



INTELLIGENCE ARTIFICIELLE POUR L'INGENIERIE

Génie Industriel

Dr Hanane ZERMANE

Maitre de conférences B

COURS ET TRAVAUX DIRIGES

Intelligence Artificielle pour l'Ingénierie

POUR LES CYCLES DE MASTER EN

Génie industriel

Management industriel

Table de matières

Introduction	3
Programme proposé.....	3
Objectif du cours	4
Les prérequis	4
Chapitre 1 : Introduction à L'intelligence Artificielle	5
1.1. Introduction	5
1.2. Définitions de l'IA.....	5
1.3. Historique	6
1.4. Domaines d'application.....	8
1.5. Le raisonnement dans l'IA	8
1.6. Quelques techniques de l'IA.....	9
1.7. Conclusion.....	9
Chapitre 2 : Les systèmes experts	10
2.1. Introduction	10
2.2. Définitions	10
2.3. Rôle des Systèmes Experts	10
2.4. Composantes d'un système expert	12
2.4.1. La base de connaissances	12
2.4.2. Le moteur d'inférence	14
2.5. Modes de raisonnement	16
2.5.1. Chainage avant	16
2.5.2. Chainage arrière	16
2.5.3. Chainage mixte.....	17
2.6. Comment Construire un Système Expert	17
2.7. Exercices corrigés.....	19
2.8. Exercices proposés	21
2.9. Conclusion	22
Chapitre 3 : La logique floue.....	23
3.1. Introduction	23
3.1.1. Origine.....	23
3.1.2. Définition de la logique floue.....	24
3.1.3. Historique d'application	25
3.2. Les ensembles classiques.....	25
3.2.1. Les ensembles classiques : rappel	25

3.2.2.	Limite des ensembles classiques	26
3.3.	Les ensembles flous	27
3.3.1.	Définitions	27
3.3.2.	Opérations sur les ensembles flous	31
3.4.	Création d'un système de contrôle flou	33
3.4.1.	La fuzzification	34
3.4.2.	L'inférence floue	36
3.4.3.	L'agrégation et l'activation des règles	36
3.4.4.	La défuzzification.....	39
3.5.	Exercices corrigés.....	40
3.6.	Conclusion.....	44
Chapitre 4 : Les approches d'apprentissage		45
4.1.	Introduction	45
4.2.	Historique des réseaux de neurones artificiels	46
4.3.	Le neurone formel	48
4.4.	Perceptron (premier réseau de neurones)	49
4.4.1.	Types d'apprentissage	50
4.4.2.	Règle d'apprentissage	51
4.4.3.	Exemple (ET, OU et XOR)	52
4.4.4.	Limitation du perceptron.....	53
4.5.	Réseau de neurones multicouches	54
4.5.1.	Perceptron multicouches (MLP)	56
4.5.2.	Réseaux à fonction radiales de bases (RBF)	57
4.6.	SVM (Machine à Vecteur de Support)	58
4.6.1.	Définition	58
4.6.2.	Principe.....	59
4.6.3.	Fonction du noyau (kernel)	60
4.6.4.	Extensions aux cas multi-classe	61
4.6.5.	Exemple de classification.....	61
4.7.	Exercices corrigés.....	62
4.8.	Conclusion.....	67
Références Bibliographique		68
Références bibliographiques proposées		68

Introduction

Bien que les ordinateurs soient aujourd'hui capables de résoudre de nombreux problèmes, il reste encore des domaines où les humains ont plus de facilité que les machines. Le principe de l'intelligence artificielle (IA) est d'imiter l'intelligence humaine, afin de construire des machines plus puissantes et plus pratiques.

Dans ce contexte nous allons présenter ce document, qui est composé de quatre chapitres et basé sur le programme proposé pour les étudiants de Master en Génie Industriel.

Programme proposé

L'objectif de ce module est de présenter les techniques de l'intelligence artificielle et les domaines où elles sont utilisées, et ce dans le but d'offrir un panorama de cette discipline.

Parmi les prérequis et les connaissances préalables, on recommande : les systèmes formels, la logique des propositions et la logique des prédicats. Le programme proposé contient les chapitres suivants :

Chapitre 1 : Introduction à l'intelligence Artificielle	(3 semaines)
Chapitre 2 : La Logique des prédicats et Systèmes Experts	(4 semaines)
Chapitre 3 : La Logique Floue	(3 semaines)
Chapitre 4 : L'Apprentissage Automatique (K-means, K-ppv, RNA, SVM)	(4 semaines)

Les différents travaux pratiques proposés dans le cadre de ce cours sont présentés comme suit :

TP 1 : Système expert

L'objectif du premier TP est l'intégration du système expert dans le contrôle d'un système de production.

TP 2 : Logique floue

L'objectif du deuxième TP est l'application de la logique floue pour la création d'un contrôleur flou utilisant les boîtes à outils de Matlab, Fuzzy Toolbox et Simulink.

TP 3 : Apprentissage Automatique

L'objectif du troisième TP est l'utilisation des techniques K-means, K-ppv, RNA et SVM pour la classification utilisant Python.

Objectif du cours

Le but de ce cours est de présenter une initiation à l'Intelligence Artificielle (IA) essentiellement :

- ✓ Introduction à l'intelligence artificielle, ces différentes définitions, son historique, ces domaines d'application et son développement actuel.
- ✓ Présentation des techniques de l'IA : Système expert, la logique floue, les réseaux de neurones artificiels et SVM.

Les prérequis

- ✓ Les systèmes formels : dont la compréhension est un point central pour tout travail sur la pensée et les mathématiques.
- ✓ La logique des propositions et des prédicats : constitue la seule théorie mathématiquement développée et formulée du savoir, du raisonnement et de la vérité et qui donc est un outil de base pour la représentation et la manipulation des connaissances en général.

Chapitre 1 : Introduction à L'intelligence Artificielle

1.1. Introduction

L'objet de l'IA est de reconstituer des raisonnements et des actions intelligentes, à l'aide de moyens artificiels, des ordinateurs ou des machines intelligentes. Dans l'industrie, pour optimiser et évoluer le contrôle et la supervision des processus, le recours à ces techniques de l'IA est l'une des solutions efficaces. Parmi elles, les systèmes experts, la logique floue, etc. Les techniques de l'IA s'appliquent pour la surveillance des systèmes industriels et se présentent comme des méthodes utilisées en l'absence du modèle du processus ou dite des modèles symboliques.

Dans tous les secteurs d'activité, les techniques de l'IA tendent à élargir le champ d'action des machines, pour leur donner la possibilité de voir, d'entendre, de raisonner, d'agir, etc. Les systèmes de ces machines possèdent des caractéristiques associées avec l'intelligence dans le comportement humain.

L'IA a prouvé son importance, en essayant de résoudre les différents problèmes pouvant être engendrés. Ceci peut nous aider à prendre les décisions qui assurent la bonne exécution du système sujet de contrôle ou d'optimisation.

L'importance de l'intelligence artificielle se présente quand par exemple :

- La connaissance algorithmique peut être impossible à mettre en œuvre : problème d'explosion combinatoire. (jeu échec, go (75 milliards de possibilités (5 fois jeux d'échec), ...)
Jeu d'échec : en moyenne 20 coups par étapes et 50 échanges dans une partie => 20^{50} situations à explorer !
- Il n'existe pas de solution algorithmique connue : Traitement de la langue (parole).
- Les spécialistes d'un domaine font appel à des connaissances implicites (tacites) basées sur leur savoir-faire (Les heuristiques).

1.2. Définitions de l'IA

Plusieurs définitions ont été attribuées à l'intelligence Artificielle, parmi celle-ci (Boisard 2014) :

Définition 1 : « Si l'informatique est la science du traitement de l'information, l'IA s'intéresse à tous les cas où ce traitement ne peut être ramené à une méthode simple, précise, algorithmique » (Laurière 87).

Définition 2 : L'IA est « l'application de logiciels et de techniques de programmation mettant en lumière les principes de l'intelligence en général, et de la pensée humaine en particulier » (Boden 1977).

Définition 3 : « L'étude de l'intelligence, indépendamment de sa manifestation chez l'homme, l'animal ou la machine » (McCarthy)

Définition 4 : « La poursuite de la métaphysique par d'autres moyens » (Longuet-Higgins).

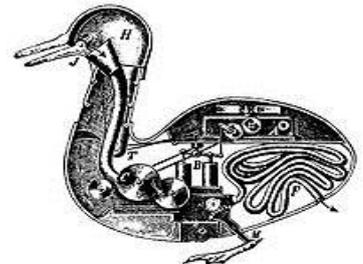
Définition 5 : « La science s'intéressant aux machines qui réalisent ce qui, fait par l'homme, nécessiterait de l'intelligence » (Minsky 1986).

Définition 6 : « L'intelligence artificielle peut être définie comme la tentative d'obtenir des machines comme celles qu'on voit dans les films. » (Russell Beale).

1.3. Historique

L'intelligence artificielle, s'est développée en parallèle avec le calcul numérique et non en concurrence avec celui-ci. Ce développement a passé par plusieurs étapes, depuis des siècles. Parmi ces développements, nous citons :

Le canard de Vaucanson : est un automate créé par Jacques de Vaucanson en 1738. Ce canard artificiel de cuivre doré boit, mange, cancanne, barbote et digère comme un vrai canard. Les ailes étaient représentées, os par os, d'un mécanisme identique à ceux des vrais oiseaux. Il était possible de programmer les mouvements de cet automate, grâce à des pignons placés sur un cylindre gravé, qui contrôlaient des baguettes traversant les pattes du canard. Le mécanisme, placé dans l'imposant piédestal, visible par tous, dans le but de montrer la complexité du travail accompli.



Le test de Turing : en 1950, le mathématicien Anglais Alan Turing publiait un article concernant l'IA. Il considérait la question : « Les machines peuvent-elles penser ? ». Le test de Turing, précurseur de l'intelligence artificielle, était fondé sur cette idée :

« on suppose qu'un individu et une machine peuvent indifféremment répondre à des questions posées par une deuxième personne située à une certaine distance et qui n'a aucun moyen matériel de savoir qui, de la machine ou de l'homme, lui répond ; »

Si cette personne ne devine pas que son interlocuteur est un ordinateur, on pourra dire que celui-ci est doté d'intelligence. Évidemment, toutes les réalisations actuelles sont bien loin de pouvoir passer ce test. La réponse proposée par Turing dans son article (1950) est :

« Oui, je crois qu'il sera possible, dans une cinquantaine d'années, de programmer des ordinateurs pour les faire jouer au jeu de l'imitation. L'ordinateur aura plus de 70% de chance de procéder à l'identification exacte après 5 minutes d'interrogation ».

Les années 1950 et 1960 : il faut attendre les années 1930 pour que certains mathématiciens fassent progresser la logique et les années 1950 pour que les calculateurs investissent le domaine civil. Ces deux outils rassemblés vont provoquer la naissance de l'IA, terme introduit en 1954 par McCarthy. C'est à cette époque que naissent parallèlement le calcul scientifique (manipulation de nombres) et le calcul formel (manipulation de symboles). A ses débuts, l'IA s'est caractérisée par la description exhaustive de problèmes simples et généraux. On compte sur les capacités de mémoire des ordinateurs et leur capacité de calcul pour engendrer beaucoup de solutions, les comparer et choisir la meilleure. Cela a conduit rapidement à l'explosion combinatoire, même si la puissance des machines a été démultipliée en très peu de temps.

