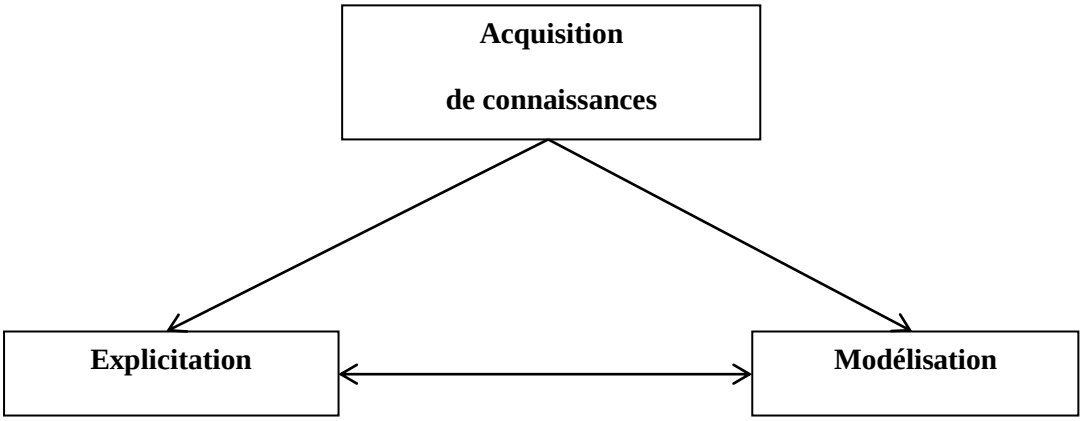
Formalismes de représentation de connaissances

Processus d'acquisition de connaissances



Problématique de l'acquisition de connaissances

- Les connaissances d'un expert sont subjectives et difficiles à formaliser.
- Les formalismes utilisés pour la représentation des connaissances ne permettent pas un bon niveau d'abstraction et restent liés à l'implantation.
- Comment organiser les connaissances en vue de leur traitement ? (pb)

Types de connaissances

a) Connaissances déclaratives (le savoir)

La population de L'Algérie était de 38.000 000 habitants en 2011.

b) Connaissances procédurales (le savoir-faire)

Une recette de cuisine =>des étapes pour faire un gâteau au chocolat.

c) Connaissances conceptuelles (combine les deux)

Le concept de référendum fera appel aux connaissances procédurales : installation des bureaux de votes et constitution des listes électorales ;

La connaissance déclarative est : un bulletin nul n'est pas compté.

Données problématiques

- Redondance : calcul du rapport (long tête/hauteur du corps) alors que l'on a les rapports (long tête/long standard) et (hauteur du corps /long standard)
- Corrélation : la taille de la fontanelle diminue en même temps que la taille du corps augmente
- Synonymie : posture du corps (presque droit, C, faiblement courbé)
- Ambiguïté : couleur (foncé, noir)
- Information cachée : forme mâle, forme femelle
- Imprécision : poisson de grande taille.
- Absence : absence de paupière adipeuse.

Évolution de processus d'acquisition de connaissances

Techniques mannuelles	Techniques semi-automatiques	Apprentissage machine

Représentation des connaissances (Types)

- Traitement de l'information numérique (premiers ordinateurs)
- Traitement alphanumérique (bases de données)
- Traitement symbolique (en intelligence artificielle on traite des faits, des énoncés des équations, des méthodes représentées par des systèmes de symboles, etc)

Formalismes de représentation de connaissances

1. Logique des propositions (pas de quantificateurs et pas de variables)
2. Logique des prédicats de premier ordre (introduction de variables et de quantificateurs)
3. Règles de production

Représentation structurées de connaissances

1. Réseaux sémantiques
2. Objets structurés
3. Frames
4. Logiques terminologiques KL-ONE
5. Graphes conceptuels
6. Réseau de Pétri
7. Réseaux de neurone

1. Logique propositionnelle

- La logique est un langage formel pour représenter l'information et permettre d'en tirer des conclusions.
- La logique des propositions (propositionnelle) une suite de symboles séparés par des conjonctions (et), des disjonctions (ou) ou des négations (non)
- La logique des propositions permet d'exprimer :

– des faits sur le monde réel: "***Ali habite à M'sila***"
– des négations: "***Mohamed n'habite pas à Msila***"
– des conjonctions et des disjonctions « ***Mohamed est un étudiant, et il habite à M'sila*** »
– des phrases avec "conséquence" logique: "***Si le fils ne rencontre pas sa mère, la mère ne le rencontre pas non plus***."

- Une proposition est une expression (phrase) à propos du monde qui est soit(Vraie) soit (Fausse).

Exemple: Représenter les expressions suivantes en logique de proposition

1. Socrate est un homme ➔HommeSocrate/ Homme(Socrate)
2. Platon et Socrate sont des hommes ➔

HommePlaton∧HommeSocrate

Homme(Platon)∧Homme(Socrate)

3. Zola a écrit Germinal ➔***(AUTEUR, ZOLA)*** **=>** ***(LIVRE, GERMINAL)***

Auteur(Zola) => Livre(Germinal)

Éléments de base de la logique propositionnelle:

– Symboles de propositions: P, Q, … (phrases)
– Phrases spéciales (valeurs): Vrai, Faux
– Opérateurs: ∧ (et), ∨ (ou), ¬ (non), ⇒ (implique), ⇔ (équivalent)

Formation de phrases (propositions) composées:

1. Négation: ¬ P
2. Conjonction: P ∧ Q
3. Disjonction: P ∨ Q
4. Implication: P ⇒ Q (si P alors Q)
5. Equivalence: P ⇔ Q (P et seulement si Q)

Exemples:

1. Soient les propositions composées suivantes :

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

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

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

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

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

OBS: Ordre de précédence : ¬, ∧, ∨, ⇒

Exemples:

¬ A ∨ B ⇒ C est équivalent à((¬A)∨B)⇒C

A ⇒ B ⇒ C est incorrect

2. Supposons: P = "le fils rencontre sa mère" Q = "La mère rencontre son fils"

Comment représenter la phrase ?

"Si le fils ne rencontre pas sa mère, alors elle ne le rencontre pas non plus"➔ ¬ P ⇒¬ Q

Table de vérité

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: La table de vérité aide à calculer la valeur logique de n'importe quelle phrase (proposition composée).

Logique propositionnelle et inférence

Définition : l'inférence est un mécanisme qui établit la sémantique d'une phrase (la valeur de vérité) en lui faisant subir une suite des transformations syntaxiques (sans utiliser la TV) en réduisant le problème original en un problème ne dépendant que de faits connus.

Exemple :

Prémisses:

1. "Si le fils ne rencontre pas sa mère, elle ne le rencontre pas non plus"

2. "La mère rencontre son fils"

Conclusion:

Alors "Le fils rencontre sa mère" est vrai

Correspondance en logique propositionnelle

Soit: P ≡ "Le fils rencontre sa mère"

Q ≡ "La mère rencontre son fils"

Inférence

Prémisses en logique propositionnelle

1. ¬ P ⇒ ¬ Q
2. Q

Conclusion

Alors P

Composantes de base de l'inférence:

- ***Axiomes:*** faits connus du monde
- ***Règles:*** transformations syntaxiques qui propagent la valeur sémantique

Pourquoi l'inférence est-elle intéressante?

• Elle fournit un formalisme pour modéliser le raisonnement

• Elle réduit l'analyse sémantique à des manipulations syntaxiques

• Permet l’automatisation de ces manipulations syntaxiques.

Exemple d'une base de connaissance formulée en logique propositionnelle

1. Batterie-OK ∧Ampoules-OK ⇒Phares-OK (R1)
2. Batterie-OK ∧Starter-OK ∧¬Réservoir-Vide ⇒Moteur-Démarre (R2)
3. Moteur-Démarre ∧¬Pneu-Plat ⇒Voiture-OK (R3)
4. Phares-OK (f1)
5. Batterie-OK (f2)
6. Starter-OK (f3)
7. ¬Réservoir-Vide (f4)
8. ¬Voiture-OK
9. Batterie-OK ∧Starter-OK <= (5+6) (f5)
10. Batterie-OK ∧Starter-OK ∧¬Réservoir-Vide <= (9+7) (f6)
11. Moteur-Démarre <= (2+10) (f7)
12. ¬Moteur-Démarre ∨Pneu-Plat <= (8+3) ≡Moteur-Démarre ⇒Pneu-Plat f8)
13. Pneu-Plat <= (11+12) (f9)

Systèmes de preuve

- Un système de preuve est une collection d'axiomes et de règles d'inférence,
- il existe de nombreux systèmes de preuve pour la logique propositionnelle.
- les logiciens préfèrent les "petits" système de preuve (une seule règle et peu d'axiomes),
- les spécialiste en IA préfèrent les systèmes avec des règles fortes (et nombreuses) et peu d'axiomes.

Validité & adéquation des phrases

1. ***Validité:*** certaines phrases sont toujours vraies (valides) et peuvent donc être toujours supposées comme telle.
2. ***Adéquation:*** une règle d'inférence est "adéquate" si elle produit à partir de prémisses une conclusion toujours vraie.

Système de preuve incomplet Vs système de preuve complet

1. ***Système incomplet :*** si on supprime n'importe quel axiome ou règle d'inférence, on ne peut plus rien prouver
   ⇒On obtient alors un système de preuve incomplet
2. ***Système complet :*** un système possédant suffisamment d'axiomes et de règle pour prouver n'importe quelle phrase (même si on supprime des axiomes et des règles).

Base de connaissances

Base de connaissance = ensemble de phrases exprimées dans un langage formel (logique par exemple).

2. Logique des prédicats de premier ordre

Motivation

- La logique propositionnelle a un pouvoir expressif limité (contrairement au langage naturel par exemple)

Exemple :

Comment exprimer les phrases suivantes en logique des propositions?

