Intelligence Artificielle

Année universitaire 2018/2019

Pr. Jaouad Boumhidi

Plan du cours

Introduction à l'Intelligence Artificielle

Chap 1: Représentation des connaissances et inférence en logique des propositions et des prédicats

Chap 2: Représentation des connaissances et inférence en logique floue

Chap 3: Introduction à la théorie d'Apprentissage:

- Arbre de décosion
- Réseau de neuronnes,
- SVM

Définir l'Intelligence Artificielle ?

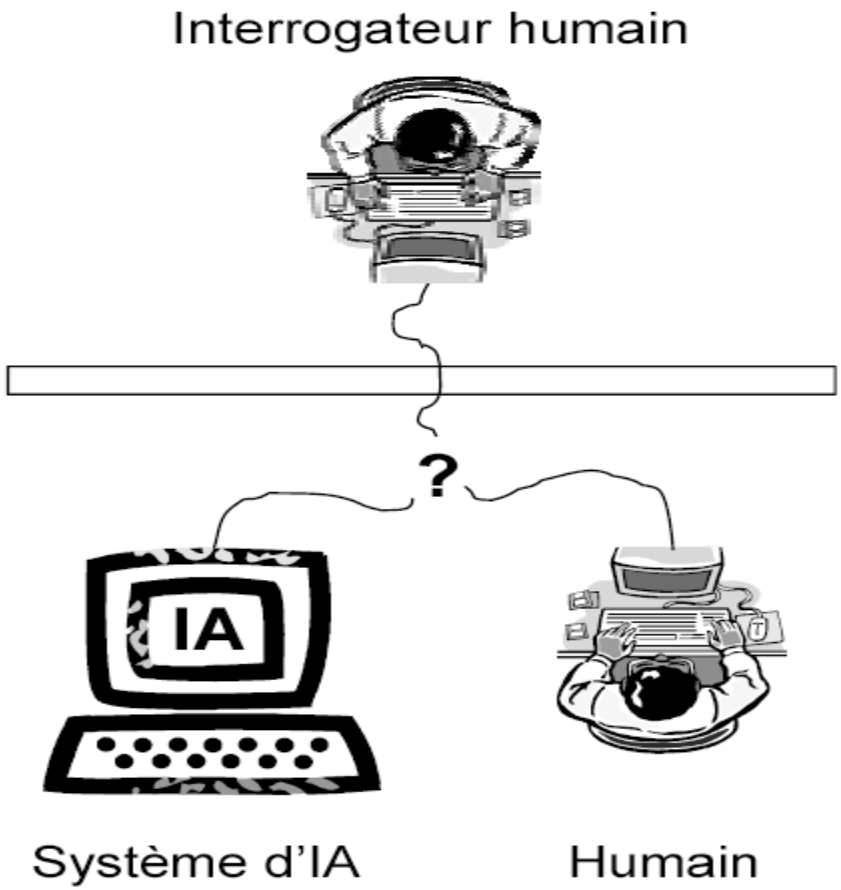
IA: **Science qui étudie des agents intelligents** (Programme intelligent+ Architecture matériel) :

Agents intelligents: Quatre types de définitions:

- **1)** Système qui **pense** comme les **humains (cognitive)**
- **2)** Système qui **agit** comme les **humains (Système passant le test de Turing)**
- **3)** Système qui **pense rationnellement (Logicisme: pensée logique)**
- **4)** Système qui **agit rationnellement**

Rationalité: consiste à faire l'action appropriée, cela dépend essentiellement de la mesure de performance qui définit le critère de succès.

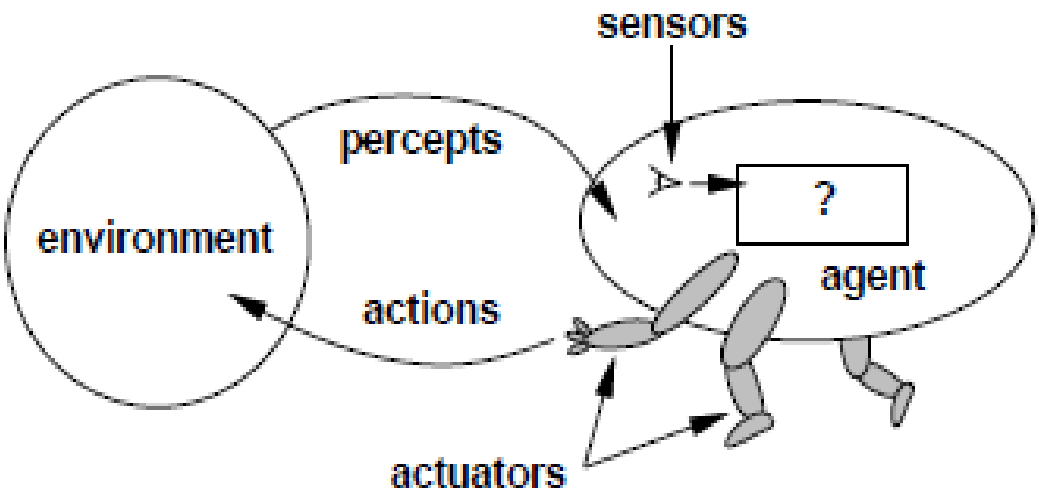
Le test de turing: Un testeur humain doit identifier s'il dialogue avec un humain ou une machine : la machine réussit le test si elle n'est pas identifiée en tant que telle.



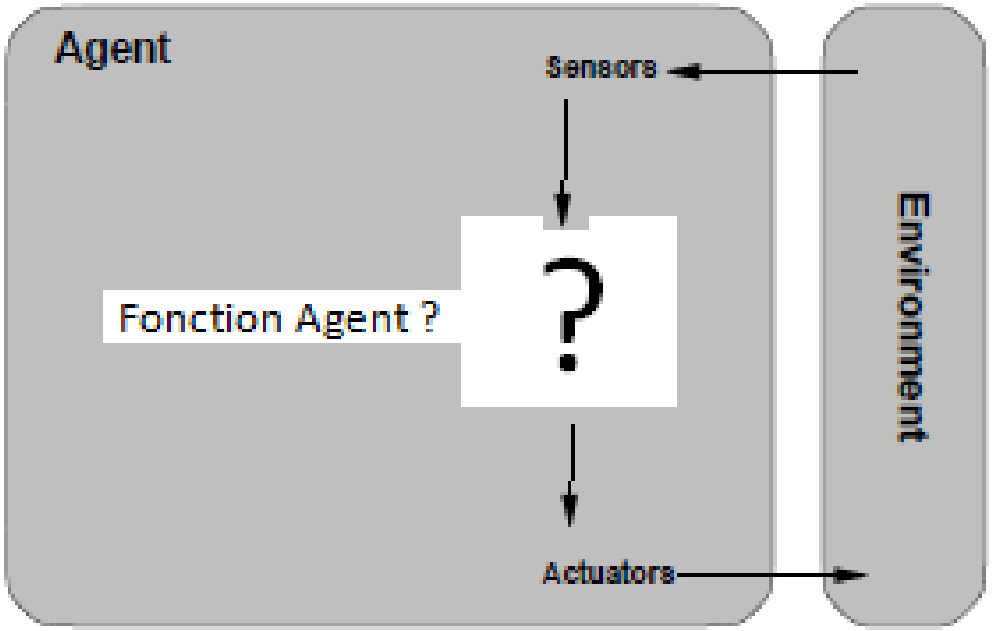
Agent intelligent
(**Programme** + Architecture)

Définition d'agent:

Entité considéré comme percevant(perception) son environnement par des capteurs (sensors)et agit rationnellement sur cet environnement par des effecteurs(actuators).



Agents interagissant avec l'environnement



<Percepts, *Fonction Agent*, *Programme Agent* et Action(Décision)>

- Percepts: Entrées perceptives d'un agent à un instant donnée.

- Actions: Action à un instant donnée, (qui dépend de la séquence de percepts prise jusqu'à ce moment).

- ***Fonction Agent:***

Fonction mathématique décrivant le comportement d'un agent et qui fait lier une action à chaque séquence de percepts.

Un agent est complètement déterminé par la fonction d'agent qui étant donnée une séquence de perception **renvoie une action**.

- ***Programme Agent:***

Implémentation de la fonction Agent qui s'exécute sur l'architecture Agent

Chap. 1 Représentation des connaissances en *Logique* des propositions et des prédicats

Plan:

- Logique des propositions
- Logique des prédicats
- Moteurs d'inférences à base de règles
- Application en Prolog

- **Deux aspects dans une représentation des connaissances:**

–La syntaxe: définit les énoncées valides

–La sémantique: Définit les règles de détermination de la vérité d'un énoncé (***Manière par laquelle l'agent atteint son but***)

Logique des propositions(LP)

La logique des propositions est ***un langage formel*** constitué d'une ***syntaxe*** et d'une ***sémantique***

Syntaxe:

Proposition?

C'est une expression (énoncé) qui prend ses valeurs(de vérité) dans l'ensemble {**Vrai, Faux**}

Exemples:

Mohammed est le père de Ali

Les droites D1 et D2 sont parallèles.

Syntaxe de LP

SYMBOLES PROPOSITIONNELS : (p, q, r,….) :

Pour représenter et manipuler une proposition on utilise des symboles propositionnels dont la valeur peut être vraie ou faux.

Nous utilisons ***des lettres en minuscules*** pour chaque symbole (p, q, r,….)

Exemple:

p=Driss est le père de Rachid

CONNECTEURS:

¬	Négation
∧	Conjonction
∨	Disjonction
→	Implication
↔	Équivalence

Syntaxe de LP

- Une énoncée tel que $p \rightarrow q$ est une implication,

P: sa **permisse** ou **antécédent**

Q: sa **conclusion** ou **conséquence**

ENONCÉ ATOMIQUE

C'est une **proposition élémentaire**, constitué d'un seul symbole propositionnel **ne contenant pas de connecteurs logique**.

- Exemples

p: représente la proposition : «Driss est père de Ali»

LITTÉRAUX (ATOME OU SA NÉGATION)

- Un littéral **positif est un atome**
- Un littéral **négatif est la négation d'un atome**
- Exemples
- Littéraux positifs : p, q
- Littéraux négatifs: ¬p, ¬q

ENONCÉE COMPLEXE (FORMULE):

Construite à partir **de symboles propositionnel et de connecteurs**

Exemple: $(P \wedge Q) \rightarrow \neg R$

Si Rachid est malade, il ne sort pas $P \rightarrow \neg Q$

AMBIGUÏTÉ SYNTAXIQUE :

Pour assurer l'absence d'ambiguïté syntaxique, toute énoncée complexe doit être encadré par des parenthèses

Exemple: $(A \wedge B) \rightarrow C$ ou lieu de $A \wedge B \rightarrow C$

On peut utiliser les règles de priorité (par ordre décroissant) des connecteurs:

$\neg$, ^, V, $\rightarrow$, $\leftrightarrow$

Mais la priorité ne résout pas les ambigüités des énoncées tel que:

a → b → c

Qui peut se lire: (a → b) → c ou encore a →(b → c) qui ne signifie pas la même chose

Cepedant,

a ^ b ^ c

Qui peut se lire: (a ^ b) ^ c ou encore a ^ (b ^ c) est autorisée

Sémantique de LP

SÉMANTIQUE

Elle ***définit les règles*** de détermination de la valeur de vérité d'une Formule(***énoncé quelconque***) dans ***un modèle quelconque*** (c.à.d un énoncé est il vrai ou faux).

BASE DE CONNAISSANCES (KNOWLEDGE BASE):

C'est l'ensemble des énoncés (**supposées tous vrais**) à utiliser pour réaliser l'inférence(pour vérifier si un certain énoncé est vrai)

- **MODÈLES ASSOCIÉ À UNE KB (KNOWLEDGE BASE).**

Considérant par exemple les énoncées de KB qui utilisent les symboles p ,q et r.

un modèle possible est m1= {p=faux , q=faux, r= vraie }

- Avec 3 symboles il existe 2^3 modèles possibles.

Le calcul de la valeur de vérité d'un énoncé est réalisé selon la table de vérité suivante :

Sémantique de LP

P	Q	¬P	P∧Q	P∨Q	P→Q	P↔Q
F	F	V	F	F	V	V
F	V	V	F	V	V	F
V	F	F	F	V	F	F
V	V	F	V	V	V	V

Exemple: Dans le modèle **m1= {p=faux , q=faux, r= vraie }**
Quel est la valeur de vérité de: **(p v q) ^ (¬ r)**

BASE DE CONNAISSANCE (KB)

C'est l'ensemble des énoncés (**supposées tous vrais**) à utiliser pour réaliser l'inférence(pour vérifier si un certain énoncé est vrai)

Cette base est représentée aussi par la conjonction de ces énoncés.

En effet : si la KB se compose des énoncés E1, ……,EN on peut les considérer comme un seule énoncée :

E1^ E2 ^…… ^EN on affirme que tous les énoncés individuels sont vrais.

Sémantique de LP

INFÉRENCE : (DÉCIDER LA BONNE CONSÉQUENCE)

- Le but est de décider si la conjonction de la KB a pour conséquence Une énoncée α : $KB \models \alpha$

Cela signifie que : Si KB vrai alors α ? Vraie (ie α est –elle conséquence ?)

L'opérateur $\models$ est dit de conséquence.

Algorithme d'inférence par énumération de modèles

Cet algorithme permet de :

- Enumérer les modèles
- Et vérifier si α est vraie dans tous les modèles pour lequel KB est vrai.

ENUMÉRATION DE MODÈLES :

Revient à affecter des Vrai et Faux pour tout symbole propositionnel.

- Soit par exemple KB est :

R1 ¬ P (vraie)

R2 P→Q (vraie)

On a: 2 symboles P et Q

Il existe 2^2=4 modèles possibles.

m1 = {P=V, Q=V}

m2= {P=V, Q=F}

m3= {P=F, Q=V}

m4= {P=F, Q=F}

Quel sont les modèles pour lequel R1 ^R2 est vrai (c.à.d) R1 vrai et R2 vrai.

DRESSER LA TABLE DE VÉRITÉ :

P	Q	R_1	R_2	KB ($R_1 \wedge R_2$)
F	F	V	V	V
F	V	V	V	V
V	F	F	F	F
V	V	F	V	F

Les modèles pour lesquels KB est vraie sont :

m3 = {P=F, Q=V}

m4 = {P=F, Q=F}

Si on veut par exemple décider si pour KB= { R1 et R2 }

α =Q est elle conséquence (Q est elle vraie)

On remarque : α n'est pas toujours vrai dans m3 et m4 alors α = Q n'est pas conséquence. Cependant, l'énoncée $\alpha = (\neg P) \wedge (\neg P)$

c'est toujours vraie dans m3 et m4 alors α est conséquence de KB

càd $KB \models \alpha$

Exercice1

Soit la base de connaissance suivante:

KB={

E1: $p \wedge q$

E2: $\neg p \rightarrow q$

}

Vérifier si les énoncée suivant sont vrai?

$((\neg p) \vee (\neg q)) \rightarrow p$

$(p \vee q) \wedge (\neg p)$

Exercice2

Soit la base de connaissance suivante:

KB={

E1: $p \wedge q$

E2: $\neg p \rightarrow r$

E3:$(q \vee r) \rightarrow p$

}

Vérifier si les énoncée suivant sont vrai?

$((\neg p) \vee (\neg q)) \rightarrow r$

$(p \vee q) \wedge (\neg r)$

REPRÉSENTATION DE CONNAISSANCES EN LOGIQUE DES PRÉDICATS DU PREMIER ORDRE

La logique des propositions ne permet pas de formaliser des énoncés généraux: comme suit:

x est père de y et x est père de z alors y et z sont frères (ou x, y et z sont des variables).

Ceci est réalisé par la logique des prédicats qui est indépendante des ***noms propres*** sous forme:

$$\forall(x, y, z)\ (père(x, y) \wedge père(x, z)) \rightarrow frère(y, z)$$

Où père et frère sont maintenant **des prédicats** (relations entre les différents objets)

PRÉDICAT :

$$P : D_1 \times D_2 \times ..... \times D_n \rightarrow \{vrai, faux\}$$
$$(x_1, x_2, ..., x_n) \rightarrow P(x_1, x_2, ...x_n)$$

Exemple: père(X,Y)

- Le prédicat P appliqué à un certains objet constants C1 , C2 , ……Cn : P(C1 ,C2 ,…Cn) devient une proposition.

Exemple : Père (Ali ,Youssef)

Ensemble de départ de dim 1:

Soit: Ali est un homme: peut être représenté par: H(Ali)

X est un homme: H(X) ou Homme(X)

REMARQUES:

Un prédicat peut être appliqué à des variables et /ou des constantes

- Le nombre de paramètres d'un prédicat est appellé: '***arité***'
- Un prédicat d'arité 0 est une proposition.
- **Les connecteurs** de la LProposition $(\wedge, \vee, \neg, \rightarrow, \leftrightarrow)$

sont aussi ceux de LPrédicats

- Les **quantificateurs** $(\forall)(\exists)$: sont les éléments de L Prédicats

SYMBOLES ET INTERPRÉTATIONS :

Quatre types de symboles :

- Les symboles de constants (Ali, Mohammed,….)
- Les symboles de variables (**X**, **Y**,…)
- Les symboles de prédicats : représentent les relations(liens) entre les objets (***décrit les propriétés des objets***)

Exemples: frère, père , mère….

- Les symboles de ***fonctions*** (chaque objet est caractérisé par un ensemble de fonctions

Exemple: voiture(x)

ATOME : Prédicat appliqué à des termes: p(t1 , …….,tm)

FORMULE BIEN FORMÉE (F.B.F)

C'est une expression formé d'atomes auquel sont appliqués les connecteurs(et/ou quantificateurs):

$$\wedge, \vee, \neg, \rightarrow, \leftrightarrow, (\forall), (\exists)$$

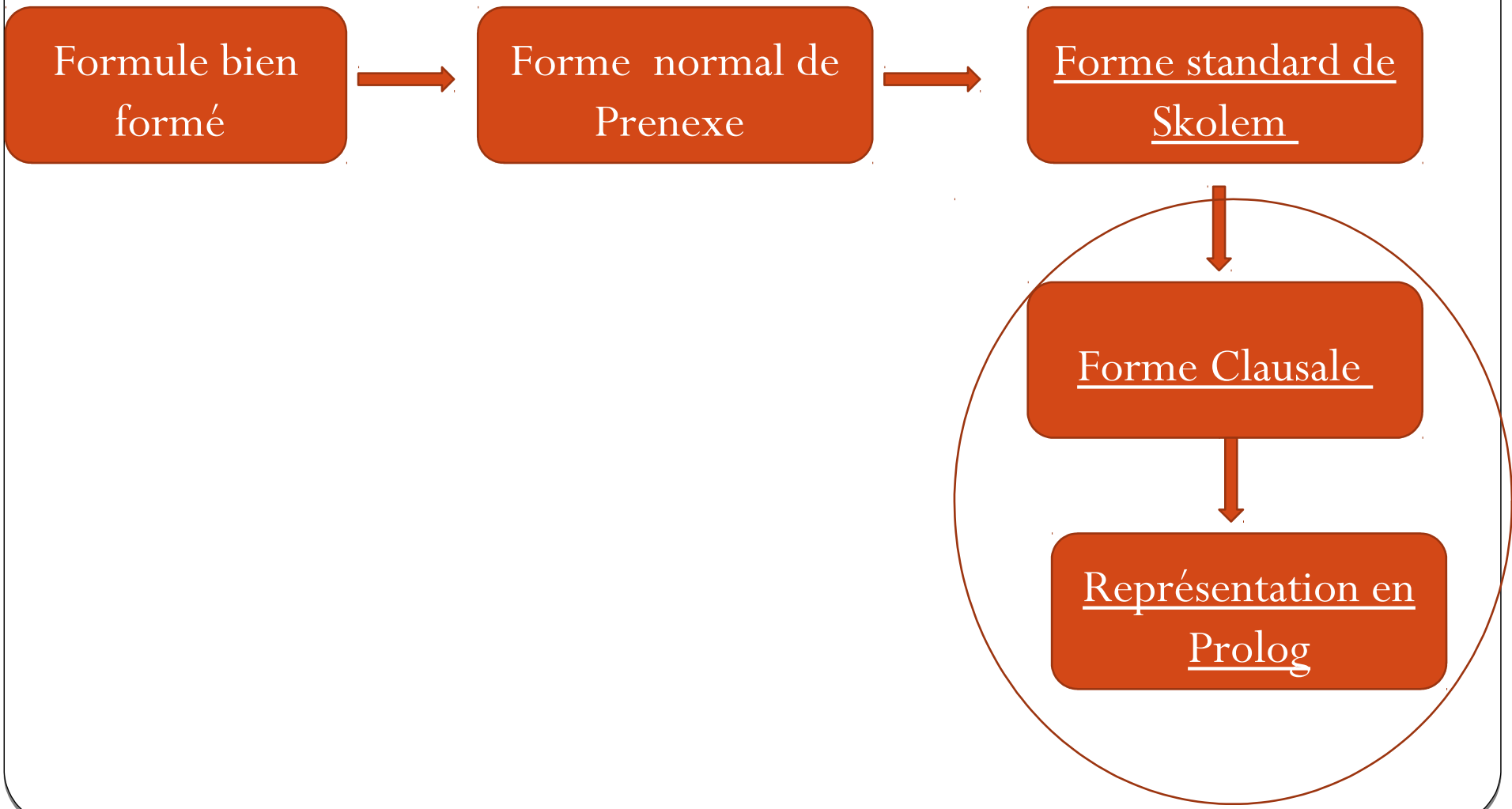
Exemple:

$$F = ((\ (\forall x)\ P(x)\ ) \wedge ((\exists y) Q(y))) \rightarrow ((\exists x)\, R(x))$$

Tous les philosophes sont assis $F = (\forall x)\ (P(x) \rightarrow A(x))$

Quelques philosophes sont assis $F = (\exists x)\ (P(x) \wedge A(x))$

Objectif:


Formule bien formé
Forme normal de Prenexe
Forme standard de Skolem
Forme Clausale
Représentation en Prolog

FORME NORMALE DE PRENEXE :

Une f.b.f nommée F est normal de Prenexe ssi :

$$F = (Q_1 x_1)(Q_2 x_2)....(Q_n x_n)M$$

Où Q_i i=1,2 sont des quantificateurs $\forall$ *ou* $\exists$

M est une formule sans quantificateurs

Exemple : $F = (\forall x)(\exists y)(\exists z)(f(x,y) \rightarrow Q(y,z))$

Remarque : **Toute f.b.f admet une forme normale de Prenexe**

EQUIVALENCES DE BASE UTILISÉES POUR ATTEINDRE LA FORME CLAUSALE:

$$\neg((\forall x)\, p(x)) \equiv (\exists x)\neg p(x)$$

$$\neg((\exists x)\, p(x)) \equiv (\forall x)\neg p(x)$$

- Soit M et N deux f.b.f sans quantificateurs

$$((\forall x)M) \wedge ((\forall x)N) \equiv (\forall x)(M \wedge N)$$
$$((\exists x)M) \vee ((\exists x)N) \equiv (\exists x)(M \vee N)$$
$$((\forall x)M) \vee ((\forall x)N) \rightarrow (\forall x)(M \vee N) \equiv$$
$$((\exists x)(M \wedge N) \rightarrow ((\exists x)M) \wedge (\exists x)N)$$

Soient Q_1 et Q_2 deux quantificateurs qcq

$$((Q_1 x)\ M(x)) \wedge ((Q_2 y)\ N(y)) \equiv (Q_1 x)(Q_2 y)(M(x) \wedge N(y))$$
$$((Q_1 x)\ M(x)) \vee ((Q_2 y)\ N(y)) \equiv (Q_1 x)(Q_2 y)(M(x) \vee N(y))$$

Forme standard de Skolem (la Skolemisation)

Consiste à supprimer les quantificateurs existentiel ($\exists$)

- Soit F une f.b.f, Pour mettre F sous forme standard de SKolem on réalise les étapes suivantes :
- Mettre F sous forme normale de Prenexe:

$F=(Q_1X_1)(Q_2 X_2)\ldots\ldots(Q_N X_N)M$

Pour tous quantifieurs: $Q_i = \exists$

- S'il n'y a aucune $\forall$ à gauche de Q_i alors :

Supprimer (Qi xi) et remplacer xi dans M par une constante non déjà existante.

EXEMPLE :

$$F \equiv (\exists y)(\forall x)\, P(x) \wedge Q(y)$$
$$\equiv (\forall x)\, P(x) \wedge Q(a)$$

- Si Qj , Qj+1 ,……., Qe sont des quantificateurs $\forall$ à gauche de Qi ,alors :

  Supprimer (Qi xi) et remplacer xi dans M par une fonction f de xj , xj+1,… xe : (f(xj , xj+1,… xe)

Exemple :

$$F \equiv (\forall x)(\exists y)\, P(x) \wedge Q(y)$$
$$\equiv (\forall x)\, P(x) \wedge Q(f(x))$$

- Les constantes et fonctions introduits dans M sont appellées fonctions de Skolem.