Les années 1970 : voient naître le concept de système expert. C'est un système dédié à la résolution d'un problème très particulier, grâce à la représentation des connaissances spécialisées du domaine. L'intelligence devient alors synonyme de prise de décision dans un contexte difficile, en gérant des incertitudes ou des informations incomplètes, voire contradictoires. Ceci, on le résume par expertise, par opposition aux connaissances simples et rigoureuses que l'on traitait dans les premières applications.

Les années 1980 et 1990 : les années 1980 sont les années de la pénétration industrielle. Les Japonais lancent leur projet d'ordinateurs de la 5^{ème} génération. Ainsi, le langage PROLOG, implémentant directement des inférences logiques, se développe et se répand rapidement.

Nos jours :

Véhicules autonomes, Machine Learning, Deep Learning, ...

Durant toute cette période, l'intelligence artificielle est intégrée presque dans tous les domaines. Son application aujourd'hui s'élargie surtout dans le domaine industriel.

1.4. Domaines d'application

L'intelligence artificielle s'applique dans de différents secteurs et domaines. Nous pouvons citer les principaux :

- Le calcul formel,
- La vision artificielle : analyse de texte, d'images,
- La robotique : génération de plans,
- Les machines autonomes : perception, interprétation, décision, action,
- Le langage naturel : traduction, compréhension, synthèse,
- La démonstration de théorèmes,
- Les jeux,
- La représentation des connaissances dans diverses disciplines,
- Médecine,
- Automatisme,
- L'apprentissage automatique (RNA, Deep learning, ...).



L'application de l'intelligence artificielle dans ces différents domaines a prouvé son utilité. Ceci n'est possible qu'à base des différents raisonnements de l'IA.

1.5. Le raisonnement dans l'IA

On rencontre quelques catégories de raisonnement génériques, adaptées à des applications dans n'importe quel domaine :

- Le raisonnement déductif (rendre explicites des connaissances implicites),
- Le raisonnement abductif (rechercher des causes plausibles à des manifestations),
- Le raisonnement inductif (construire des lois, des concepts à partir d'exemples),
- Le raisonnement analogique (rapprochement entre situations particulières),
- Le raisonnement hypothétique ou par défaut (pallier le manque d'information par des hypothèses explicites ou implicites).

Il existe d'autres catégories de raisonnement spécifiques, adaptées à des applications bien particulières :

- ✓ Le raisonnement temporel (gérer le temps pour la planification ou la simulation d'événements datés, conduite d'ateliers de production, par exemple),
- ✓ Le raisonnement spatial et géométrique (pour la robotique, par exemple),

- ✓ Le raisonnement causal (propager des causes pour en trouver les effets ou encore remonter de l'observation d'un effet à sa cause possible),
- ✓ Le raisonnement qualitatif (décrire le comportement de systèmes physiques à l'aide d'ordres de grandeurs symboliques).

1.6. Quelques techniques de l'IA

Dans tous les secteurs d'activité, les techniques de l'IA tendent à élargir le champ d'action des machines, en leur donnant la possibilité de voir, d'entendre, de raisonner, de parler, d'agir, etc. C'est-à-dire des systèmes qui possèdent des caractéristiques associées avec l'intelligence dans le comportement humain :

- ✓ Le langage de compréhension,
- ✓ L'apprentissage,
- ✓ Le raisonnement,
- ✓ Les solutions des problèmes, etc.

Répondant aux questions :

- Comment le cerveau humain fonctionne-t-il ?
- Est-ce qu'on peut extraire l'intelligence humaine vers des machines ?
- Des machines peuvent-elles vraiment être intelligentes ?

1.7. Conclusion

L'IA est un ensemble de réalisations et de recherches qui essaient d'imiter artificiellement les performances humaines. Donc, on essaie d'obtenir un comportement intelligent avec des ordinateurs, par le biais des différentes techniques de l'intelligence artificielle. Dans le cadre de ce cours, nous allons voir les techniques suivantes :

- ✓ Les Systèmes Experts,
- ✓ La Logique Floue,
- ✓ Les techniques d'apprentissage automatique supervisé et non supervisé (reconnaissance de formes, régression ou classification) : K-means, K-ppv, Réseau de Neurones Artificiels et SVM.

Chapitre 2 : Les systèmes experts

2.1. Introduction

L'IA est sans conteste, une des aventures intellectuelles de la fin du vingtième siècle et le début de notre siècle, de telle sorte que, les machines se comportent comme des humains. Ce qui nous intéresse, c'est le raisonnement dans l'IA. Est-ce qu'on peut vraiment réaliser des systèmes ou des machines qui peuvent raisonner pour faire le même travail voulu ? Pour répondre à cette question, les systèmes experts sont indispensables, ceux sont eux qui peuvent aider ces machines à exploiter les connaissances acquises, et donner les réponses aux requêtes posées par les utilisateurs, dans un domaine spécifié.

Hypothèse :

Est-ce qu'on peut réaliser des systèmes ou des machines qui peuvent raisonner afin de faire le même travail désiré qu'un être humain ?

Réponses :

- Les systèmes experts : peuvent aider ces machines à exploiter les connaissances acquises, et donner les réponses aux requêtes posées par les utilisateurs.
- La méthodologie des systèmes experts soulève de nombreux espoirs, et semble s'accorder avec le développement accéléré de l'informatique, et la complexité des problèmes et des systèmes existants.
- Les systèmes experts viennent d'exploiter les connaissances, ainsi que les expériences des experts pour résoudre ou essayer d'englober le tout d'un problème.

2.2. Définitions

Définition 1 : un système expert peut être défini de deux façons. La première est fonctionnelle, pour décrire la fonction du système, qu'est-ce qu'il fait ou qu'il devrait faire. La seconde, est technique, s'intéresse au système comment est-il construit. Si on combine ces deux formules : «Un système expert est un programme conçu pour simuler le comportement d'un humain qui est un spécialiste ou un expert dans un domaine très restreint » (Denning 1986).

Définition 2 : « un ensemble de programmes capable de reproduire la démarche d'un expert humain confronté à un problème dans son domaine de compétence » (Denis 2006).

2.3. Rôle des Systèmes Experts

Lorsque le processus intellectuel par lequel un humain évalue une situation, ou prend une décision est précisément modélisé, il est relativement facile de le programmer, c'est le cas par

exemple, dans des domaines tels que la comptabilité, le calcul scientifique ou la commande numérique de machines-outils. Les systèmes experts se sont développés comme une technique informatique visant à atteindre trois objectifs :

- Capturer aisément les unités de savoir-faire : pour faciliter l'expression la plus directe possible des règles, par rapport à leur forme d'émergence chez les experts.
- Exploiter l'ensemble des unités de savoir-faire : donc combiner et/ou chainer des groupes de règles pour inférer (ou dériver) des connaissances telles que des jugements, plans, preuves, décisions, prédictions, nouvelles règles, Et rendre compte de la manière dont les nouvelles connaissances ont été inférées.
- Supporter aisément la révision de l'ensemble des unités de savoir-faire : c'est d'offrir des facilités pour les ajouts et les suppressions de règles.

Le schéma d'un système expert est représenté par la figure 2.1.

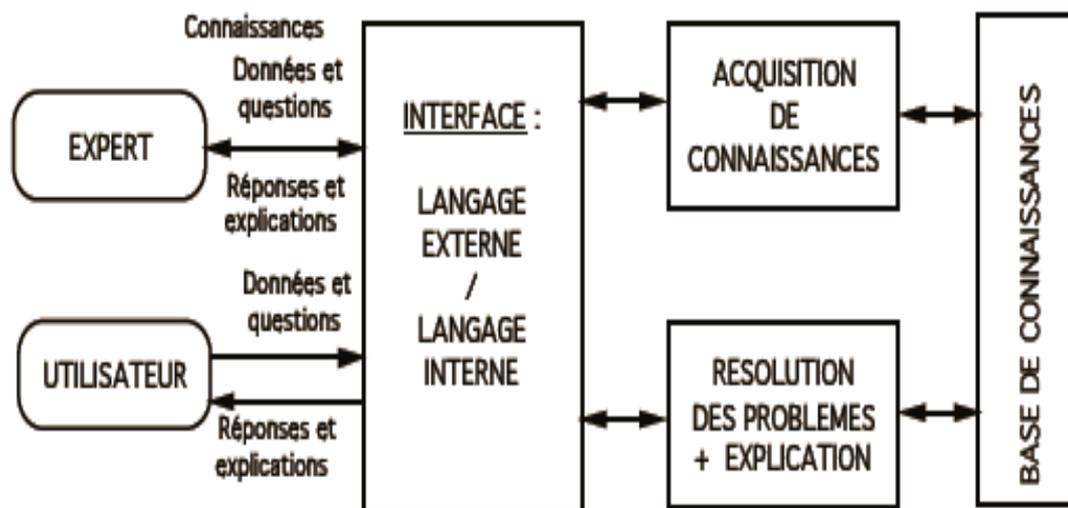


Figure 2.1. Schéma d'un Système Expert.

Il existe plusieurs systèmes experts créés et appliqués dans plusieurs domaines, médical, industriel, chimie, Parmi eux :

- **SEDIAG, DIAGNEX, MAINTEX, SOLVEUR/AMIDEC** : des systèmes experts pour le diagnostic.
- **DENDRAL** : 1969, chimie, recherche la formule développée d'un corps organique à partir de la formule brute et du spectrogramme de masse du corps considéré. Identifier les constituants chimiques d'un matériau.
- **CRYVALIS** : 1979, chimie, recherche la structure de protéines à partir de résultats d'analyse cristallographique.

- **MOLGEN** : 1977, biologie, engendre un plan de manipulations génétiques en vue de construire une entité biologique donnée.
- **PROSPECTOR** : 1978, géologie, aide le géologue à évaluer l'intérêt d'un site en vue d'une prospection minière. (1600 règles)
- **AM** : 1977, mathématiques, proposition de conjectures, de concepts intéressants. (500 règles)
- **MUSCADET** : 1984, mathématiques, démonstration de théorèmes.
- **MYCIN** : 1974, médecine, système d'aide au diagnostic et au traitement de maladies bactériennes du sang (Denis 2006).

2.4. Composantes d'un système expert

Un système expert est composé de deux parties principales, une base de connaissances qui contient deux parties : une base de faits et une base de règles, et un moteur d'inférence qui change entre elles les mises à jour. L'architecture d'un système expert est présentée sur la figure 2.2.

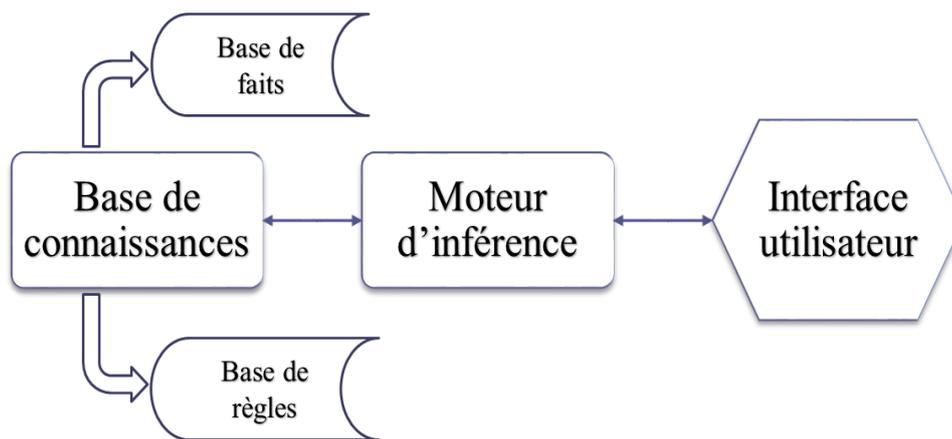


Figure 2.2. Architecture d'un système expert.