- Tous les étudiants du master sont des algériens
- Tous les étudiants étaient soit satisfaits par le cours de leur enseignant soit ils n'étaient pas.
- Chaque étudiant a une relation d'amitié avec un autre.

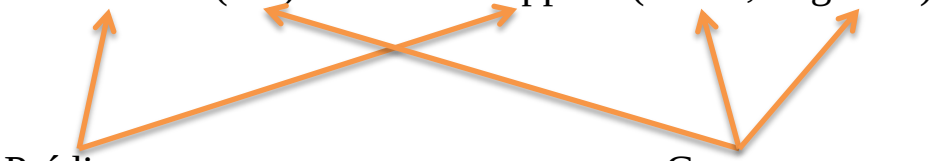
Définition de la logique des prédicats

- Une suite de symboles, de variables et de relations avec des quantificateurs universels et existentiels.
- Au lieu des symboles propositionnels P, Q, …, utiliser des ***prédicats*** et des ***termes:***

Exemples

1) Ali est une personne Samir supporte Feghouli

Personne (Ali) Supporte(Samir, Feghouli)

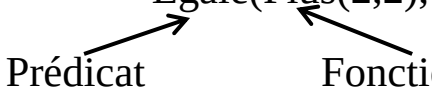


Prédicats Constantes

2) Mourad donne un livre à Salim

Donne (Mourad, un livre, Salim)

3) Egale(Plus(2,2), 4)



Prédicat Fonction

Eléments de base de la logique des prédicats

- Constantes : Ali, 2, Unix, ...
- Prédicats :Père, Frère, >, Egale, supporte, ...
- Fonctions : Sqrt(.), JambeGaucheDe(.), Plus, ...
- Variables : x, y, a, b, ...
- Connecteurs :∧, ∨, ¬, ⇒, ⇔
- Égalité : =
- Quantificateurs :∀,∃

Forme d'une expression en logique des prédicats

Phrases atomiques = Prédicat(terme$_1$, … , terme$_n$)

OU(***terme$_1$ = terme$_2$ =...***)

Avec : ***Terme = Fonction(terme$_1$, … , terme$_n$)***

OU ***Constante*** ou ***Variable***

Exemples

1) Frère(Mourad, Nassim)

2) > (Longueur(JambeGaucheDe(Omar)),1.2)

Phrases complexes = phrases atomiques réunies par des connecteurs

Tells ques: ¬ S, S1∧ S2 , S1∨S2 , S1 ⇒ S2 , S1 ⇔ S2

Exemples

1) Frère(Mourad, Nassim) ⇒ Frère(Nassim, Mourad)

2) > (2, 1) ∨ ≤ (1, 2)

3) > (2, 1) ∧ ¬ > (1, 2)

Quantification

La quantification permet d'exprimer des propriétés sur une collection d'objets sans avoir à les désigner chacun par un nom,

Deux types de quantificateurs:

- Quantificateur existentiel: ∃
- Quantificateur universel: ∀

1. Quantification existentielle

Forme générale:***∃ <variables><phrase>***

Exemple:

1. Il y a quelqu'un d'intelligent à Unix

➔∃ x A(x, Unix) ∧ Intelligent(x)

∃ x P: est équivalent à la disjonction d'instanciations de P

A (Ali, Unix) ∧ Intelligent(Ali)

∨ A(Omar, Unix) ∧ Intelligent(Omar)

∨ A(Said, Unix) ∧ Intelligent(Said)

∨ …

- Typiquement ∧ est le connecteur principal avec ∃
- Erreur fréquente: utiliser ⇒ comme connecteur principal avec ∃:

Ex : ∃ x A(x, Unix) ⇒ Intelligent(x)

2. il y a une personne"

⇔ ***∃x Personne(x)***

Variable liée par ∃

- Les variables dans une expression désignent des emplacements possibles pour des constantes:

  – x est une variable libre dans ***Personne(x)***

  – x est une variable liée dans ***∃ x Personne(x)***

Exemples :

1. ∃ y (Mange(Walid, y) ∧ Biscuit(y))

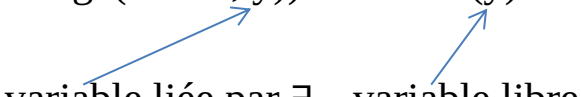
⇔"Walid mange un biscuit"

- Sans les parenthèses extérieures il y a une ambiguïté:

2. ∃ y Mange(Walid, y) ∧ Biscuit(y)

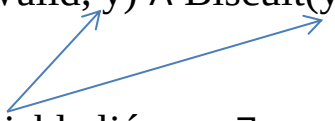
Pourrait être interprété comme:

a) (∃ y Mange(Walid, y)) ∧ Biscuit(y)



variable liée par ∃ variable libre

b) ∃ y (Mange(Walid, y) ∧ Biscuit(y))



variable liée par ∃

2. Quantification universelle

Forme générale: ***∀<variables><phrase>***

Exemple:

1. Toute personne à Unix est intelligente

⇔ ***∀x A(x, Unix) ⇒ Intelligent(x)***

∀ x P : est équivalent à la conjonction d'instanciations de P

A(Ali, Unix) ⇒ Intelligent(Ali)

∧ A(Omar, Unix) ⇒ Intelligent(Omar)

∧ A(Said, Unix) ⇒ Intelligent(Said)

∧ …

- Typiquement ⇒ est le connecteur principal avec ∀
- Erreur fréquente: utiliser ∧ comme connecteur principal avec ∀ :

Ex : ***∀x A(x, Unix) ∧ Intelligent(x)***

⇨ Signifie: "***Tout le monde est à Unix et tout le monde est intelligent***"

Lois de Morgan

- ***¬∃x φ*** est équivalent à ***∀x ¬φ***
- ***¬ ∀x φ*** est équivalent à ***∃x ¬φ***
- ***¬ (φ ∨ψ)*** est équivalent à $\neg\varphi \wedge \neg\psi$
- ***¬ (φ ∧ ψ)*** est équivalent à ***¬φ ∨¬ψ***

Propriétés des quantificateurs

- ***∀x ∀y*** est équivalent à ***∀y ∀x (permutation)***
- ***∃x ∃y*** est équivalent à ***∃y ∃x (permutation)***
- ***∃x ∀y*** n'est pas identique à ***∀y ∃x***

Ex: ***∃x ∀y Aime(x, y)***⇔ "Il y a quelqu'un qui aime tout le monde"

∀y ∃x Aime(x, y)⇔ "Chacun est aimé au moins par une personne"

Dualité des quantificateurs

Chaque quantificateur peut être exprimé à l'aide de l'autre:

1. ***∀x Aime(x, CrêmeGlacée)***⇔ ¬ ∃x ¬ Aime(x, CrêmeGlacée)
2. ***∃x Aime(x, Kiwi)***⇔ ¬ ∀x ¬ Aime(x, Kiwi)

Puissance de la logique des prédicats du 1^{èr} ordre

- Presque toutes les phrases du langage naturel peuvent être représentées en logique des prédicats du 1er ordre.
- Il n'y a pas de correspondance unique entre une phrase en langage naturel et une expression logique,

Exemples:

1. La mère de Ali est mariée au père de Ali

⇔***Marié(Père(Ali), Mère(Ali))***

2. Ali vit dans une maison jaune

⇔***a) Vit(Ali, Maison) ∧ Couleur(Maison, Jaune)***

b) ∃x Maison(x) ∧ Couleur(x, Jaune) ∧Vit(Ali, x)

3. Si la voiture appartient à Ali, alors elle est verte

⇔***a) Possède(Ali, Voiture) ⇒ Couleur(Voiture, Vert)***

b) ∃x Voiture(x) ∧Possède(Ali, x) ⇒ Couleur(x, Vert)

4. Certaines personnes aiment les serpents

⇔***a) ∃x (Personne(x) ∧ Aime(x, Serpent))***

b) ∃x ∀y (Personne(x) ∧ Serpent(y)) ⇒ Aime(x,y)

5. Tous les étudiants passent des examens

⇔ ***∀x Étudiant(x) ⇒ Passe-examen(x)***

b) ∀x (Étudiant(x) ⇒∃y Examen(y) ∧ Passe(x,y))

6. Si x est parent de y, alors x est plus vieux que y

⇔ ***∀x ∀y Parent(x,y) ⇒ PlusVieux(x, y)***

8. Si x est la mère de y, alors x est un parent de y"

⇔ ***∀x ∀y Mère(x,y) ⇒ Parent(x, y)***

9. Chacun est loyal envers quelqu'un"

⇔ ***∀x ∃y Personne(x) ∧ Personne(y) ∧ LoyalEnvers(x,y)***

Ou ***∃y ∀x Personne(x) ∧ Personne(y) ∧ LoyalEnvers(x,y) ?***

10. il existe quelqu'un envers qui chacun est loyal"

→ ***phrase ambiguë !!***

10. Les gens tentent d'assassiner les dirigeants envers lesquels ils ne sont pas loyaux

⇔ ***∀x ∀y Personne(x) ∧ Dirigeant(y) ∧TenterAssassiner(x,y) ⇒ ¬LoyalEnvers(x,y)***

11. la seule chose que les gens tentent de faire est d'assassiner ceux envers qui ils ne sont pas loyaux"

→ ***une autre phrase ambiguë !!***

Égalité

Terme1 = Terme2 est vrai étant donné une interprétation si etseulement si terme1 et terme2 font référence au même objet

Exemples:

1. ***Père(Omar) = Mohamed***
2. ***Téléphone(Mourad) = 0770576344***

Inférence en logique du 1er ordre

Prémisses:

1. Si x est un parent de y, alors x est plus âgé que y
2. Si x est la mère de y, alors x est un parent de y
3. Fatma est la mère de Zahra

Conclusion:

– Fatma est plus âgé que Zahra

Correspondance en logique du 1er ordre:

Prémisses:

1. $\forall x \forall y$ Parent(x,y) $\Rightarrow$PlusAgé(x, y)
2. $\forall x \forall y$ Mère(x,y) $\Rightarrow$ Parent(x, y)
3. Mère(Fatma, Zahra)

Conclusion:

PlusAgé(Fatma, Zahra)

L'inférence est obtenue par des axiomes et des règles (i.e. par des transformations syntaxiques) qui étendent ceux de la logique propositionnelle.