Exemple 2

$$\forall x \, \exists y \, (\text{Etudiant}(x) \rightarrow (\text{Fac}(y) \wedge \text{Inscrit}(x, y)))$$

Peut être écrite sous forme skolem suivante:

$$\forall x (\text{Etudiant}(x) \rightarrow (\text{Fac}(f(x)) \wedge \text{Inscrit}(x, f(x))))$$

Remarque : une f.b.f peut avoir plusieurs formes de Skolem différents

FORME CLAUSALE OU ENSEMBLE DE CLAUSES :

Pour obtenir une Clause d'une f.b.f, , il suffit de la mettre sous forme standard de Skolem

$(\forall x1)(\forall x2) \ldots\ldots(\forall xn)\ F1$

F1: est dite une clause

Une forme clausale est une conjonction de clauses Noté aussi:

$F1 \wedge F2 \wedge \ldots\ldots \wedge Fk$

Ou:

$\{ F_1, F_2, \ldots\ldots, F_k \}$

C'est ce qu'on appelle en prolog ***la base des faits***

En logique des prédicats: la base de connaissances est constituée d'une base de faits= ensemble de clauses + base de règles (**utilisée pour déduire un but**)

Sémantique de la logique des prédicats:

Utilisation des moteurs d'inférences (chainage avant et arrière) pour la déduction d'un but.

MOTEUR D'INFÉRENCE :

- LES MI À CHAINAGE AVANT (FORWARD CHAINING)

Détermine le résultat ou (But) à partir de (BDF : base de fait)+ (la base de règles)

Exemple :

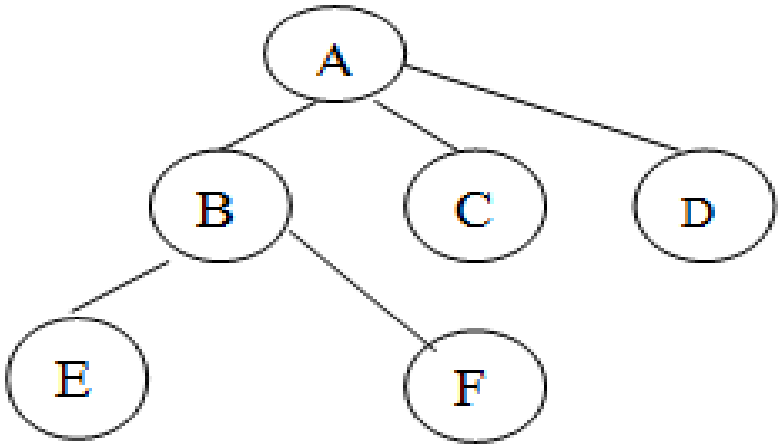
Soit BDF= {A, B, C, D}

Et soit la règle A, C→G

On obtient une nouvelle BDF= {A, B ,C, D ,G}

TYPES D'EXPLORATION

Exploration en profondeur d'abord et l'exploration en largeur d'abord (Aveugle)



Résultats d'exploration eu profondeur d'abord A, B, E, F, C, D….
Par contre l'exploration en largeur d'abord → A B C D E F

EXEMPLE : M.I À CHAINAGE AVANT EN PROFONDEUR D'ABORD :

Soit la base de connaissance suivant BDF= {A, D, E,G}

Et soit la base de règles suivante :

BDR= {

R1 : A,B,C → H

R2: A, U, C → F

R3: E, G, B → S

R_4: D, G → C

R_5: A, E → B

R_6: U, S, T → F

R_7: G, H → R

R_8: D, E → T

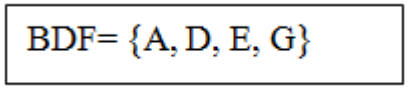
R_9: R, S, H → F

R_{10}: A, U → B

}

- Le problème est d'établir le fait F
- Le problème du MI peut être schématisé par l'ordre suivant qui illustre l'exploration ***en profondeur d'abord*** de l'arbre de recherche

Racine de l'arbre

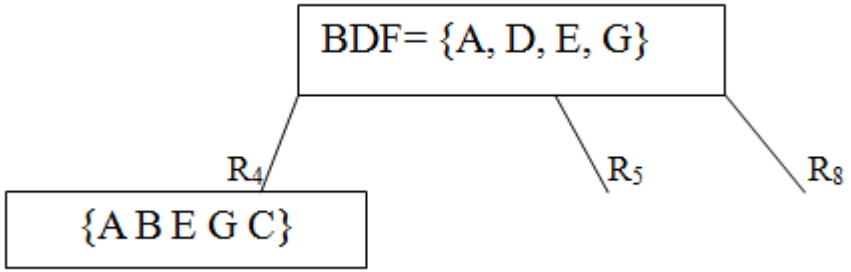


A partir de cette base , on peut aboutir directement , Selon

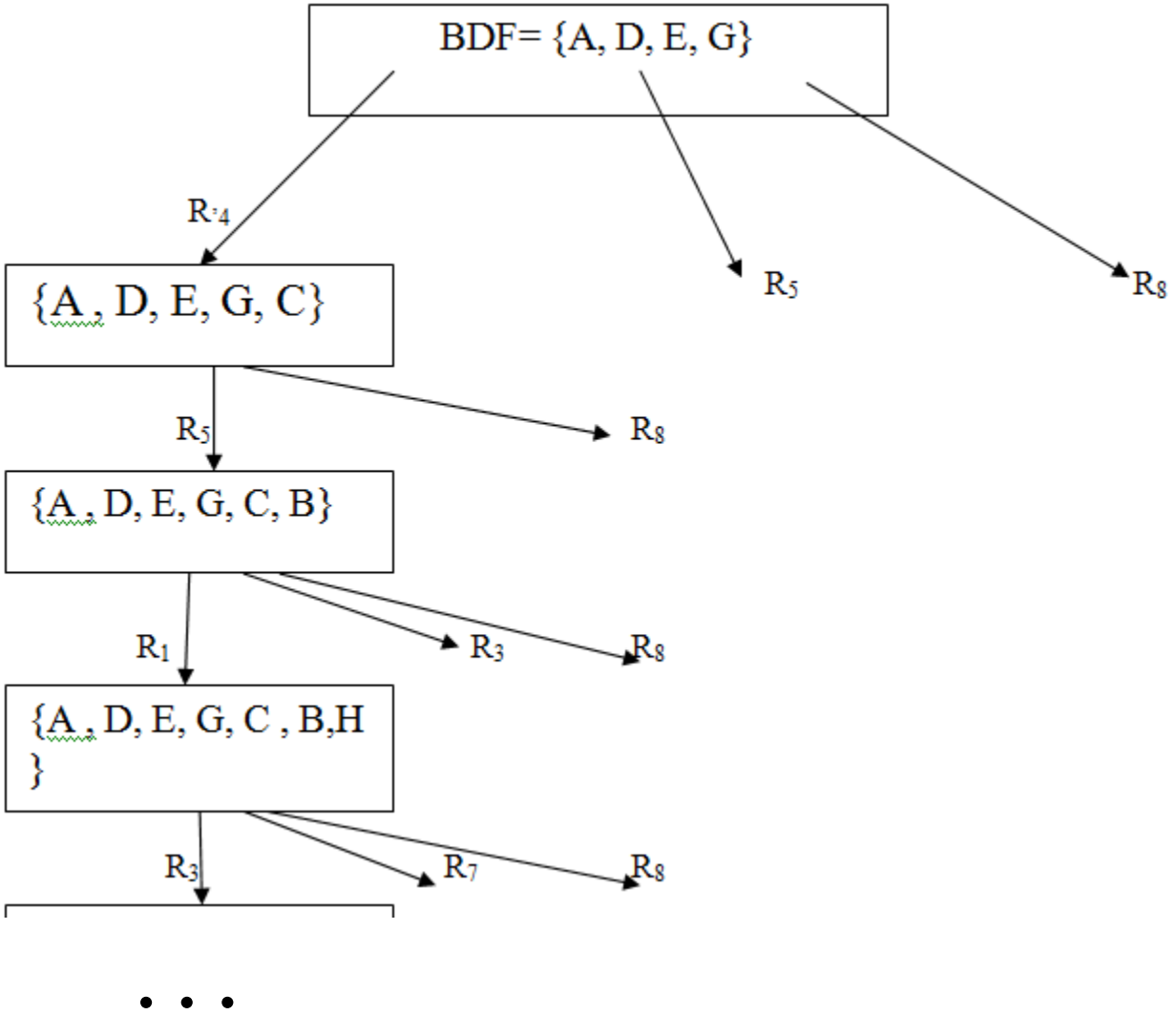
$R_4 \rightarrow C$

$R_5 \rightarrow B$

$R_8 \rightarrow T$

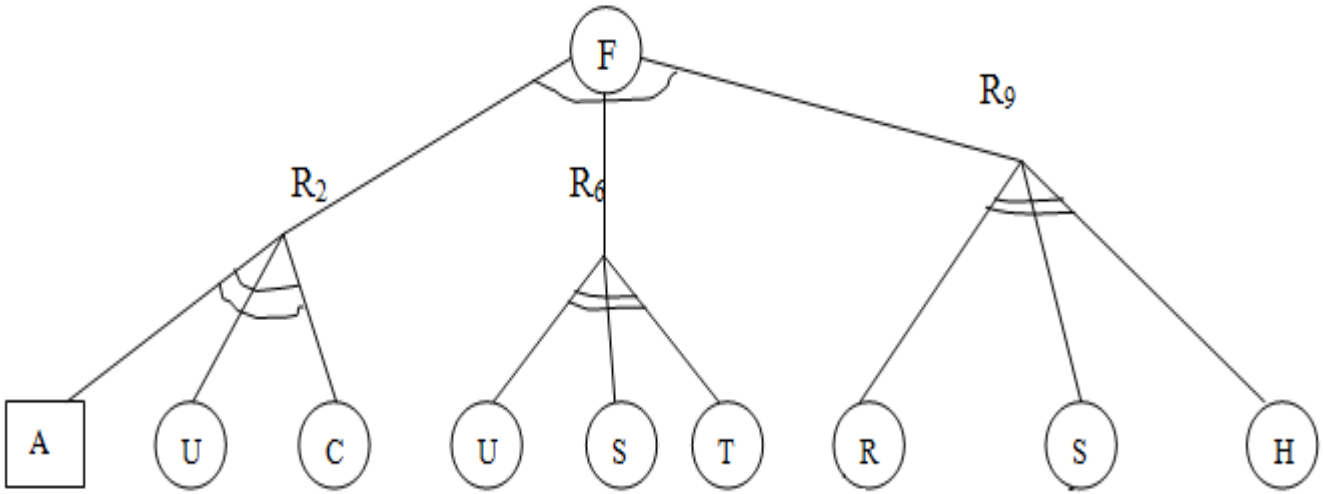


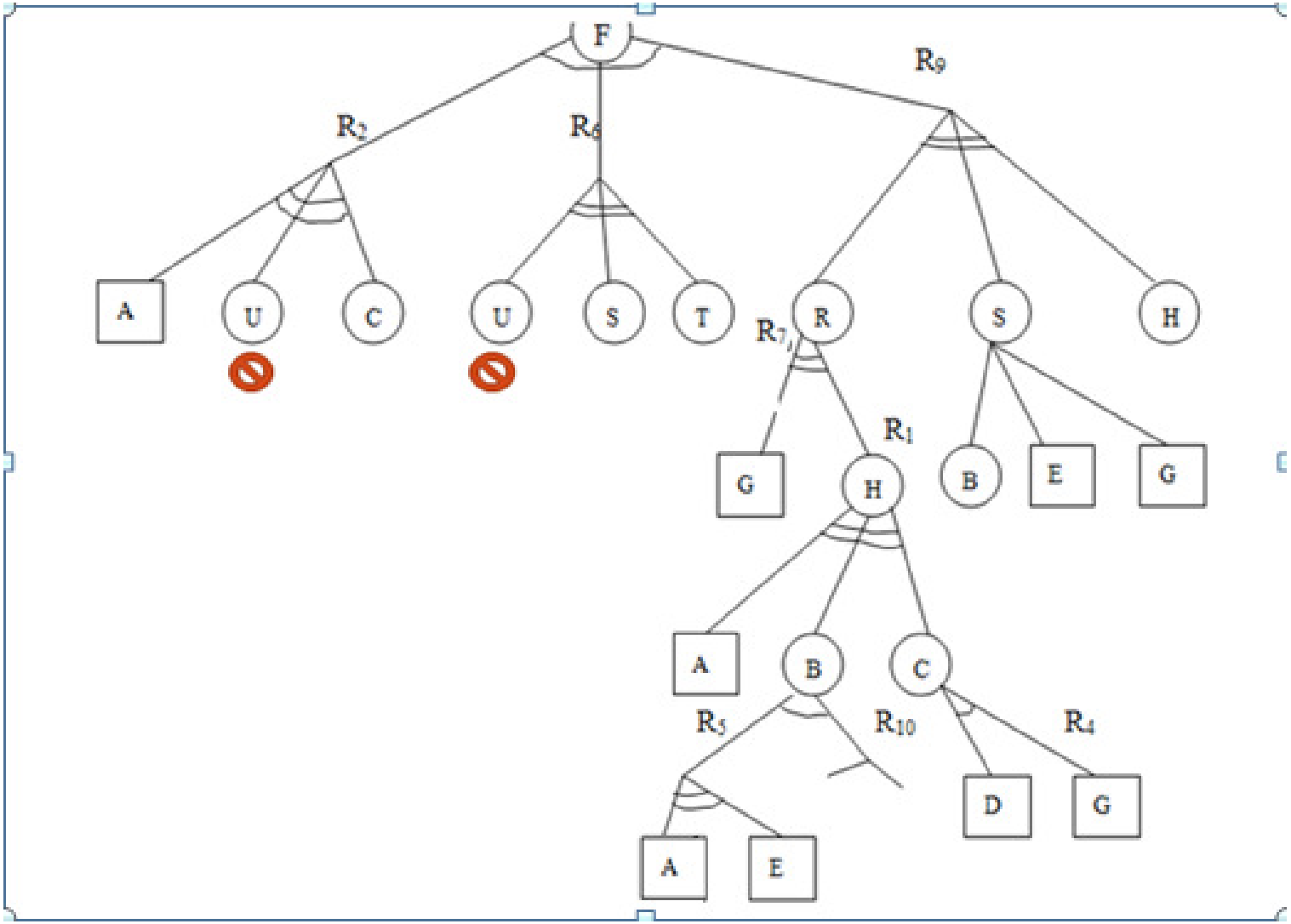
LA RÈGLE R_4 EST EXPLORÉ (ELLE NE SERA PLUS APPLIQUÉE)



Moteur d'inférence à chainage arrière-en profondeur d'abord-

Correspond ou chemin inverse effectué par MI à chainage avant, on commence du fait à établir et regarde les autres faits qui le produit (se sont les nouveaux problèmes à résoudre)




F
R9
R3
R6
A
U
C
U
S
T
R7
R
S
H
R1
G
H
B
E
G
A
B
C
R5
R10
R4
D
G
A
E

Exercice

Soit la base des faits suivante : $BF = \{B, C\}$

Et la base de règles suivante :

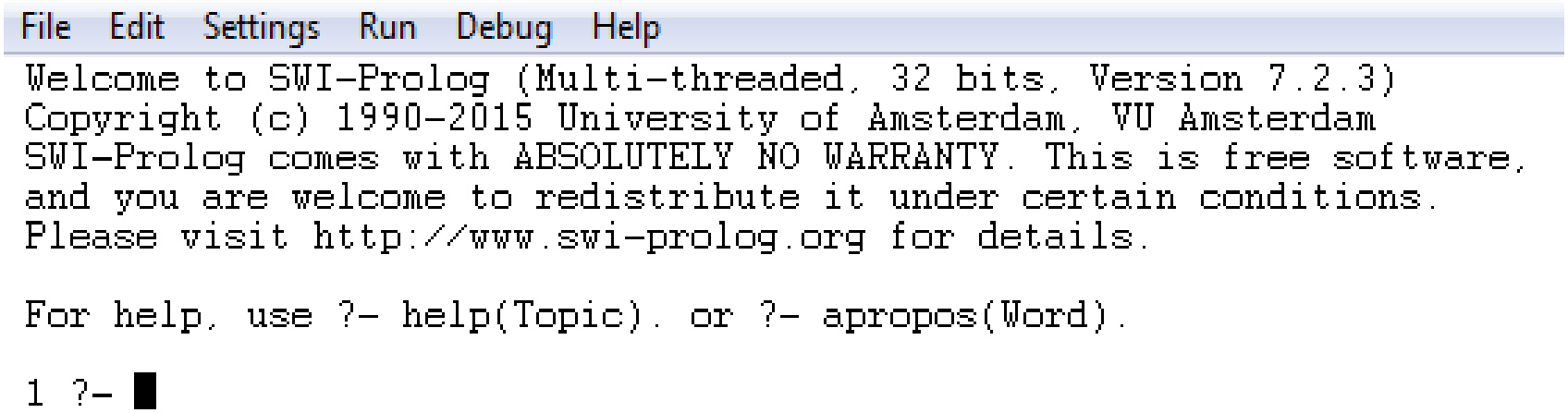
1) $B, D, E \rightarrow F$
2) $G, D \rightarrow A$
3) $C, F \rightarrow A$
4) $B \rightarrow X$
5) $D \rightarrow E$
6) $X, A \rightarrow H$
7) $C \rightarrow D$
8) $X, C \rightarrow A$
9) $X, B \rightarrow D$

Déduire le fait ***H*** en utilisant les moteurs d'inférence **chainage avant et chainage arrière** par exploration en profondeur d'abord.

Langage Prolog

- Un langage de programmation logique (déclarative)
- Comporte: **un compilateur** et **une bibliothèque**

- Si Prolog démarre normalement, vous verrez apparaître ce qui suit:



Le symbole **?-** est appelé l'*invite de commande (ou encore prompt). Elle indique* que Prolog attend une requête

Exemple:

```
3 ?- write("Hello").
Hello
true.

4 ?-
```

On met toujours un point en fin de requête.

Le mot **true** imprimé ensuite par Prolog indique que la commande a réussi, du point de vue logique.

Constitution d'un Programme Prolog

- Programme Prolog=Base de connaissance={faits+Règles}

- Les faits: se sont des clauses **supposées toujours vrais**
- Les règles: sont des relations qui permettent d'établir de nouveau faits.

  Exemple: si on est le père du père de quelqu'un alors on est son grand-père se traduit par :

pere(X,Z) ^ pere(Z,Y) →grandpere(X,Y) se traduit en prolog par:

grandpere(X,Y):-pere(X,Z),pere(Z,Y).

grandpere(X,Y):-pere(X,Z),pere(Z,Y).

Est interprété comme:

$\forall X \forall Y \forall Z$ pere(X,Z)^ pere(Z,Y) $\rightarrow$grandpere(X,Y)

Conventions de SWI-PROLOG

- Il existe plusieurs éditeur/débuggeur PROLOG. Nous utiliserons ici SWI-PROLOG.
- Tous les faits et règles doivent se terminer par un « . ».

- Les variables utilisées dans les faits ou les règles commencent toujours par une majuscule ou « _ »

  Exemple : X Genre _nombre _

- A l'inverse, tout le reste commence par une minuscule

- Il ne faut pas mettre d'espace entre le prédicat et la parenthèse ouvrante qui l'accompagne
- Pas d'accent dans le programme

- Les commentaires sont mis entre « /* » et « */ » ou commencent par « % » et se terminent sur la même ligne.

Exemple de programmation d'une base de connaissance

- Soit le prédicat nommé **famille** :
- **famille(enfant, sexe, pere).**
- Considérant une table dont les attributs sont le ***nom de l'enfant***, ***son sexe*** et le nom de son ***père***.

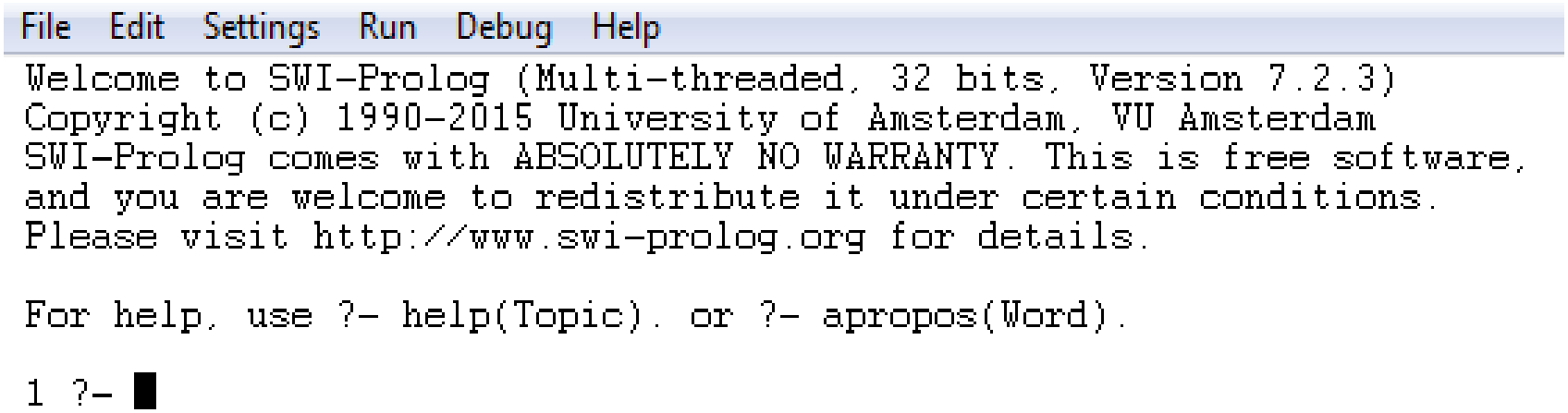
Considérant la table des faits suivants:

Table des faits (ou base de faits)

- famille(mohammed, h, driss).
- famille(sanae, f, ahmed).
- famille(nagib, h, rachid).
- famille(rajae, f, said).
- famille(mourad, h,driss).
- famille(jamal, h, ismail).
- famille(meriem, f, ahmed).
- famille(jaouad, h,samir).
- famille(driss, h,jaber).
- famille(ahmed, h,jaber).

Lancer SWI-PROLOG

- Si Prolog démarre normalement, vous verrez apparaître ce qui suit:



Le symbole **?-** est appelé l'*invite de commande (ou encore prompt). Elle indique* que Prolog attend une requête

- Créer un nouveau fichier (File/new…
- Puis saisir la base des faits précédente
- Et enregistrer le fichier sous l'extention .pl
- Fermer votre fichier

Remarque :

- On peut travailler sans règles (comme une base de donnée) pour la recherche ou la vérification de la présence d'une donnée).
- Avec les règles on déduit d'autres faits vrais, C'est ce qui fait la puissance d'un langage comme PROLOG par rapport à SQL où toutes les données de la base doivent être explicitement énoncées.

Revenir à swi prolog puis Compiler votre fichier:
(File/consulter….)

répondre aux questions suivantes:

1) Vérifier la présence ou non d'une donnée dans la table (sans utiliser les règles)

- Najib, homme et son père est rachid
- Rajae femme et son père est ahmed
- Quelle sont les femmes qui figurent comme enfants
- Quelle sont les enfants de driss

```
3 ?- famille(_,h,driss).
true
```

- Le système répond par ***true*** s'il trouve une façon de prouver le nouveau fait a partir des faits et des relations donnes dans le programme.

- L'utilisation de _ dans le fait signifie que sa valeur ne nous importe pas

- Ajouter maintenant les règles suivantes:
- /****************** Les regles **********************/
- /*R1*/ pere(X,Y):-famille(Y,_,X).
- /*R2*/ frere(X,Y):-famille(X,_,Z),famille(Y,_,Z).
- /*R3*/ oncle(X,Y):-pere(Z,Y),frere(X,Z).

répondre aux questions suivantes:

- Mohammed et mourad sont ils des frères
- Ahmed est il oncle de mourad
- Ahmed est il oncle de najib

Exercice

Soit la base de connaissance suivante (faits+règles):

```
 /****************** Les faits*********************/
/*Arguemnts du predicat 'gestion' :
      gestion(etudiant, sexe, annee_bac, moy)*/
 gestion (mohammed, h, 2008, 12).
gestion (sanae, f, 2009, 10.5).
gestion (nagib, h, 2007, 13).
gestion (rajae, f, 2009, 12.5).
gestion (mourad, h, 2008, 11).
gestion (ahmed, h, 2009, 10).
gestion (meriem, f, 2010, 14).
gestion (rachid, h, 2009, 11).
 /****************** Les règles *********************/
/*R1*/  mention(X,Y):- gestion (X,_,_,Y),Y>=12.
```

Vérifier qu'un étudiant figure ou non dans la table

Afficher la liste de tous les étudiants (leurs noms)

Afficher les noms des étudiants ayant la moyenne supérieure à 10

Afficher les noms des étudiants ayant la moyenne entre 12 et 14

Afficher les noms des étudiants ayant la moyenne supérieure à 13 ou inférieure à 11

Afficher uniquement les noms des étudiants ayant une mention

Afficher les noms et les moyens des étudiants ayant une

Unification et affectation.

Le symbole « = » en PROLOG signifie l'unification et pas l'affectation.

Pour affecter une valeur numérique à une variable, en évaluant mathématiquement le résultat, il faut utiliser « is »

Exemple:

?- M=1+1, N is 1+1.

M = 1+1

N = 2 ;

write(),

Il existe quelques actions prédéfinies comme write(), qui est applicable à une constante, une variable ou une chaîne de caractères entre guillemets simples, nl (passage à la ligne)

Exemple:

?- write('bonjour'),nl,write('bonsoir').

bonjour

bonsoir

true.

?- X is 2,write(X).

2

X = 2.

En PROLOG, on ne définit pas de fonction ou de procédures mais des prédicats

- calculer la somme de deux nombres ?

```
somme(X,Y,S) :- S is X+Y.
```

Requête:

```
?- somme(2,4,X).
X = 6.
```

- connaître le plus grand parmi 2 nombres ? parmi 3 ?

```
max2(X,Y,X) :- X>=Y.
 max2(X,Y,Y) :- X<Y.
max3(X,Y,Z,M) :- max2(X,Y,N),max2(N,Z,M).
```

Récursivité

- En PROLOG, il n'y a pas d'autres moyens de faire des boucles que d'utiliser la récursivité:

- Exemple:

Afficher N fois 'bonjour'. (Ou : ecrit(N) est vrai si le message 'bonjour' est écrit N fois.):

```
ecrit(N) :- N>0, write('bonjour'), nl, N1 is N-1,ecrit(N1).
```

Requête:

```
?- ecrit(4).
```

Récursivité

Afficher les nombres de 1 à N.

```
/* de N à 1 */
decroissant(0).
decroissant(N) :- N>0, write(N), nl, N1 is N-1,
   decroissant(N1).
/* de 1 à N */
 croissant(0).
croissant(N) :- N>0, N1 is N-1, croissant(N1), write(N), nl.
```

un nombre est pair?

- pair(0).
- pair(X) :- X>0, X2 is X-2, pair(X2).

- somme des N premiers entiers. (Ou : som(N,X) est vrai si X est la somme des entiers de 1 à N.):

som(0,0). som(N,X) :- N>0, N1 is N-1, som(N1,X1), X is N+X1.

factorielle d'un nombre. (Ou : fact(N,X) est vrai si X vaut N!.)

fact(0,1). fact(N,X) :- N>0, N1 is N-1, fact(N1,X1), X is N*X1.

fibo(N,X) est vrai si X est la valeur de la suite de *Fibonacci* au rang N.

fibo(1,1). fibo(2,1). fibo(N,X) :- N>2, U is N-1, V is N-2, fibo(U,U1), fibo(V,V1), X is U1+V1.

La logique floue

- La logique floue propose des modes de raisonnement approximatifs plutôt qu'exacts

- C'est principalement le mode de raisonnement utilisé dans la plupart des cas par **les humains**.

En 1965, Lotfi Zadeh, introduit la notion d'ensembles flous (Fuzzy sets)

La logique floue a été introduite en 1973,

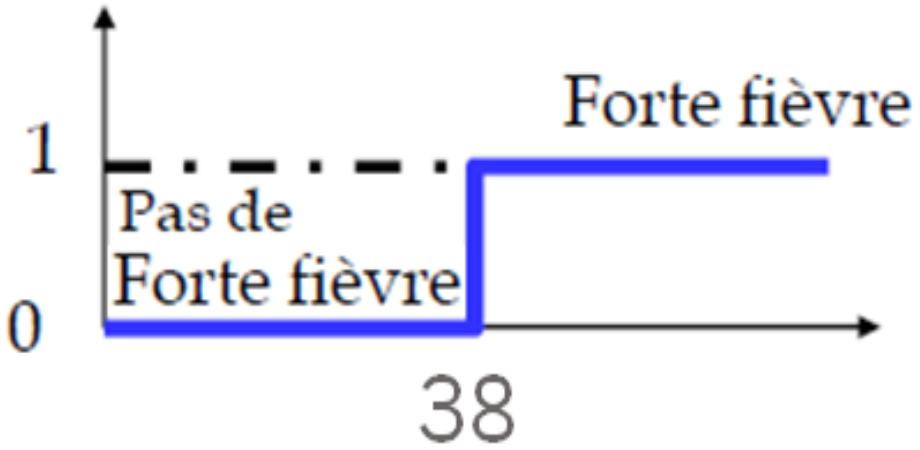
Elle est longtemps restée marginale.

L'idée de Zadeh consiste à utiliser le modèle de l'esprit humain

Approche Classique

Logique Propositionnelle

Les propositions ont des valeurs dans l'intervalle { Vrai, Faux}
ou {0,1}



Ensemble Classique

- La relation d'appartenance à un ensemble classique A est représentée par une fonction μ qui prend des valeurs de vérité dans la paire $\{0,1\}$. Ainsi,

la fonction d'appartenance d'un ensemble classique A est définie par:

$$\mu_A(x) = \begin{cases} 1 & si\ x \in A \\ 0 & si\ x \notin A \end{cases}$$

Inconvinients

- Les variables décrivant les états sont booléennes,
- Ne peuvent prendre que deux valeurs
- Sont donc mal adaptées à représenter la plupart des phénomènes courants

En effet:

Si la température est de 37,98 ou si la température est de 38,02 ??

- L'incertain et l'imprécis

Je crois que la température est élevée (***donc pas sure***)

Logique floue: Théorie des ensembles flous

Selon Zadeh, **la logique floue est la théorie des ensembles flous** (Zadeh, 1994).

La théorie des ensembles flous est une théorie mathématique dont l'objectif principal est la **manipulation des notions incertaines du langage naturel.**

Ainsi, elle évite les inadéquations de la théorie des ensembles classiques

Exemple:

l'ensemble *A représentant les* ***PC qui sont trop chers*** pour une population d'étudiants. Après une enquête menée au sein de cette population, un PC ayant un prix supérieur ou égal à 8000 DH sera déclaré ***trop cher***, *quand un prix inférieur ou égal à 5000* ***n'est pas trop cher***.

- Il existe un nombre important de PCs ayant un prix entre ces deux limites. Dans cet intervalle, on peut utiliser des valeurs, comprises strictement entre 0 et 1, pour classifier ces prix comme étant ***partiellement trop cher***.

- Cette classification permettra de définir:

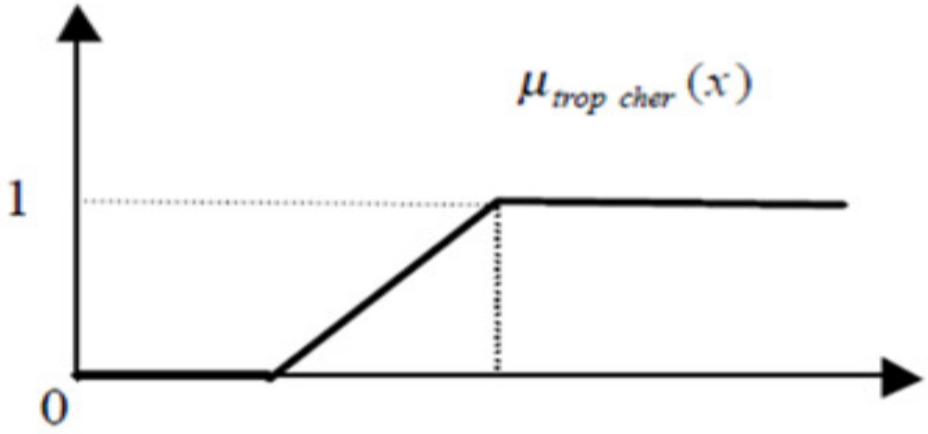
une nouvelle fonction d'appartenance, $\mu A(x)$,

associée à l'ensemble A représentant les PC trop chers.

- $\mu A(x)$ *est strictement* ***entre 0 et 1*** *implique que x appartient à A* avec un degré de vérité égal à $\mu A(x)$.

A est donc le sous ensemble flou *associé à la* ***valeur*** **linguistique** ***trop cher***

Chaque ensemble flou est caractérisé par sa fonction d'appartenance flou



Concepts fondamentaux

Si par exemple , $\mu A(x)=0.1$, *x appartient à l'ensemble flou A avec un degré d'appartenance de 10%*

⇔ Faible appartenance ⇔ Traduction de la valeur linguistique « Faible »

Si par exemple , $\mu A(x)=0.9$, *x appartient à l'ensemble flou A avec un degré d'appartenance de 90%*

⇔ Forte appartenance ⇔ Traduction de la valeur linguistique « Fort»

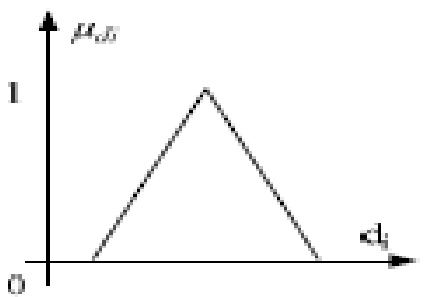
Un degré d'appartenance constitue une mesure d'incertitude

Un ensemble flou est totalement déterminé par sa fonction d'appartenance

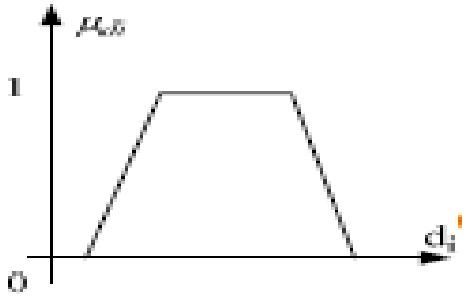
Caractéristiques de la fonction d'appartenance

La fonction d'appartenance décrivant un ensemble flou est caractérisée par 4 propriétés:

Type: forme qui peut être triangulaire, trapézoïdale, gaussienne,…



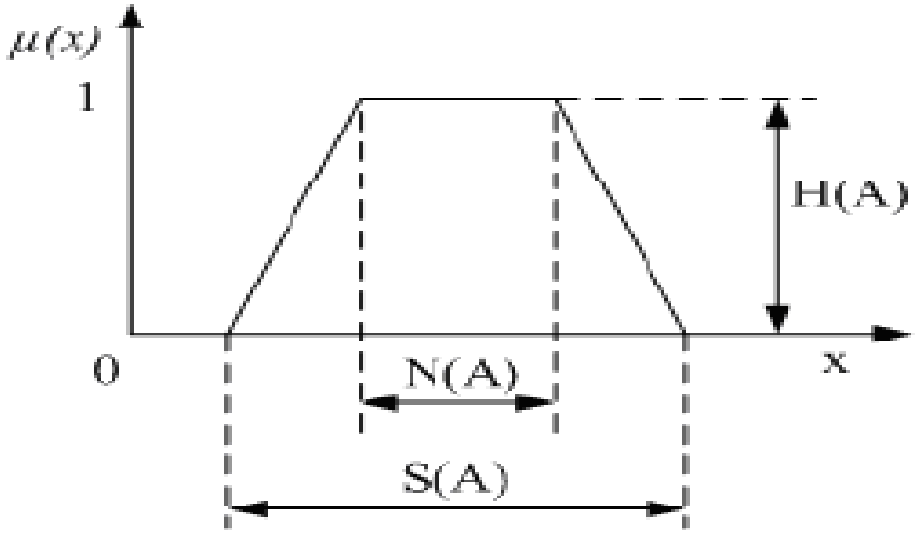
ou



La hauteur : $H(A) = Sup_{x \in X}(\mu_A(x))$ de la fonction d'appartenance. Un sous-ensemble flou est dit normalisé s'il est de hauteur 1.

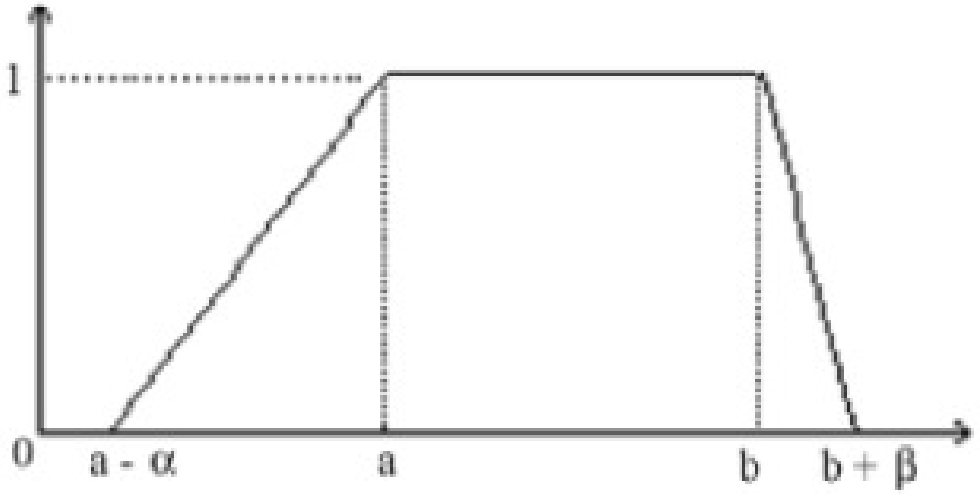
Le noyau : $N(A) = \{x / \mu_A(x) = 1\}$ est l'ensemble des éléments qui appartiennent totalement à A. Pour les fonctions de type triangulaire, le noyau est un singleton qui est appelé aussi valeur modale.

Le support : $S(A) = \{x / \mu A(x) \neq 0\}$; cet ensemble décrit l'ensemble des éléments qui sont partiellement dans A.



Un nombre flou trapézoïdale est notée généralement par: (a, b, α, β) :

$$\mu_A(x) = \begin{cases} 0 \text{ si } x < a - \alpha \text{ ou } b + \beta < x, \quad (x \text{ hors du support de } A) \\ 1 \text{ si } a < x < b, \quad (x \text{ dans le noyau de } A) \\ 1 + (x - a) / \alpha \text{ si } a - \alpha < x < a, \\ 1 - (b - x) / \beta \text{ si } b < x < b + \beta \end{cases}$$



Triangulaire (Très utilisée)

- Un nombre flou triangulaire est un cas particulier de trapézoïdale est notée par: (a, α, β)

- Déterminer l'expression de la fonction d'appartenance dans ce cas?

VARIABLES LINGUISTIQUES ET VALEURS LINGUISTIQUES

- Une variable linguistique est une variable dont les valeurs associées sont **linguistiques plutôt que numériques** (Zadeh, 1997).

Par exemple, la **variable linguistique *âge*** *peut être évaluée* par les **trois valeurs linguistiques** suivantes:

jeune, moyen et vieux

- La notion de variable linguistique suppose donc l'existence d'un **univers de discours *X*** *et d'un ensemble de valeurs* linguistiques D associé à la variable linguistique étudiée

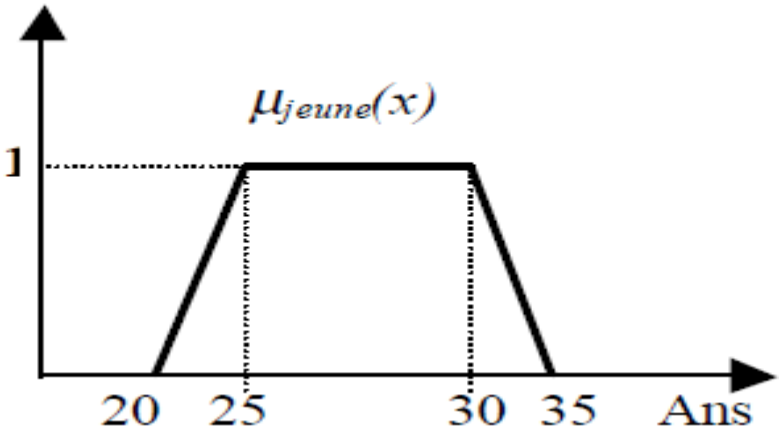
- Pour la variable linguistique *âge, X* est l'ensemble des nombres réels positifs IR+

Exemple: **[0 120]**

et D = {*jeune, moyen, vieux*}.

- En logique floue, **les valeurs linguistiques sont représentées par des sous ensembles flous**

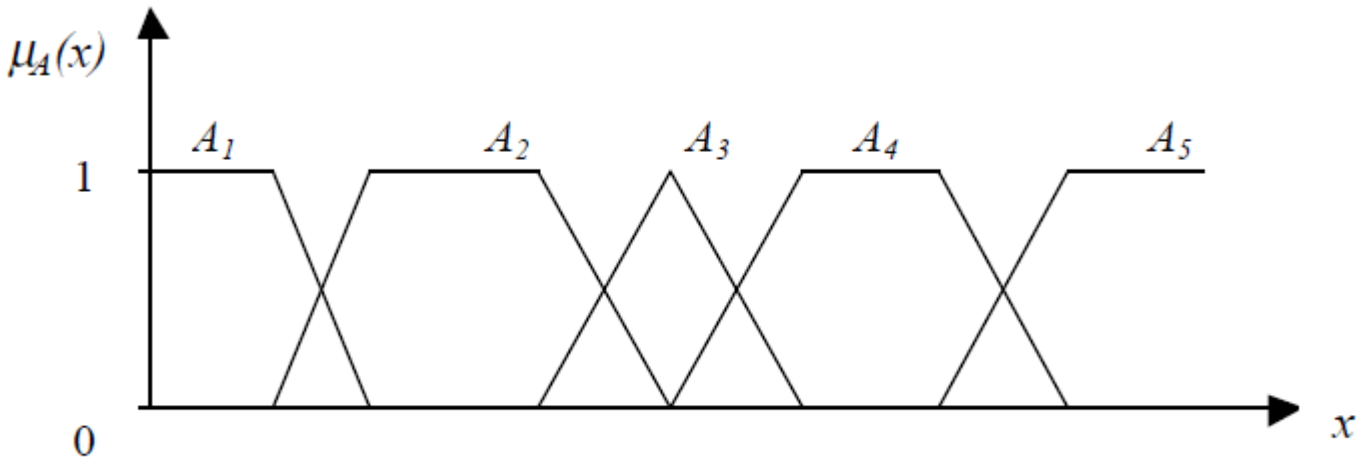
- Considérons, par exemple, le cas de la valeur linguistique ***jeune***; *elle* peut être représentée de la façon suivante:



Partition floue

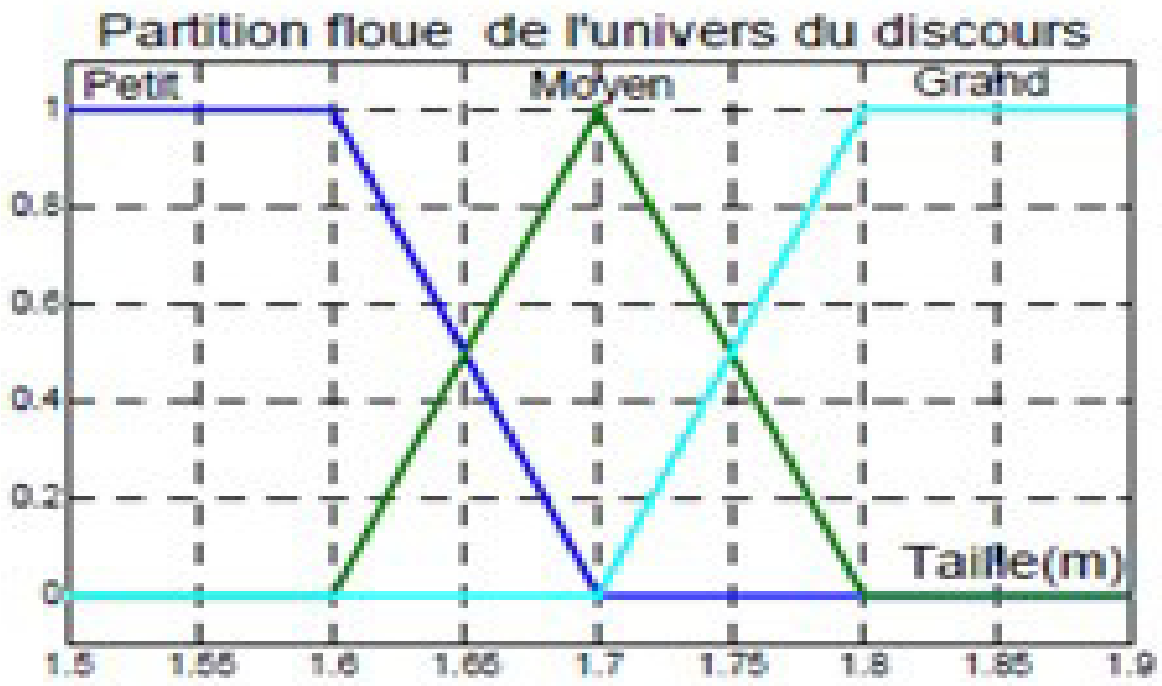
- Partition floue: Soient N ensembles flous *Aj du référentiel X*. *(A1, A2,.., Aj,., AN) est* dite une *partition floue si:*

$$\forall x \in X \quad \sum_{j=1}^{N} \mu_{A_j}(x) = 1 \quad avec \quad A_j \neq \phi \quad et \quad A_j \neq X \quad \forall \quad 1 \leq j \leq N$$



Exemple d'une partition floue formée de cinq ensembles flous.

Exemple, la variable linguistique *taille peut être évaluée* par les trois valeurs linguistiques suivantes: petit, moyen et grand (3 ensembles flous)



Compléter la partition floue, de l'age sur un univers de discours de votre choix et avec 5 valeurs linguistiques de votre choix

Les opérateurs flous

Extention des Opérateurs de la théorie des ensembles classiques : $=, \cup, \cap, \subset,$ et Complément.

Soit A et B deux sous ensembles flous de X définis par les fonctions d'apprtenance: μ_A et μ_B :

Égalité

$$A = B \text{ ssi } \forall x \in X, \mu_A(x) = \mu_B(x)$$

Inclusion

$$A \subset B \text{ ssi } \forall x \in X, \mu_A(x) < \mu_B(x)$$

Intersection

$$\forall x \in X, \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

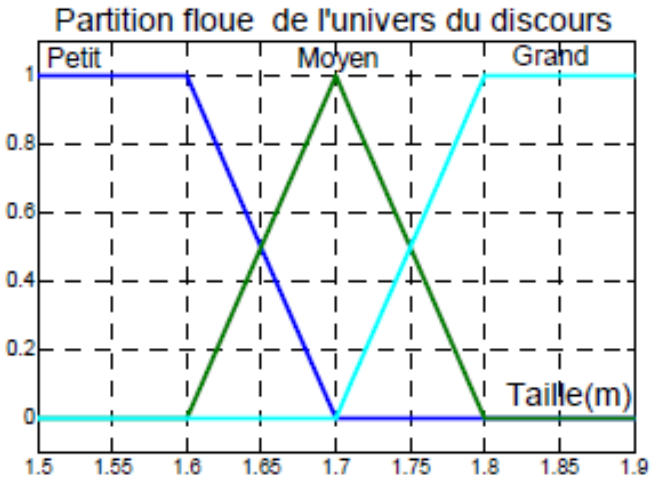
Union

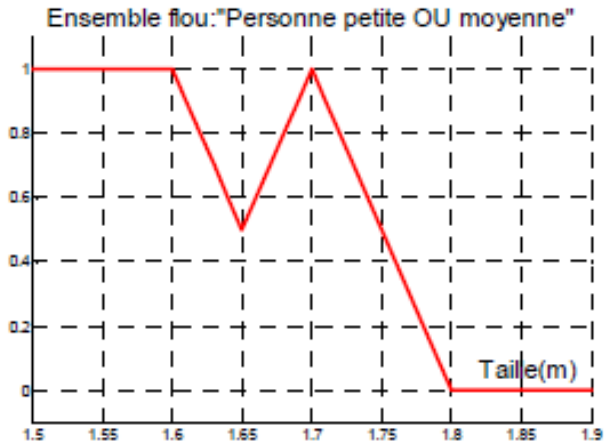
$$\forall x \in X, \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

Opérateur flou Union

L'ensemble des personnes petites OU moyennes est un ensemble flou de fonction d'appartenance :

$$\mu_{A\cup B}(x) = max\left(\mu_A(x), \mu_B(x)\right) \quad \forall x \in U$$

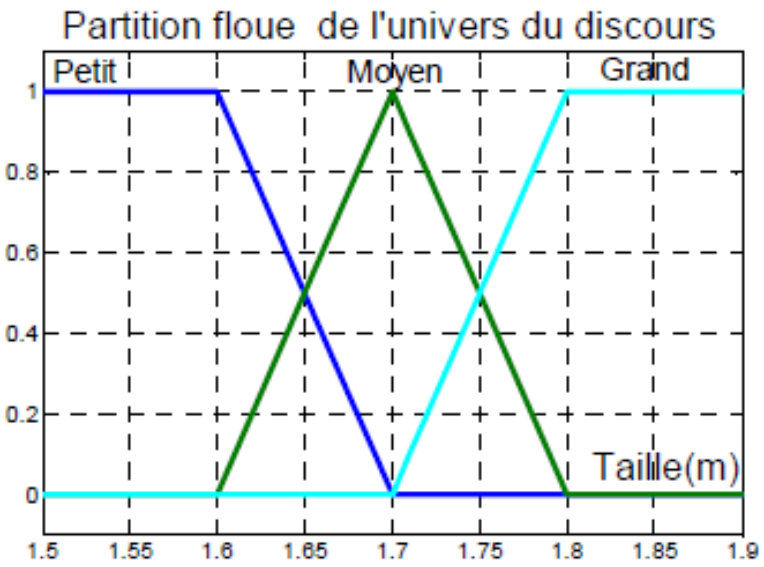


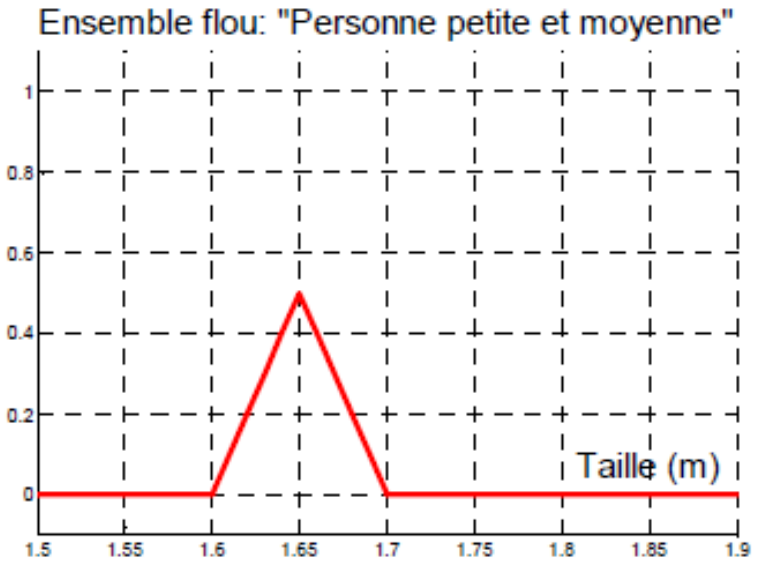


Opérateur flou Intersection

L'ensemble des personnes petites ET moyennes est un ensemble flou de fonction d'appartenance :

$$\mu_{A\cap B}(x) = min\left(\mu_A(x), \mu_B(x)\right) \quad \forall x \in U$$

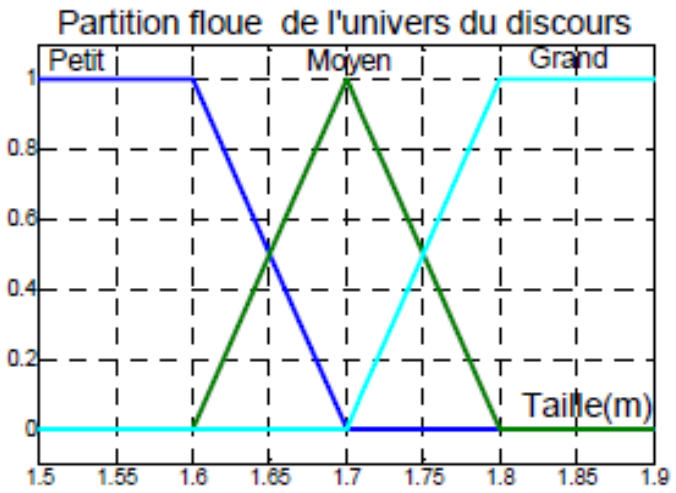


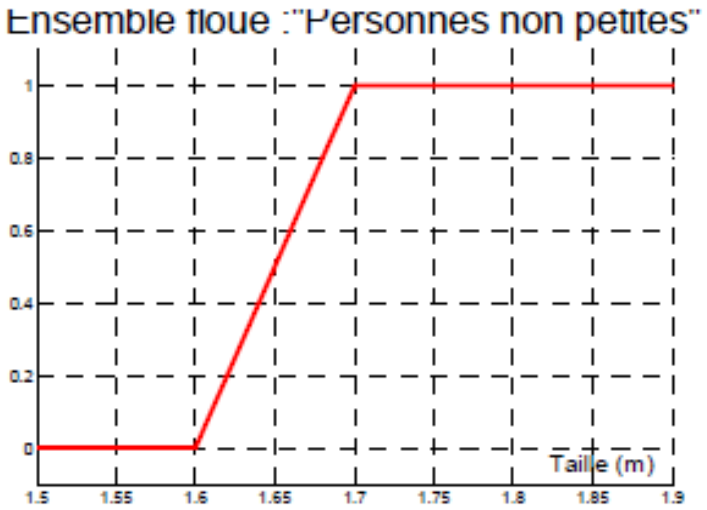


Complément

L'ensemble des personnes NON petites est un ensemble flou de fonction d'appartenance :

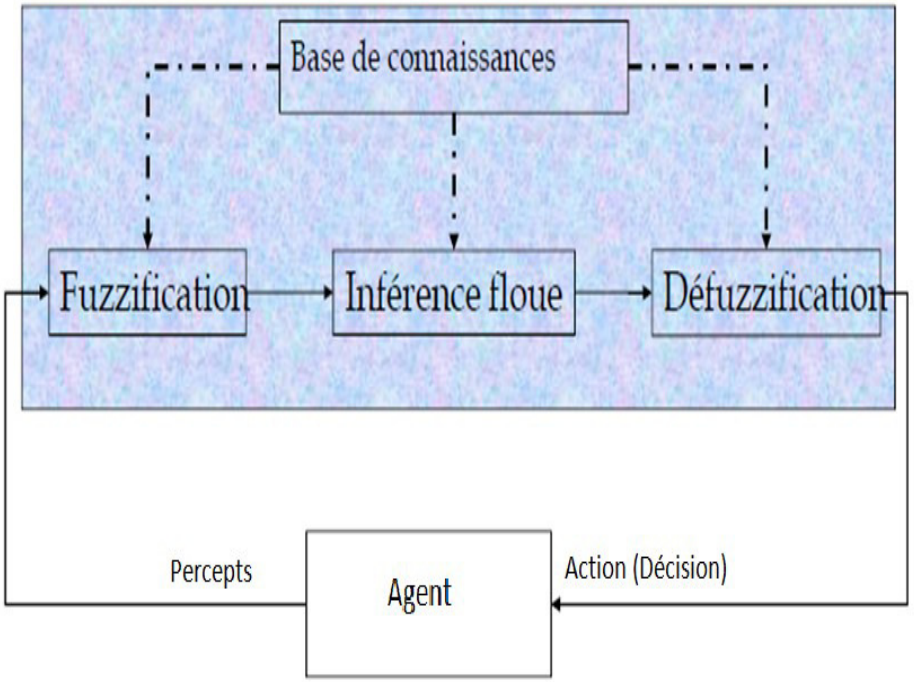
$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \quad \forall x \in U$$

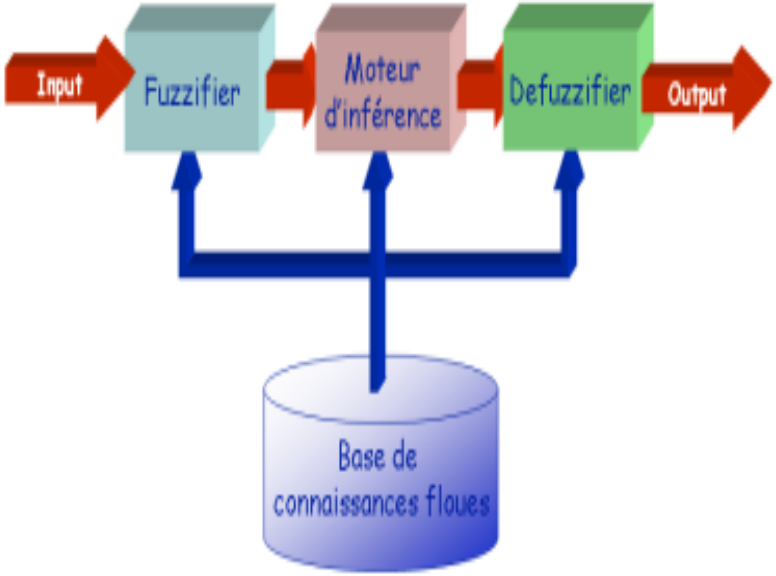




Prise de décision par logique floue
Fuzzy decision making

Il ya 4 étapes nécessaires lors de conception d'une **décision flou**:





Fuzzification

C'est le processus qui consiste à transformer une ***grandeur numérique*** en un ***ensemble flou***.

Revient à qualifier une ***variable numérique*** par une ***variable linguistique***

Comment fuzzifier?

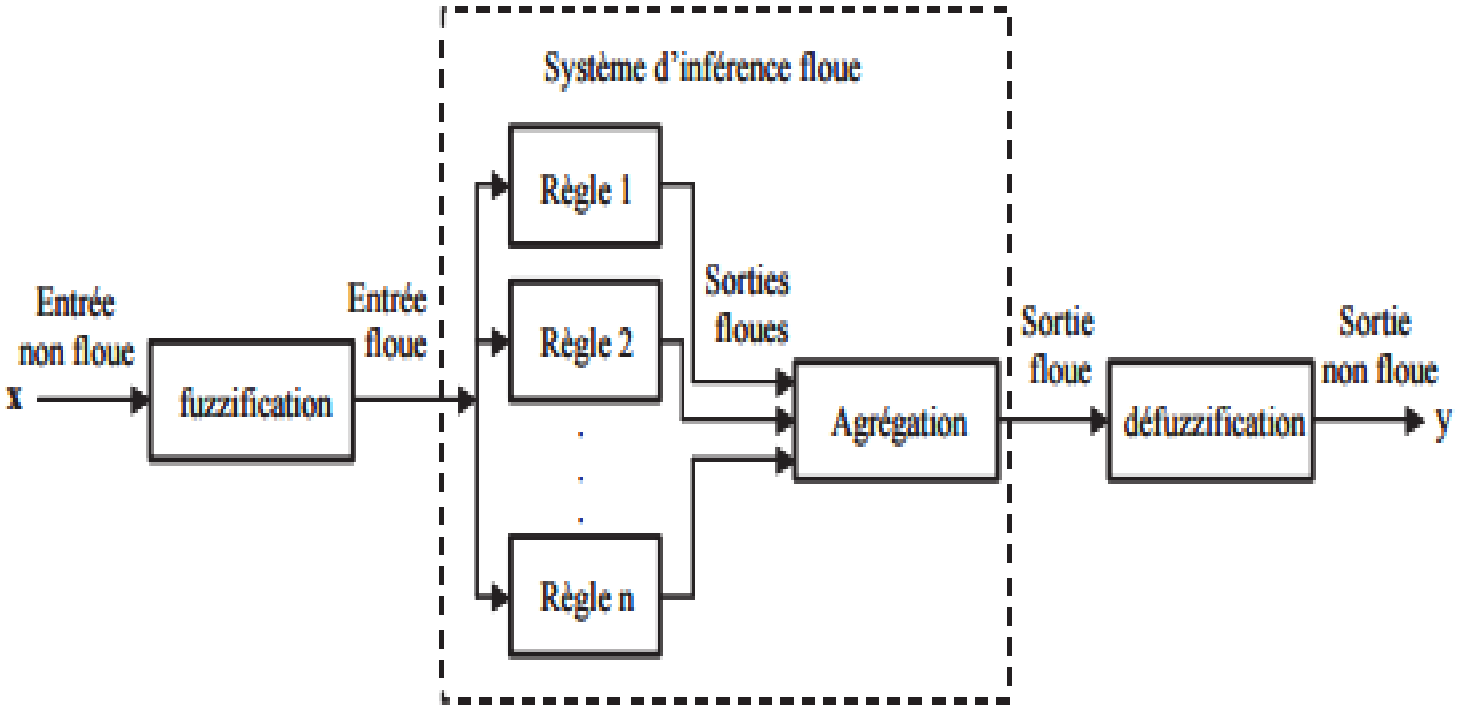
- Donner **l'univers de discours**: plage de variations possibles de l'entrée considérée(des percepts).
- **Une partition** en classe floue de cet univers

Exemple:

Une entrée *taille* (petit; moyenne;grand)

Une entrée *taille* (troppetit;petit; moyenne;grand; trog grand)

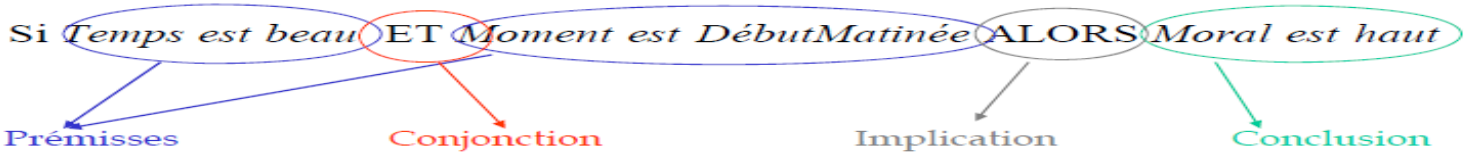
Système d'Inférence Floue(SIF) ou (FIS en Anglais)



Règle floue et proposition floue

Si x est A ET y est B Alors z est C

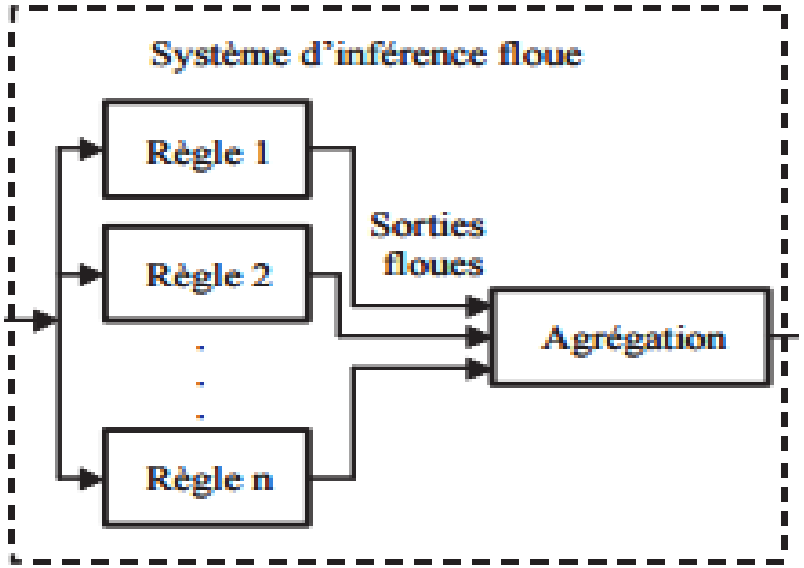
Exemple 1



Exemple 2

SI ***distance*** entre vihécules est ***petite*** et ***vitèsse*** est ***fort*** alors ***freiner*** ***tres*** ***fort***;

Étapes d'établissement de la sortie floue



Étape1 du SIF:
Calcul du degré d'activation de chaque règle

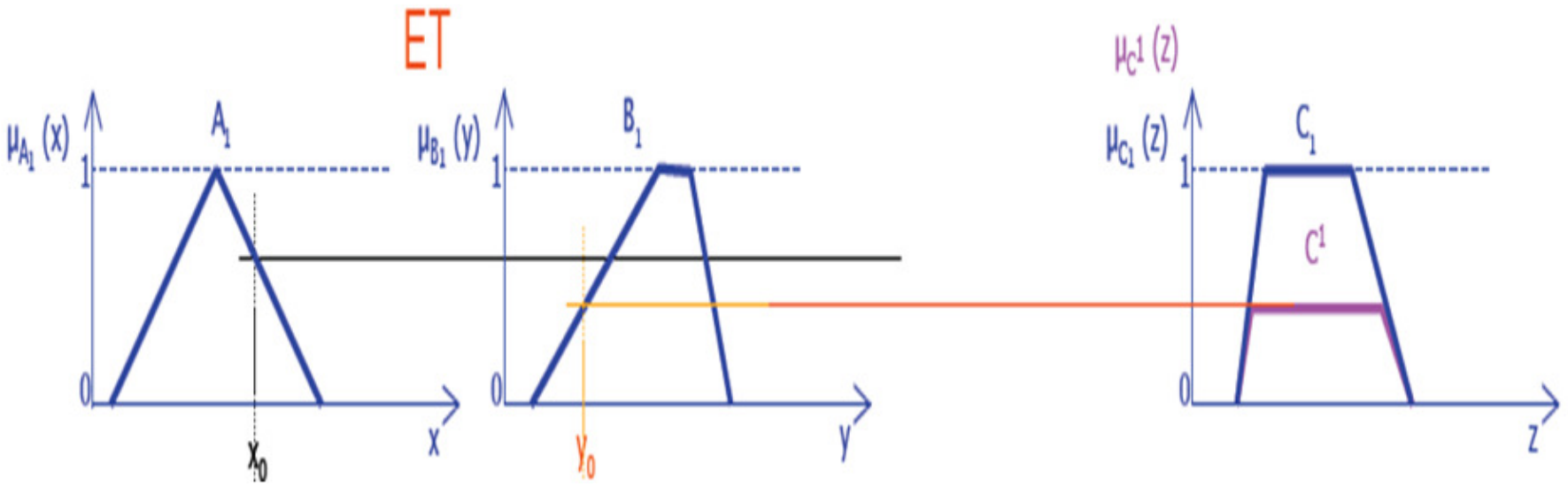
- L'activation des règles consiste à calculer le degré d'activation de chacune. C'est une valeur comprise entre 0 et 1
- **min(u,v)** permet de représenter la conjonction **ET**

Règle1 : ***Si*** *X1 est A1* ***ET*** *X2 est A2 alors y est B*

- ***Degré des deux permises (Si X1 est A1 ET X2 est A2) est le min des degrés d'appartenances des deux)***
- ***Dans ce cas:***

$\mu B(y)$ = poids de la règle 1 = min($\mu A1$ (x1), $\mu A2$ (x2))

$\Re_1$: si x est A_1 et y est B_1 alors z est C_1



Exemple d'illustration du cas d'une inférence avec une seule règle,

Nous supposons que nous voulons connaître la force de freinage d'une voiture qui roule dans une ville si le feu est rouge.

Pour cela nous avons deux caractéristiques associées : **Vitesse** et **Distance**. Ces caractéristiques sont alors nos variables d'entrée et la variable de sortie est **freinage**. Si nous considérons une voiture qui roule à une vitesse de *65 Km/h* et le feu se trouve à *20 mètres*,

Soit la règle suivante :

R1 : Si Distance est Courte et Vitesse est Maximale Alors Freinage est Fortement

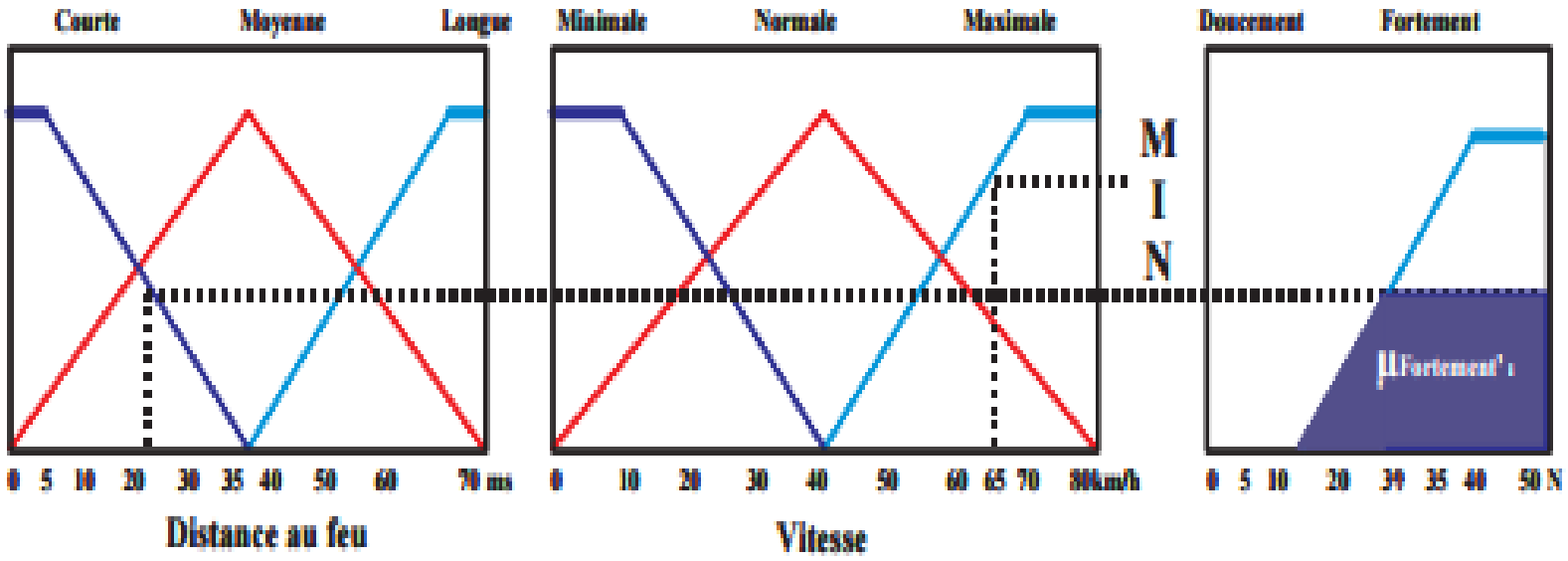
Le degré d'accomplissement ou degré d'appartenance (μ) pour la variable Distance (20) et Vitesse ($x_2 = 65$) est comme suit :

$$x_1 = 20 \rightarrow \mu_{Courte}(20) = 0.42 \text{ , } \mu_{Moyenne}(20) = 0.57 \text{ , } \mu_{Longue}(20) = 0.0$$

$$x_2 = 65 \rightarrow \mu_{Minimale}(65) = 0.0, \mu_{Normale}(65) = 0.375, \mu_{Maximale}(65) = 0.833$$

$$R_1 : Freiner_{\mu Fortement_1} = [min(\mu_{Courte}(20), \mu_{Maximale}(65))] = min(0.42, 0.833) = 0.42$$

Règle 1 : Si distance est Courte et Vitesse est Maximale Alors freiner Fortement



Activation des règles(calcul des degrés de toutes les règles

R1: $Si\ (X_1\ est\ A_{11})\ et\ (X_2\ est\ A_{12})$ $alors\ Y\ est\ B_1$

R2: $Si\ (X_1\ est\ A_{21})\ ou\ (X_2\ est\ A_{22})$ $alors\ Y\ est\ B_2$

R3: $Si\ (X_1\ est\ A_{31})\ et\ (X_2\ est\ A_{32})\ et\ (X_3\ est\ A_{33})$ $alors\ Y\ est\ B_3$

.....

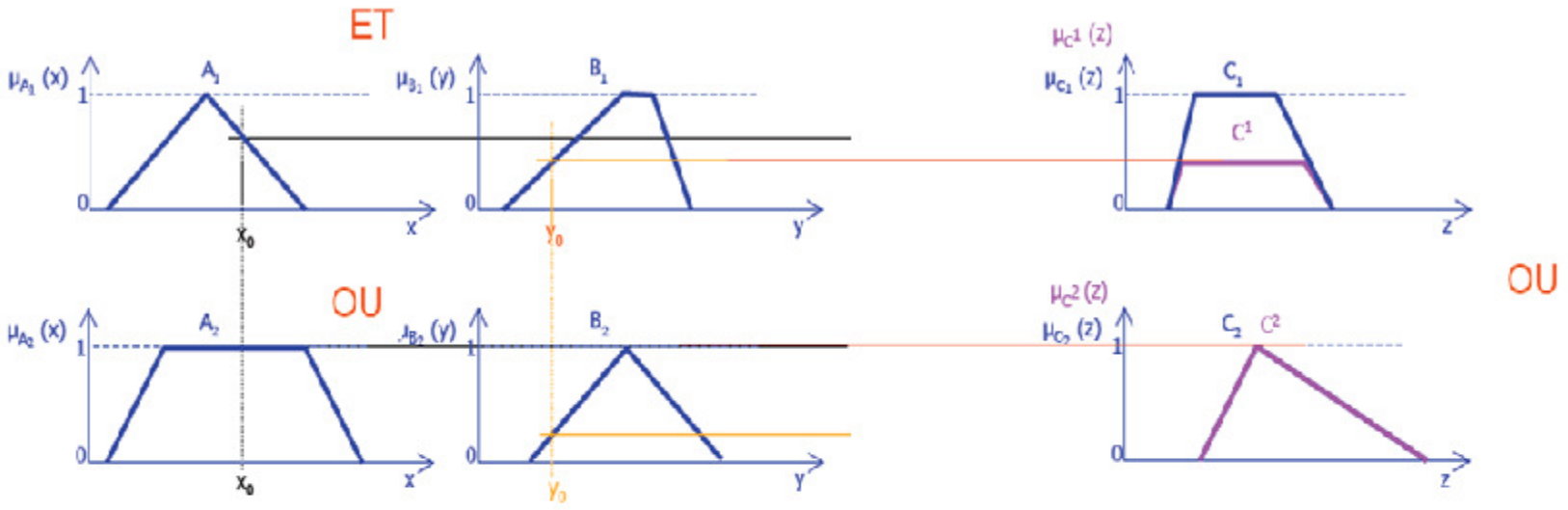
Agrégation

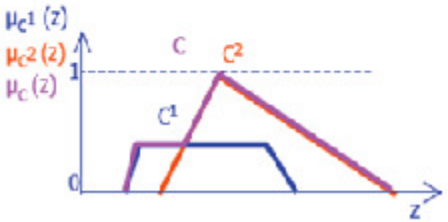
Les règles sont liées par un opérateur OU

c.à.d si R1 ou R2 ou ….ou Ri ou ….Rq

$$\mu_B(y) = MAX\left[\mu_{B_i}(y)\right] \qquad i \in \{indices\ des\ règles\ activées\}$$

y: sortie floue (décision floue)





$\Re_1$: si x est A_1 et y est B_1 alors z est C_1

$\Re_2$: si x est A_2 ou y est B_2 alors z est C_2

Fait : x est $\overline{x}_0$ et y est $\overline{y}_0$

Conséquence : z est C

Exemple d'illustration du cas d'une inférence avec deux règles,

- Nous continuons notre exemple de la voiture avec les mêmes valeurs, c'est-à-dire *20 mètres pour la distance au feu* et *65 km/h pour la vitesse de la voiture .*
- Nous considérons les deux règles suivantes :
- *R1 : Si Distance est Courte et Vitesse est Maximale Alors Freinage est Fortement*
- *R2 : Si Distance est Moyenne et Vitesse est Normale Alors Freinage est Doucement*