2.4.1. La base de connaissances

Chaque expert acquis au cours de son travail une expertise, cette expertise (connaissances) est stockée sous forme de faits et de règles de production. Cet ensemble de connaissances, est conservé sur un support de stockage, appelé « une base de connaissances ».

2.4.1.1. La base de fait :

Contient les différents faits utiles à l'application. Les faits sont des variables décrivant le monde, représentés par un nom et un état. On les utilise pour conditionner l'exécution des règles par le moteur d'inférence. La base de fait possède les caractéristiques suivantes :

- Mémoire de travail** : elle est variable au cours de l'exécution et vidée lorsque l'exécution est terminée. Au début, elle contient ce que l'on sait du cas examiné avant toute intervention

du moteur d'inférence (faits initiaux). Puis elle est complétée par les faits déduits par le moteur ou demandés à l'utilisateur (méta-faits).

Par exemple, dans le domaine médical, la base de faits pourra contenir une liste de symptômes au début et un diagnostic lorsque celle-ci se terminera.

ii. Le type d'un fait : les faits peuvent prendre des formes plus ou moins complexes. Les valeurs possibles sont :

- ✓ Booléennes : vrai, faux
- ✓ Symboliques ou nominales : appartenant à un domaine fini de symboles
- ✓ Réelles : pour représenter les faits continus.

Un système expert qui n'utilise que des faits booléens est dit d'ordre 0. Un système expert qui utilise des faits symboliques ou réels, sans utiliser de variables, est d'ordre 0+. Et un système utilisant la logique du premier ordre (calcul) est d'ordre 1.

iii. Les formules et les conditions : Dans un système expert d'ordre 0, on pourra par exemple écrire des formules d'un fait booléen ou sa négation de la forme :

actif ou \neg actif

Dans un système d'ordre 0+, on pourra trouver les formules : Si X est actif et (profession = médecin) et (salaire \leq 20000).

Illustration : « actif » est un fait booléen, profession est un fait symbolique et salaire est un fait réel.

Dans un système d'ordre 1, on pourra trouver : $\forall X$ maladie(X) et (X = grippe) et (symptôme(X) = forteFièvre).

Ces formules sont appelées conditions lorsqu'elles servent à déclencher des règles. On remarque que les faits booléens peuvent être interprétés comme des formules puisqu'ils possèdent une valeur de vérité (1 ou 0).

iv. Méta-faits et Méta-valeurs : Pour qu'un système expert puisse modéliser un raisonnement humain, il est indispensable qu'il puisse raisonner sur ses propres raisonnements, réfléchir aux faits qu'il manipule, aux formules qu'il peut construire, etc.

Il n'est pas suffisant que le système ait des connaissances, il faut aussi qu'il ait des méta-connaissances.

Ainsi, valeur(profession) est un méta-fait symbolique => médecin

valeur(salaire) = connue est une méta-condition => ≤ 20000

Quand un fait est inconnu, on peut envisager de demander sa valeur à l'utilisateur, si possible. Mais il n'est pas envisageable par exemple qu'un médecin demande à son patient : "quelle maladie avez-vous ?", ni qu'un juge demande à la personne comparaissant devant lui : "à quelle peine dois-je vous condamner ?" (Denis 2006).

2.4.1.2. La Base de Règles :

La base de règles (BR) représente les raisonnements effectués par un expert. Ces règles sont appelées les unes à la suite des autres afin de créer des enchaînements de raisonnements. Tous ces raisonnements peuvent être représentés sous la forme de règles de production de type « Si la condition est vraie alors exécuter action ».

Une règle est de la forme « **Si** conjonction de conditions **Alors** conclusion » où une conclusion est de la forme : Fait = valeur

La conjonction, comprise entre Si et Alors, est appelée **prémisse** ou le **déclencheur** de la règle. La **conclusion**, est appelée le **corps** de la règle.

En formalisme logique, en notant les conditions C_1, \dots, C_n ,

On écrit une règle ainsi : $C_1 \wedge C_2 \wedge \dots \wedge C_n \Rightarrow [\text{Fait} = \text{valeur}]$

(en langage Prolog, la notation est en général : $[\text{Fait} = \text{valeur}] : - C_1, \dots, C_n$)

Déclencher une règle consiste à remplacer ses prémisses par sa conclusion (chaînage avant), ou sa conclusion par ses prémisses (chaînage arrière) (Denis 2006).

2.4.2. Le moteur d'inférence

Un moteur d'inférence est un mécanisme qui permet d'inférer des connaissances nouvelles à partir de la base de connaissances du système. Selon différentes stratégies, le moteur d'inférence utilise des règles, les interprète, les enchaîne jusqu'à arriver à un état représentant une condition d'arrêt. L'exécution des règles par le moteur d'inférence influe sur l'état des faits et éventuellement sur les autres règles. Le cycle de base d'un moteur d'inférence est présenté sur la figure 2.3.

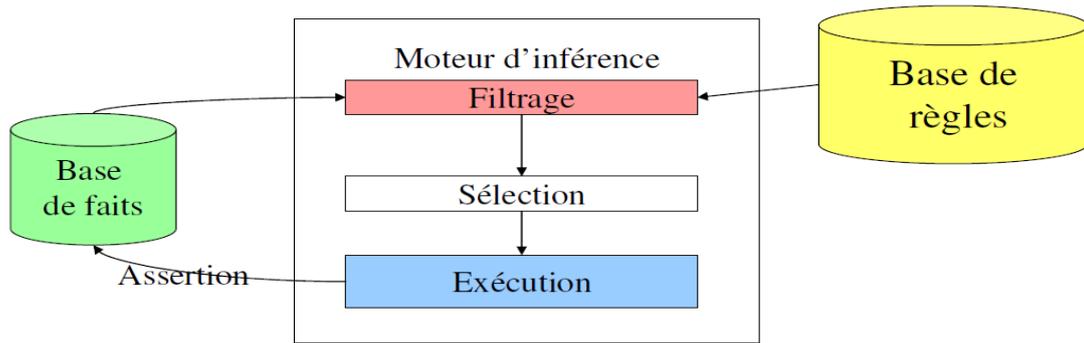


Figure 2.3. Cycle de base d'un moteur d'inférence.

1) L'évaluation

La sélection (restriction) : on cherche à déterminer un sous-ensemble F_1 de BF et un sous-ensemble R_1 de BR qui méritent a priori d'être mis en présence lors de l'étape de filtrage.

Le filtrage (pattern-matching) : En chaînage avant, on compare les prémisses des règles de R_1 avec les faits. En chaînage arrière, on compare les conclusions des règles de R_1 avec le sous-but à atteindre. Dans les deux cas, on détermine un ensemble R_2 de règles qu'il est possible de déclencher, appelé ensemble de conflit.

La résolution de conflit : consiste à choisir dans R_2 un sous-ensemble R_3 de règles qu'il va falloir effectivement déclencher.

2) Régime irrévocable et retour en arrière

Lorsque R_3 est vide, les systèmes les plus simples se contentent de s'arrêter. On dit que ces moteurs d'inférence sont en régime irrévocable (invariable). Un fonctionnement un peu plus compétent consiste à réexaminer l'ensemble de conflit R_2 du cycle précédent. Certaines règles y ont déjà été déclenchées. On peut décider de ne pas remettre en question les modifications de BF consécutives à ces déclenchements et de déclencher d'autres règles de R_2 . Dans ce cas, on dira que le système expert est en régime irrévocable.

Par contre, si on annule les modifications de BF avant de choisir d'autres règles à déclencher dans R_2 , on dit qu'on a effectué un retour arrière. Un système expert fonctionnant de cette manière est en régime par tentatives (branch and bound).

3) L'exécution : mode de raisonnement

On distingue trois modes principaux de fonctionnement pour la phase d'exécution des moteurs d'inférences. C'est à dire trois façons de raisonner : le chaînage avant, le chaînage arrière et le chaînage mixte (Denis 2006).

2.5. Modes de raisonnement

2.5.1. Chainage avant

Un moteur d'inférence fonctionne dans ce mode lorsque les faits représentent des informations dont la vérité a été prouvée. Ce mode de fonctionnement va des faits vers les buts :

- Détecter les règles dont les prémisses sont vérifiées (filtrage)
- Sélectionner la règle à appliquer
- Appliquer la règle
- Recommencer jusqu'à ce qu'il n'y ait plus de règle applicable

Le processus de chainage avant suit l'algorithme suivant :

Algorithme chaînage avant :

ENTREE: *BF, BR, F*

F est le fait à déduire

Tant que *F n'est pas dans BF et qu'il existe dans BR une règle applicable faire*

choisir une règle applicable R

BR = BR - R (désactivation de R)

fin tant que

si F appartient à BF alors

F est établi

sinon

F n'est pas établi

finsi

2.5.2. Chainage arrière

Le moteur d'inférence part d'un fait que l'on souhaite établir, il recherche toutes les règles qui concluent sur ce fait, il établit la liste des faits qu'il suffit de prouver pour qu'elles puissent se déclencher, puis il applique récursivement le même mécanisme aux autres faits contenus dans cette liste. L'exécution de l'algorithme de chaînage arrière peut être décrit par un arbre dont les nœuds sont étiquetés soit par un fait, soit par un des deux mots ET, OU.

Le processus de chainage arrière suit l'algorithme suivant :

But initial placé au sommet d'une pile

Détection des règles qui concluent à ce but

Résolution de conflits

Application de la règle, i.e, les éléments des prémisses deviennent de nouveau sous- buts à atteindre.

Arrêt : pile vide ou aucune règle applicable

Plusieurs différences apparaissent entre le chaînage avant et le chaînage arrière sont présentées dans le tableau 2.1.

Tableau 2.1. Différence entre chaînage avant et arrière.

	Chaînage avant	Chaînage arrière
Points forts	Fonctionne bien lorsque le problème se présente « naturellement » avec des faits initiaux ; Produit une grande quantité de faits à partir de faits initiaux très peu nombreux ; Adapté à la planification, le contrôle, l'interprétation.	Fonctionne parfaitement lorsque le problème consiste à prouver une hypothèse ; Il est focalisé sur le but à prouver et pose donc des questions pertinentes, qui ne déroutent pas l'utilisateur ; Contrairement au chaînage avant, il recherche dans la base de connaissances les informations intéressantes pour le problème courant ; Adapté au diagnostic et à la prescription.
Points faibles	Souvent ne perçoit pas certaines évidences ; Le système peut poser de nombreuses questions, qui parfois s'avèrent non pertinentes.	Poursuit une ligne de raisonnement même s'il s'avère qu'il devrait l'abandonner pour une autre. Les facteurs de croyance et les métarègles peuvent aider à résoudre ce problème.

2.5.3. Chaînage mixte

L'algorithme de chaînage mixte combine, comme son nom l'indique, les algorithmes de chaînage avant et de chaînage arrière.

Algorithme chaînage mixte :

ENTREE: BF, BR, F

F est le fait à déduire

Tant que F n'est pas déduit mais peut encore l'être faire

Saturer la base de faits par chaînage avant (déduire tout ce qui peut être déduit)

Chercher quels sont les faits encore éventuellement déductibles

Déterminer une question pertinente à poser à l'utilisateur et ajouter sa réponse à la base de faits

fin tant que

2.6. Comment Construire un Système Expert

La réalisation d'un système expert est un travail long, et un peu complexe dans sa programmation et dans la collection et la formalisation des connaissances. Le système expert apporte un intérêt financier aux entreprises qui l'utilise, chaque entreprise devra réaliser un tel système lorsqu'elle possède un expert compétent dans son domaine. Même si cet expert vient de

quitter l'entreprise, les connaissances acquises ne disparaissent plus, parce que le transfert d'expertise est réalisé, et l'entreprise ne perd pas du temps et d'argent. La réalisation d'un tel système expert passe par quatre étapes :

1) La Première Étape : Le choix d'un moteur d'inférence

Le choix d'un moteur d'inférence se fait selon l'ordre du moteur d'inférence si d'ordre 0 (si les faits sont des propositions) ou 1 (si on va calculer des prédicats). Un moteur d'ordre 0 peut suffire, mais si les problèmes mis en jeu sont plus complexes, un moteur d'ordre 1 est nécessaire ; et à l'informaticien de choisir ce type.

2) La Deuxième Étape : Le travail de l'expert

L'expert doit arriver à extraire de ses méninges ses connaissances et les traduire sous une forme accessible par le moteur. Ce travail est long et difficile ; il n'est pas habitué à ce genre de réflexion et quelquefois, son propre raisonnement ne lui est pas complètement accessible. Il est alors aidé par un ingénieur cognitif chargé d'organiser les explications de l'expert. Cette étape aboutit à la réalisation d'une maquette système expert sur une partie restreinte du domaine choisi, suit une série de tests permettant à l'expert et au cognitif de vérifier la véracité des diagnostics proposés par le système.

3) La Troisième Étape : Extension au domaine

Il s'agit d'étendre la version prototype au domaine initialement choisi. Là, l'expert doit continuer le travail de mise à plat de sa connaissance et également tester le travail du système expert.

4) La Quatrième Étape : Vers un produit fini

Un système expert peut être utilisé par des non-experts ou des non-informaticiens. Il est indispensable de lui associer un ensemble de logiciels d'interface du type dialogue en langage naturel en respectant l'ergonomie et l'interface homme/machine, et par l'explication de raisonnement, (Denis 2006).

2.7. Exercices corrigés

Quelques exercices sont proposés pour bien expliquer les modes de raisonnement et différencier entre le chaînage avant et arrière.

Exercice 1 : A partir de la base de faits : B, C et des règles suivantes, cherchez le but H par un chaînage avant.

R1	Si B et D et E	Alors F
R2	Si G et D	Alors A
R3	Si C et F	Alors A
R4	Si B	Alors X
R5	Si D	Alors E
R6	Si X et A	Alors H
R7	Si C	Alors D
R8	Si X et C	Alors A
R9	Si X et B	Alors D

Lorsque plusieurs règles sont en compétition, on choisira la première.

Solution : Chainage avant

Seule la règle **R6** possède **H** comme conséquence, donc : Nouveaux buts : X, A

La règle **R4** possède X comme conséquence, donc : Nouveaux buts : A, B

B est un fait, donc : Nouveaux buts : A

Trois règles possèdent A comme conséquence, ce qui détermine trois possibilités **R2**, **R3**, ou **R8**.

Application de **R2** Nouveaux buts : G, D

G n'est jamais une conséquence donc : Echec ---

Application de **R3** Nouveaux buts : C, F => C est un fait, Nouveau but : F

Application de **R1** Nouveaux buts : B, D, E => B est un fait,

Nouveaux buts : D, E

Application de **R7** Nouveaux buts : C, E => C est un fait,

Nouveau but : E

Application de **R5** Nouveaux buts : D

Application de **R7** Nouveaux buts : C, **C est un fait, donc Succès.**

Exercice 2 : A partir de l'énoncé de l'exercice 1, cherchez le but H par un chaînage arrière.

Solution : Chainage arrière

Etape 1 : Règles applicables : 4 ou 7. On choisit 4.

BF = {B,C,X}, La règle 4 est désactivée.

Etape 2 : Règles applicables : 7, 8 ou 9. On choisit 7.

BF = {B,C,X,D}, La règle 7 est désactivée.

Etape 3 : Règles applicables : 5, 8 ou 9. On choisit 5.

BF = {B,C,X,D,E}, La règle 5 est désactivée.

Etape 4 : Règles applicables : 1, 8 ou 9. On choisit 1.

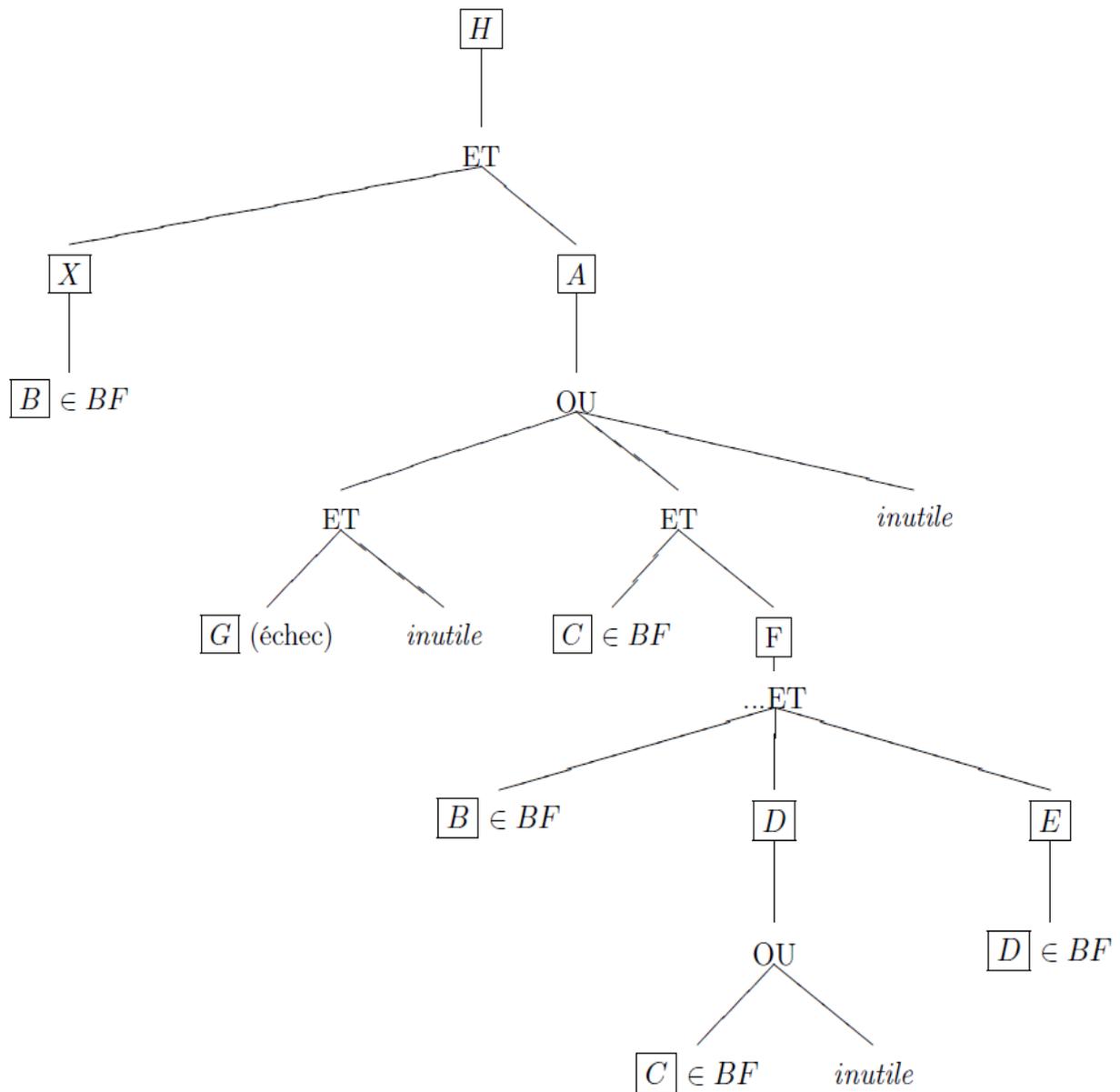
BF = {B,C,X,D,E,F}, La règle 1 est désactivée.

Etape 5 : Règles applicables: 3, 8 ou 9. On choisit 3.

BF = {B,C,X,D,E,F,A}, La règle 3 est désactivée.

Etape 6 : Règles applicables : 6, 8 ou 9. On choisit 6. **H est établi**

On peut obtenir le même résultat en utilisant un arbre :



2.8. Exercices proposés

Exercice 1 : Explorez toute la base de règles à partir de la base de faits pour déduire tous les faits.

BF = {H,B,X,C}

BR :

1. P \vdash B,E,D
2. A,B \vdash D,G
3. A \vdash C,P,K
4. D,L \vdash C,Y
5. E \vdash D,M
6. H \vdash A,L
7. H,W \vdash Z
8. P,L \vdash W

Exercice 2 : à partir de la base de faits : A, K et des règles :

- | | | | |
|----|---------|---------------|---|
| R1 | A, B, C | \rightarrow | D |
| R2 | I, H | \rightarrow | B |
| R3 | H, F | \rightarrow | B |
| R4 | A | \rightarrow | I |
| R5 | E, F | \rightarrow | D |
| R6 | A | \rightarrow | F |
| R7 | K, L | \rightarrow | E |
| R8 | A | \rightarrow | L |

1. Réaliser du chaînage avant jusqu'à ce qu'aucune règle ne puisse plus donner de faits supplémentaires.
2. Utiliser le chaînage arrière pour répondre au but : D

Exercice 3 : Un expert a construit la base de règles suivante :

- | | | |
|------------------|---------------|---|
| R1 : A et B | \rightarrow | C |
| R2 : D | \rightarrow | A |
| R3 : E | \rightarrow | F |
| R4 : G | \rightarrow | H |
| R5 : I | \rightarrow | F |
| R6 : H et F et J | \rightarrow | B |
| R7 : H et K | \rightarrow | J |
| R8 : G et F | \rightarrow | K |

La base initiale de faits est : (D, G, I),

- 1- Prouvez le fait **C** par chaînage arrière. quelle est la suite de règles essayées pour prouver le fait **I**, on indiquera si chaque règle essayée a été un succès ou un échec.
- 2- On veut prouver le fait **C** en chaînage avant ; quelle est la différence entre les deux modes de raisonnement (chaînage arrière et chaînage avant).

2.9. Conclusion

Les systèmes experts, une des applications de l'intelligence artificielle les plus utilisées dans le monde de l'entreprise. De nombreux systèmes experts ont été implantés avec succès pour résoudre des problèmes concrets, comme les systèmes de contrôle et de supervision dans les cimenteries.

Parfois, les systèmes experts souffrent d'une faiblesse intrinsèque, toutes les expertises ne sont pas facilement formalisables sous forme de règles ou ne sont pas capables de traiter les systèmes non linéaires. La solution à ce problème est la logique floue, plusieurs problèmes non linéaires ont été résolus et plusieurs systèmes ont été réalisés afin d'aboutir à un contrôle globale d'une entreprise.

Chapitre 3 : La logique floue

3.1. Introduction

La logique floue suscite un intérêt général de la part des ingénieurs et des industriels, mais plus de la part de tous ceux qui éprouvent le besoin d'automatiser la prise de décision dans leur domaine, de construire des systèmes artificiels pour effectuer les tâches habituellement prises en charge par les humains. Les connaissances dont nous disposons sur une situation quelconque sont généralement imparfaites, parce que nous avons un doute sur leur validité, elles sont alors incertaines, ou parce que nous éprouvons une difficulté à les exprimer clairement, elles sont alors imprécises (Meunier 2007). Le fait d'utiliser des connaissances floues permet d'exprimer des situations graduelles. Le contrôle flou s'est surtout montré robuste (Gacogne 2003).

L'intérêt essentiel de la logique floue, réside dans le fait que les notions linguistiques sont bien adaptées et traduisent le raisonnement qualitatif humain dans un processus industriel. Elle décrit des situations avec des règles qui représentent des informations. Dans ce cas, un opérateur ne prend pas des décisions seulement sur des situations spécifiques dont il n'a qu'une connaissance incomplète, mais qu'il agrège le long de son expérience dont il utilise les notions linguistiques (Zermane 2017).

3.1.1. Origine

Les connaissances dont disposent les humains sur le monde ne sont presque jamais parfaites. Ces imperfections peuvent être distinguées en deux classes :

Incertitudes : pour désigner les connaissances dont la validité est sujette à question.

Par exemple, si nous savons qu'une personne s'est cognée la tête au plafond, nous devinons qu'il est probable qu'elle soit très grande => Probabilité.

Imprécisions : pour désigner les connaissances qui ne sont pas perçues ou définies nettement.

Par exemple, au lieu de dire qu'une personne mesure 2.3 m, nous disons usuellement que cette personne est très grande => Logique floue.

La figure 3.1 illustre deux situations différentes où la différence entre la précision et la signification est très importante et parfois critique.

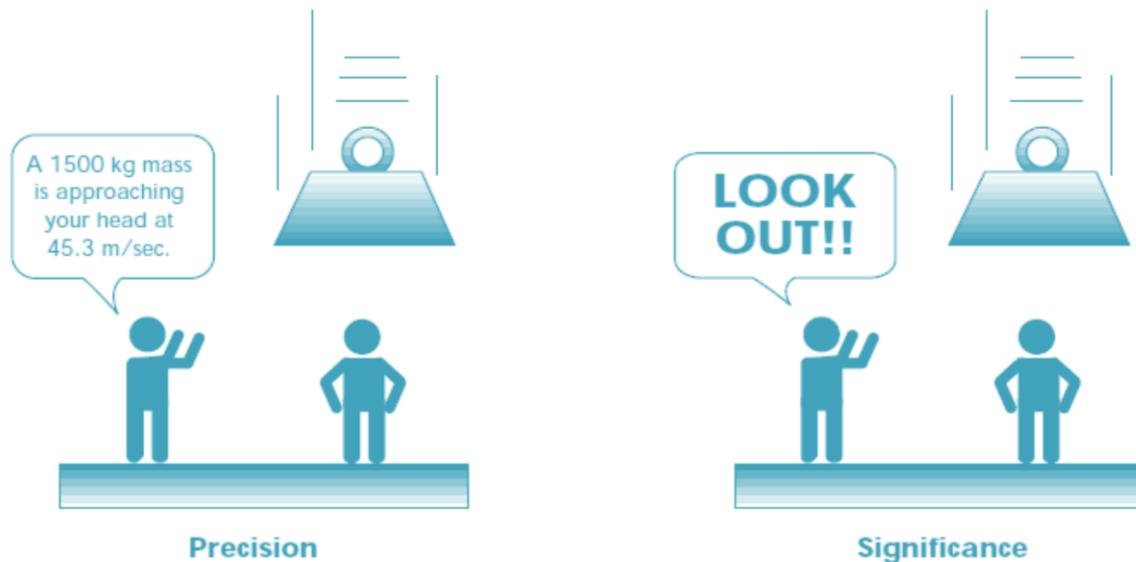


Figure 3.1. Différence entre précision et signification.

Comment faire en sorte d'exprimer ces imprécisions en termes logiques ?

En logique classique, une proposition est vraie ou fausse.

Exemple : une personne est grande. Vrai ou faux ? => Pas flexible...

En logique floue, une proposition peut avoir autant de valeurs que l'on veut.

Exemple : une personne est grande. C'est vrai à 30% => Flexible !

Règle de déduction Modus Ponens :

Si A et (A => B) sont vraies Alors on déduit que B est vraie.

Soit : m.p : A, (A => B) ⊢ B

Question : si A et B n'acceptent ni faux ni vrai ?

3.1.2. Définition de la logique floue

C'est une extension (généralisation) de la logique classique qui permet la modélisation des imperfections des données et se rapproche dans une certaine mesure de la flexibilité du raisonnement humain. La logique floue a été créée par Lotfi Zadeh en 1965 en se basant sur sa théorie des ensembles flous, qui est une généralisation de la théorie des ensembles classiques. En introduisant la notion de degré dans la vérification d'une condition (degré d'appartenance), nous permettons à une condition d'être dans un autre état que vrai ou faux. Un des intérêts de la logique floue pour formaliser le raisonnement humain est que ces règles sont énoncées en langage naturel. La logique floue est appliquée dans plusieurs domaines, parmi eux :

- Aide à la décision, diagnostic : (domaine médical, orientation professionnelle...)
- Base de données : (objets flous et/ou requêtes floues)
- Reconnaissance de forme,

- Agrégation multicritère et optimisation,
- Commande floue dans les systèmes de supervision, ...

3.1.3. *Historique d'application*

1965 : Concept introduit par Pr. Lotfi Zadeh (Berkeley) : «Fuzzy set theory » Définition des ensembles flous et opérateurs associés.

1970 : Premières applications : Systèmes experts, Aide à la décision en médecine, commerce...

1974 : Première application industrielle. Régulation floue d'une chaudière à vapeur réalisée par Mamdani.

1975 : le professeur Mamdani à Londres développe une stratégie pour le contrôle des procédés et présente les résultats très encourageants qu'il a obtenus sur la conduite d'un moteur à vapeur.

1978 : la société danoise F.L. Smidth a réalisé le contrôle d'un four à ciment. C'est là la première véritable application industrielle de la logique floue.

Japon, la recherche n'est pas seulement théorique mais également très applicative, que la logique floue connaît son véritable essor à la fin des années 1980.

1985 : Les japonais sont les premiers à introduire des produits grand public « Fuzzy Logic Inside ».

1987 : Métro de Sendai (Hitachi) (Japon).

1990 : c'est en Allemagne que des applications apparaissent en grand nombre comme la conduite de hauts-fourneaux Dunkerque (Allemagne).

1992 : Usine de papier au Portugal, Produits de consommation courante : Aspirateurs, machines à laver, système de climatisation..., Appareils photos, Caméras Canon (autofocus, stabilisateur d'image), Photocopieurs (qualité d'image, distribution d'encre et industrie automobile (régulation du moteur, système de transmission).

3.2. *Les ensembles classiques*

3.2.1. *Les ensembles classiques : rappel*

Le concept d'appartenance dans la théorie des ensembles désigne le fait qu'un élément fasse partie ou non d'un ensemble. Par exemple, l'entier $7 \in \{6; 7; 9\}$ et $7 \notin \{1; 3; 8; 5\}$.

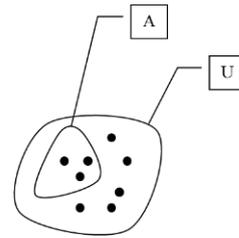
Soient U l'univers du discours, A un sous-ensemble de U , μ la fonction caractéristique de l'ensemble et x un élément. Une fonction d'appartenance est une fonction qui explicite l'appartenance de x ou non à l'ensemble A et on écrit : $\mu_A(x)$.

Dans la théorie des ensembles classiques :

Si μ_A est la fonction d'appartenance de l'ensemble A

$$\forall x \in U \quad \mu_A(x) = 0 \quad \text{si } x \notin A$$

$$\mu_A(x) = 1 \quad \text{si } x \in A$$



3.2.2. Limite des ensembles classiques

Exemple : un patient atteint d'hépatite présente généralement les symptômes suivants :

- Le patient a une forte fièvre, son degré de température est 39 °C,
- Sa peau présente une coloration jaune,
- Il a des nausées.

Question : Si le patient à 38,9 °C de température, est ce qu'il a de l'hépatite ou non ?

Selon la logique classique (Figure 3.2), la représentation graphique du degré de la température est présentée comme dans le graphe de la figure. Cette représentation montre que le patient n'a pas de forte fièvre ($38.9 < 39$ °C), donc le patient n'a pas d'hépatite.

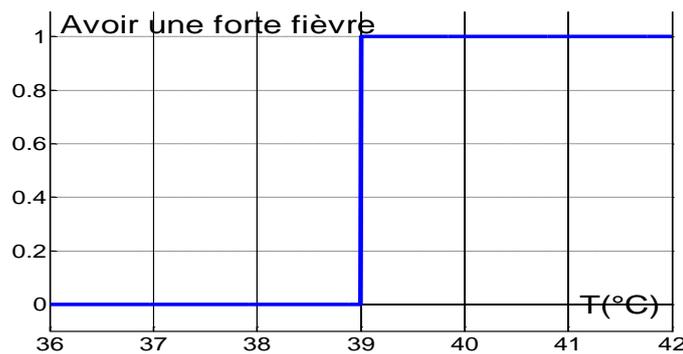


Figure 3.2. Ensemble classique.

Si on peut avoir une autre représentation comme celle présentée sur la figure 3.3, on peut dire que le patient possède une forte fièvre à **48%** \Rightarrow y a une possibilité qu'il ait une hépatite à x %.

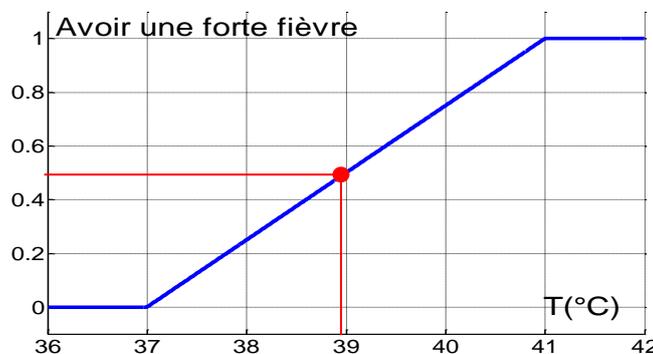


Figure 3.3. Ensemble flou.

Cette représentation conduit aux ensembles flous.

3.3. Les ensembles flous

3.3.1. Définitions

La théorie des sous-ensembles flous est une théorie mathématique du domaine de l'algèbre abstraite. Elle a été développée par Lotfi Zadeh en 1965 afin de représenter mathématiquement l'imprécision relative à certaines classes d'objets et sert de fondement à la logique floue.

Les sous-ensembles flous (ou parties floues) ont été introduits afin de modéliser la représentation humaine des connaissances, et ainsi améliorer les performances des systèmes de décision qui utilisent cette modélisation. Elles sont utilisées soit pour modéliser l'incertitude et l'imprécision, soit pour représenter des informations précises sous forme lexicale assimilable par un système expert.

Si $\mu_A(x)$ est la fonction d'appartenance de l'ensemble flou A , $\forall x \in U, \mu_A(x) \in [0 ; 1]$.

Si $\mu_A(x) = 0,30$: x appartient à l'ensemble flou A avec un degré d'appartenance (valeur de vérité) de 30%. Un ensemble flou est totalement déterminé par sa fonction d'appartenance.

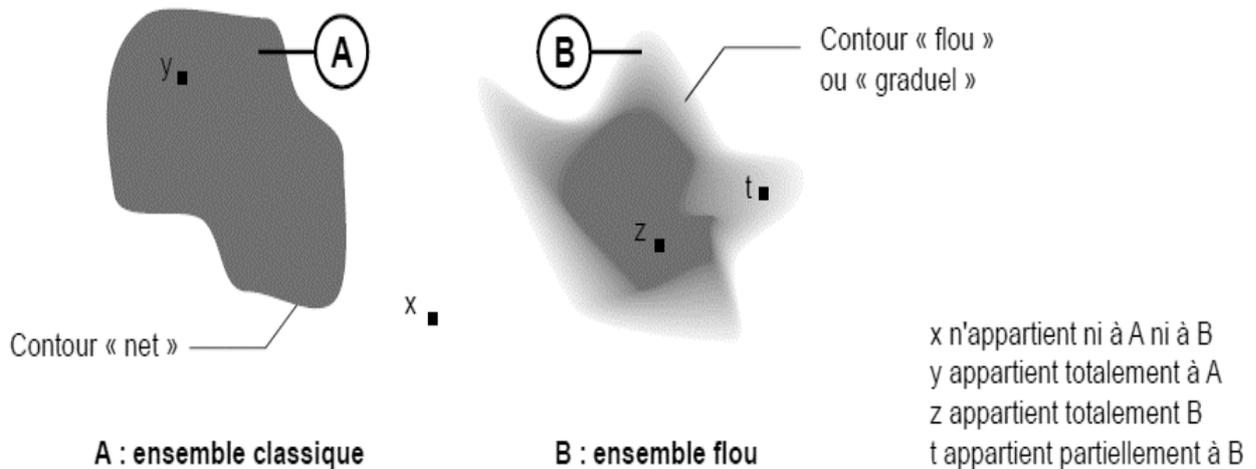


Figure 3.4. Ensemble classique et flou.

Un ensemble flou A est défini sur un univers de discours U (ensemble d'éléments discrets ou continus) par sa fonction d'appartenance μ_A . La grandeur $\mu_A(x)$ définit le degré d'appartenance de l'élément x à l'ensemble A .

$$\mu_A: U \rightarrow [0, 1], x \rightarrow \mu_A(x)$$

$$A = \{(x, \mu_A(x)) \mid x \in U\}$$

Pour pouvoir décrire facilement un sous-ensemble flou, on utilise certaines de ses caractéristiques représentées par la figure 3.5. La première caractéristique est le *support* d'un ensemble flou A , c'est-à-dire l'ensemble des éléments x qui appartiennent à A . il est noté $Supp(A)$ et présenté par la formule : $Supp(A) = \{x \in U \mid \mu_A(x) > 0\}$. La deuxième caractéristique est le noyau

qui présente l'ensemble de tous les éléments appartenant de façon absolue (avec un degré = 1). Le noyau est présenté par : $Noy(A) = \{x \in U \mid \mu_A(x) = 1\}$.

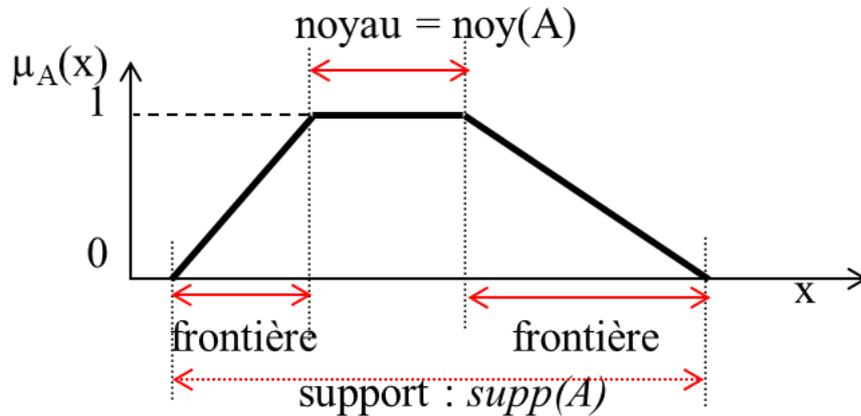


Figure 3.5. Support et noyau d'un ensemble flou.

L'ensemble flou vide est noté \emptyset , il est défini par : $\mu_{\emptyset}(x) = 0, \forall x \in U$.

Le plus grand ensemble flou sur U est noté I_U , il est défini par : $\mu_{I_U}(x) = 1, \forall x \in U$.

La logique floue est basée sur des variables floues dites *variables linguistiques* divisés en ensembles flous dites valeurs linguistiques présentés dans un univers du discours qui indique l'intervalle de mesure de la variable. Par exemple sur la figure 3.6, on définit :

- Univers du discours est l'intervalle de mesure de la variable linguistique : de 0 à 30 °C.
- Variable linguistique : Température.
- Valeurs linguistiques : « Froid », « Tempéré » et « Chaud ».

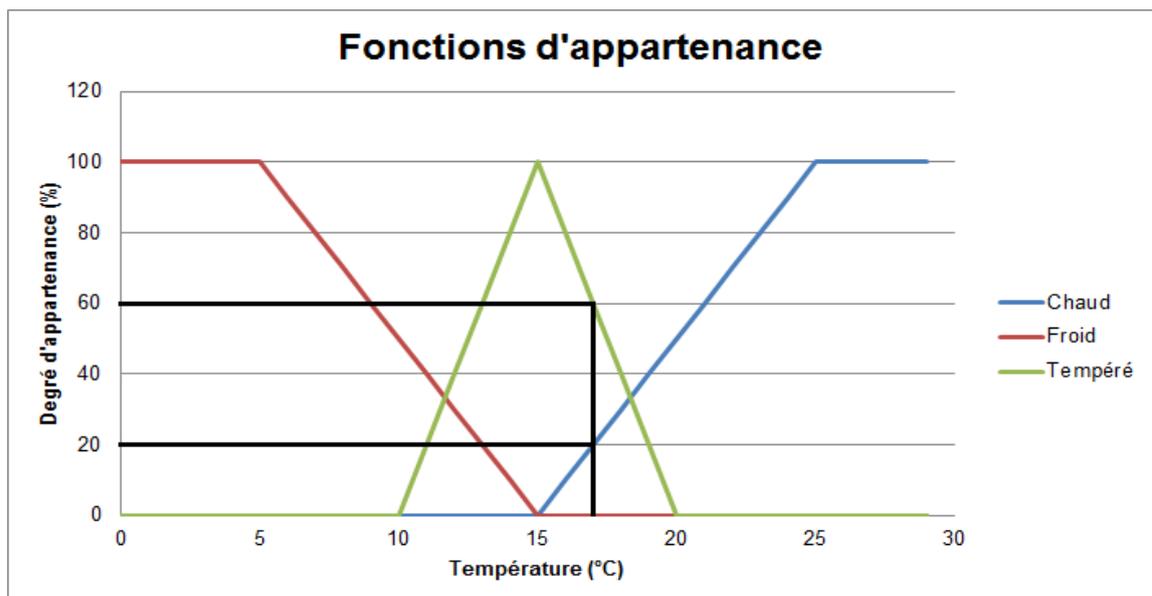
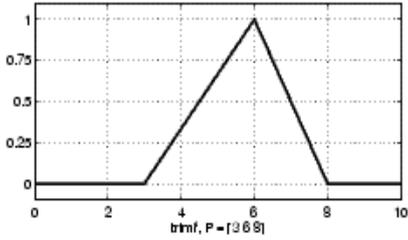
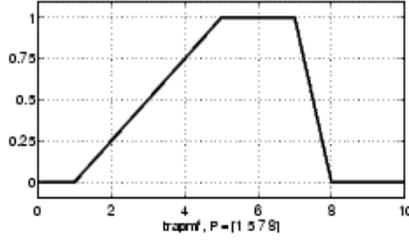
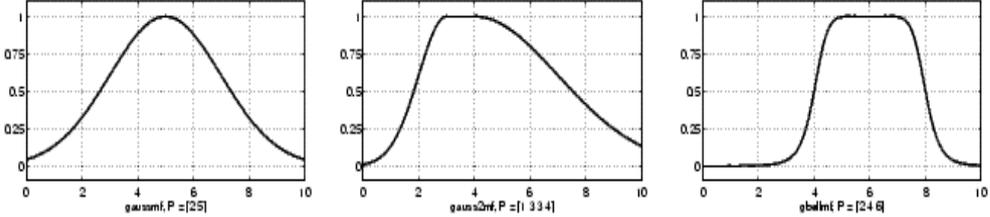
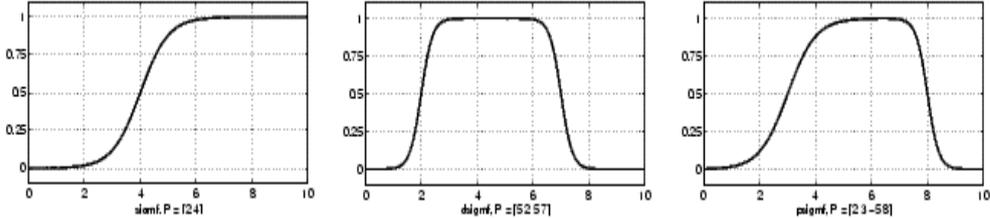


Figure 3.6. Univers du discours, variables et valeurs linguistiques.

Les fonctions d'appartenance peuvent avoir diverses formes selon leurs définitions. Ces formes sont représentées dans le tableau 3.1.

Tableau 3.1. Les fonctions d'appartenance.

Fonction d'appartenance	Forme de la fonction d'appartenance
Triangulaire (trimf)	
Trapézoïdale (trapmf)	
Gaussienne (gaussmf)	
Sigmoides (sigmf)	

La figure 3.7 illustre un exemple de présentation de quelques ensembles flous en forme triangulaire.

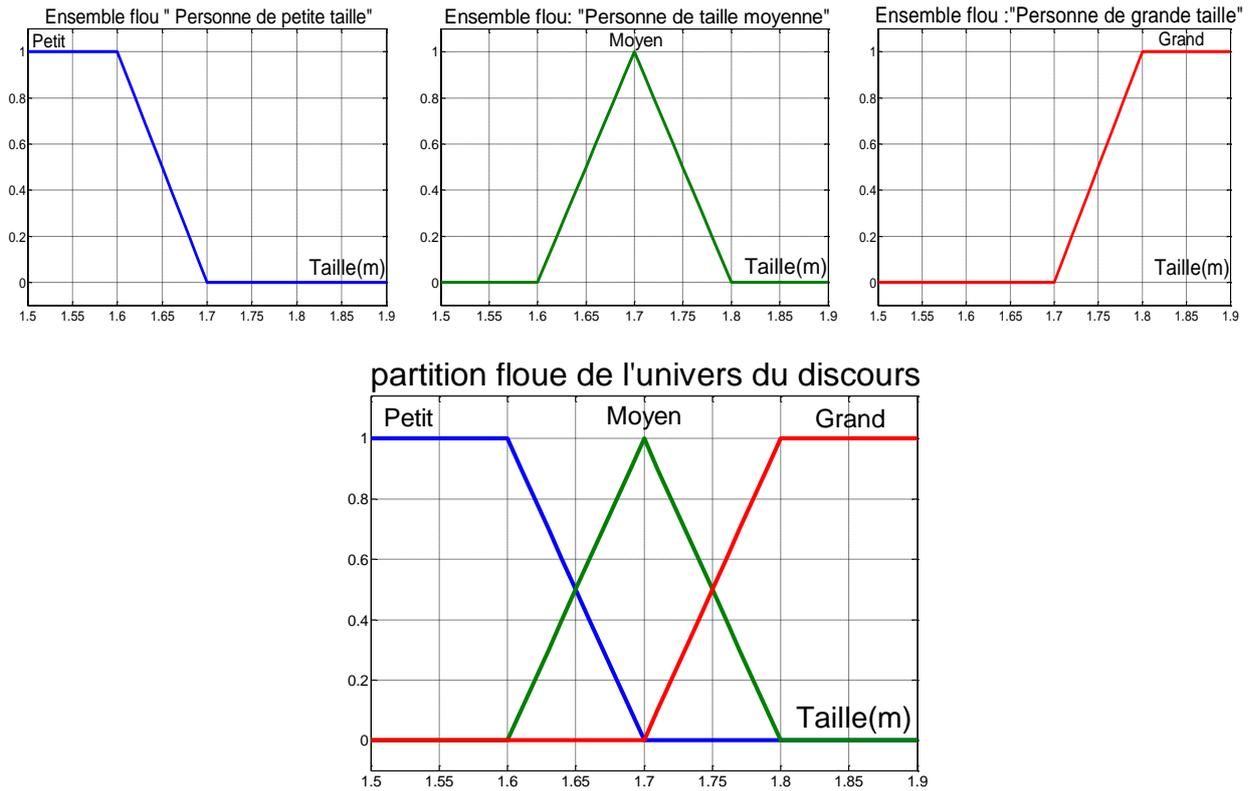


Figure 3.7. Exemple de représentation d'ensembles flous.

Les ensembles classiques sont des cas particuliers d'ensembles flous. Leur fonction d'appartenance valant 0 ou 1, sont en créneaux. Comme conclusion, la logique floue englobe la logique classique où les données sont certaines. Lorsqu'un fait certain correspond à l'énoncé de la valeur d'une variable, on a un singleton :

$$\begin{cases} \mu_{x_0}(x_0) = 1 \text{ pour } x = x_0 \\ \mu_{x_0}(x) = 0 \text{ pour } x \neq x_0 \end{cases}$$

Exemples :

- 1- Fonction d'appartenance de la classe « le feu est rouge » (Figure 3.8) :



Figure 3.8. Fonction d'appartenance de la classe « le feu est rouge ».

- 2- Fonction d'appartenance de la classe « la température est tiède » (Figure 3.9) :

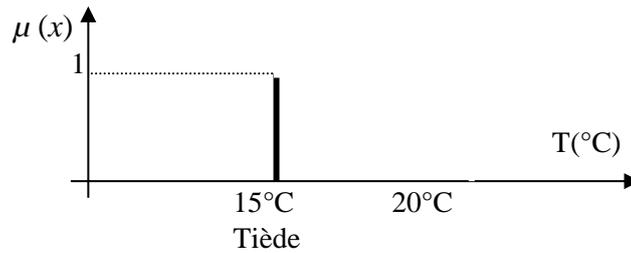


Figure 3.9. Fonction d'appartenance de la classe « la température est tiède ».

3.3.2. Opérations sur les ensembles flous

Comme la logique booléenne standard est un cas particulier de la logique floue, Tous les résultats obtenus en logique classique doivent être retrouvés par la logique floue. Comme dans le cas des ensembles classiques, On définit l'union, l'intersection, le complément d'ensembles flous.

Les définitions les plus souvent rencontrées sont :

Mamdani : $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$ et $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$

Sugeno : $\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$ et $\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x)$

Dans les deux cas : $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

pour $x \in U$

1. L'union (Mamdani) : A est l'ensemble flou des personnes « petit ». B est l'ensemble flou des personnes « moyen ». L'ensemble des personnes **petit OU moyen** est un ensemble flou de fonction d'appartenance (Figure 3.10) : $\mu_{A \cup B}(x) = \text{Max}(\mu_A(x), \mu_B(x)) \quad \forall x \in U$.

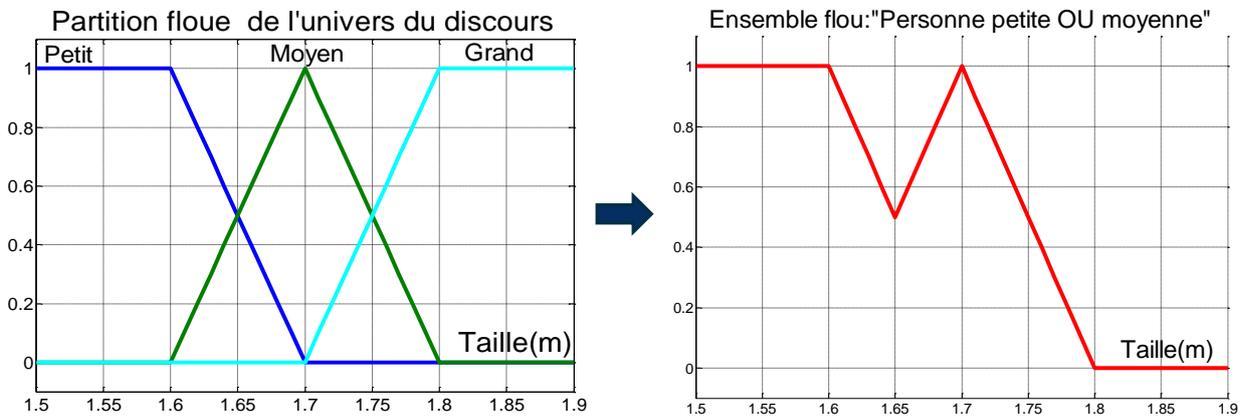


Figure 3.10. Union de deux ensembles flous.

2. L'intersection : L'ensemble des personnes petit **ET** moyen est un ensemble flou de fonction d'appartenance (Figure 3.11) : $\mu_{A \cap B}(x) = \text{Min}(\mu_A(x), \mu_B(x))$.

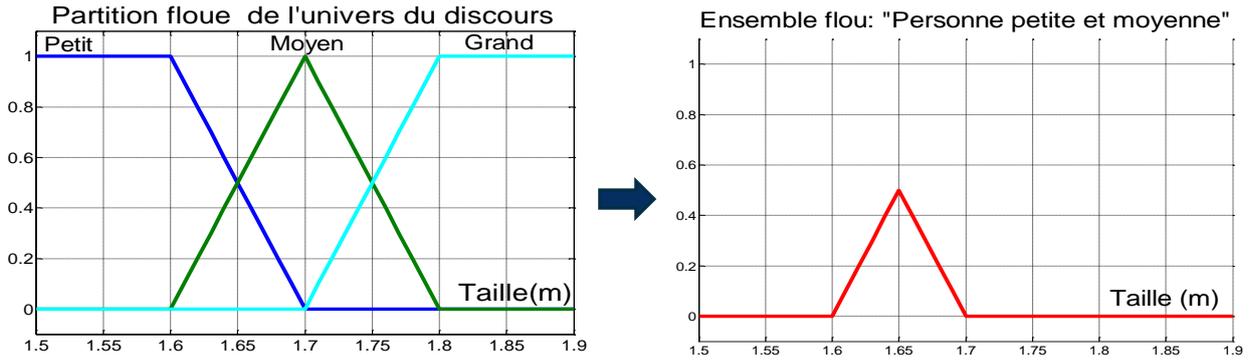


Figure 3.11. Intersection de deux ensembles flous.

3. Le complément

A est l'ensemble flou des personnes petites. L'ensemble des personnes NON petites est un ensemble flou de fonction d'appartenance (Figure 3.12) : $\mu_{\bar{A}}(x) = 1 - \mu_A(x) \quad \forall x \in U$.

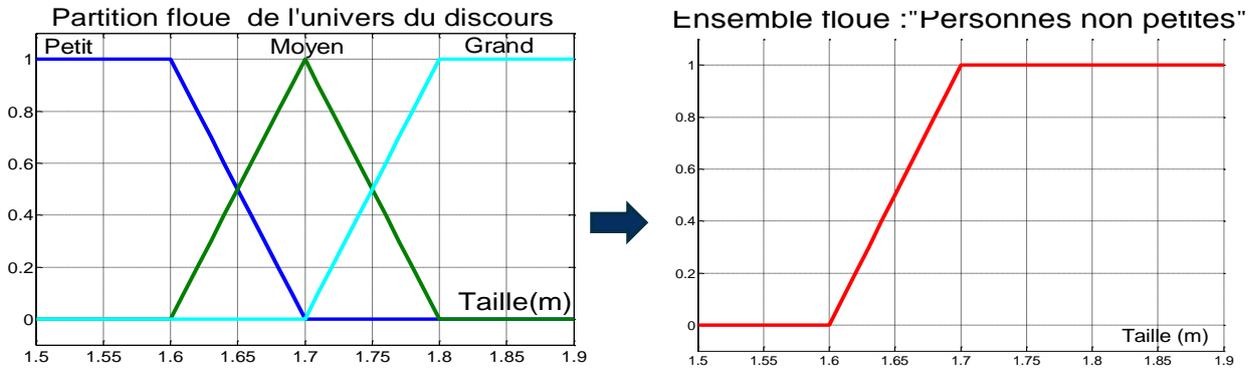


Figure 3.12. Complément d'un ensemble flou.

4. Autres propriétés

Commutativité : $A \cup B = B \cup A, A \cap B = B \cap A$

Associativité : $A \cup (B \cap C) = (A \cup B) \cap C, A \cap (B \cup C) = (A \cap B) \cup C$

Distributivité : $A \cup (B \cap C) = (A \cup B) \cap (A \cup C), A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Idempotence : $A \cup A = A, A \cap A = A$

Identité : $A \cup \emptyset = A, A \cup 1_U = 1_U, A \cap \emptyset = \emptyset, A \cap 1_U = A$

2 exceptions notables :

En logique floue, le principe du tiers exclu est contredit (Figure 3.13) : $A \cup \bar{A} \neq U$ i.e $\mu_{A \cup \bar{A}}(x) \neq 1$.

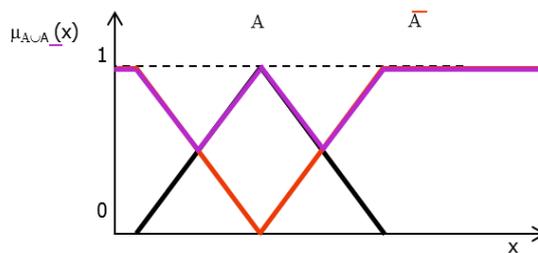


Figure 3.13. Union de l'ensemble flou et son complément.

Ainsi, on peut être A et $\neg A$ en même temps (Figure 3.14) : $A \cap \bar{A} \neq \emptyset$ i.e $\mu_{A \cap \bar{A}}(x) \neq 0$.

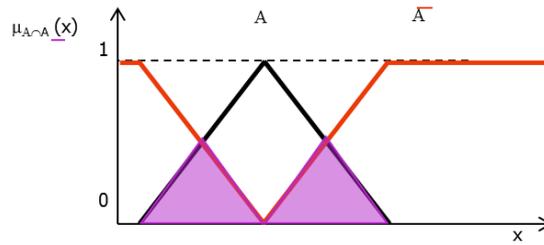


Figure 3.14. Intersection de l'ensemble flou et son complément.

3.4. Création d'un système de contrôle flou

Les premières applications de la logique floue étaient dans le domaine du contrôle des systèmes fait par des experts humains. Le contrôle de ces systèmes fait apparaître deux types d'informations :

- Des informations numériques obtenues par les mesures des capteurs.
- Des informations linguistiques obtenues par les experts humains.

Le contrôle flou (Figure 3.15) utilise la logique floue comme une démarche qui peut couvrir la stratégie du contrôle linguistique. Il est intégré dans la partie qui gère les données de commande et de contrôle de la partie opérative du système, et appelée contrôleur flou.

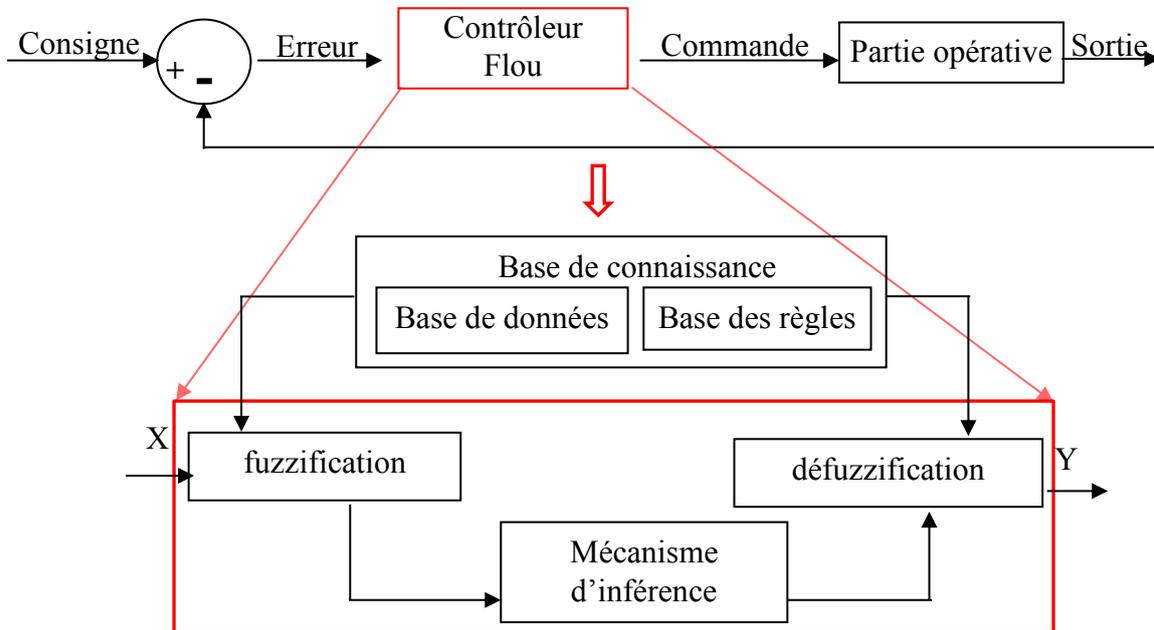


Figure 3.15. Schéma d'un contrôleur flou.

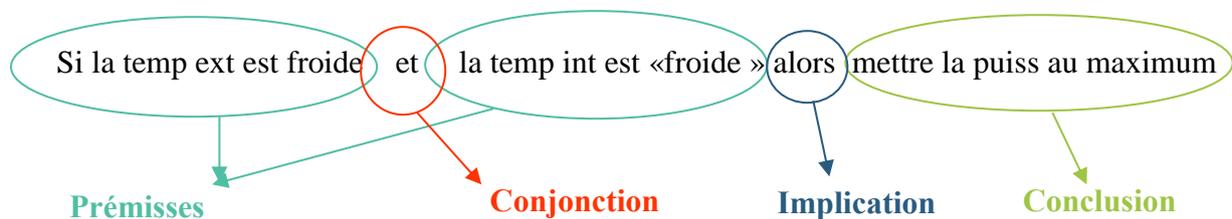
Avant de créer un système de contrôle flou, on doit définir la base de connaissance qui comprend les connaissances de l'expert humain pour le contrôle du système ainsi que le domaine de variation des variables d'E/S. Elle est donc constituée de :

1. **Base de données (faits)** : On regroupe dans ce bloc, l'ensemble des définitions utilisées dans les règles du contrôleur flou (univers du discours, partition flou, choix des opérateurs, ...)
2. **Base de règles floues** : La base de l'expert est généralement exprimée par des règles floues de la forme SI – ALORS.

Les systèmes à logique floue utilisent une expertise exprimée sous forme d'une base de règles floues de type : *Si (X est A) Alors (Y est B)*.

Exemple : quelques règles de la base de règle d'un contrôleur flou de puissance sont comme suit :

1. Si la température extérieure est « **froide** » et la température intérieure est « **froide** » alors mettre la puissance au « **maximum** »
2. Si la température extérieure est « **froide** » et la température intérieure est « **bonne** » alors mettre une puissance « **moyenne** », ...



Les systèmes à logique floue traitent de variables d'entrées floues et fournissent des résultats sur des variables de sorties elles-mêmes floues. Ce processus passe par plusieurs étapes :

3.4.1. La fuzzification

La fuzzification est la première étape dans la réalisation d'un contrôle flou. Elle transforme chaque valeur réelle d'entrée (mesure) en un ensemble flou. En lui attribuant sa fonction d'appartenance à chacune des classes préalablement définie. Donc c'est l'étape qui consiste en la quantification floue des valeurs réelles d'une variable qui indique le degré d'appartenance d'une valeur réelle à chaque ensemble flou.

Hypothèse : sur la figure 3.16, Pierre mesure 1.62 m ?

Est-ce que Pierre est :

- 1- Petit ?
- 2- Moyen ?
- 3- Grand ?

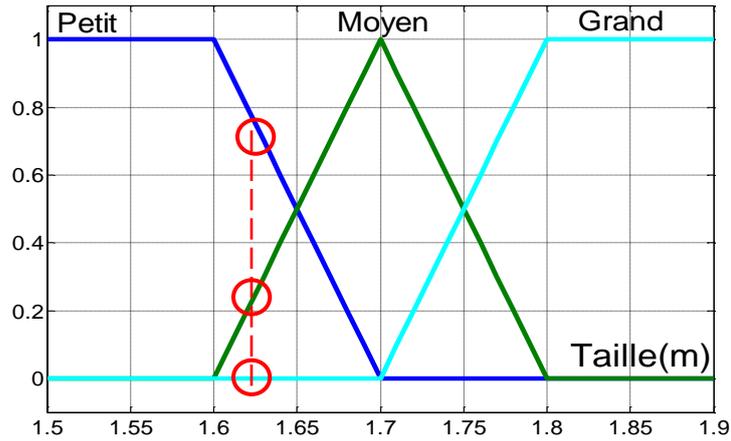
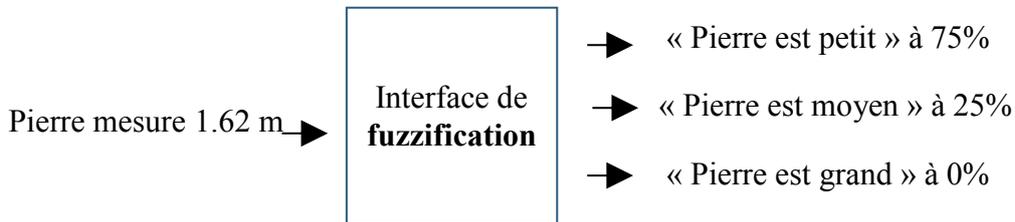


Figure 3.16. Appartenance d'une valeur réelle aux ensembles flous.

Réponse :



Généralisation : on prend l'exemple des ensembles flous présentés sur la figure 3.17. Pour n'importe quelle valeur de x , on calcule ses degrés d'appartenance en valeur absolue par :

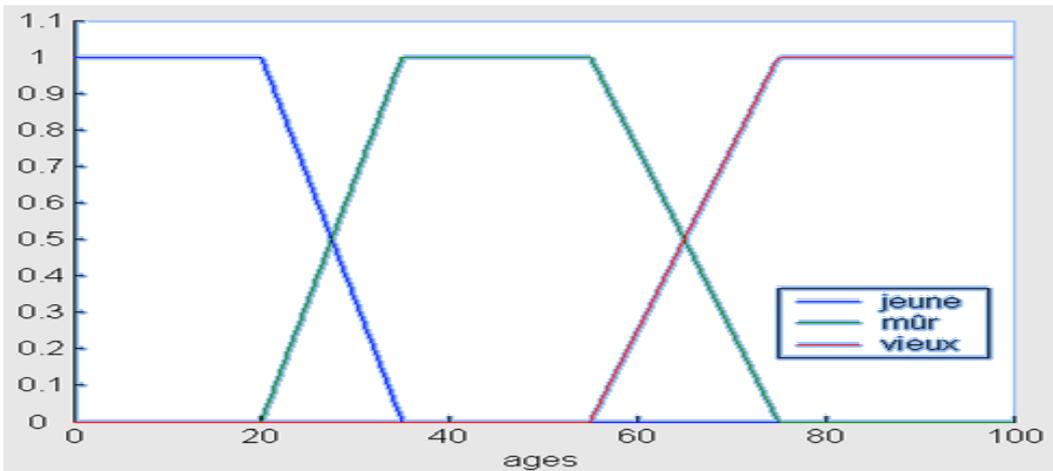


Figure 3.17. Généralisation de calcul des degrés de vérités.

$$\mu_{jeune} : [0, 100] \rightarrow [0, 1]$$

$$x \rightarrow \begin{cases} \mu_{jeune}(x) = 1 & \text{si } x \leq 20 \\ \mu_{jeune}(x) = \frac{20-x}{15} & \text{si } 20 < x < 35 \\ \mu_{jeune}(x) = 0 & \text{si } x \geq 35 \end{cases}$$

$$\mu_{\hat{m}ur} : [0, 100] \rightarrow [0, 1]$$

$$x \rightarrow \begin{cases} \mu_{\hat{m}ur}(x) = 0 & \text{si } x \leq 20 \text{ ou } x \geq 75 \\ \mu_{\hat{m}ur}(x) = \frac{x-20}{15} & \text{si } 20 < x < 35 \\ \mu_{\hat{m}ur}(x) = 1 & \text{si } 35 \leq x \leq 55 \\ \mu_{\hat{m}ur}(x) = \frac{55-x}{20} & \text{si } 55 < x < 75 \end{cases}$$

$$\mu_{\hat{v}ieux} : [0, 100] \rightarrow [0, 1]$$

$$x \rightarrow \begin{cases} \mu_{\hat{v}ieux}(x) = 0 & \text{si } x \leq 55 \\ \mu_{\hat{v}ieux}(x) = \frac{x-55}{20} & \text{si } 55 < x < 75 \\ \mu_{\hat{v}ieux}(x) = 0 & \text{si } x \geq 75 \end{cases}$$

3.4.2. L'inférence floue

C'est une opération logique par laquelle on admet une proposition sous sa liaison avec d'autres propositions tenues pour vraies (condition vérifiée). Elle exprime la relation qui existe entre les variables d'entrée (Variables linguistiques) et les variables de sortie (Variable linguistique).

- La variable floue X appartient à la classe floue A avec un degré de validité $\mu_A(x_0)$.
- La variable floue Y appartient à la classe floue B à un degré qui dépend du degré de validité $\mu_B(y_0)$ de la prémisse.

L'appartenance dans les sorties dépend de :

- 1) L'ensemble flou de sortie considérée.
- 2) Le degré de validité de la prémisse $\mu_{\text{prémisse}}(x_0)$.
- 3) La méthode d'implication choisie.
 - a. