be activated depending on the intensity of the transmitted impulses.

Artificial/Formal Neural Networks (ANN)

Applications

1. Control
   - Autopilot:

     Alvinn: vehicle control network based on video images,
   - Speech synthesis:

     NETtalk: a network that learns to pronounce a text in English
   - Manufacturing/production process,
2. Recognition/classification/analysis
   - Printed texts, handwritten characters (postcodes), speech,
   - Images (e.g. faces), animated images or video clips,
   - Predict Risks (financial, natural, etc.)
3. Prediction
   - Economics, finance, market analysis, medicine,

Relationship with biological neural network

- Massive parallelism
- Local calculation
- Simple calculation element of the "artificial neuron" type.

How does it Work?

- Learning (by back-propagation).

Areas of success

- Speech recognition (TDNNs)
- Robotic control (ALVINN)
- Character recognition (OCR),
- Financial forecasting (FF).

Some historical landmarks

1943: W.S. McCulloch & W. Pitts

- Proposed a simple model of a neuron capable of producing a Turing machine.
- Demonstrated that a synchronous assembly of such neurons is a universal computing machine (i.e. any logical/arithmetic function can be realized by threshold units).

Warren S. McCulloch

William Pitts

1948: D. Hebb

- Proposed a learning rule for neural networks.

1958: F. Rosenblatt

- Proposed the Perceptron model and demonstrated its convergence theorem.

1969: M. Minsky & S. Papert

- Demonstrated the limitations of the Perceptron model.

1985:

- Apparition of back-propagation learning for multi-layer networks.

Formal neuron architecture

McCulloch and Pitts (1943) modeled how a biological neuron works and imagined an artificial reproduction that mimicked human reasoning.

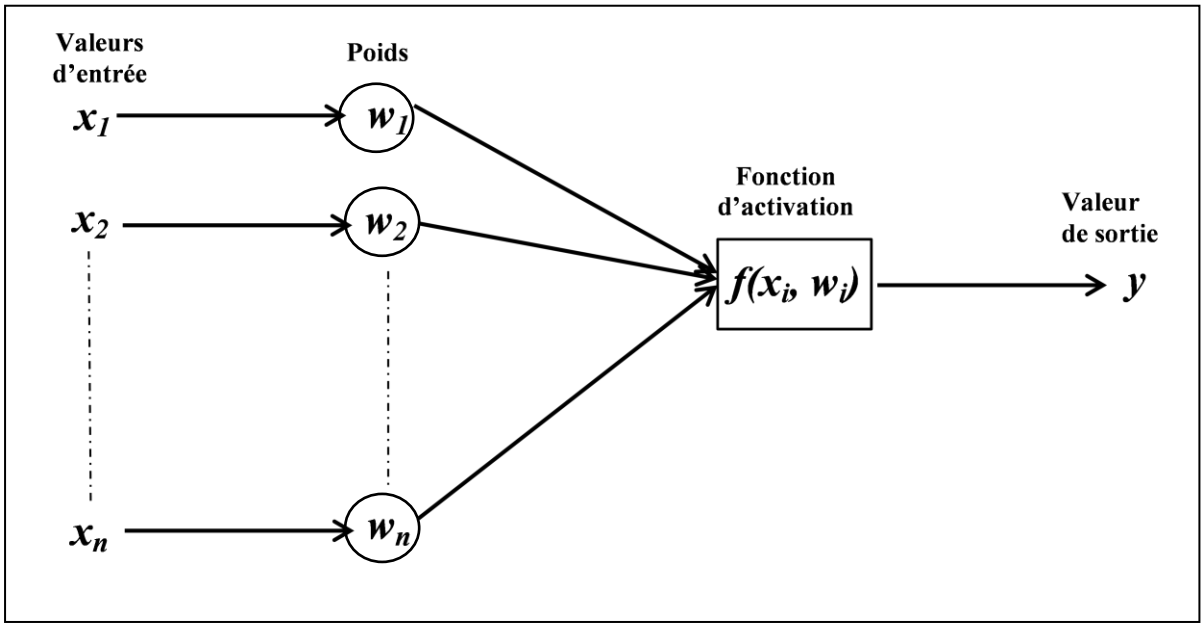
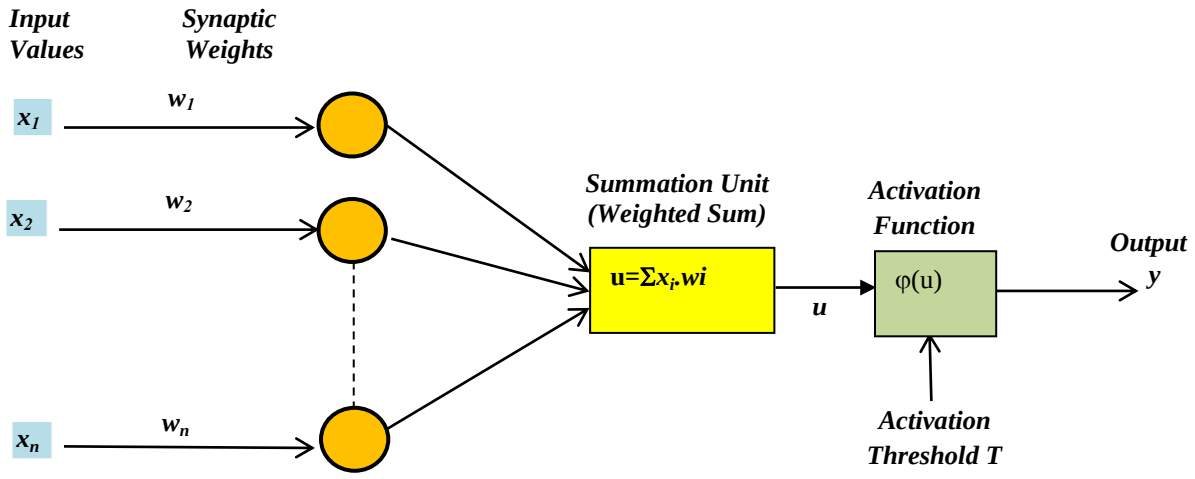