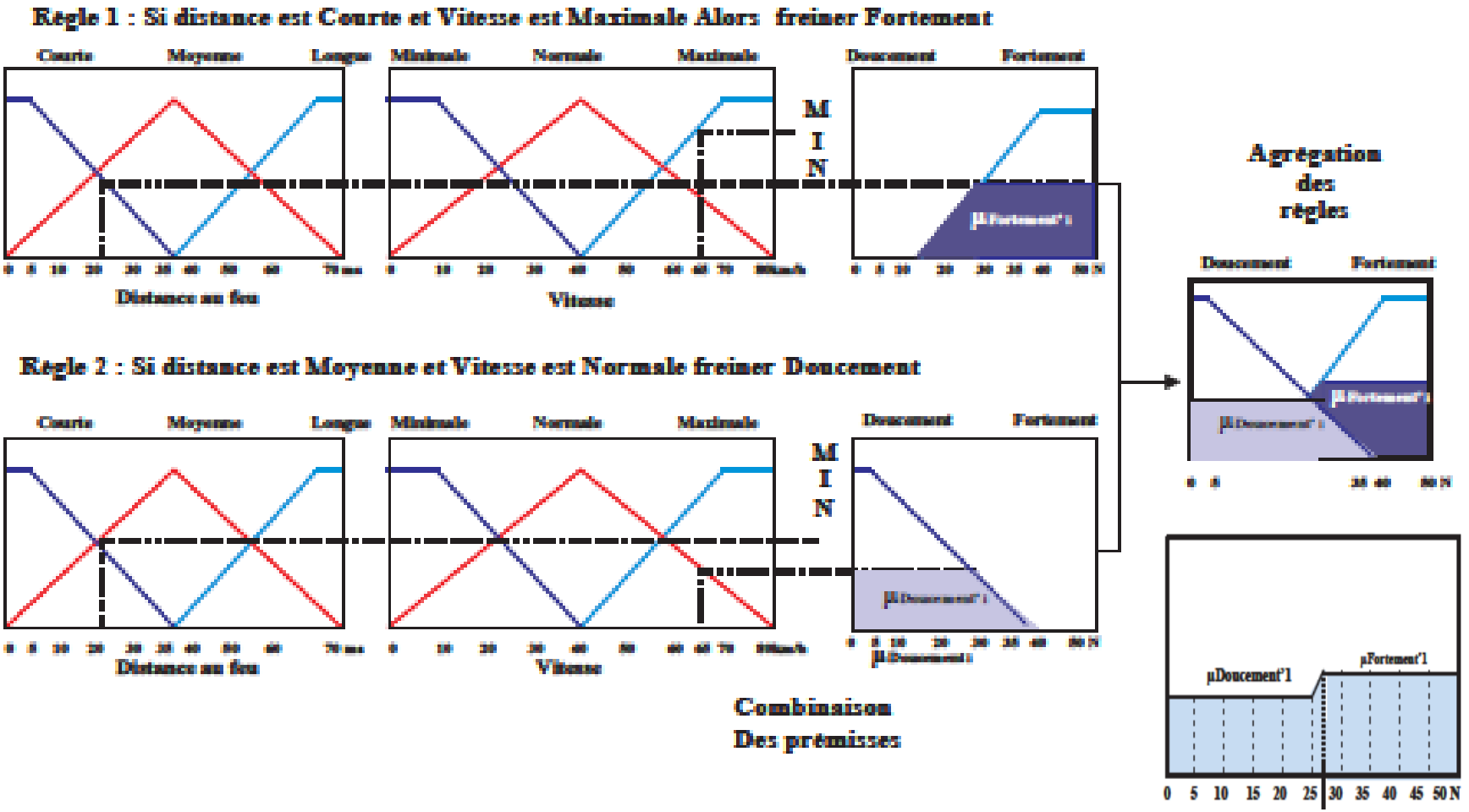
Comme on a vu que:

$$x_1 = 20 \rightarrow \mu_{Courte}(20) = 0.42\ ,\ \mu_{Moyenne}(20) = 0.57\ ,\ \mu_{Longue}(20) = 0.0$$

$$x_2 = 65 \rightarrow \mu_{Minimale}(65) = 0.0,\ \mu_{Normale}(65) = 0.375,\ \mu_{Maximale}(65) = 0.833$$

$$R_1 : \mu_{Fortement_1} = [min(\mu_{Courte}(0.42), \mu_{Maximale}(0.833))] = min(0.42, 0.833) = 0.42$$

$$R_2 : \mu_{Doucement_1} = [min(\mu_{Moyenne}(0.57), \mu_{Normale}(0.375))] = min(0.57, 0.375) = 0.375$$



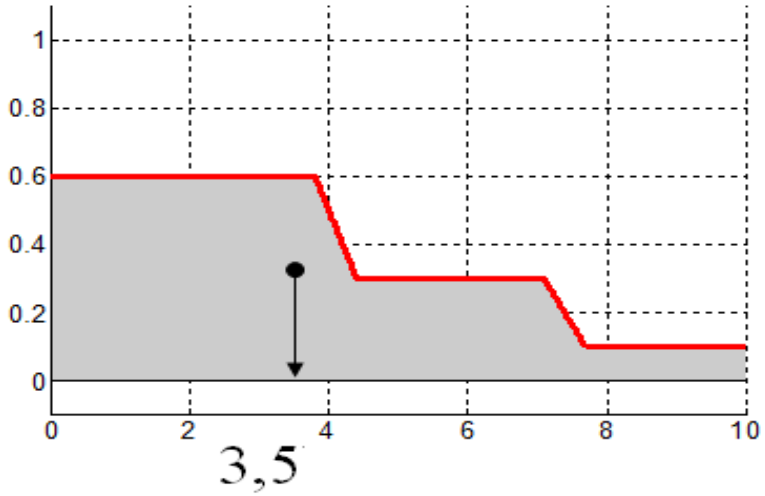
Méthode de défuzzification

Méthode du centre de gravité (COG) C'est l'abscisse du centre de gravité de la surface du sous ensemble flou obtenu après agrégation des règles.

$$sortie = \frac{\int_U y \cdot \mu(y) \cdot dy}{\int_U \mu(y) \cdot dy}$$

U = *Univers du discours*

= *Toutes les valeurs de sorties considérées*



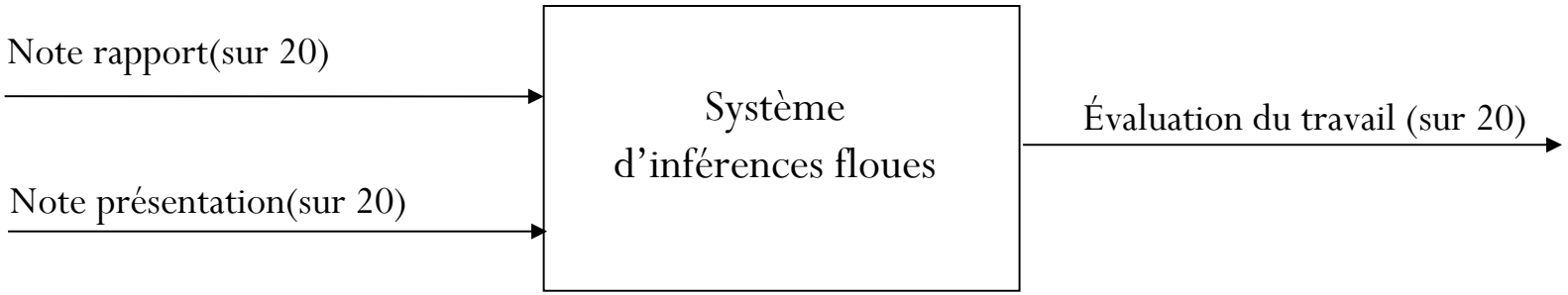
Remarque:

Il existe d'autres méthodes de défuzzification [Ross, 2005] :

méthodes des maximums,

- ✓ somme−prod,
- ✓ moyenne−pondérée,
- ✓ moyenne des maximums.
- ✓ Etc…

Exercice : Système d'évaluation floue des PFE Master



1. Choix des entrées/sorties

2 entrées: Rapport; Présentation.
1 sortie: Évaluation

2. Univers des discours

[0..20] pour chacune des E/S

3. Classes d'appartenances:

Notes Rapport et Présentation $\{$Médiocre; Moyen; Excellent$\}$

$$\text{Evaluation} \in \{\text{Médiocre; Mauvais; Moyen; Bon; Excellent}\}$$

Bases de règles (maximum 9 règles)

1. If (Rapport is excellent) and (Présentation is excellent) then (Evaluation is excellent)
2. . If (Rapport is moyen) and (Présentation is excellent) then (Evaluation is bon)
3. . If (Rapport is médiocre) and (Présentation is excellent) then (Evaluation is moyen)
4. . If (Rapport is médiocre) and (Présentation is moyen) then (Evaluation is mauvais)
5. . If (Rapport is médiocre) and (Présentation is médiocre) then (Evaluation is médiocre)
6. …

Travaux Pratique Décision floue

Utilisation de la laibrairie JFuzzyLogic en java

```
FUNCTION_BLOCK tipper// Block definition
VAR_INPUT// Define input variables
    service : REAL;
    food : REAL;
END_VAR
VAR_OUTPUT// Define output variable
    tip : REAL;
END_VAR
FUZZIFY service// Fuzzify input variable 'service': {'poor', 'good' , 'excellent'}
    TERM poor := (0, 1) (4, 0) ;
    TERM good := (1, 0) (4,1) (6,1) (9,0);
    TERM excellent := (6, 0) (9, 1);
END_FUZZIFY
// voir explication des valeurs
FUZZIFY food// Fuzzify input variable 'food': { 'rancid', 'delicious' }
    TERM rancid := (0, 1) (1, 1) (3,0) ;
    TERM delicious := (7,0) (9,1);
END_FUZZIFY
```

```
DEFUZZIFY tip// output variable 'tip' : {'cheap', 'average', 'generous' }
TERM cheap := (0,0) (5,1) (10,0);
TERM average := (10,0) (15,1) (20,0);
TERM generous := (20,0) (25,1) (30,0);
METHOD : COG;// Use 'Center Of Gravity' defuzzification method
DEFAULT := 0;// Default value is 0 (if no rule activates defuzzifier)
END_DEFUZZIFY

RULEBLOCK No1
AND : MIN;// Use 'min' for 'and' (also implicit use 'max' for 'or' to fulfill
    DeMorgan's Law)
ACT : MIN;// Use 'min' activation method
ACCU : MAX;// Use 'max' accumulation method

RULE 1 : IF service IS poor OR food is rancid THEN tip IS cheap;
RULE 2 : IF service IS good THEN tip IS average;
RULE 3 : IF service IS excellent AND food IS delicious THEN tip is generous;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

Quelques bibliothèques necessaires:

- **import net.sourceforge.jFuzzyLogic.FIS;**
- **import net.sourceforge.jFuzzyLogic.rule.*; (ou** .rule.FuzzyRuleSet)

```
public static void main(String[] args) throws Exception {
String fileName = "c:/tipper.fcl";
    FIS fis = FIS.load(fileName,true);
if( fis == null ) {
        System.err.println("Can't load file: ‘ »          + fileName + """);
        return;
    }
```

```
fis.chart(); // Visualiser les partitions floues de fuzzification
//Test:  // Set inputs
fis.setVariable("service", 4);
fis.setVariable("food", 7);
// Evaluate calcul de la décision pour la valeur de test
fis.evaluate();
// affichage de la décision:
System.out.println(fis.getVariable("tip").defuzzify());
fis.getVariable("tip").chartDefuzzifier(true);
```

Exercice

- Réaliser sous Java (JFuzzyLogic) une application intelligente à base de logique floue pour prédire le montant d'un crédit que peut obtenir un client bancaire, en se basant sur son âge et son salaire. On suppose que le montant maximal de crédit accordé est de 100 000 DH.
- On suppose que l'âge des clients est compris entre [20 et 65] , et le salaire est entre 2000 et 30 000 DH.

Et On considère que :

$$Age \in \{Petit, Moyen, Grand\}$$

$$Salaire \in \{Petit, Moyen, Grand\}$$

$$Mon\tan t\_ crédit \in \{TrèsPetit, Petit, Moyen, Grand, TrèsGrand\}$$

Exercice2:

Réaliser, sous JFuzzyLogic, une application d'aide à la conduite d'un véhicule, (système de freinage intelligent):

- On considère que la vitesse et la distance entre deux véhicules seront représentées par 5 valeurs linguistiques et le freinage par 7 valeurs linguistiques.
- On suppose que la vitesse maximale du véhicule est de 200 Km/h , la distance maximale qui nous intéresse entre les deux véhicules est de 100 m et le freinage varie entre 0 et 100.

ET flou	: MIN
OU flou	: MAX
Implication floue	: MIN
Agrégation des règles	: MAX
Défuzzification	: COG

Introduction à l'Apprentissage Artificiel

Agent intélligent

À base de règles

d'inférence:

- Logique des prédicats
- Logique floue

Agent intélligent

À base d'Apprentissage:

- Arbres de décision
- Réseaux de Neuronnes
- Support vecteur machines

…..

Apprentissage Naturel (Humain)

Dès sa naissance l'enfant apprend:

- La voix de sa mère
- Apprend à marcher
- Apprend à lire
- …

Apprentissage Artificiel (ou Automatique)

Les machines (agents) ont besoin d'apprendre

L'apprentissage est essentiel pour des environnements inconnus, où le concepteur ne peut pas tout savoir à l'avance.

Ce qui permet d'améliorer la capacité de l'agent à agir dans le futur

Domaines d'applications

- Elles s'appliquent à un grand nombre d'activités humaines

Il s'agira, par exemple:

- d'établir un **diagnostic** médical à partir de la description clinique d'un patient,
- Reconnaissance de l'écriture, des formes, paroles, de visages…
- Approximation des fonctions incertaines
- BIG DATA
- …

Les données pour l'apprentissage:
Ensemble d'apprentissage S et un *ensemble test T*

- On partitionne l'échantillon en un ***ensemble d'apprentissage S*** **et un *ensemble test T*.**

- **Exemple:**

reconnaissance de chiffres: Les données?

Types d'apprentissage

- **Apprentissage supervisé (Existence de superviseur)**

(**sortie désirée(Target): connue**)

- **Apprentissage non-supervisé(Clustering)**

sortie désirée inconnue

- **Apprentissage par renforcement(par experiences)**

Apprentissage supervisé

- S = {(X1, y1), (X2, y2), … , (Xi , yi), … , (Xm, ym)}
- T = {Xt1, Xt2,, … , Xti , … , Xtp}

Objectif est de chercher Une fonction de décision (classifieur)

Deux types de problèmes

Régression(prédire des valeurs numériques continues)

et

Classification(les valeurs à prédire sont discrètes)

Apprentissage supervisée:

Formalisme de la de la classification supervisée

- Un ensemble X = X1 x X2x…Xn ou chaque Xi est le domaine d'un attribut ***Xi symbolique***, ***numérique*** ou ***structuré***.
- Un ensemble fini de classes ***Y(classes désirées)***
- Les exemples sont des couples $(x; y) \in X$x Y
- Un échantillon S est un ensemble fini d'exemples

Classifieur : $f : X \rightarrow Y$

Le problème général de la classification supervisée :
étant donné un échantillon S = {(x1; y1); …. ; (xn ; yn)}, trouver un classifieur f qui minimise une fonction objectif (risque) R(f).

Fonction risque (ou fonction erreur)

Fonction de perte : $L(y, f(x)) = \begin{cases} 0 \text{ si } y = f(x) \\ 1 \text{ sinon.} \end{cases}$

Où y sortie désiré donné par le superviseur et f(x) sortie du classifieur

L'espérance mathématique de cette mesure permet de définir la ***fonction risque*** suivante(***dans le cas continu***):

$$R(f) = \int L(y, f(x)) dP(x, y)$$

Le problème est de trouver un classifieur f qui minimise le risque R(f).

Pour le problème régression

Dans un problème de régression, *y* prend des valeurs continues et l'on cherche également à exprimer par une fonction la dépendance entre x et y. La fonction de perte qu'on considère principalement est l'écart quadratique défini par:

$$L(y, f(x)) = (y - f(x))^2.$$

L'erreur d'une fonction f est alors l'´ecart quadratique moyen défini par :

$$R(f) = \int_{\mathcal{X}\times\mathcal{Y}} (y - f(x))^2 dP(x, y).$$

Pratiquement: utiliser Risque empirique

- Le risque empirique Remp(f) d'une fonction f sur l'échantillon S = {(x1, y1), . . . (xl, yl)} est la moyenne de la fonction de perte calculée sur S:

$$R_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l} L(y_i, f(x_i)).$$

Remp(f) est la moyenne du nombre d'erreurs de prédiction de f sur les éléments de S :

$$R_{emp}(f) = \frac{Card\{i | f(x_i) \neq y_i\}}{l}.$$

Lorsque f est une fonction de **régression** et L la fonction de perte quadratique, Remp(f) est la moyenne des carrés des écarts à la moyenne de f sur S :

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} (y_i - f(x_i))^2.$$

Exemple de problème de régression:

Approximation de fonctions

- Données : N(10 par exemple) points sur la courbe
  $x \rightarrow 2*\sin(x) + 4$ sur [0 10]

Chapitre 3

Apprentissage par arbres de décisions

Apprentissage à l'aide d'arbres de décisions

But: d'identifier les classes auxquelles appartiennent des objets à partir de certains traits ***descriptifs et symbolique***.

Exemple:

Accord d'un prêt bancaire : à partir de la situation d'un ***client*** (sa **description) la procédure de** classification donne la réponse à la demande de prêt : oui / non (sa **classe).**

L'exemple précédent fait déjà apparaître trois objets essentiels:

1. **La population : les clients.**
2. **Les descriptions : les revenus, âge, statut marital d'un client, sa domiciliation...**
3. **Les classes : les réponses à la demande de prêt (oui / non).**

Remarque:

La donnée d'une description avec la classe correspondante est un **exemple.**

Exemples:

(30 ans, marié(oui), 8000 DH, oui)

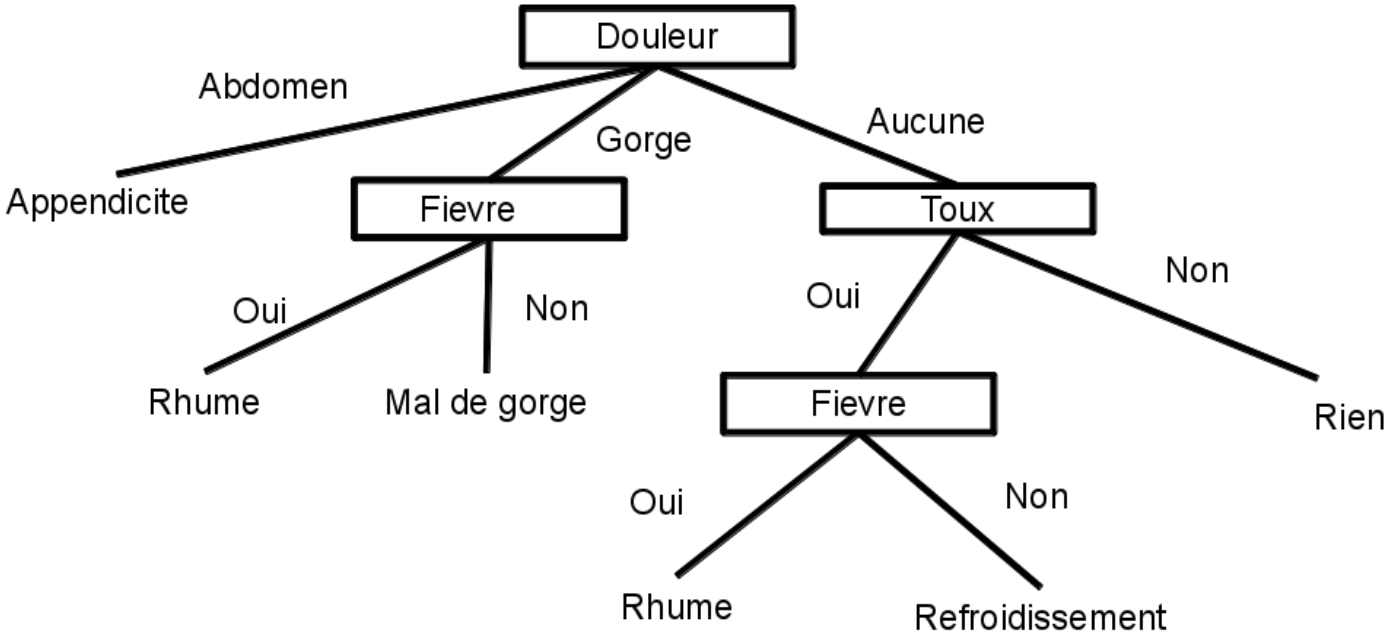
(50 ans, marié(oui), 3000 DH, Non)

Fonctionnement:

Un arbre de décision accepte en entrée une situation ou un Objet décrit par un ensemble d'attributs et retourne une décision

Un arbre de décision atteint une décision en exécutant une séquence de tests.

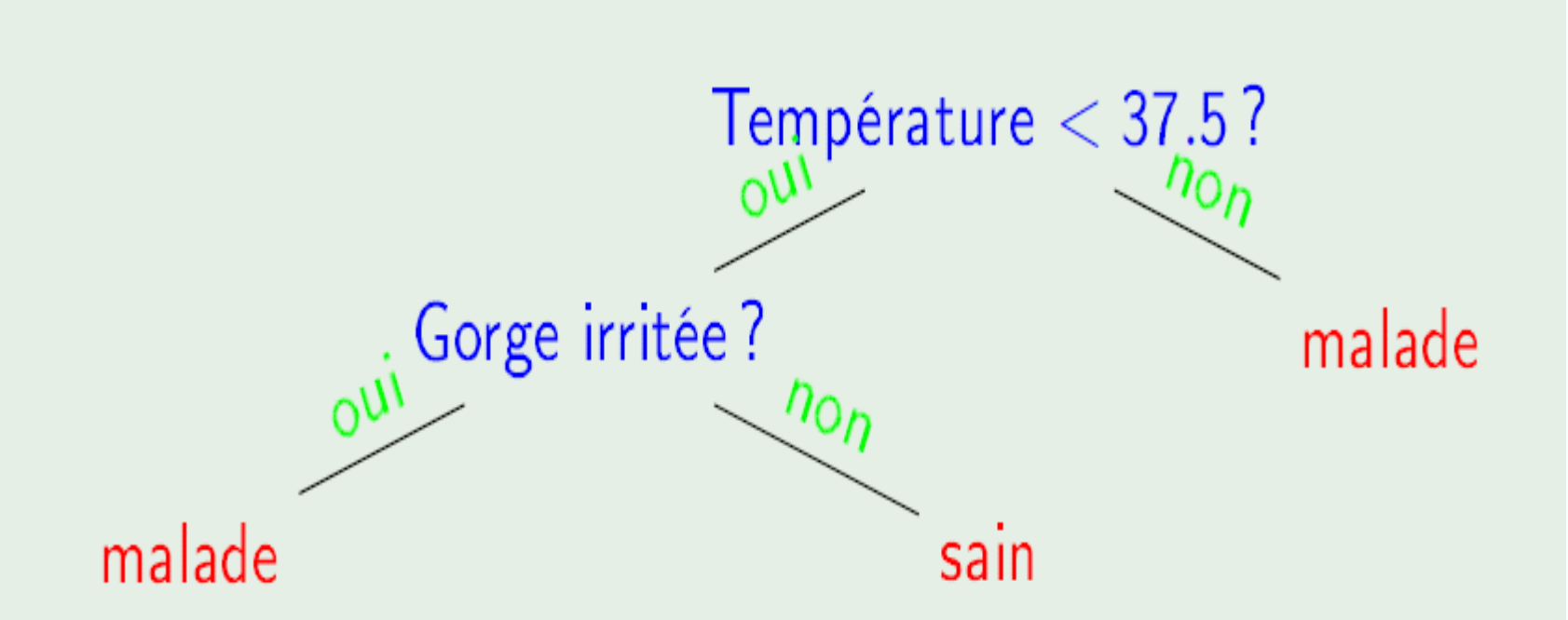
Exemple d'arbre de décision:



L'objectif est de construire un arbre de décision qui classifie les patients en utilisant un **algorithme de classification** et un **ensembles d'exemples bien choisis.**

Noeuds: Attributs
Branches: Valeurs prises par les Attributs
Feuilles: Décisions (Classes)
Arbre: Classifieur

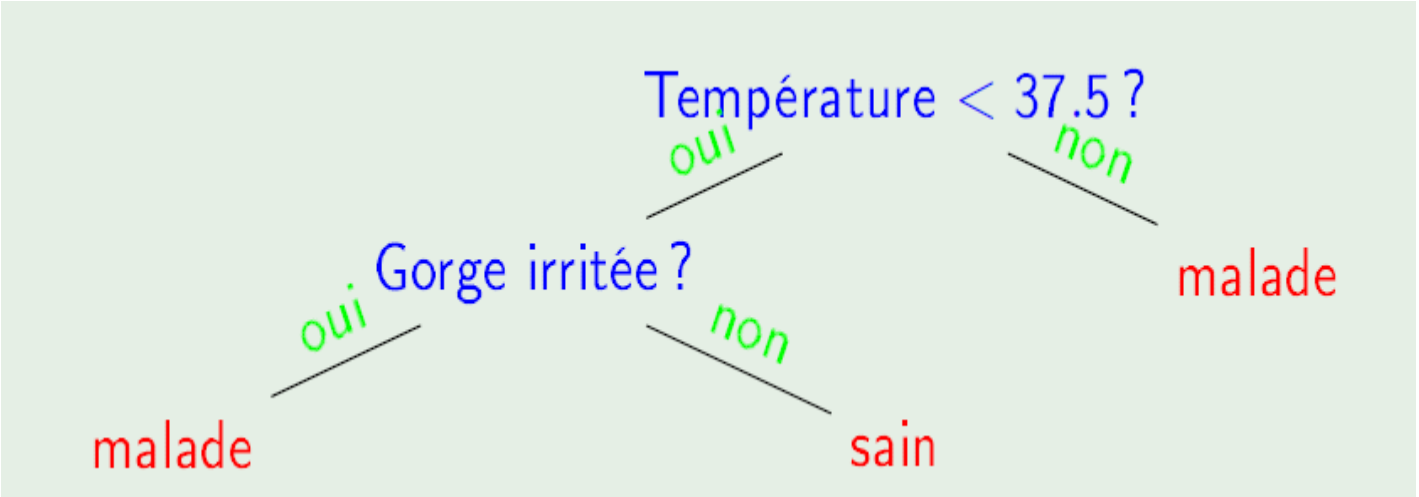
Des objets (des patients),
des attributs (Température et Gorge irritée),
des classes (malade ou sain).



Chaque nœud interne correspond à un test sur un attribut,
et les branches partant du nœud sont annotées avec les valeurs possibles du test (de l'attribut)

Après construction de l'arbre, Un nouveau exemple est classé depuis la racine jusqu'à une feuille

Exemple: l'arbre suivant Permet de classer un nouveau exemple : (37.2,oui),



Remarques

- Les arbres de décision ont deux qualités appréciables :
  - ***les décisions sont aisément interprétables,***
  - ***la classification est très rapide.***

? Décider si on joue au tennis ou non.

Attributs considérées:

Ciel: (Soleil, couvert, pluie)

Temp: (chaux, doux, froid)

Humidité: (élevée, normale)

Vent: (fort, faible)

Sortie: jouer ou non (oui, non)

Ensembles d'exemples à utiliser pour l'apprentissage d'arbre de décision

jour	ciel	temp.	humidité	vent	jouer
1	soleil	chaud	élevée	faible	non
2	soleil	chaud	élevée	fort	non
3	couvert	chaud	élevée	faible	oui
4	pluie	doux	élevée	faible	oui
5	pluie	froid	normale	faible	oui
6	pluie	froid	normale	fort	non
7	couvert	froid	normale	fort	oui
8	soleil	doux	élevée	faible	non
9	soleil	froid	normale	faible	oui
10	pluie	doux	normale	faible	oui
11	soleil	doux	normale	fort	oui
12	couvert	doux	élevée	fort	oui
13	couvert	chaud	normale	faible	oui
14	pluie	doux	élevée	fort	non

Les exemples positifs sont ceux pour lesquels le but (jouer est vrai: oui).
Les exemples négatifs sont ceux pour lesquels le but (jouer est faux: non).

Les arbres sont construits **récursivement** de façon descendante.

Lorsqu'un test est choisi, on divise l'ensemble d'apprentissage pour chacune des branches et on réapplique **récursivement l'algorithme.**

- Sur notre exemple:

On **initialise avec l'arbre vide**.

L'échantillon contient 14 éléments dont 9 ont pour classe oui et 5 ont pour classe non.

l'échantillon peut être caractérisé par le couple (9,5).

Par exemple, si nous avions le couple (0,14) c'est-à-dire qu'aucun jour ne favorise le jeu, **on attribuerait une feuille contenant <u>non</u> au noeud courant**

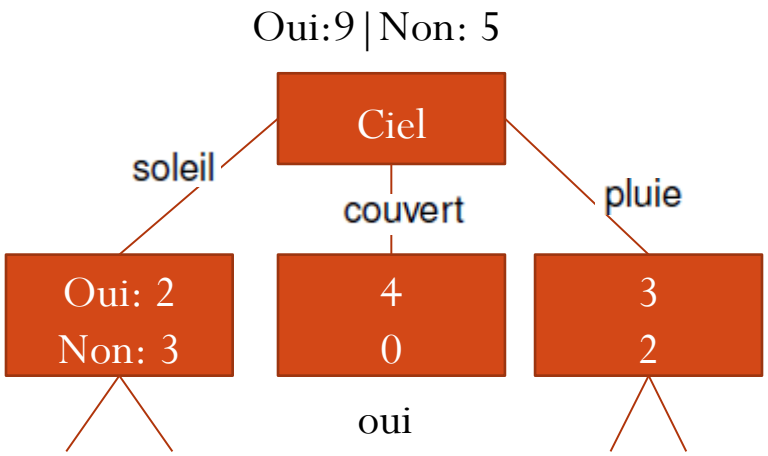
Ici, ce n'est pas le cas.

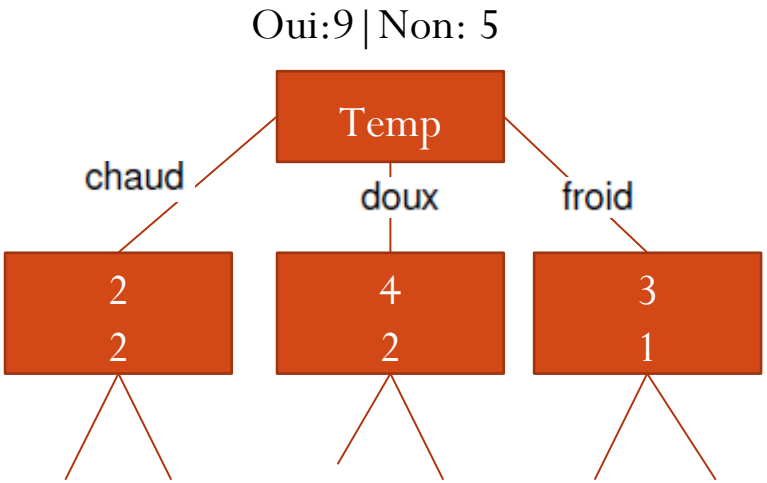
Quelle est alors l'attribut qui va initialiser l'arbre?

Nous devons choisir un test parmi les 4 attributs possibles:

Lequel de ces 4 tests est le plus discriminant ?

Choix de la racine (initialisation de l'algorithme)





Temp est choix médiocre: 3 résultats possibles dont chacune nécessite d'autres tests

Ciel: Attribut important: car si sa valeur est couvert terminée on répond oui ou non

Comment alors choisir le meilleur attribut? Utilisation d'un critère

Principaux algorithmes d'apprentissage par arbres de décision :

Deux catégories:

CART [Breiman84], ***utilise l'indice de Gini***

ID3 [Quinlan83]. C4.5 [Quinlan94] ***utilise la notion d'entropie***

ID3 développée par Quinlan en 83, améliorée en 94 par une nouvelle version C4.5

Algorithme ID3

4 cas à considérer pour le problème récursif

1. S'il ya des exemples positifs et négatifs, choisir le **meilleur attribut** pour les différentier

? Comment choisir un attributs

2. Si les exemples restant sont tous positifs ou tous négatifs on a terminé on peut répondre oui ou non
3. s'il ne reste **pas d'exemples** cela signifie qu'on n'a observé aucun exemple de cette sorte (***manque d'exemples pertinents***) on retourne une **valeur par défaut** calculée à partir de la classification majoritaire au niveau du **parent du noeud**
4. s'il ne reste pas d'attributs mais qu'il reste des exemples positifs et négatifs c'est le cas où certaines données sont incorrectes (bruité), une solution consiste à utiliser un vote à la majorité.

Choix de L'attribut à tester

L'attribut ciel n'est pas parfait mais assez satisfaisant. Cependant temp est moins satisfaisant.

Il faut donc une mesure formel de ce qui est satisfaisant et inutile

Cette mesure doit avoir sa valeur maximale quand l' attribut est parfait et sa valeur minimale quand il n'est pas utile.

Le gain d'information par rapport au test de l'attribut est:

$$\text{Gain (A)} = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) \text{ - moyenne(A).}$$

$$\text{Moyenne (A)} = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

Dans l'apprentissage d'un arbre de décision, si **l'ensemble d'apprentissage** contient P exemples positifs et n exemples négatifs, une estimation de l'information contenue dans une réponse correcte:

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right)=-\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right)-\left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right)$$

Dans le cas du jeu de tennis:

P=9 et n=5

On a: …

Chaque attribut A divise l'ensemble d'exemples d'apprentissage E en sous ensembles E1,…Ev en fonction de leurs valeurs pour A.

Exemple: ciel (E1,E2,E3)

E1={exemples/ciel=soleil , 5 exemples (2positifs et 3 négatifs)}

E2={exemples/ciel=couvert , 4 exemples (4positifs et 0 négatifs)}

E3={exemples/ciel=pluie , 5 exemples (3positifs et 2 négatifs)}

Chaque Ei à pi exemples positifs et ni exemples négatifs

Si l'on regarde de nouveau les attributs considérés au tableau, on a:

Gain (ciel) = ?

p=9, n=5;

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right)=I\left(\frac{9}{14},\frac{5}{14}\right)$$

$$=-\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right)-\frac{5}{14}\log_2\left(\frac{5}{14}\right)$$

Moyenne (ciel) $=\sum_{i=1}^{3}\frac{p_i+n_i}{p+n}I\left(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right)$

$$=\frac{1}{14}\left[(2+3)I\left(\frac{2}{5},\frac{3}{5}\right)+4I\left(\frac{4}{4},\frac{0}{4}\right)+(3+2)I\left(\frac{3}{5},\frac{2}{5}\right)\right]$$

$$\log_a(x)=\frac{\ln(x)}{\ln(a)}$$

Exercice

jour	ciel	temp.	humidité	vent	jouer
1	soleil	chaud	élevée	faible	non
2	soleil	chaud	élevée	fort	non
3	couvert	chaud	élevée	faible	oui
4	pluie	doux	élevée	faible	oui
5	pluie	froid	normale	faible	oui
6	pluie	froid	normale	fort	non
7	couvert	froid	normale	fort	oui
8	soleil	doux	élevée	faible	non
9	soleil	froid	normale	faible	oui
10	pluie	doux	normale	faible	oui
11	soleil	doux	normale	fort	oui
12	couvert	doux	élevée	fort	oui
13	couvert	chaud	normale	faible	oui
14	pluie	doux	élevée	fort	non

1) Déterminer l'attribut qui initialise l'arbre de décision

2) Réaliser l'apprentissage par arbre de décision à partir des exemples de tableau précédent

Serie_TD1

TP Arbre de décision

Diviser data set

Applications aux arbres de décision

- Le package **sklearn.tree** propose des algorithmes d'apprentissage fondés sur les arbres de decision (classication, regression)
- ***Exemple1:***