Unification

C'est le processus qui rend 2 expressions identiques.

Remarque: En logique propositionnelle 2 expressions sont les mêmes seulement si elles sont syntaxiquement identiques, la présence de variables en logique des prédicats complique ce fait.

Exemples :

1. Humain(x) = Humain(Socrate) **si et seulement si** x=Socrate

Il faut parfois substituer une fonction à une variable:

2. Humain(x) → Mortel(x)

Humain(PèreDe(Platon))

↓ Avec: x/PèreDe(Platon)

Humain(PèreDe(Platon)) → Mortel(PèreDe(Platon))

3. Une substitution σ unifie les phrases atomiques p et q ssipσ = qσ

P	Q	σ	P=Q après unification
Connaît(Ali, x)	Connaît(Ali, Omar)	{ *x/Omar* }	Connaît(Ali, Omar)=Connaît(Ali, Omar)
Connaît(Ali, x)	Connaît(y, Karim)	{ *y/Ali, x/Karim*}	Connaît(Ali, Karim)=Connaît(Ali, Karim)
Connaît(Ali, x)	Connaît(y, Mère(y))	{ *y/Ali, x/Mère(Ali)*}	Connaît(Ali, Mère(Ali))=Connaît(Ali, Mère(Ali))
Connaît(Ali, x)	Connaît(x, Karim)	{échec}	

Principe de l'unification :

Unifier les prémisses des règles avec des faits connus, puis appliquer l'unificateur à la conclusion

Exemple:

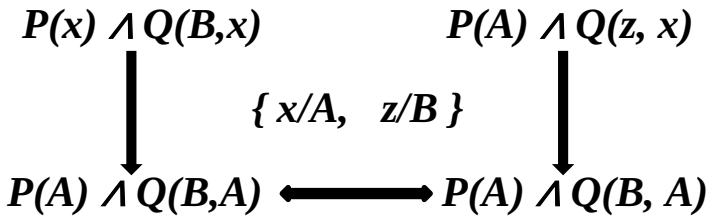
si on connaît q (dernière colonne) et on a la règle

Connaît(Ali, x) ⇒ Aime(Ali, x)

Alors on peut conclure :

- Pour q=Connaît(Ali, Omar)➔ Aime(Ali, Omar)
- Pour q= Connaît(Ali, Karim) ➔Aime(Ali, Karim)
- Pour q= Connaît(Ali, Mère(Ali)) ➔Aime(Ali, Mère(Ali))

Règle : Une variable doit être substituée de manière consistante pour toutes ses occurrences dans les expressions à unifier



Preuve par chaînage avant (Forward Chaining)

Trouver une preuve pour la conclusion ***PlusVieux(Fatma, Zahra)*** à l'aide d'un chaînage avant :

1. Mère(Fatma, Zahra) (donné)

2. Vivant(Fatma) (donné)

3. ∀x∀y Mère(x,y) ⇒ Parent(x, y) (donné)

4. ∀x∀y (Parent(x,y)∧Vivant(x)) ⇒PlusVieux(x, y) (donné)

Par remplacement du fait (1) dans la règle (3) on obtient le nouveau fait suivant:

5. Parent(Fatma, Zahra)

Remplaçons les faits (5), (2) dans la règle (4) on obtient le nouveau fait:

6. PlusVieux(Fatma, Zahra) (ce qui est demandé)

Principe du chaînage avant: effectuer une série de remplacement des faits donnés dans des règles aussi données jusqu'à arriver au fait à vouloir prouver (il s'agit d'un chaînage guidé par les données de départ Data-Driven)

Preuve par chaînage Arrière (Backward chaining)

Principe: Partir d'un objectif (à prouver) et dériver de nouveaux sous-objectifs jusqu'à n'avoir que des sous-objectifs connus comme vrais.

➔ Semblable à la réduction de problème (chaînage guidé par l'objectif à prouver).

Exemple: Prenons le même exemple précédent

Preuve par chaînage arrière de ***PlusVieux(Fatma, Zahra)*** à partir des prémisses:

1. Mère(Fatma, Zahra)

2. Vivant(Fatma)

3. $\forall x \forall y$ Mère(x,y) $\Rightarrow$ Parent(x, y)

4. $\forall x \forall y$(Parent(x,y) $\wedge$ Vivant(x)) $\Rightarrow$PlusVieux(x, y)

objectif: ***(i) PLusVieux(Fatma, Zahra)***

on constate que l'objectif (i) correspond à la partie droite de (4)

⇨ Obtenir de nouveaux sous-objectifs:

(ii) Parent(Fatma, Zahra)

(iii) Vivant(Fatma)

(iii) correspond à (2) (déjà prouvé ou donné)

(ii) correspond à la partie droite de (3)

⇨ On obtient un nouveau sous-objectif:

(iv) Mère(Fatma, Zahra)

(iv) correspond à (1) (qui est vrai)

On constate que tous les sous objectifs sont vrais ➔ l'objectif de départ est aussi vrai

Remarques :

- Le chaînage arrière est dirigé par les objectifs ("goal directed").
- Le chaînage arrière est à la base de la programmation logique (Prolog)

3. Autres Logiques

a) Logique de croyance

Ex : Croire(Ali, Père(Mourad, Ibrahim))

b) Logique temporelle

Permet de raisonner à propos du temps

c) Logique non-monotone

Permet aux valeurs de vérité associées aux faits de changer

d) Logique floue

Accompagne les faits de valeurs de vraisemblance

4. Règles de production

- Le formalisme le plus utilisé par les systèmes experts.
- Des formes simple et efficace.
- Base sur des règles de la forme « Si … Alors … »

Exemples :

1. S'il n'y a pas d'image mais que vous entendez le son, vérifiez le réglage d'intensité de l'écran"

Si *Non Image* ***et*** *Son* ***Alors*** *Intensité-Écran*

2. Un client est insolvable, si son compte bancaire est négatif, et qu'il ne possède aucun titre

Si ((*Solde-Compte < 0****)et (****Montant_Titre=0****)) Alors***
Client_Insolvable

Avantages

- Facilité d'expression
- Efficacité et clarté de représentation
- Développement aisé => large diffusion de logiciels
- Formation de programmeurs moins exigeante (Ing.Connais).
- Facilité de M.à.j
- Possibilité de vérification et de validation de connaissances.
- Implication des utilisateurs dans le développement.
- Possibilité d'appeler d'autres formalismes tels que les objets structurés.

5. Les réseaux sémantiques

Quelques caractéristiques

- Un graphe orienté
- Les nœuds représentent les concepts
- Les flèches représentent les liens sémantiques entre les concepts tels que :

- Est un : relation d'inclusion (permet l'héritage)
- Instance de : relation d'appartenance entre une instance et la classe à laquelle il appartient.

- Utilisés surtout dans l'interprétation du langage naturel
- Une instance/Sous classe hérite toutes les propriétés sémantiques de la classe parent.
- Une instance partage tous les liens sémantiques de la classe à laquelle elle appartient
- L'héritage peut être simple ou multiple. C.à.d une sous classe possède plusieurs classes parents.