Fig II.6. Architecture of the formal neural network

Principle of a formal neuron (McCulloch and Pitts model/Linear model)

A formal neural network or also called threshold automaton based on the same principle of a biological neural network as explained previously



This model is based on two equations:

$$u = \Sigma xi \,.wi \quad (1) \qquad \textit{(weighted sum)}$$

$$y = \varphi(u-S) \quad (2) \qquad \textit{(activation function)}$$

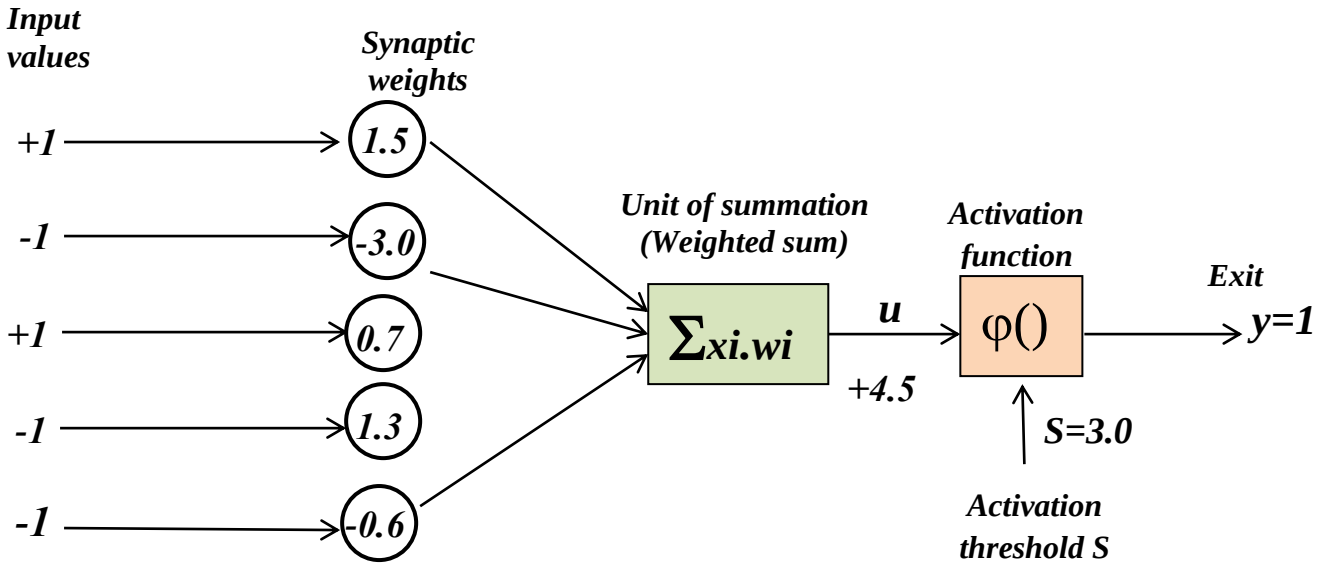
with :

$x_1, x_2, ..., x_n$: are the inputs,
$w_1, w_2, ..., w_n$: are the synaptic weights of neuron k,
u: the output of the summing unit,
S: the threshold,
$\phi(.)$:the activation function,
y: the output signal of neuron k.