```
from sklearn import tree
X = [[0, 0], [1, 1]]
Y = [0, 1]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)
print(clf.predict([[0.9, 0.9]]))
```

- **DecisionTreeClassifier**: est utilisable soit dans le cas binaire où les étiquettes sont [-1, 1]. Soit dans le cas multiclasses où les étiquettes sont [0, ..., K-1] classification.

- sklearn.tree.DecisionTreeClassifier(*criterion='gini'*,*splitter='best'*, *max_depth=None*, *min_samples_split=2*,*min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*,*max_leaf_nodes=None*, *class_weight=None*)

Parameters:

1) **criterion** : string, optional (default= “gini”)(si non l’argument criterion prend la valeur “entropy”); The function to measure the quality of a split
2) **splitter** : string, optional (default=”best”) ”)(si non l’argument **splitter peut** prendre la valeur “random”);
3) **max_features** : int, float, string or None, optional (default=None). Le nombre de caractéristiques à considérer lors de la recherche de la meilleure partition
4) **max_depth** : int or None, optional (default=None): La profondeur maximale de l'arbre
5) ….

Accuracy= clf.score(X,y)

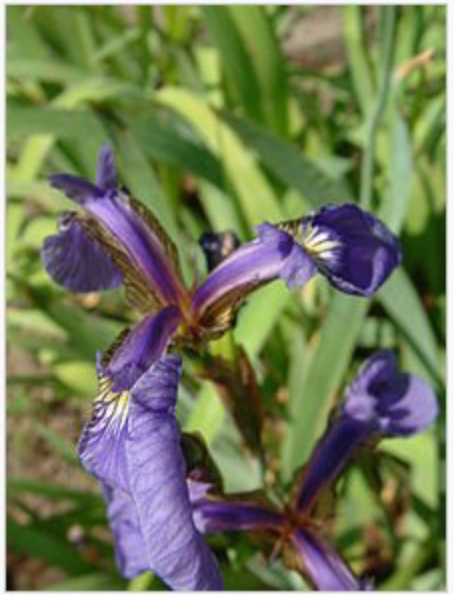
La méthode clf.score(X,y) permet de calculer le taux de bonne classification du modèle appris par la méthode clf, sur un ensemble de données stocke dans X et dont les classes sont stockées dans y. Le taux de bonne classification a, appelé en anglais accuracy, est tel que

a = 1-e où e est le taux d'erreur.

Jeux de données Iris

- Iris est un ensemble de données qui est disponible dans scikit-learn dans le package ***datasets***.
- Comme tout datasets, il est constitué de deux dictionnaires
- .data qui stocke un tableau de dimensions nxm ou n est le nombre d'instances, et m le nombre d'attributs
- .target qui stocke les classes, cibles (étiquettes) de chaque instance (dans le cas supervisé).

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*
4.4	2.9	1.4	0.2	*I. setosa*
4.9	3.1	1.5	0.1	*I. setosa*
5.4	3.7	1.5	0.2	*I. setosa*
4.8	3.4	1.6	0.2	*I. setosa*



Iris setosa

7.0	3.2	4.7	1.4	*I. versicolor*
6.4	3.2	4.5	1.5	*I. versicolor*
6.9	3.1	4.9	1.5	*I. versicolor*
5.5	2.3	4.0	1.3	*I. versicolor*
6.5	2.8	4.6	1.5	*I. versicolor*
5.7	2.8	4.5	1.3	*I. versicolor*
6.3	3.3	4.7	1.6	*I. versicolor*
4.9	2.4	3.3	1.0	*I. versicolor*
6.6	2.9	4.6	1.3	*I. versicolor*
5.2	2.7	3.9	1.4	*I. versicolor*
5.0	2.0	3.5	1.0	*I. versicolor*
5.9	3.0	4.2	1.5	*I. versicolor*
6.0	2.2	4.0	1.0	*I. versicolor*
6.1	2.9	4.7	1.4	*I. versicolor*

Iris versicolor

6.3	3.3	6.0	2.5	*I. virginica*
5.8	2.7	5.1	1.9	*I. virginica*
7.1	3.0	5.9	2.1	*I. virginica*
6.3	2.9	5.6	1.8	*I. virginica*
6.5	3.0	5.8	2.2	*I. virginica*
7.6	3.0	6.6	2.1	*I. virginica*
4.9	2.5	4.5	1.7	*I. virginica*
7.3	2.9	6.3	1.8	*I. virginica*
6.7	2.5	5.8	1.8	*I. virginica*
7.2	3.6	6.1	2.5	*I. virginica*
6.5	3.2	5.1	2.0	*I. virginica*
6.4	2.7	5.3	1.9	*I. virginica*
6.8	3.0	5.5	2.1	*I. virginica*
5.7	2.5	5.0	2.0	*I. virginica*
5.8	2.8	5.1	2.4	*I. virginica*

Iris virginica

Partition aléatoire des données d'un dataset

- Scikit learn propose un package **cross_validation** qui comprend de nombreuses fonctionnalités, dont la fonction:

cross_validation.train_test_split qui partitionne aléatoirement un jeu de données en un ensemble d'apprentissage et un ensemble de test.

- ***from sklearn.cross_validation import train_test_split***
- ***import random*** #pour pouvoir utiliser un generateur de nombres aleatoires

Soit X données d'apprentissage et Y données désirées:

- partition de l'échantillon de travail en un échantillon d'apprentissage (70 %) et un échantillon test (30 %):

X_train, X_test, Y_train, Y_test

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size
    =0.3,random_state=random.seed())

clf.fit(X_train, Y_train) # entrainement de clf sur l'échantillon
    d'apprentissage
# estimation de l'erreur de clf calculée sur l'échantillon test
print("score estimé du classifieur appris", clf.score(X_test,
    Y_test))
```

- Sous ***scikit-learn***, dans tout classifieur il existe deux méthodes essentielles:

- 1. fit(tableau_data,tableau_target) qui apprend un modèle à partir des données

- 2. La fonction predict(tableau_data) qui renvoi un tableau qui stocke, pour chaque nouvelle donnée en entrée, la classe prédite.

Exercice:

- Produire Sous *scikit-learn* un programme Python qui:
- charge le jeu de données Iris,

```
from sklearn import datasets
irisData = datasets.load_iris()
print (irisData.data)
print (irisData.target)
```

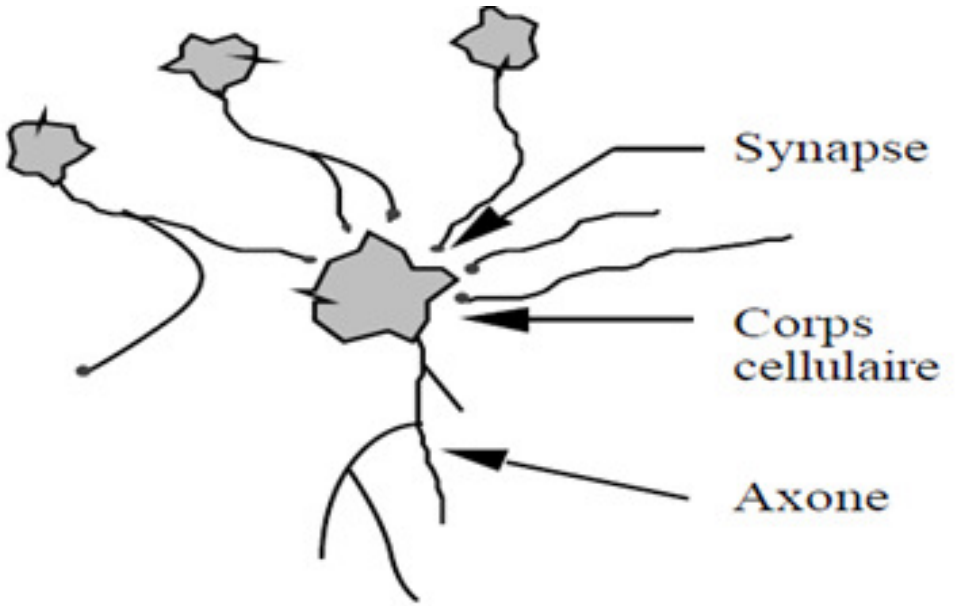
- Apprend un arbre de décision en utilisant le paramétrage par défaut
- de séparer les données de travail en des données d'apprentissage et de test,
- Faire l'apprentissage sur les données d'apprentissage en utilisant les arbres de décision : (DecisionTreeClassifier())
- d'afficher la précision de cet arbre évalué sur l'échantillon test.

Réseaux de neurones Une origine biologique (Système nerveux)

- Comment l'homme fait-il pour apprendre raisonner, calculer à travers son cerveau …?
- Comment s'y prendre pour créer une intelligence artificielle ?
- La physiologie du cerveau montre que celui-ci est constitué de cellules (les neurones) interconnectées:

- Un neurone = cellule cérébrale dont la fonction principale consiste à **collecter**, **traiter** et **transmettre** des informations

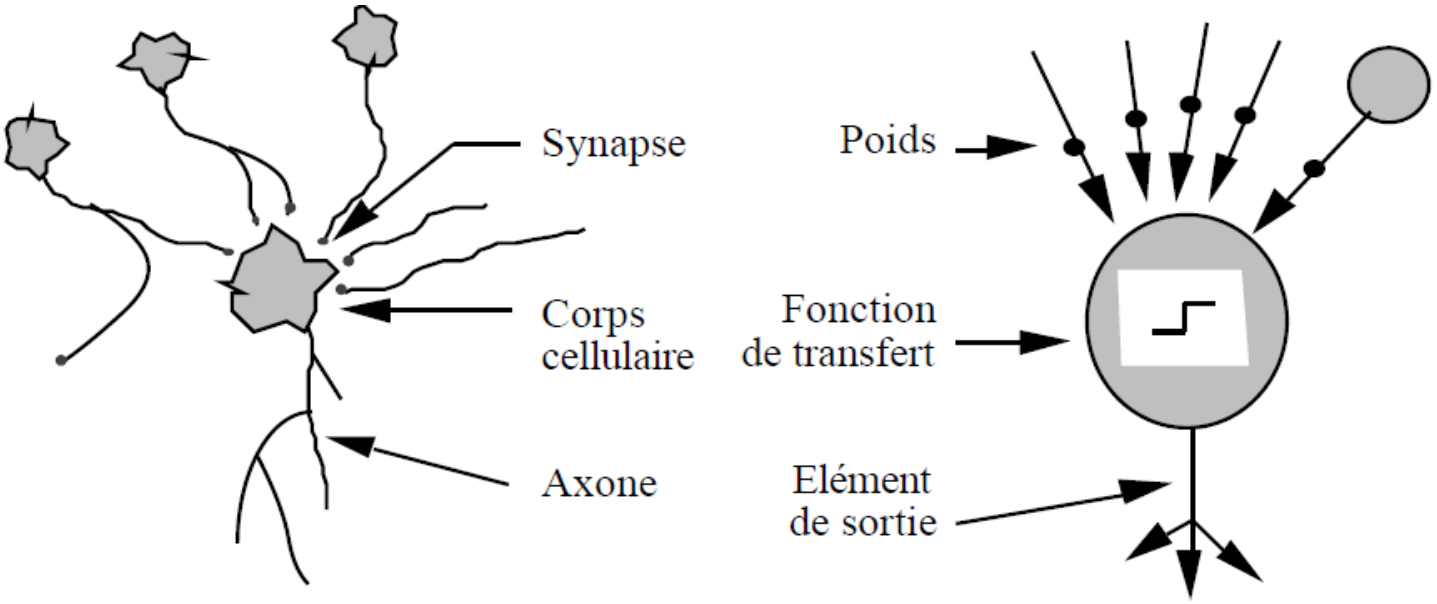
- le cerveau contient environ **100 milliards** de neurones
- nombre total de connexions est estimé à environ **10^15** .
- le nombre de neurones actifs décroit,



L'apprentissage est indispensable à son développement

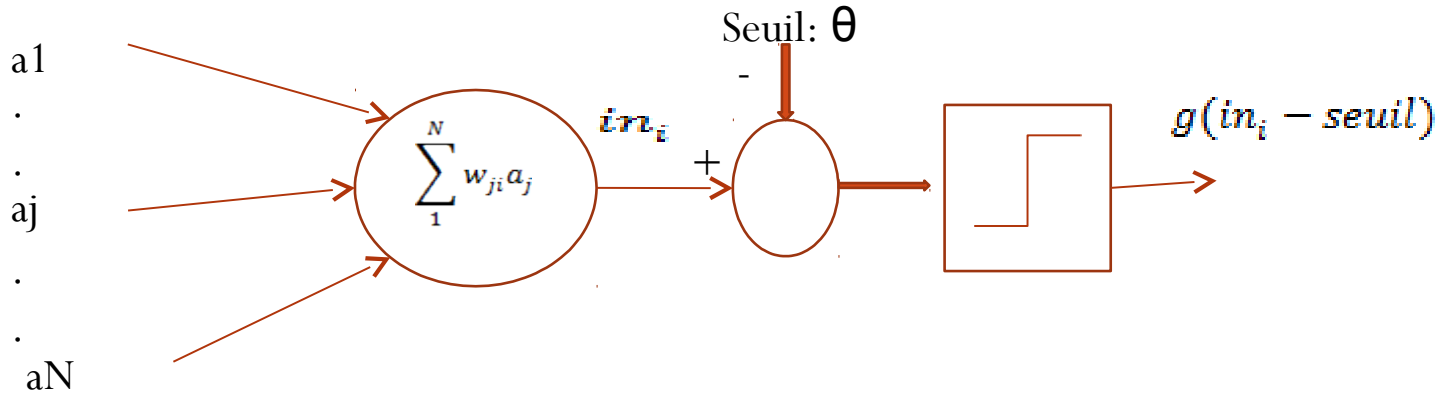
Du neurone biologique au neurone artificiel

Les *neurones* ***reçoivent*** *des informations par les dendrites* (par l'intermédiaire des ***synapses***) ***traitent*** *et* ***envoient*** *l'information par les axones.*



- Dendrites : Signaux d'entrée
- Axone : Signal de sortie

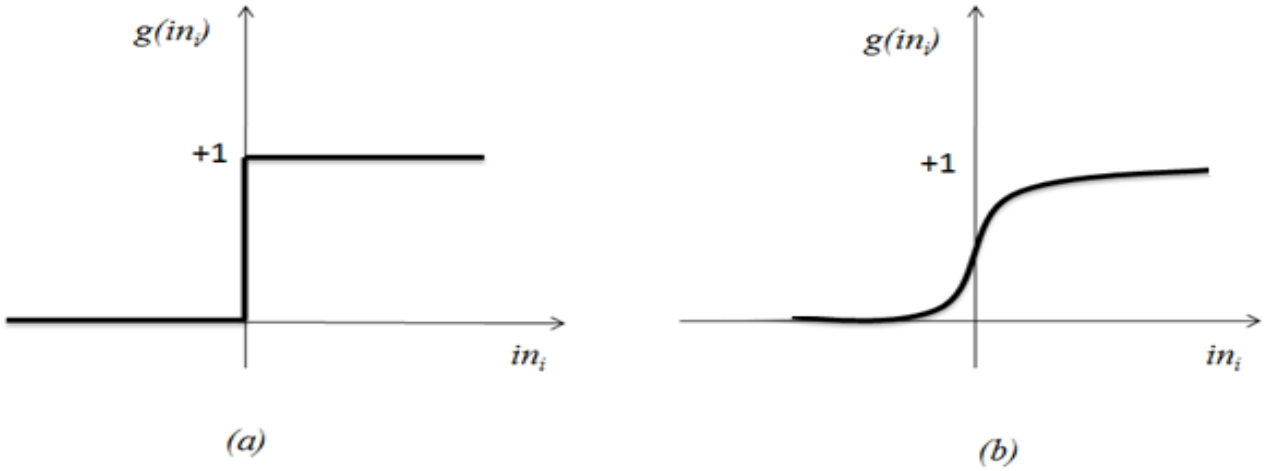
- Le neurone reçoit les entrées a1, …, ai, …, an.
- Le potentiel d'activation du neurone est défini comme la somme pondérée (les poids sont les coefficients synaptiques wi) des entrées.
- La sortie y est alors calculée en fonction du seuil θ.



$$s_i = g((\sum_{j=1}^{n} W_{j,i} a_j) - seuil)$$

Modèle d'un neurone artificiel

Fonctions d'activations(seuillage)



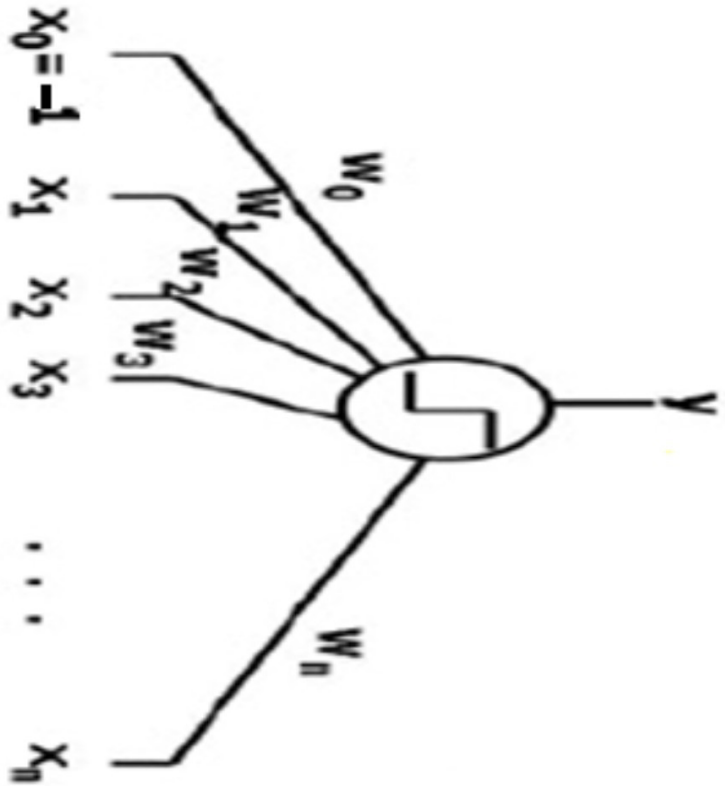
- (a) : seuil (fonction de Heavyside)
- (b) : sigmoïde $g(x) = (1 + e^{-\beta x})^{-1}$

La fonction d'activation g est conçue pour que l'unité soit « active » (proche de +1) quand les « bonnes » entrées sont données, et « inactive » (proche de 0) quand elles sont « mauvaises ».

Forme Standard Utilisé (Neurone à Biais)

On ajoute une entrée supplémentaire x0 (le biais), avec le coefficient synaptique suivant : w0 = θ

Coefficient de biais w0 connecté à une entrée x0=-1



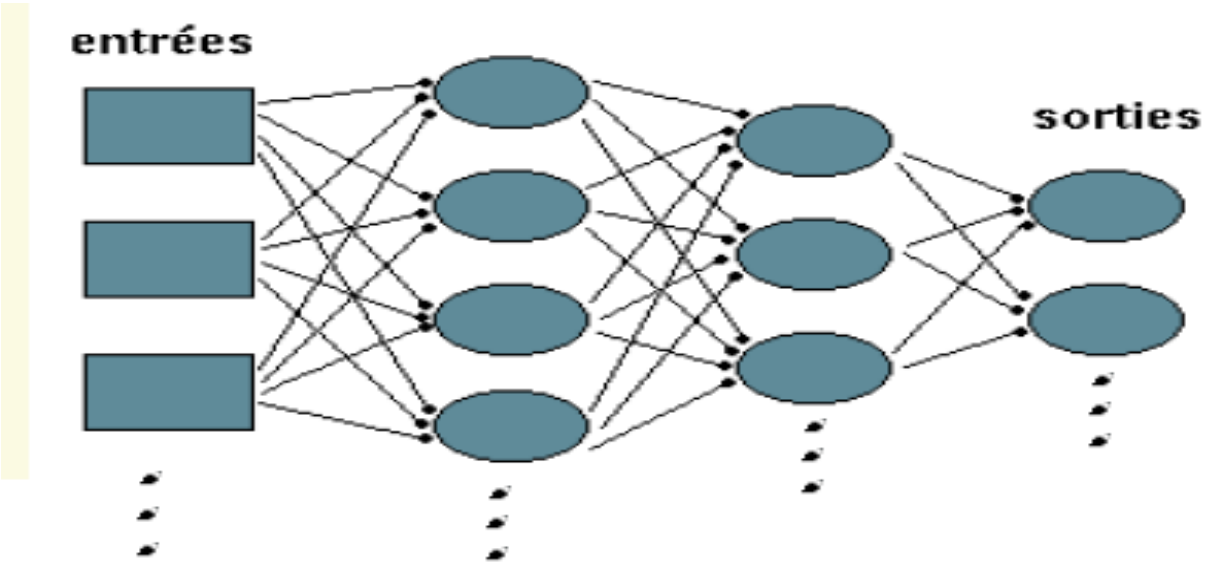
Définitions

• Déterminer un réseau de neurones = Trouver les coefficients synaptiques. On parle de phase d'***apprentissage : les caractéristiques du réseau*** sont modifiées jusqu'à ce que le comportement désiré soit obtenu.

• ***Base d'apprentissage : exemples représentatifs du comportement ou*** de la fonction à modéliser. Ces exemples sont sous la forme de couples (entrée ; sortie) connus.

• ***Base d'essai : pour une entrée quelconque (bruitée ou incomplète),*** calculer la sortie. On peut alors évaluer la performance du réseau.

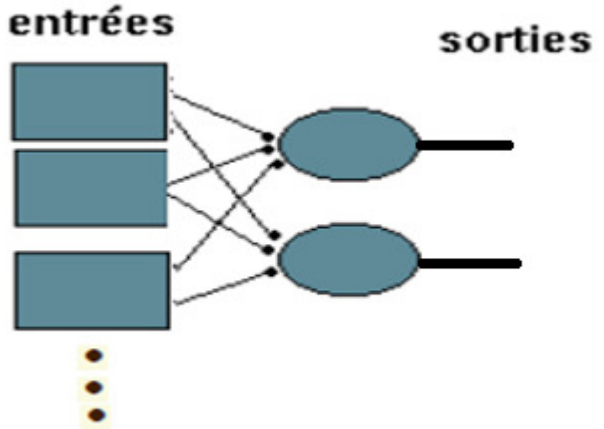
Structures des réseaux de neurones



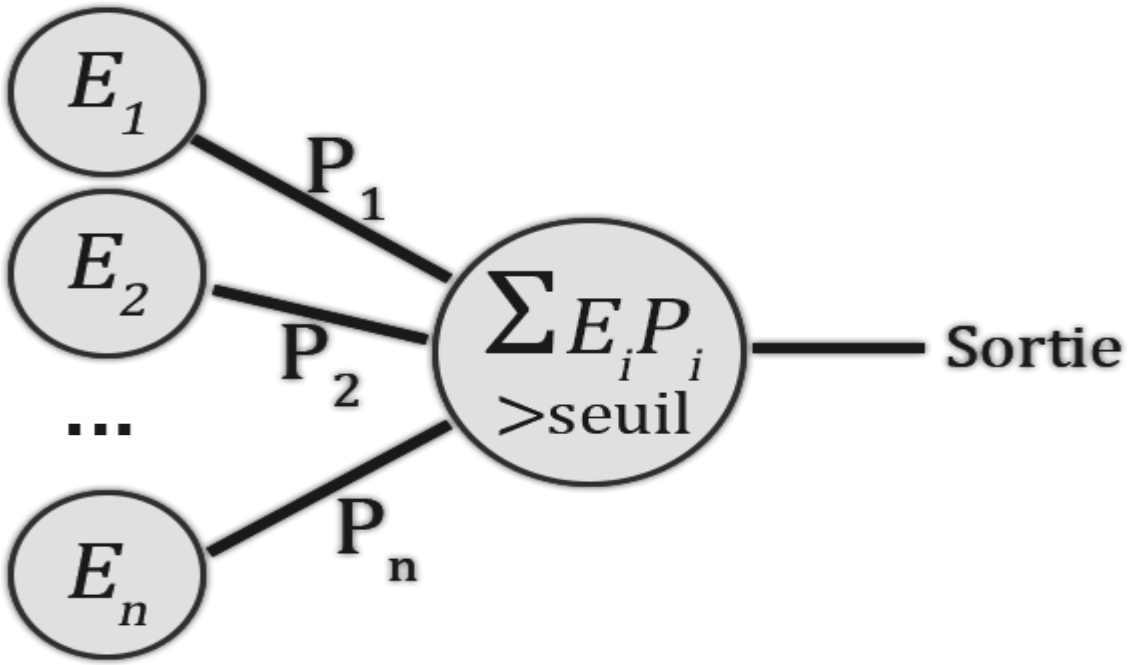
Réseaux de neurones monocouche
–Le perceptron–

Toutes les entrées sont directement connectées aux sorties

Schéma:



Comme chaque unité de sorties est indépendante des autres on limite notre étude à une sortie.



On distingue généralement deux types:
Perceptron à seuil et Perceptron basé sur la descente du gradient

Perceptron à seuil

Algorithm Perceptron à seuil:

1/ Initialisation des poids (y compris le bias) à des valeurs (petites) choisies au hasard.

2/ Présentation d'une entrée X = (x0,x1, … xn) de la base d'apprentissage.

3/ Calcul de la sortie obtenue y pour cette entrée :

$a = \sum(wi \cdot ei)$

y = signe (a) (si a > 0 alors y = +1 sinon alors y = -1)

4/ Si la sortie y du Perceptron est différente de la sortie désirée yd pour cet exemple d'entrée

alors modification des poids (μ le pas de modification choisi entre 0 et 1) :

$wi(t+1) = wi(t) + \mu.((Err).xi)$ où l'erreur : err= yd – y.

Ainsi de suite jusqu'à terminer l'ensemble d'exemples d'apprentissage avec une erreur nulle pour ***tous les exemples***, on retient alors les derniers poids ;

Considérant une fonction d'activation à ***seuil*** (fonction majorité) qui retourne 1 ssi

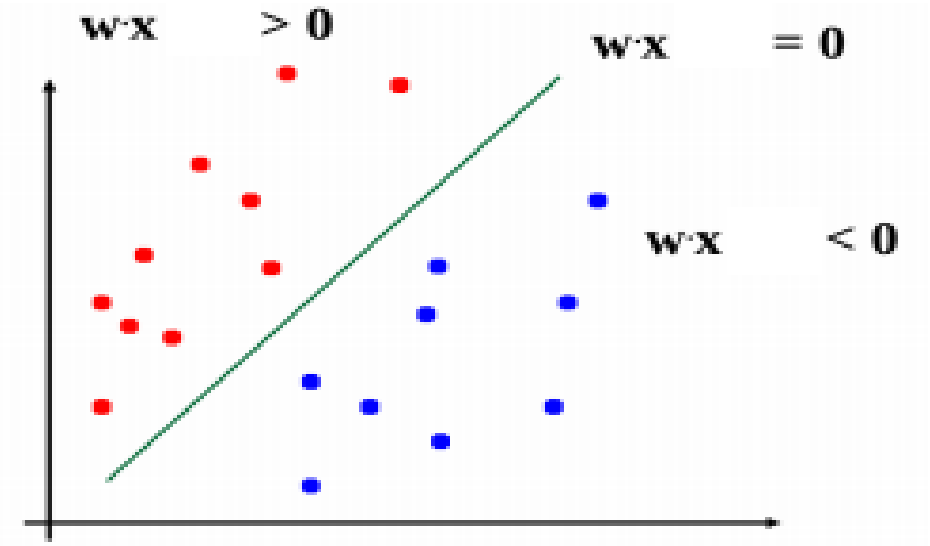
$$\sum_{j=0}^{n} wj\ xj > 0 \quad \text{ou} \quad W.X>0$$

Or: l'équation W.X=0 définit un hyperplan dans l'espace des entrées le perceptron ne retourne 1 que ssi l'entrée se trouve sur un côté de cet hyperplan

Déf: Si E est un espace vectoriel de dim n, alors tous sous espace H de dim n-1 s'appelle hyperplan de E

Càd: le perceptron ne retourne 1 que ssi l'entrée se trouve sur un coté de cet hyperplan

Perceptron à seuil = qualifié de séparateur linéaire

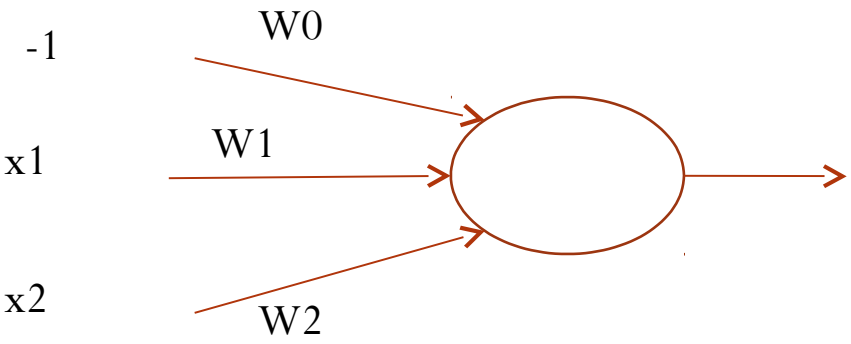


Exemple de fonctionnement de l'algorithme d'apprentissage du Perceptron à seuil:

Application aux portes logiques(AND, OR,…)

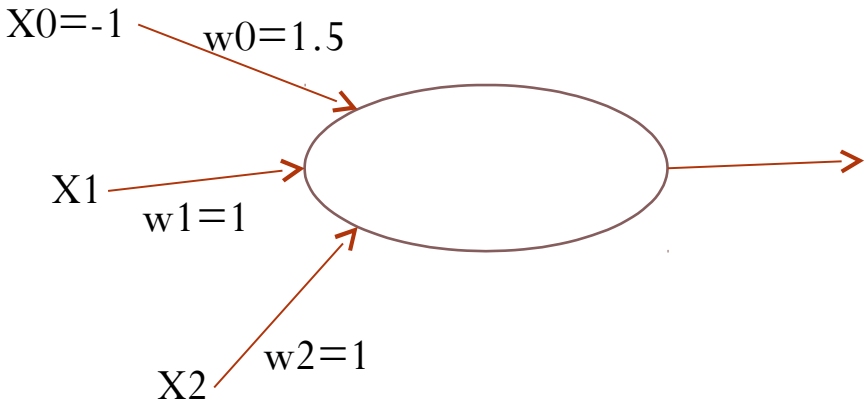
Base d'exemples d'apprentissage :X=(x1,x2), e=(X,yd)

x1	x2	yd	
1	1	1	e1
0	1	0	e2
1	0	0	e3
0	0	0	e4

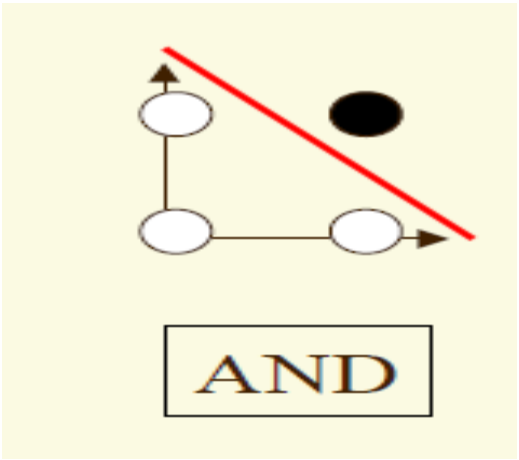


Exécuter l'algorithme à seuil sur ces exemples d'apprentissage :
Initialisation des poids: (1.3, 0.5, 0.5)

Exemple: perceptron à seuil de AND



x1	x2
1	1
0	1
1	0
0	0



Remarque: l'algorithme à seuil fonctionne dans le cas où Les données sont linéairement séparables

Dans le cas contraire exemple de xor, cet algorithme ne fonctionne pas.

Problème: ce n'est pas toutes les fonctions sont séparables

Exemple: la fonction XOR

Algorithme 1. Perceptron à sigmoïde

Algorithme d'apprentissage par rétropropagation du gradient pour les perceptrons, supposant une fonction d'activation différentiable :

entrées : *exemples*, un ensemble d'exemples, chacun avec l'entrée

$\mathbf{x} = x_1 \ldots x_n$ et la sortie y

réseau, un perceptron avec les poids W_j, $j = 0 \ldots n$,

et une fonction d'activation g

répéter

pour chaque e **dans** *exemples* **faire**

$$in \leftarrow \sum_{j=0}^{n} W_j x_j[e]$$

$$Err \leftarrow y[e] - g(in)$$

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j[e]$$

jusqu'à ce qu'un critère d'arrêt quelconque soit satisfait

Démonstration: Descente du gradient

$$g(x) = \frac{1}{(1+e^{-x})} \qquad g'(x) = g(x)(1-g(x))$$

Descente du gradient

$$W_j \leftarrow W_j - \alpha * \frac{\delta E}{\delta W_j}$$

Démonstration algorithme1:

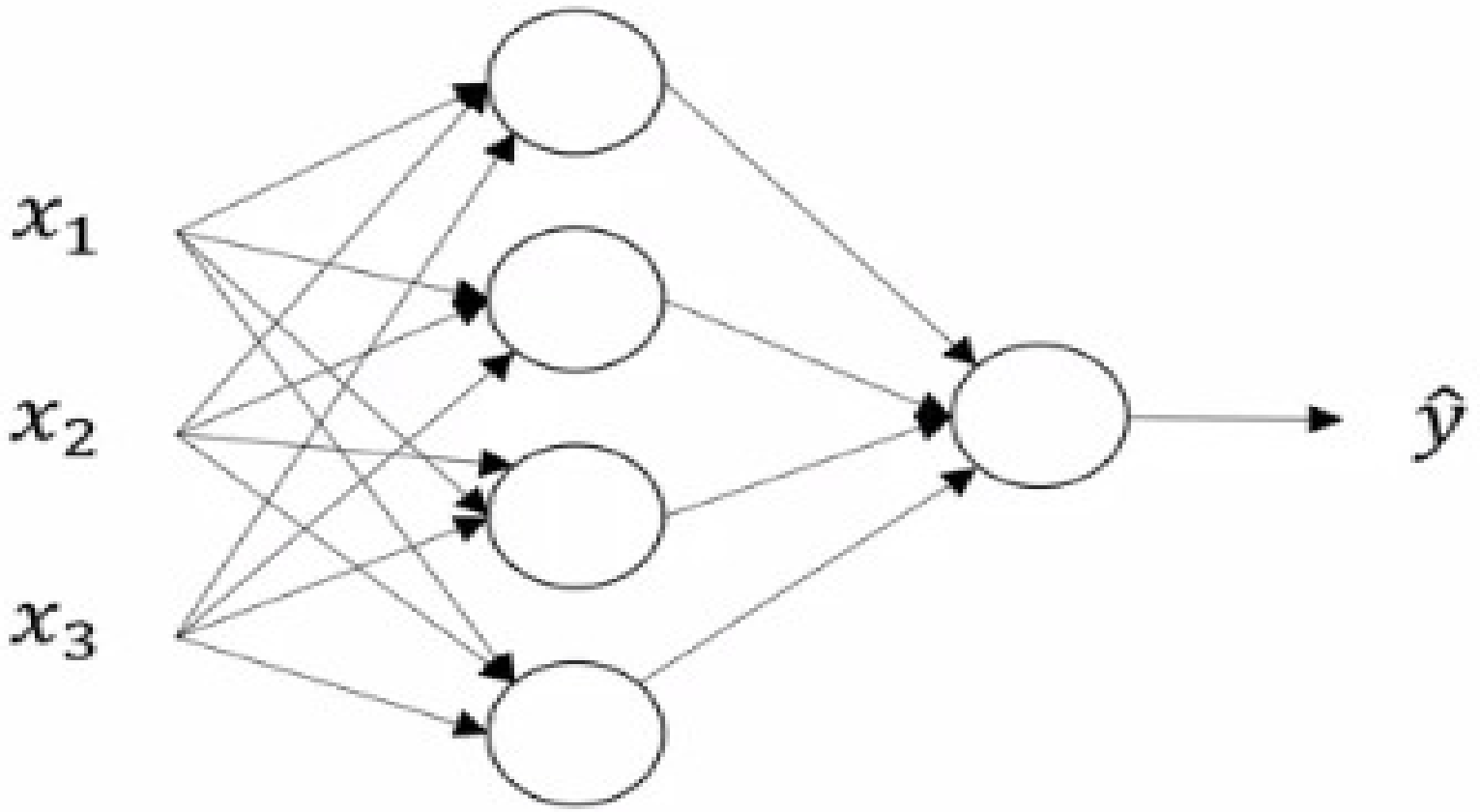
Voir démonstration en classe

Etapes de mise en œuvre d'un réseau de neurones pour la prédiction ou la classification

1. ***Identification des données en entrée et en sortie***
2. ***Normalisation des données (si nécessaire)***
3. ***Constitution de la structure du réseau à utiliser***
4. ***Apprentissage du réseau***
5. ***Dé normalisation des données en sortie (si nécessaire)***
6. ***Test du réseau***

TD -Réseaux de Neurones

Représentation des réseaux de neurones (MLP)
Et calcul de la sortie du réseau



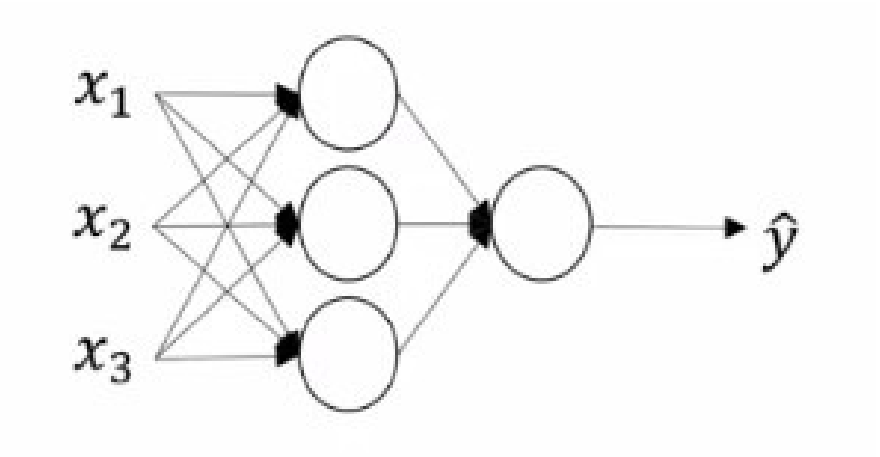
```
import numpy as np

a = np.array([1,2,3,4])
print(a)
```

```
[1 2 3 4]
```

```
import time
```

```
a = np.random.rand(1000000)
b = np.random.rand(1000000)

tic = time.time()
c = np.dot(a,b)
toc = time.time()

print(c)
print("Vectorized version:" + str(1000*(toc-tic)) +"ms")

c = 0
tic = time.time()
for i in range(1000000):
    c += a[i]*b[i]
toc = time.time()

print(c)
print("For loop:" + str(1000*(toc-tic)) + "ms")
```

Fonctions d’activations



Perceptron Multi-couches

1. **Définir la structure du MLP(taille de l'entrée, nombre de de neurones de la couche cachée, etc).**
2. **Initialiser les parameters du modèle**
3. **Boucle:**
   - **Implementer « forward propagation »**
   - **calculer la fonction du coût**
   - **Implementer « backward propagation » pour trouver les gradients**
   - **Adapter les parameters (gradient descent)**

Calcul du coût (cost):

$$J = -\frac{1}{m} \sum_{i=0}^{m} \left(y^{(i)} \log\left(a^{[2](i)}\right) + (1 - y^{(i)}) \log\left(1 - a^{[2](i)}\right) \right)$$

```
from sklearn.metrics import classification_report
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data,
iris.target)
mlp =
MLPClassifier(hidden_layer_sizes=(13,13,13),max_iter=500)
mlp.fit(X_train,y_train)
predictions = mlp.predict(X_test)
print(mlp.predict(X_test))
print(classification_report(y_test,predictions))
```

Système d'apprentissage (SVM)
SVM: Machines à vecteurs support
où: Séparateur à vaste marge

Parmi les modèles des SVM, on constate les cas:

- linéairement séparable
- les cas non linéairement séparable.

Classification linéaire binaire (linéairement séparable)

Apprentissage du perceptron

$$X \subset \mathbb{R}^n,\ Y = \{-1, 1\}$$

Définition: Un classifieur linéaire est une fonction de décision de la forme:

$$f(\mathbf{x}) = \begin{cases} 1 \text{ si } \langle \mathbf{w}, \mathbf{x} \rangle + b \geq 0 \\ -1 \text{ sinon.} \end{cases}$$

Où $\mathbf{w} \in \mathbb{R}^n$,

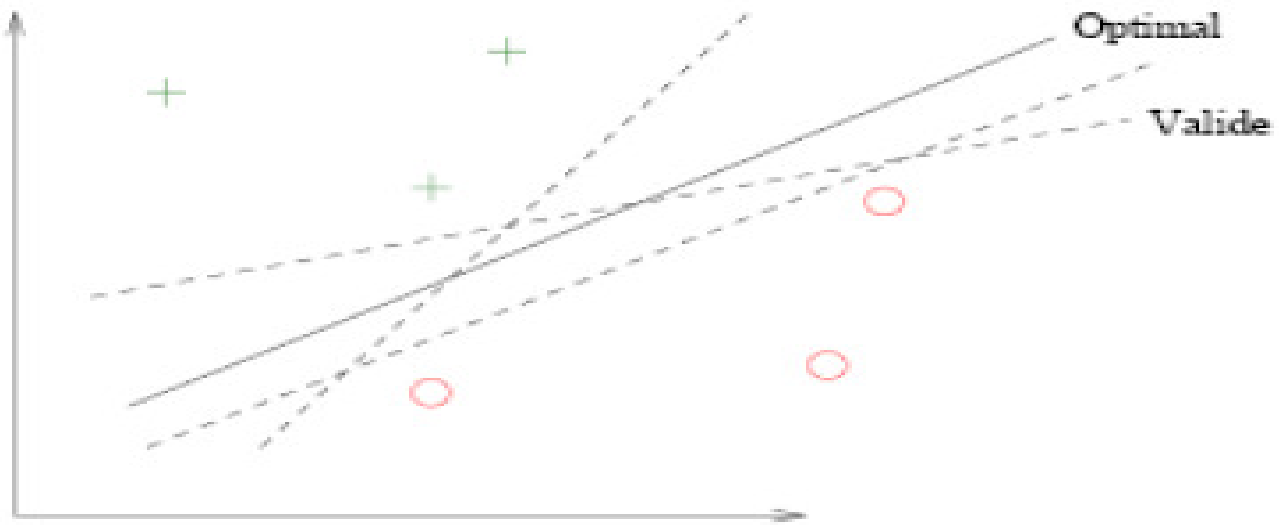
:

$b \in \mathbb{R}$, $\langle \mathbf{w}, \mathbf{x} \rangle$ désigne le produit scalaire entre w et x :

si $\mathbf{w} = (w_1, \ldots, w_n)$ et $\mathbf{x} = (x_1, \ldots, x_n)$, $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=1}^{n} w_i x_i$.

Interprétation géométrique : <w, x> + b = 0 est l''equation d'un hyperplan qui sépare X en deux demi-espaces correspondant aux deux classes

Il est évident **qu'il existe une multitude d'hyperplan** valide mais la propriété remarquable des SVM est que cet hyperplan doit être optimal.



Hyperplan ***optimal***.

- Le but de SVM est de trouver un classificateur ***optimal***(Pour deux classes d'exemples)qui va séparer les données.

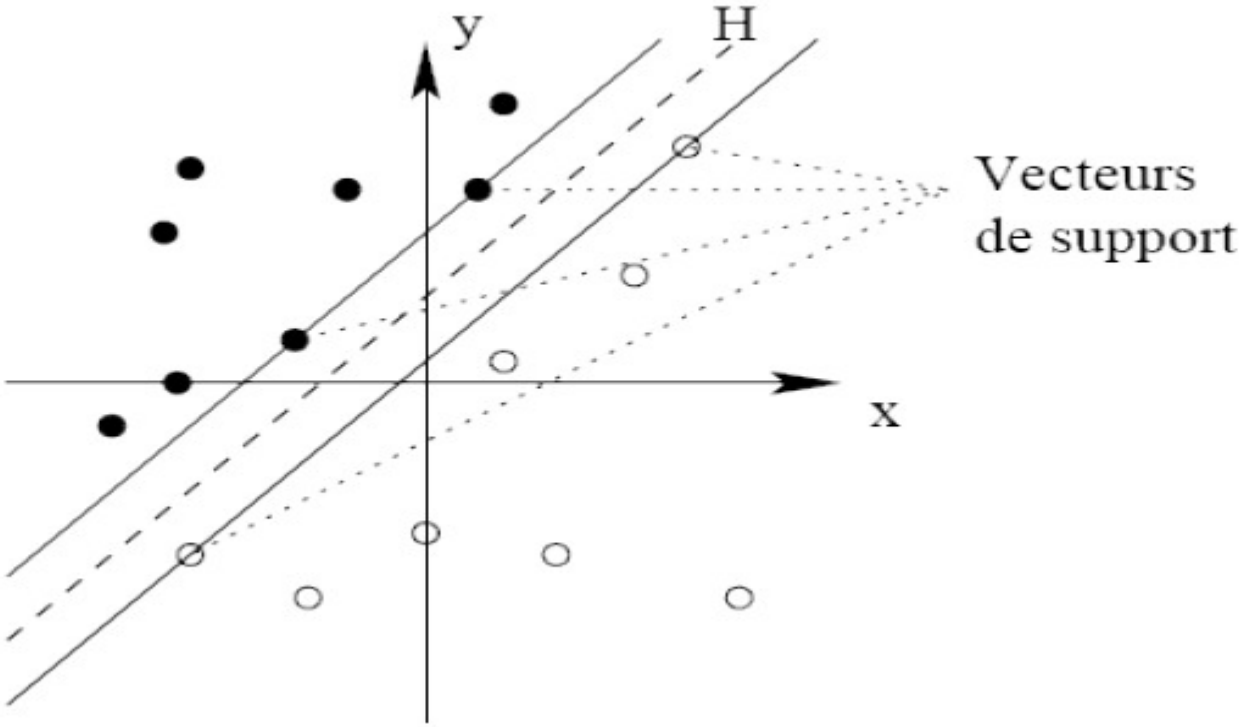
Hyperplan optimal?

Cela revient à chercher un hyperplan dont:

la distance aux exemples d'apprentissage est maximale. On appelle cette distance ≪ marge ≫

Formellement,. L'hyperplan séparateur optimal est celui qui maximise la marge.

Comme on cherche a maximiser cette marge, on parlera de *séparateurs à vaste marge*.



Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support

Hyperplan optimal

Soit S = $\{(x1, y1), \ldots, (xn, yn)\} \subset \mathbb{R}^n \times \{-1, 1\}$ un échantillon linéairement séparable.

Il existe un unique hyperplan de marge maximale qui est solution du problème d'optimisation quadratique sous contraintes linéaires suivant :

$$\begin{cases} f(x) = \langle w, x \rangle + b \\ y_i f(x_i) \geq 1 \text{ pour tout } i = 1 \ldots n \\ \text{Minimiser } ||w||^2. \end{cases}$$

Démonstration:

Soit S un échantillon linéairement séparable

et soit H un hyperplan séparateur, d'équation:

$H(x)=<w, x> + b = 0$.

Comme les données sont linéairement séparable alors:

Il n'existe aucun exemple sur l'hyperplan donc:

- Quelque soit i, H(xi) non nulle
- Donc il existe un réel a>0 telque:

Classe=1 si H(x)>=a

Classe=-1 si H(x)=<-a

Ceci revient en divisant les deux inégalités par a:(rH(x)=H(x))

Classe=1 si H(x)>=1

Classe=-1 si H(x)=<-1

Contrainte de minimisation

L'une des conditions des SVM est que tous les exemples d'apprentissage doivent être bien classées:

On aura alors la contrainte suivante:

yi(Wxi+b)>=1 i=1,..n

Cela signifie aussi: yiH(xi)>=1

Région de généralisation des SVM

C'est la région qui se trouve entre les deux hyperplans:
$wx+b=+1$ et $wx+b=-1$

L'objectif d'entrainement est de chercher un hyperplan optimal

La distance d'un point x à l'hyperplan est donnée par:

$d(x) = |w.x + b| / ||w||$

Soient x1 et x2 deux points de classes differentes (H(x1) = +1 et H(x2) = -1)

$(w.x1) + b = +1$

$et\ (w.x2) + b = -1$

$donc:\ (w.(x1 - x2)) = 2$

D'ou : $(w/||w||\ .\ (x1 - x2)) = 2/||w||$.

- On peut donc en déduire que maximiser la marge revient a minimiser $||w||$ sous certaines contraintes:

Hyperplan optimal

Soit S = $\{(x1, y1), \ldots, (xn, yn)\} \subset \mathbb{R}^n \times \{-1, 1\}$ un échantillon linéairement séparable.

Il existe un unique hyperplan de marge maximale qui est solution du problème d'optimisation quadratique sous contraintes linéaires (convexes) suivant :

$$\begin{cases} f(x) = \langle w, x \rangle + b \\ y_i f(x_i) \geq 1 \text{ pour tout } i = 1 \ldots n \\ \text{Minimiser } ||w||^2. \end{cases} \quad \textbf{(1)}$$

Remarque: On obtient le même hyperplan même si on supprime toutes les données Qui vérifie l'inégalité de la contrainte.
Les données qui vérifie l'égalité s'appellent les vecteurs supports (qui contribuent à la détermination de l'hyperplan)

Cas non linéairement séparable

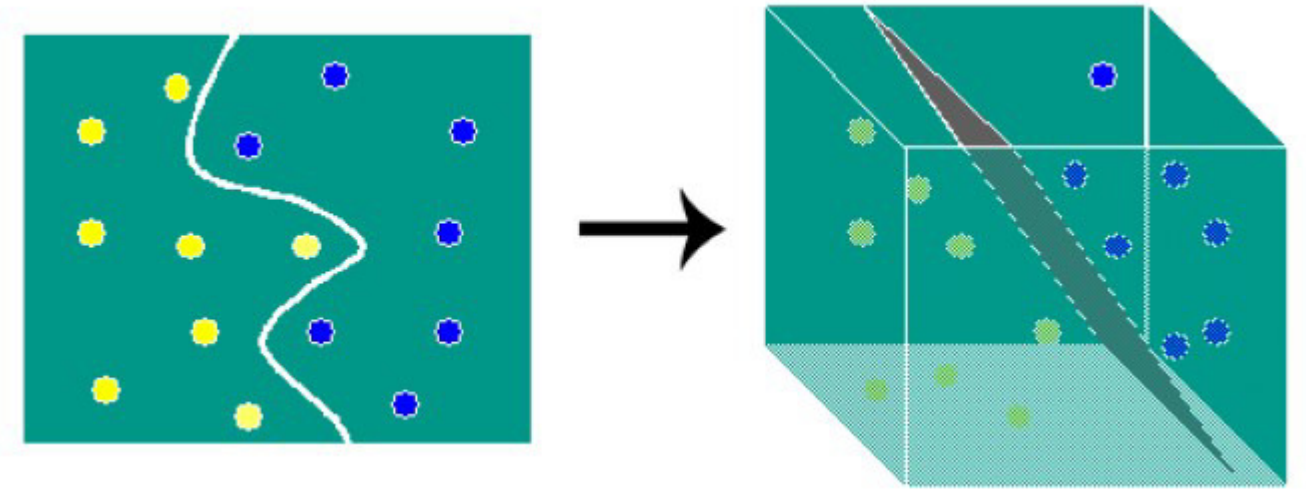
Utilisation des noyaux:

pour une représentation de dimension supérieure où les données sont linéairement séparables

Cas non linéaire

- L'idée des SVM est de ***changer l'espace des données***.
- La transformation non linéaire des données peut permettre une séparation linéaire des exemples dans un nouvel espace. On va donc avoir un changement de ***dimension***. Cette nouvelle dimension est appelé ≪ espace de re-description ≫.

En effet, intuitivement, plus la dimension de l'espace de re-description est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée. Ceci est illustré par le schéma suivant :



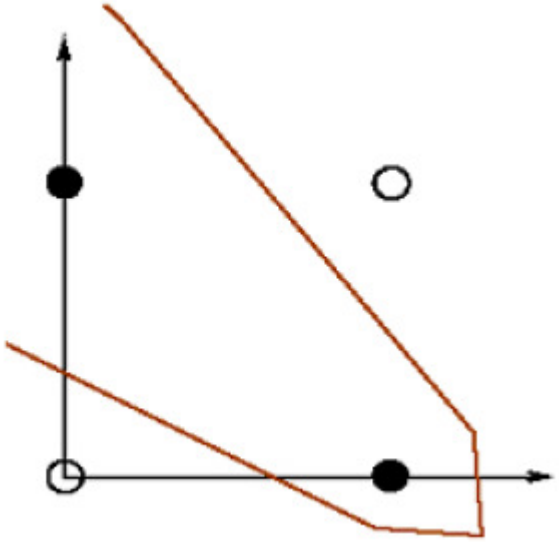
On a donc une transformation d'un problème de séparation non linéaire dans l'espace de représentation en un problème de séparation linéaire dans un espace ***de re-description de plus grande dimension***

Cette transformation non linéaire est réalisée *via* ***une fonction noyau***.

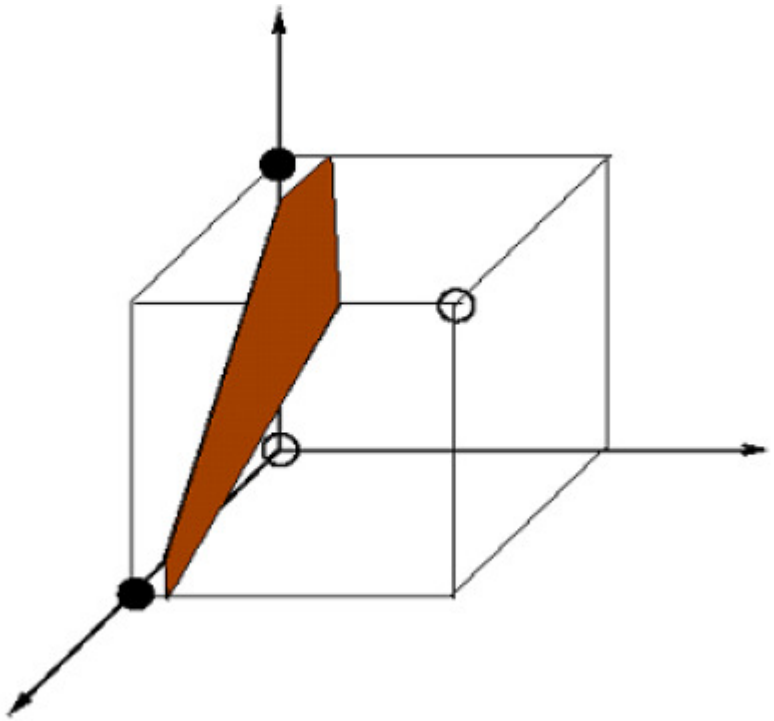
On peut citer les exemples de noyaux suivants : polynomiale, gaussien, sigmoide et laplacien.

transformation dans le cas XOR

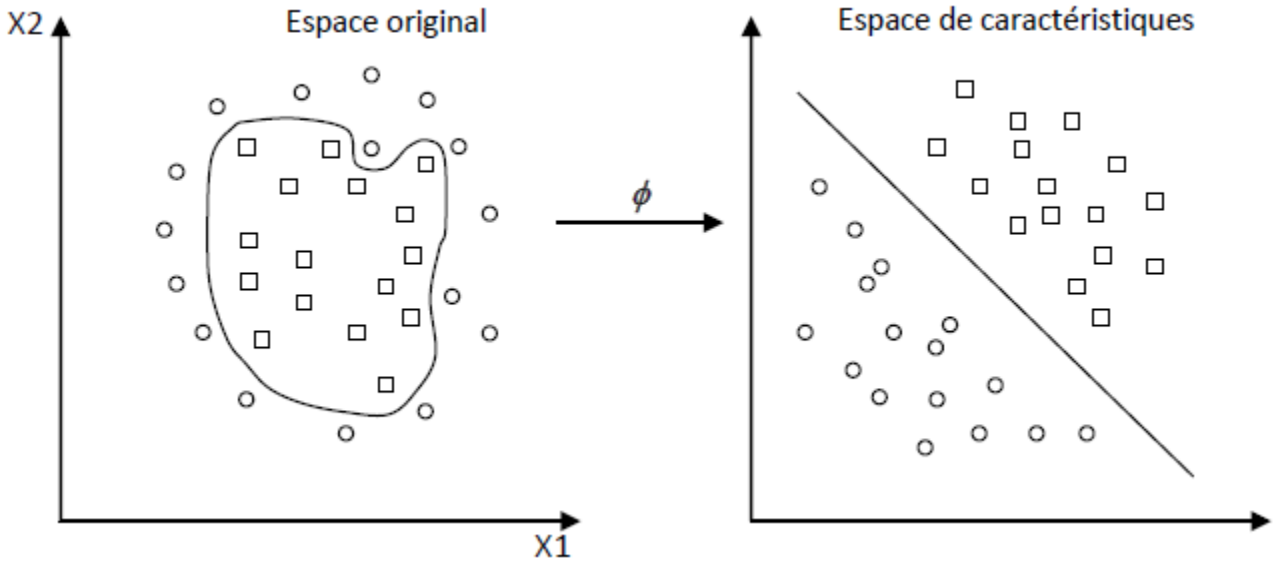
- Coordonnées des points : (0,0) ; (0,1) ; (1,0) ; (1,1)



- Si on prend une fonction polynomiale (x , y) -> (x , y , x.y) qui fait passer d'un espace de dimension 2 à un espace de dimension 3, on obtient un problème en trois dimensions linéairement séparable :
- (0,0) -> (0,0,0)
- (0,1) -> (0,1,0)
- (1,0) -> (1,0,0)
- (1,1) -> (1,1,1)



- Cette transformation d'espace est réalisée dans un nouvel espace appelé espace de caractéristiques "Features space". à l'aide d'une fonction de passage ("Mapping function"):



Noyau (Kernel)

Noyau: Kernel

$$K(x_i, x_j) = \begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_n) & \dots & K(x_n, x_n) \end{bmatrix}$$

- La fonction objective (4) peut être calculée comme suit:

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

Et la fonction de décision devient :

$$H(x) = \sum_{i \in S} \alpha_i y_i K(x_i, x) + b$$

NB/ Pour qu'une matrice (K) soit un noyau, il faut qu'elle respecte certaines conditions : être semi-définie positive (symétrique et n'a pas de valeurs propres négatives).

Exemples de noyaux

Noyau linéaire : Si les données sont linéairement séparables, le produit scalaire suffit pour définir la fonction de décision :

$$K(x_i, x_j) = x_i^T x_j$$

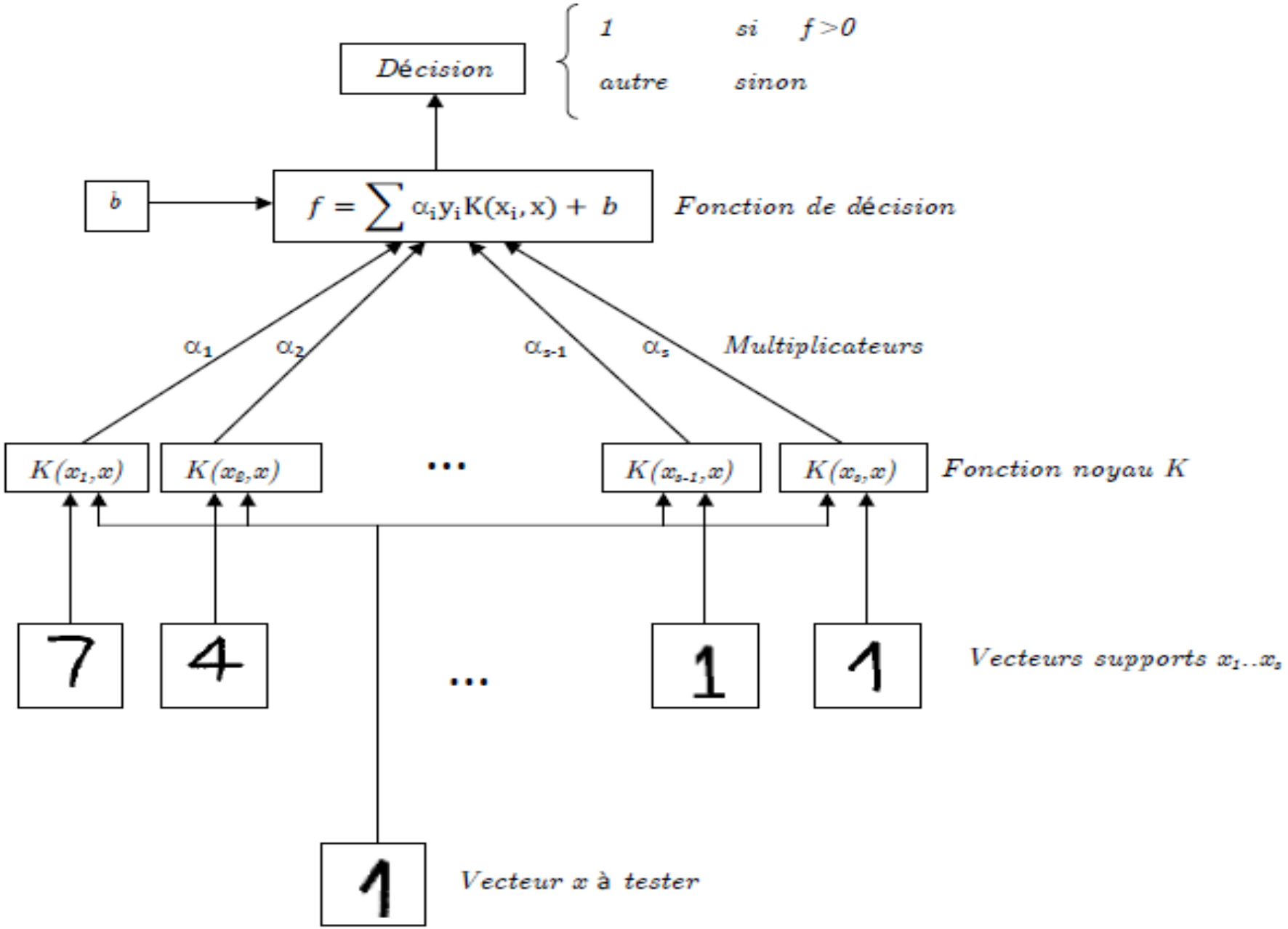
Noyau polynomial : Le noyau polynomial élève le produit scalaire à une puissance naturelle d:

$$K(x_i, x_j) = (x_i^T x_j)^d$$

Noyau RBF; Les noyaux RBF (Radial Basis functions) sont des noyaux qui peuvent êt écrits sous la forme:

$$K(x_i, x_j) = e^{\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}$$

Architecture générale d'une machine à vecteur support



TP. (sous python scikit learn)

Cas linéairement séparable:

LinearSVC

Cas non linéaire

SVC

TP. Cas linéairement séparable: sklearn.svm.LinearSVC

Exercice 1.

Produire un programme Python qui permet d'apprendre par SVM le modèle suivant:

X = [[1,0], [0,1],[0,0],[1,1]]

y = [0,0,0,1]

Tester la prédiction de quelques données

Afficher le taux de précision du modèle appris.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
cl= svm.LinearSVC()
X = [[1,0], [0,1],[0,0],[1,1]]
y = [0,0,0,1]
cl.fit(X,y)
print(cl.predict([1,1]))
```

Produire un programme Python qui permet d'apprendre par SVM la fonction non linéairement séparable XOR:

Exercice 2

1) Produire un programme Python qui charge le jeu de données digits correspondants aux chiffres manuscrits:
D= datasets.load_digits()
2) Affiche les données d'apprentissage (stockées dans le champ data de D) et les données cibles (stockées dans le champ target de D).
3) Divisez les dponnées en données d'apprentissage et données de test
3) Réalisez l'apprentissage sur toute les données d'apprentisaage
.
4) Calculer le taux de précision du modèle appris

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
digits = datasets.load_digits()
print(digits.data)
print(digits.target)
clf = svm.SVC()
#clf = svm.SVC(gamma=0.01, C=100)
X,y = digits.data[:-10], digits.target[:-10]
clf.fit(X,y)
print(clf.predict(digits.data[-5]))
plt.imshow(digits.images[-5])
#plt.imshow(digits.images[-5], cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()
```

performance du classifieur

```
from sklearn import cross_validation
sc=cross_validation.cross_val_score(clf,digits.data,digits.target,cv=4)
print(sc)
print(sc.mean())
```

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
digits = datasets.load_digits()
print(digits.data)
print(digits.target)
clf = svm.SVC()
#clf = svm.SVC(gamma=0.01, C=100)
X,y = digits.data, digits.target
clf.fit(X,y)
print(clf.predict(digits.data[1]))
plt.imshow(digits.images[1], cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()
```