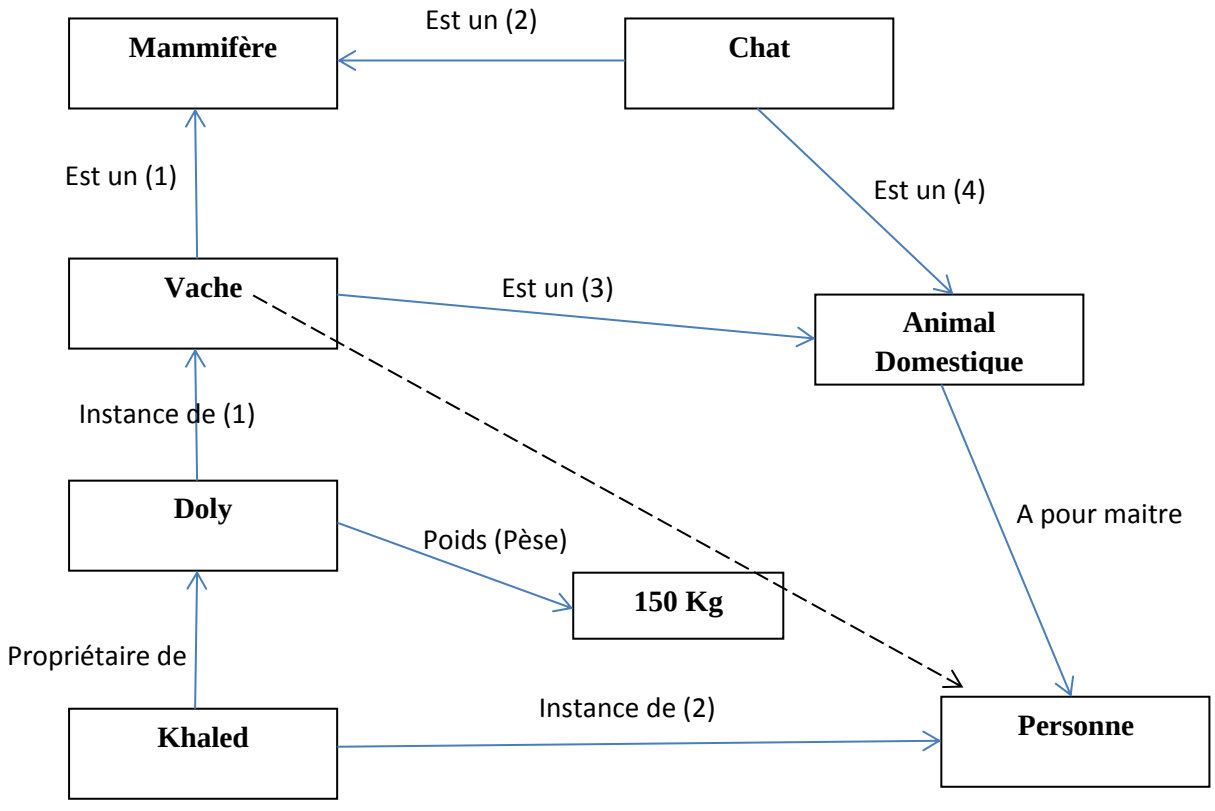


Fig II. 1. Caractéristiques du réseau sémantique

Exemple d'un réseau sémantique simple

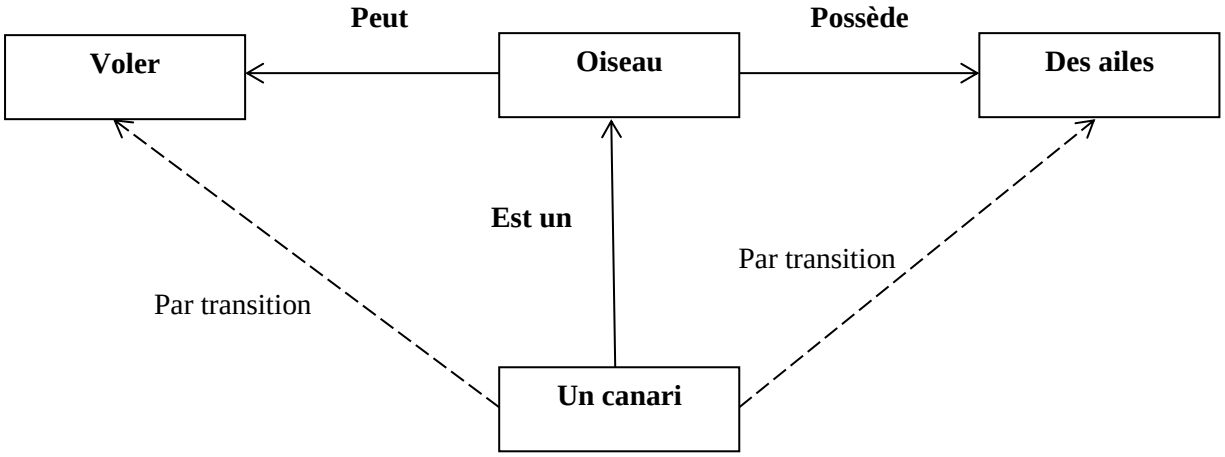


Fig II.2. Exemple d'un réseau sémantique simple

Interprétation du réseau

a) - Connaissances envisagées par le graphe

- Un canari est un oiseau
- Un oiseau possède des ailes
- Un oiseau peut voler

b)- connaissances déduites

- Un canari possède des ailes
- Un canari peut voler

Un exemple d'un réseau sémantique plus détaillé

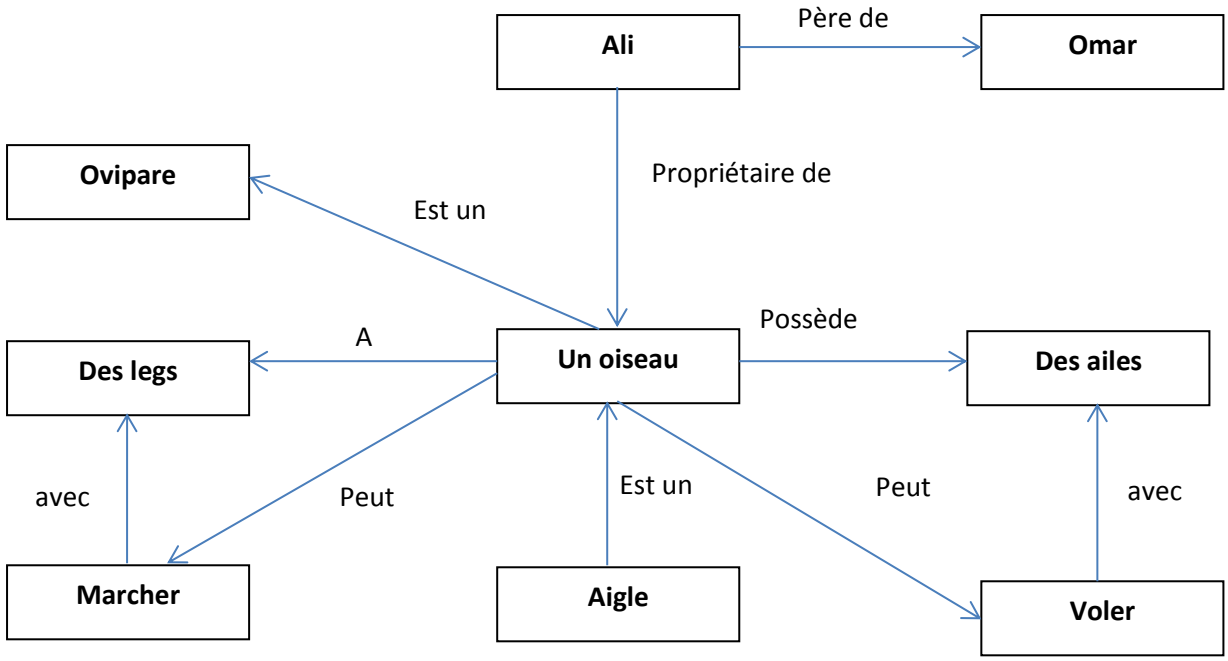


Fig II.3. Un réseau sémantique détaillé

Avantages

- La puissance d'expression
- Les propriétés d'héritage

Inconvénients

- Manque de standardisation du formalisme.
- Difficulté de M.à.J du graphe.
- Impossibilité d'exprimer des programmes.

6. Objets structurés (P.O.O)

Un objet est une structure qui contient à la fois des données (connaissances déclaratives) et des procédures ou des méthodes (connaissances procédurales)

Exemple:

Point : objet

X, Y : entier

Distance : point X point ➔ réel

Centre : point X point ➔point

Fin

Concepts de base

- Classes et sous-classes
- Objets
- Instances
- Attributs
- Méthodes

Quelques caractéristiques

- Encapsulation (Données + Méthodes)
- Héritage (instance, objet, classe, sous-classe)
- Polymorphisme
- Instanciation

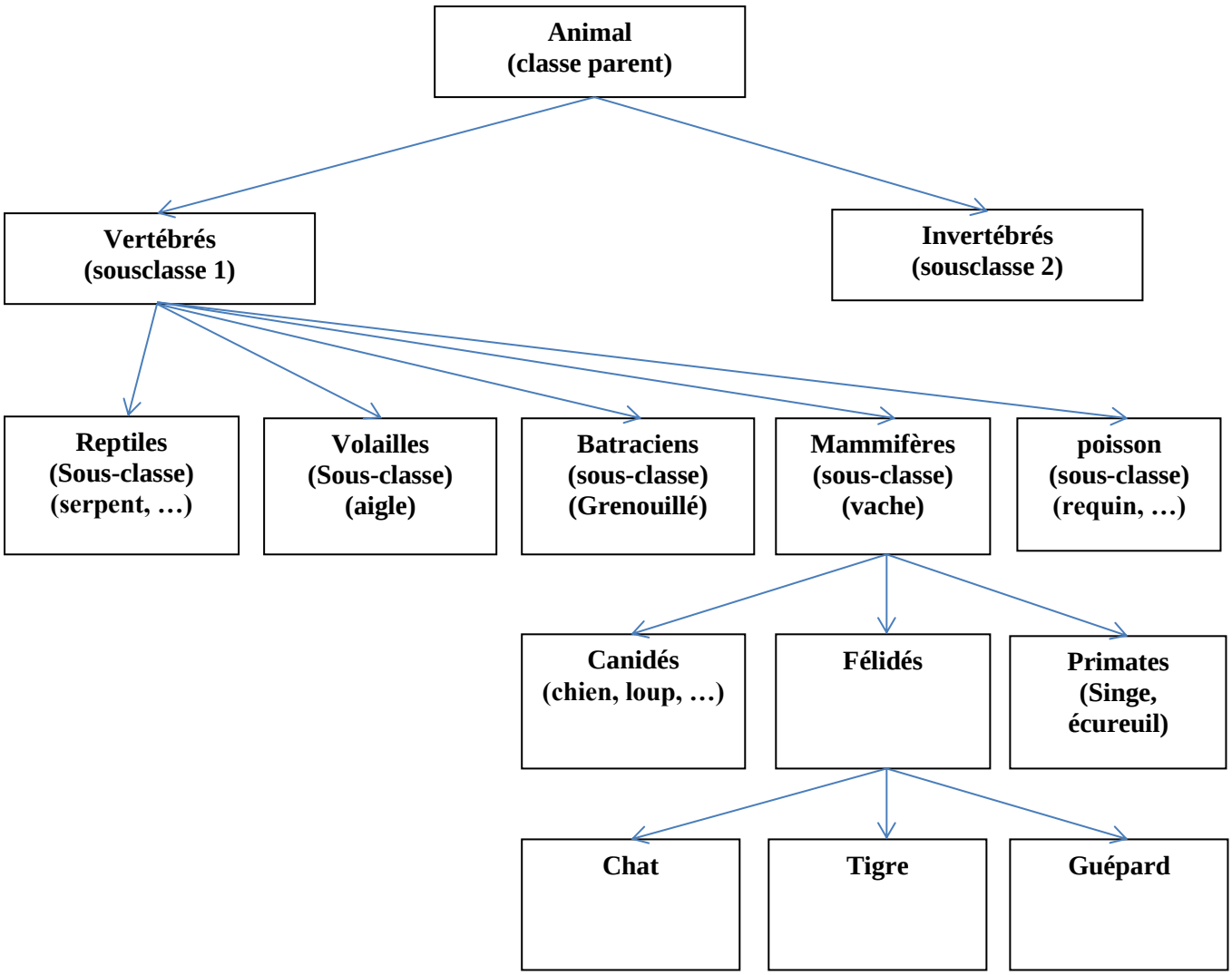


Fig II.3. Hiérarchie de la classe « Animal »