Output value: the formal neuron calculates the sum of the inputs x_1, x_2, …, x_n weighted by the synaptic weights w_1, w_2, …, w_n and compares it with the threshold *S*, if the result is greater than the threshold, then the value returned at the output is +1 (activation of the neuron), otherwise the value returned is 0

$$y = \begin{cases} +1 & si \sum xi \,.wi > S \\ 0 & sinon \end{cases}$$

Example : How an RNA works (McCulloch & Pitts model)

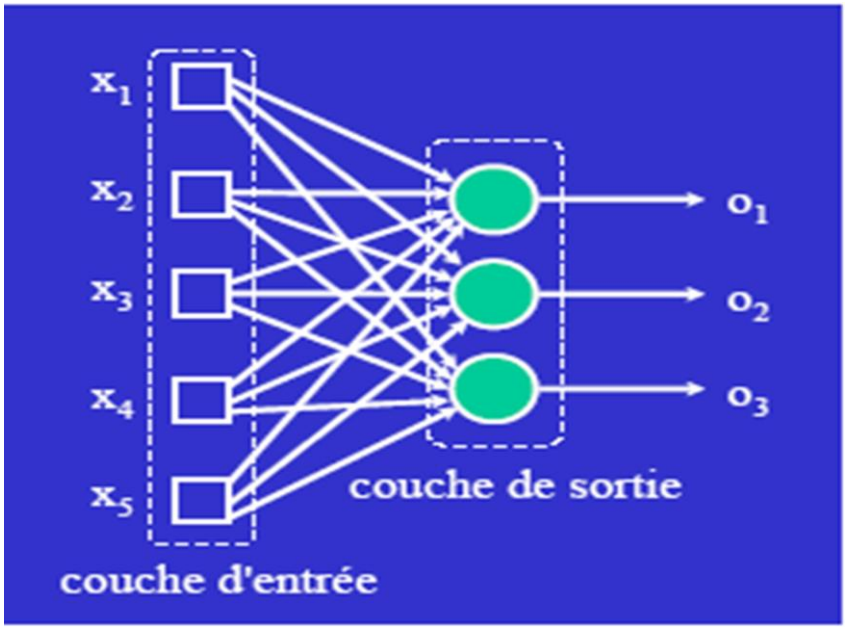


Activation function

The activation function defines the output value of a neuron in terms of the activity levels of its inputs. This function can have several forms (Linear, Exponential, Sigmoid, tanh, ReLu, Elu, Softmax, Softplus, Softsign,...)

Neural Network Topologies

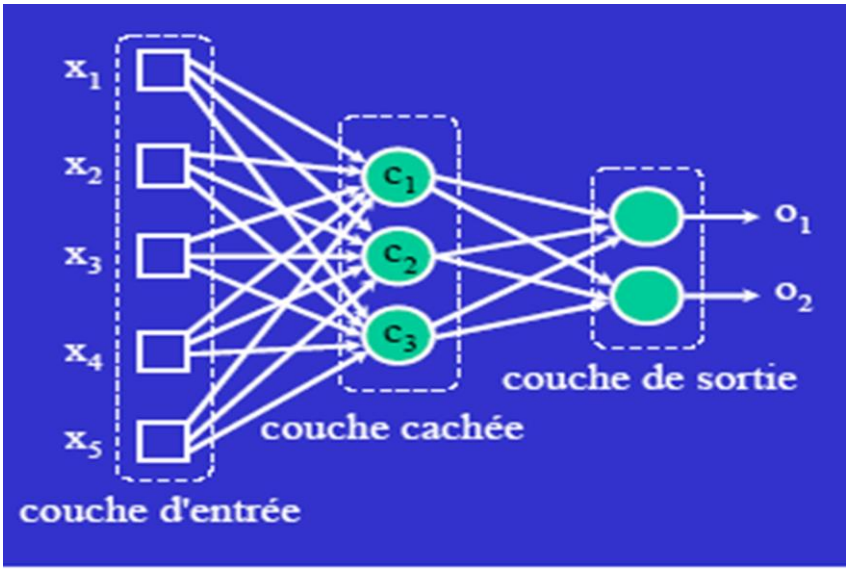
1. Single-Layer Acyclic network ("feedforward")



$$o_i = 1 \quad \text{si} \quad \sum_k w_{ik} x_k > 0$$

$$o_i = 0 \quad \text{sinon}$$

2. Multi-layer acyclic network



Output values:

Couche cachée

$$c_j = 1 \quad \text{si} \quad \sum_k w_{jk} x_k > 0$$

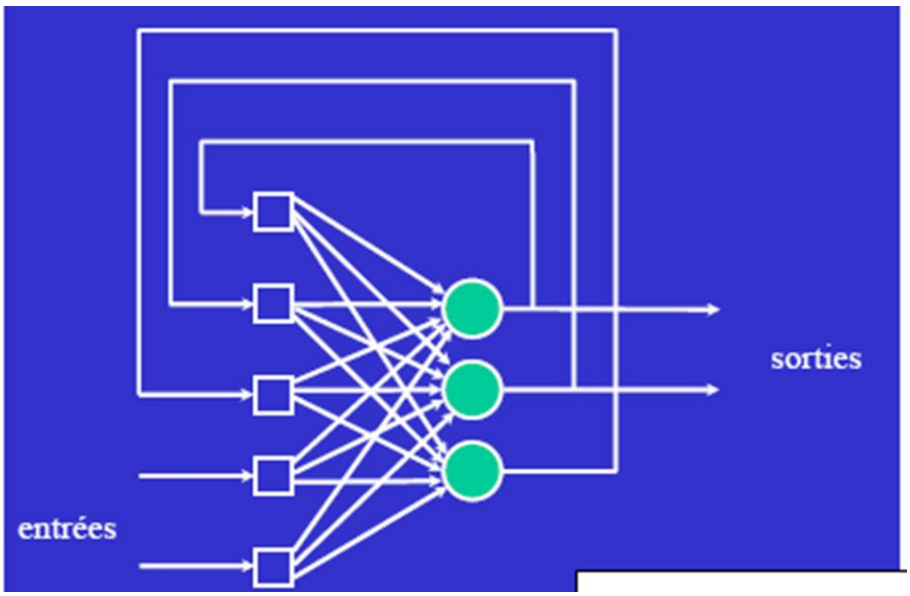
$$c_j = 0 \quad \text{sinon}$$

Couche de sortie

$$o_i = 1 \quad \text{si} \quad \sum_k w_{ik} c_k > 0$$

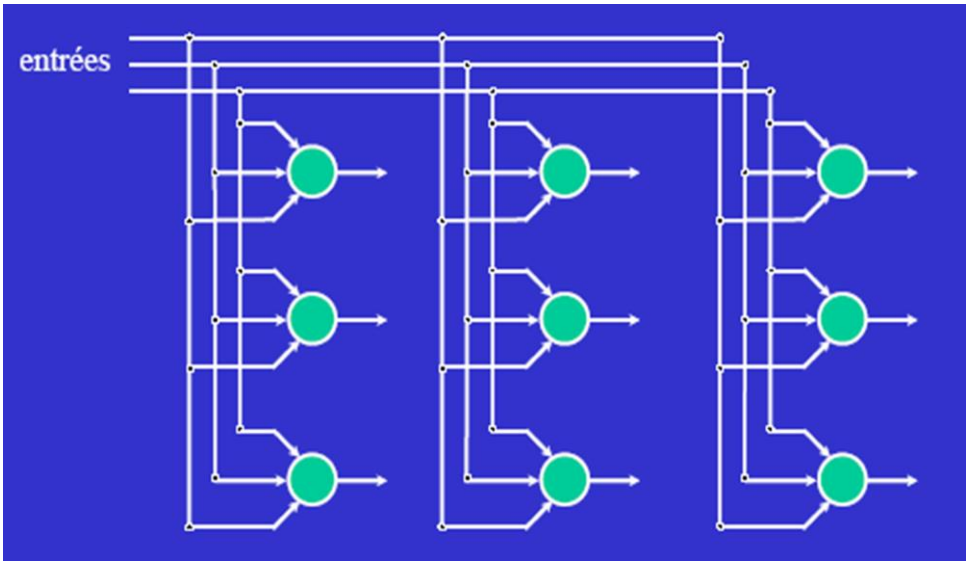
$$o_i = 0 \quad \text{sinon}$$

3. Recursive network (Hopfield network)



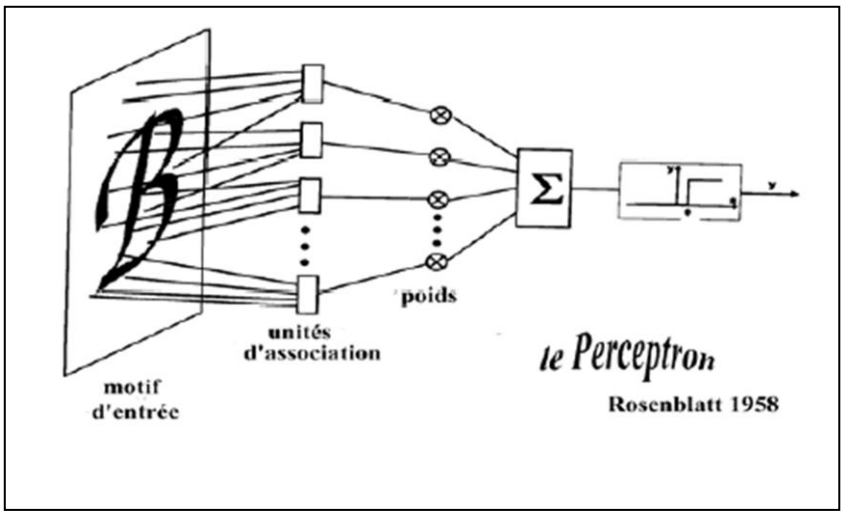
OBS: Each unit *i* is connected to every other unit *j* by a weight w_{ij} the weights are assumed to be symmetrical: $w_{ij} = w_{ji}$

4. "Lattice" network



Two-dimensional 3x3 lattice network

The Perceptron (F. Rosenblatt, 1958)

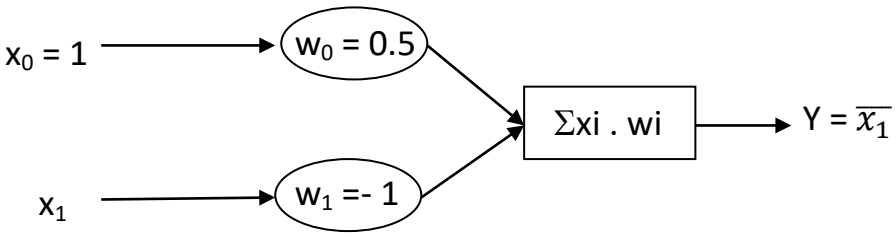


With :

$x_0 = 1$ **And** $\begin{cases} 1 & si \sum x_i . w_i > 0 \\ 0 & sinon \end{cases}$

Example : Realization of a logical NOT by a perceptron

Input x_1	Output
0	1
1	0



Exercise: Create OR, AND, XOR logic gates using a perceptron (See TD)

Learning theorem (F. Rosenblatt)

Given enough training examples, there exists an algorithm that will learn any linearly separable function.

Perceptron Learning Algorithm

Entries: *training set {(x1,x2,…,xn, t)}*

Method

randomly initialize the weights w(i), 0<=i<=n

Repeat *until convergence:*

For*each example*

Calculate the output value o of the network.

Adjust the weights:

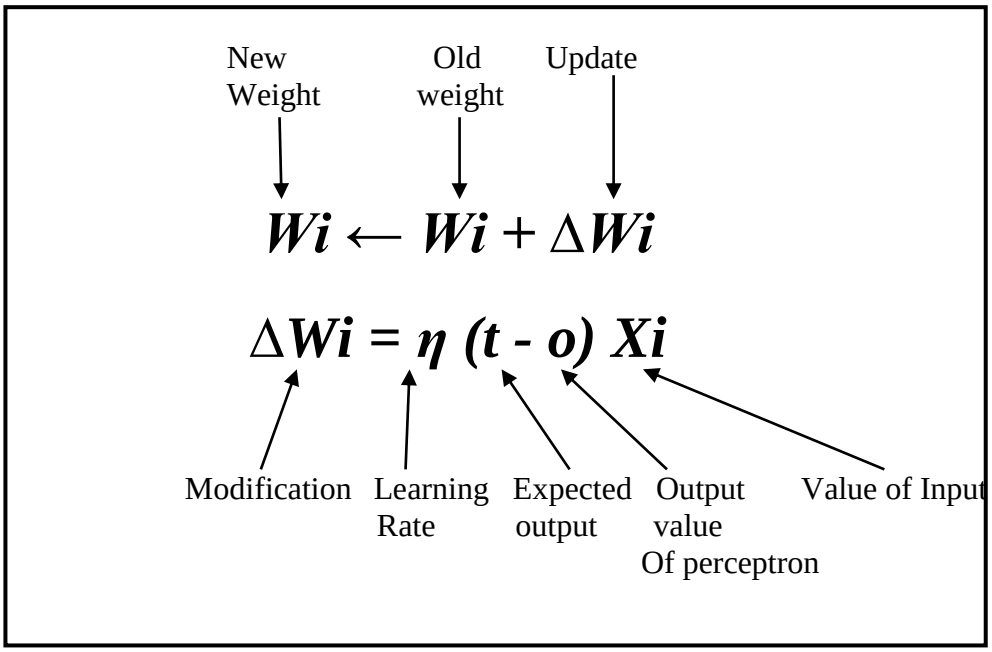
$\Delta wi = \eta(t - o)\, xi$ /* *t: expected output, O: calculated output*