Objets et classes

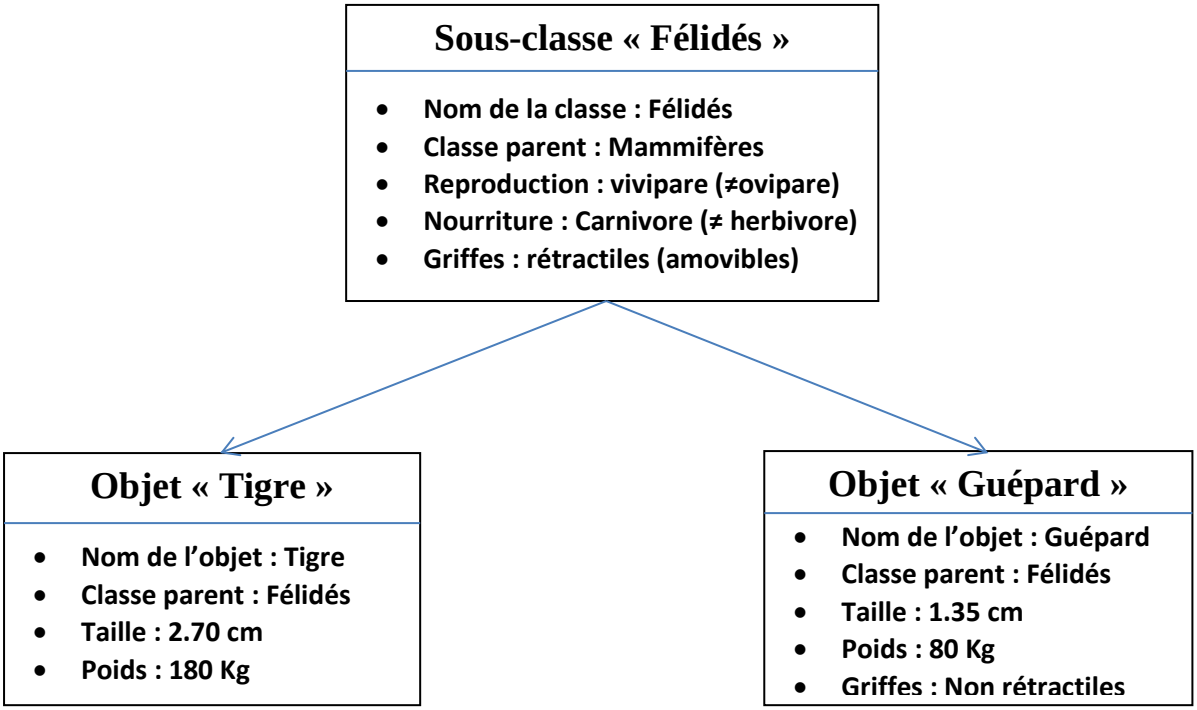


Fig II.4. Représentation des objets et mécanisme d'héritage

OBS: Un objet hérite les propriétés de la classe parent

7. Réseau de neurones (Réseau neuronaux)

Motivation

- L'humain est capable de manipuler des grandes quantités de connaissances à l'aide son cerveau
- Le cerveau est constitué de neurones (cellules nerveuses) qui constituent les éléments fondamentaux du système nerveux.
- Un neurone est constitué de : noyau, axone, dendrites

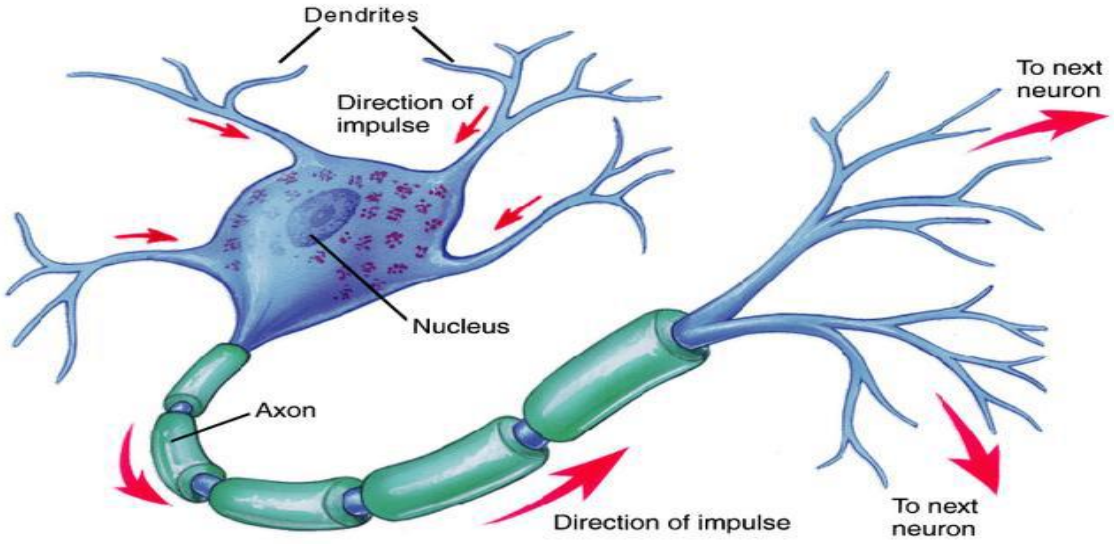


Fig II.5. Architecture du neurone biologique

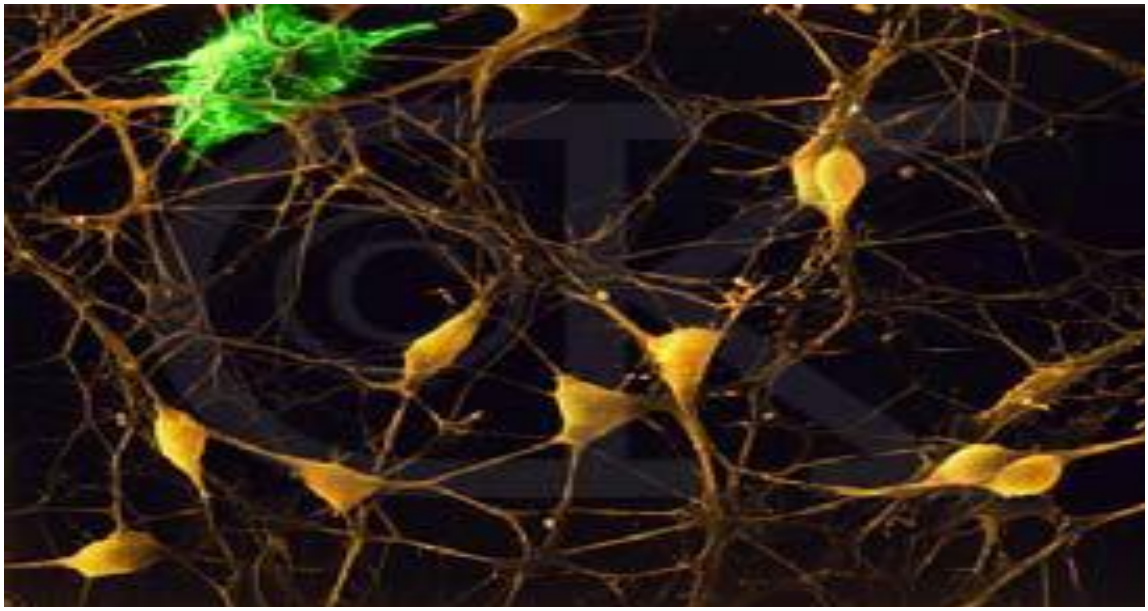
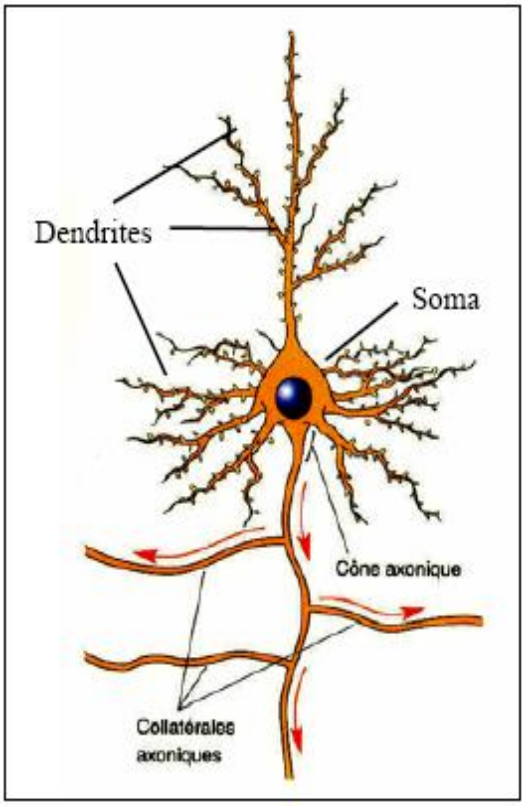


Fig II.6. Architecture du système nerveux

Caractéristiques du neurone biologique



- **Diamètre de la cellule:** 0.01 - 0.05 mm,
- **Soma (corps cellulaire)**
- formes variables (gén. sphériques),
- 20 μm de diamètre,
- contient le noyau,
- entourée d 'une membrane de 5 nm d'ép.
- **Axone**
- unique aux cellules nerveuses,
- diamètre: 1-25 μm (humain),
- longueur: 0.1 mm to 1 mm. (!!!),
- connexion vers les autres neurones (synapses),
- permet la transmission d'information,
- **Dendrites**
- Reçoivent les signaux des autres neurones,
- Chacune couverte de centaines de synapses.