$wi \leftarrow wi + \Delta wi$ /* *by the perceptron*

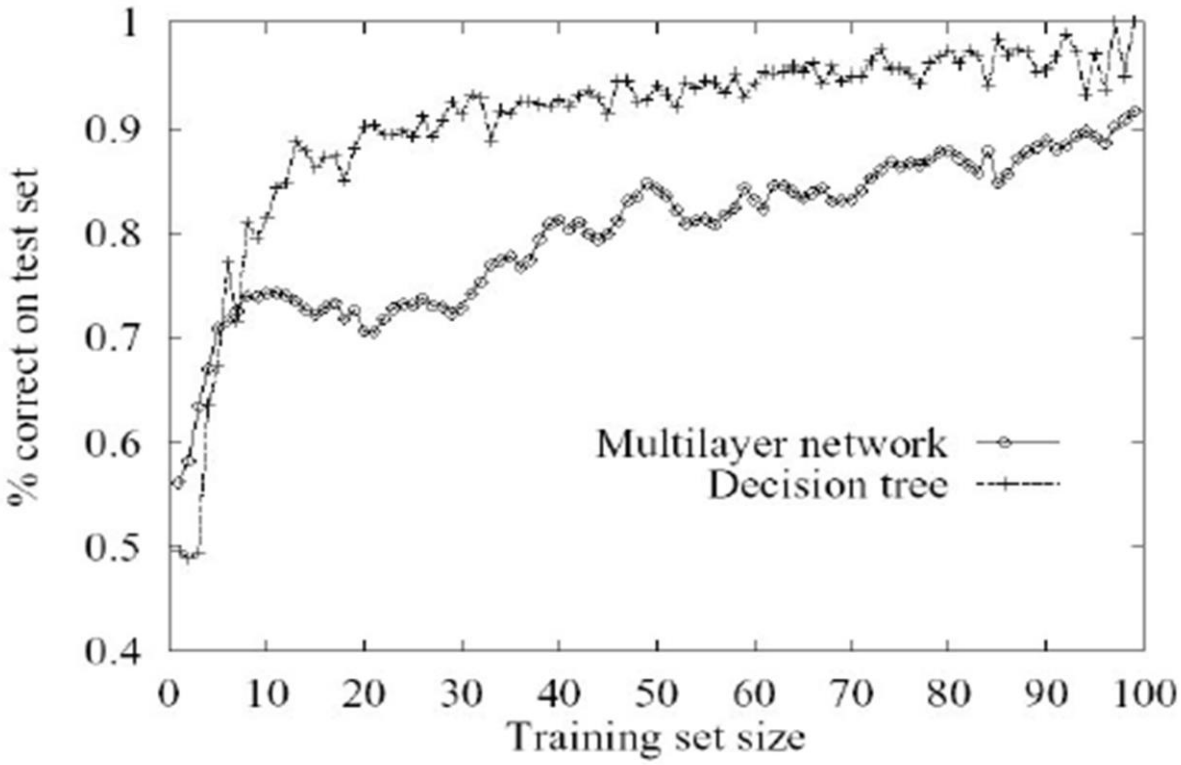
For

End Repeat

End Method



Networks Vs Decision Tree



When to use artificial neural networks to solve learning problems?

- When data is represented by attribute-value pairs,
- When the training examples are noisy,
- When (very) long learning times are acceptable,
- When a rapid assessment of the learned function is necessary,
- When the user's understanding of the learned function is unimportant.