Chiffres Importants

- Nombre de neurones dans le cerveau: ~ 10^{11}
- Nombre de connexions par neurone: ~ 10^{4} - 10^{5}
- Temps de cycle (switching time): ~ 10^{-3} seconde
- Temps moyen d'une activité cognitive: ~ 10^{-1} seconde

(ex. reconnaissance de visages)

⇨ Il n'y a donc de la place que pour 100 cycles de traitement ($10^{-1}/10^{-3}$)

ce qui est insuffisant pour une activité complexe !!!

⇨ Le cerveau doit donc effectuer des opérations en parallèle !!!

Caractéristiques du système nerveux

- Robuste et tolérant aux erreurs et aux "pannes".
- Flexible et doté de capacités d'apprentissage, pas besoin d'être explicitement "programmé" (plasticité synaptique).
- Capable de traiter de l'information partielle (floue), incertaine (probabiliste) ou incomplète.
- Très massivement parallèle.
- Les capacités (intelligence?) résident dans les connexions entre les milliards de neurones du cerveau humain.

Principe de fonctionnement d'un neurone biologique

Chaque neurone reçoit des impulsions électriques à travers ses dendrites, si ces impulsions sont suffisamment importantes, l'axone transmet une impulsion électrique pour exciter les différents neurones connectés qui seront à leur tour activés en fonction de l'intensité des impulsions transmises.

Réseaux de neurones artificiels/Formels (RNA)

Applications

1. Contrôle
- Pilotage automatique:

  Alvinn: réseau de pilotage d'un véhicule à partir d'images vidéo,
- Synthèse vocale:

  NETtalk: un réseau qui apprend à prononcer un texte en anglais
- Processus de fabrication/production,

2. Reconnaissance/classification/analyse
   - Textes imprimés, caractères manuscrits (codes postaux), parole,
   - Images fixes (ex. visages), animées ou clips vidéo,
   - Risques (financiers, naturels, etc.)
3. Prédiction
   - économie, finances, analyse de marchés, médecine,

Relation avec le réseau de neurone biologique

- Parallélisme massif
- Calcul local
- Elément simple de calcul de type "neurone artificiel".

Outil principal de fonctionnement

- Apprentissage (par rétro-propagation).

Domaines de succès

- Reconnaissance de la parole (TDNNs)
- Contrôle robotique (ALVINN)
- Reconnaissance de caractères (OCR),
- Prévisions financières.

Quelques repères historiques

1943: J. McCulloch& W. Pitts

- Proposent un modèle simple de neurone capable de produire une machine de Turing,
- Démontrent qu'un assemblage synchrone de tels neurones est une machine universelle de calcul (c.à.d. n'importe quelle fonction logique peut être représentée par des unités à seuil).

Warren S. McCulloch

William Pitts

1948: D. Hebb

- Propose une règle d'apprentissage pour des réseaux de neurones.

1958: F. Rosenblatt

- Propose le modèle du Perceptron et démontre son théorème deconvergence.

1969: M. Minsky& S. Papert

- Démontrent les limitations du modèle du Perceptron.

1985:

- Apparition d'apprentissage par rétro-propagation pour les réseaux multi-couches.

Architecture du neurone formel

McCulloch et Pitts (1943) ont modélisé le fonctionnement d'un neurone biologique et imaginé une reproduction artificielle mimétique du raisonnement humain.

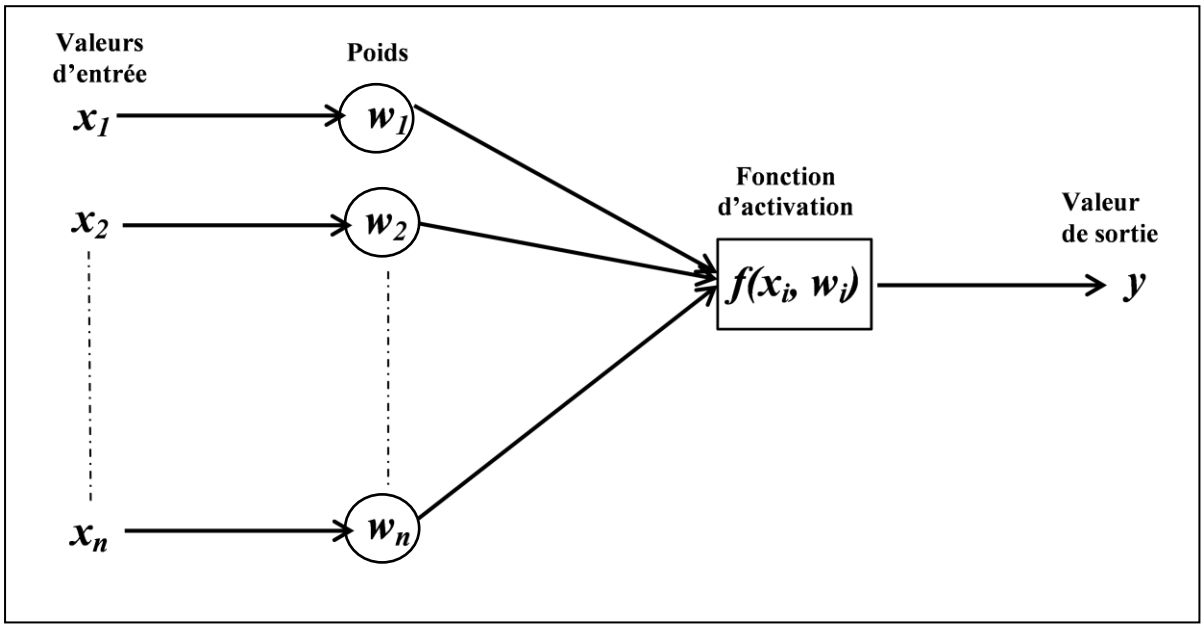
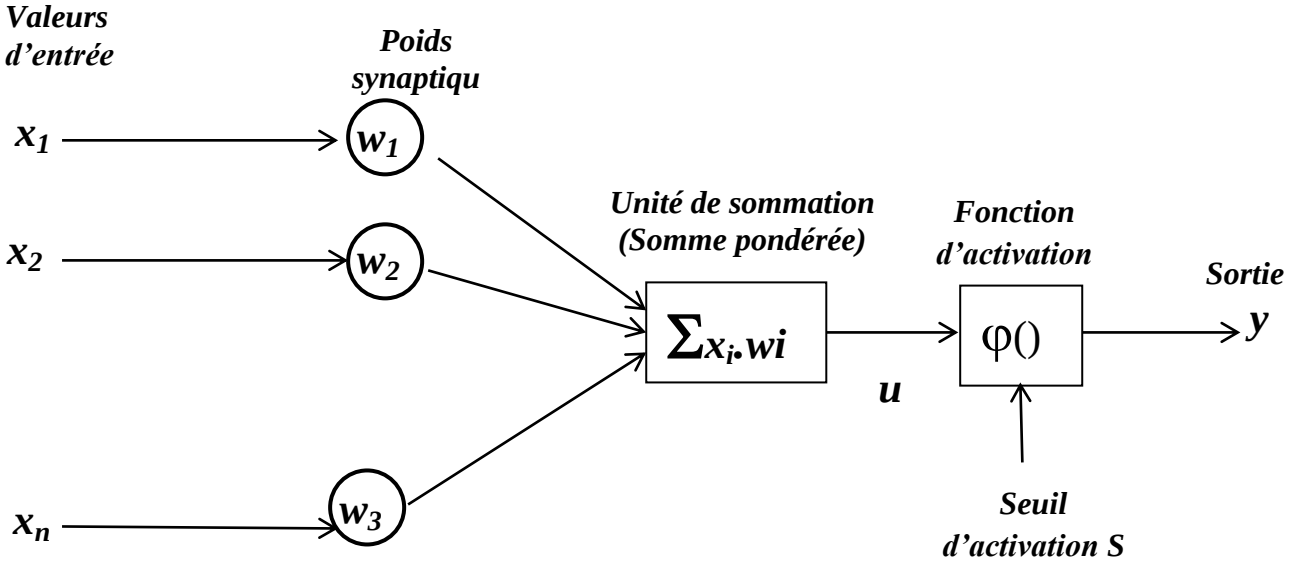


Fig II.6. Architecture du réseau de neurone formel

Principe de fonctionnement d'un neurone formel (modèle de McCulloch et Pitts/Modèle linéaire)

Un réseau de neurone formel ou aussi appelé automate à seuil base sur le même principe d'un réseau de neurone biologique comme c'était expliqué auparavant



Ce modèle base sur deux équations:

$$u = \Sigma x_i \, .wi \qquad (1)$$

$$y = \varphi(u - S) \qquad (2)$$

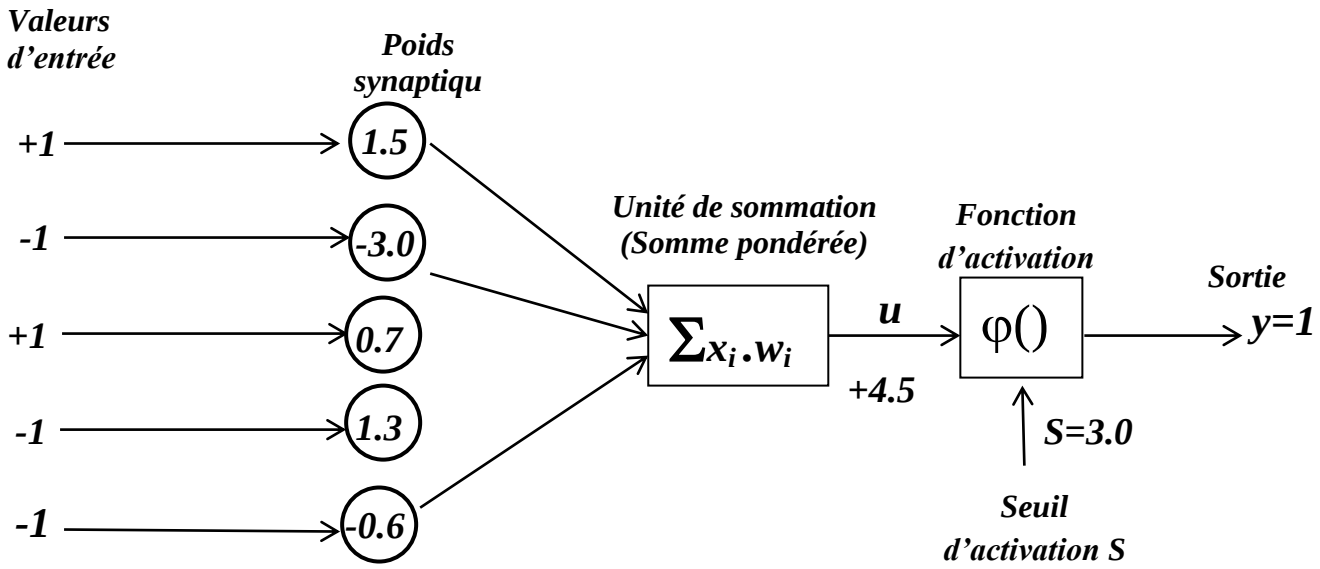
avec :

$x_1, x_2, ..., x_n$: sont les entrées,
$w_1, w_2, ..., w_n$: sont les poids synaptiques du neurone k,
u : la sortie de l'unité de sommation,
S : le seuil,
$\phi(.)$: la fonction d'activation,
y : le signal de sortie du neurone k.

Valeur de sortie : le neurone formel calcule la somme des entrées x_1, x_2, …, x_n pondérées par les poids synaptiques w_1, w_2, …, w_n et la comparer avec le seuil S, si le résultat est supérieur au seuil, alors la valeur renvoyée en sortie est +1 (activation du neurone), sinon la valeur renvoyée est 0

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

Exemple : Fonctionnement d'un RNA (modèle de McCulloch & Pitts)

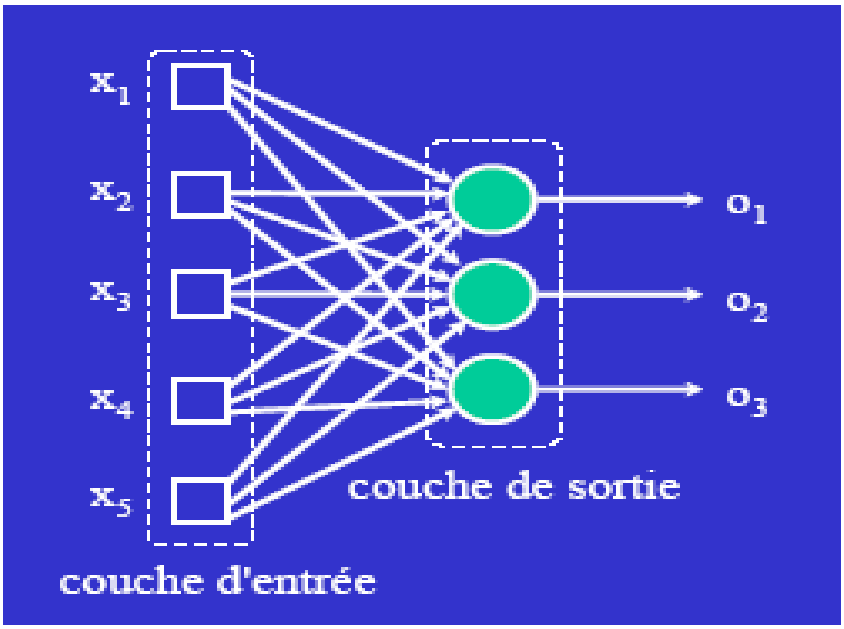


Fonction d'activation

La fonction d'activation définit la valeur de sortie d'un neurone en termes des niveaux d'activité de ses entrées. Cette fonction peut avoir plusieurs formes (linéaire, exponentielle, sigmoïde, …)

Topologies de réseaux de neurones

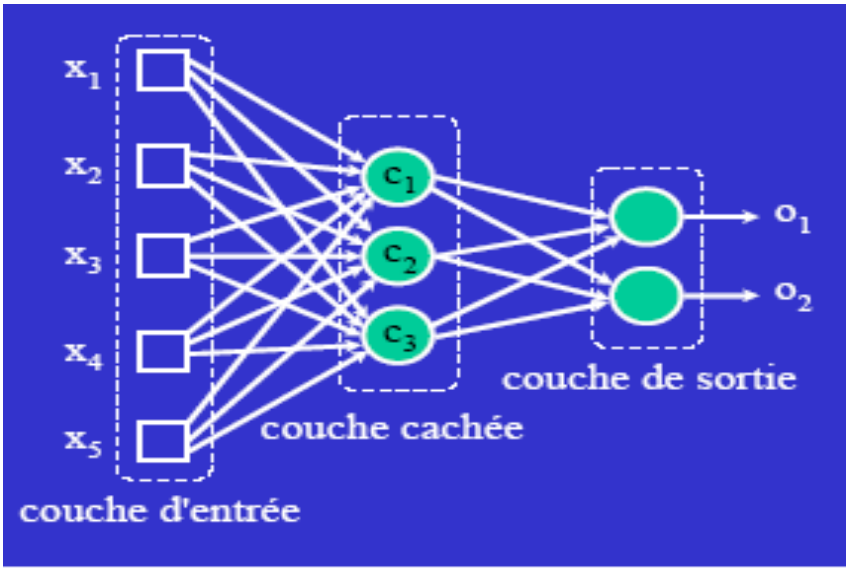
1. Réseau acyclique ("feedforward") à une seule couche



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

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

2. Réseau acyclique multi-couches



Valeurs de sortie :

Couche cachée

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

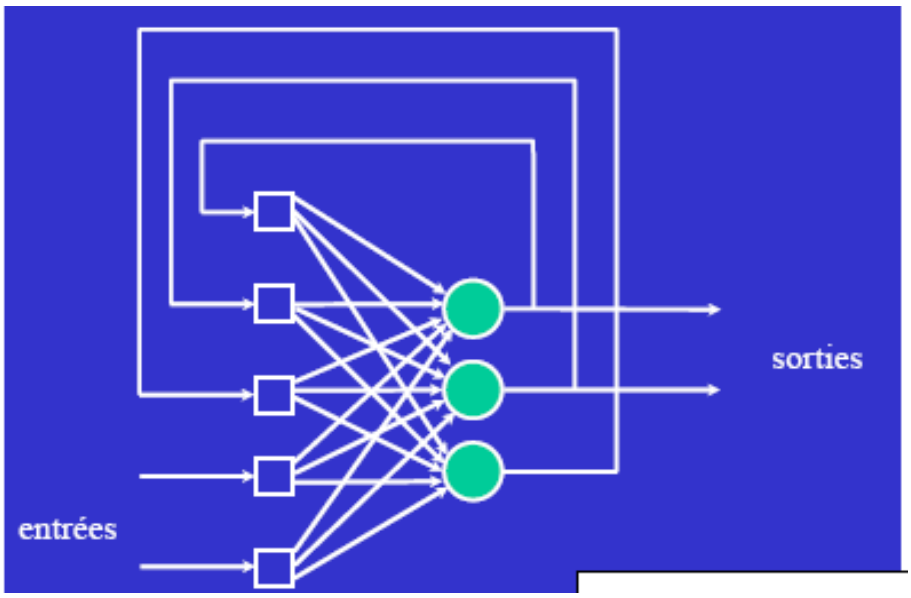
$$c_j = 0 \quad \text{sinon}$$

Couche de sortie

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

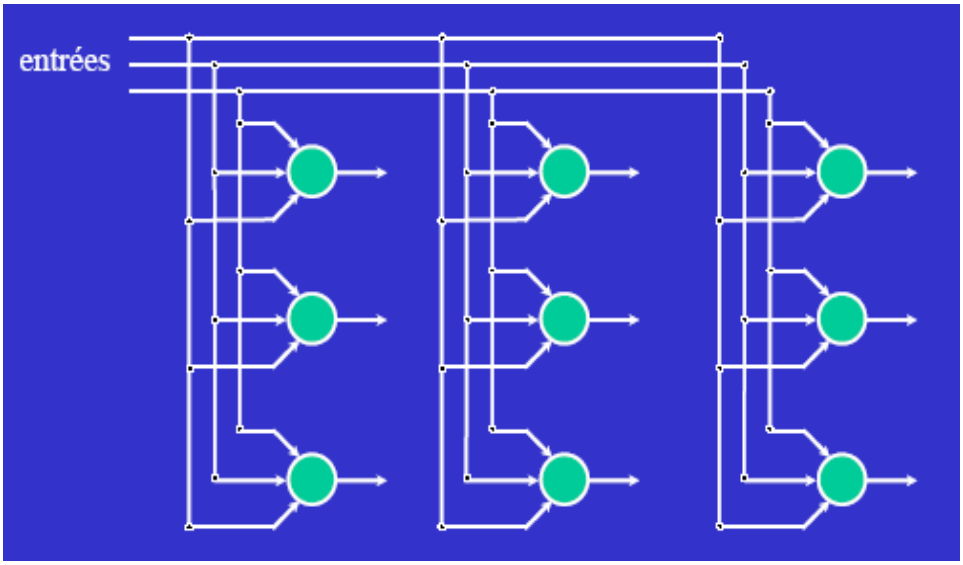
$$o_i = 0 \quad \text{sinon}$$

3. Réseau récursif (réseau de Hopfield)



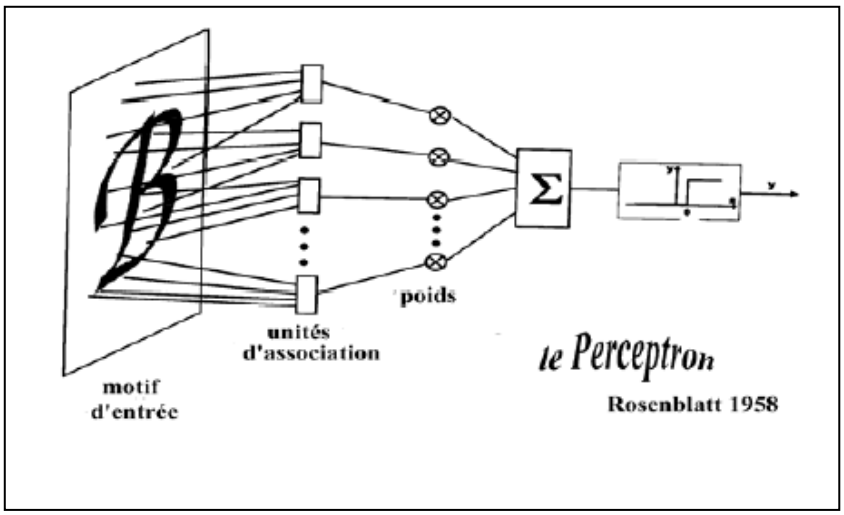
OBS : Chaque unité i est connectée à chaque autre unité j par un poids w_{ij} les poids sont supposés symétriques: $w_{ij} = w_{ji}$

4. Réseau "en treillis"



Réseau en treillis 3x3 bi-dimensionnel

Le Perceptron (F. Rosenblatt, 1958)

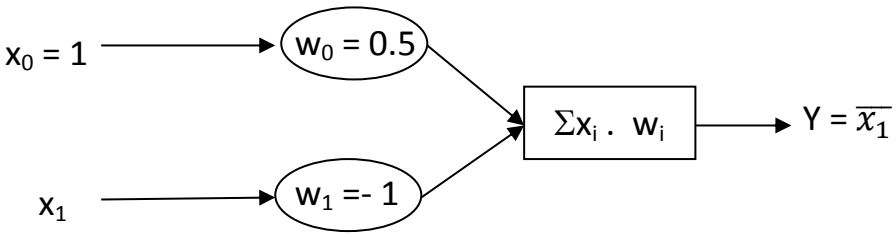


Avec :

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

Exemple : Réalisation d'un NON logique par un perceptron

Entrée x_1	Sortie
0	1
1	0



Exercice: Réaliser les portes logiques OR, AND, XOR en utilisant un perceptron (Voir TD)

Théorème d'apprentissage (F. Rosenblatt)

Étant donné suffisamment d'exemples d'apprentissage, il existe un algorithme qui apprendra n'importe quelle fonction linéairement séparable.

Algorithme d'apprentissage du Perceptron

Entrées: *ensemble d'apprentissage {(x1,x2,…,xn,t)}*

Méthode

initialiser aléatoirement les poids w(i), 0<=i<=n

Répéter *jusqu'à convergence:*

Pour *chaque exemple*

Calculer la valeur de sortie o du réseau.

Ajuster les poids:

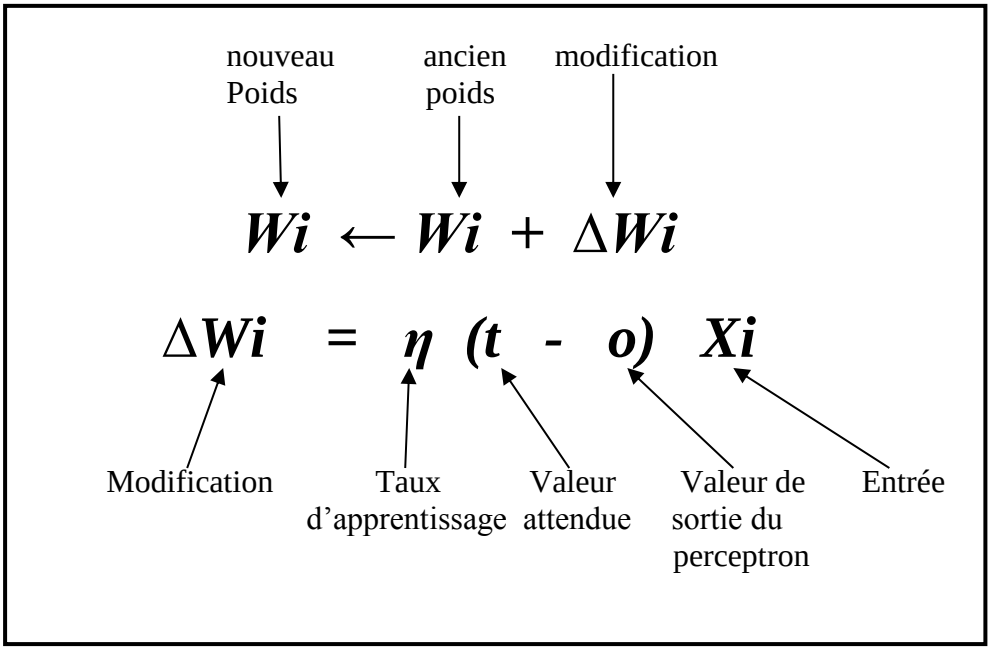
$\Delta wi = \eta(t - o)\, xi$ /* *t : sortie attendue, O : sortie calculée*

$wi \leftarrow wi + \Delta wi$ /* *par le perceptron*

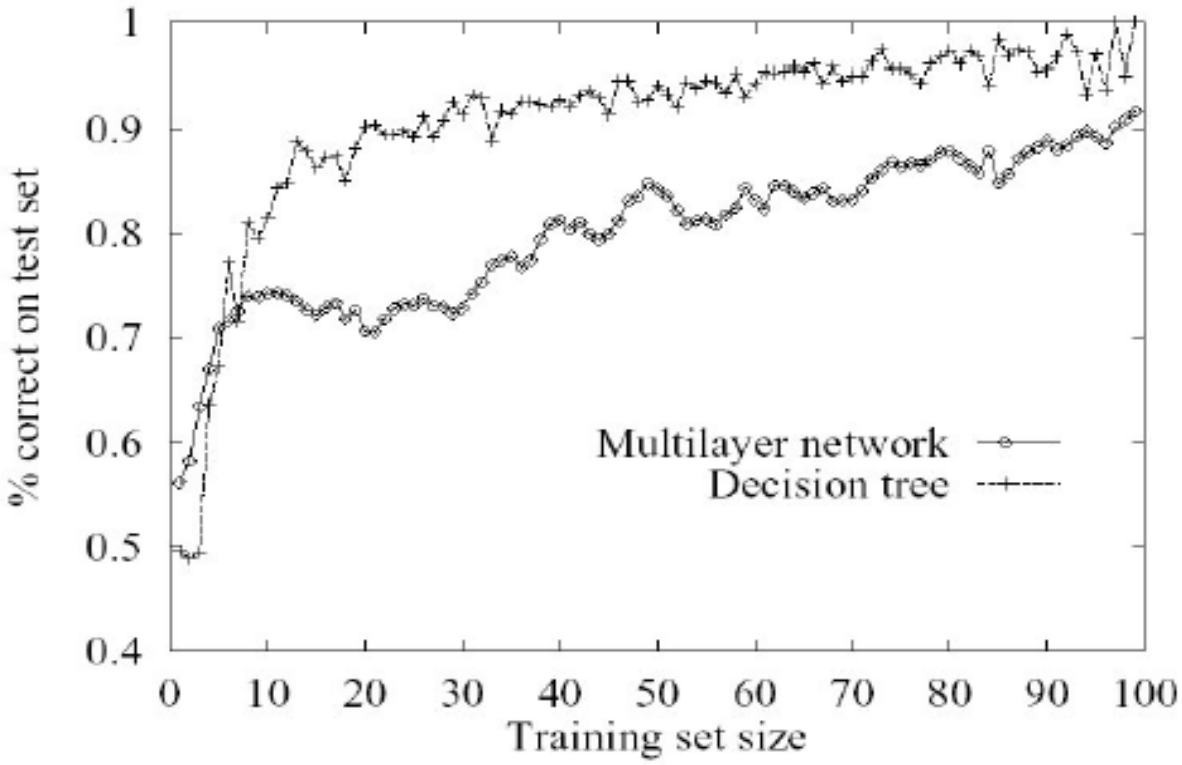
FPour

Fin Répéter

Fin Méthode



Réseaux Vs arbre de décision



Quand utiliser des réseaux de neurones artificiels pour résoudre des problèmes d'apprentissage?

- lorsque les données sont représentées par des paires attributs-valeur,
- lorsque les exemples d'apprentissage sont bruités,
- lorsque des temps d'apprentissage (très) longs sont acceptables,
- lorsqu'une évaluation rapide de la fonction apprise est nécessaire,
- lorsque la compréhension par l'utilisateur de la fonction apprise est sans importance.

Quelques applications (voir PDF associé)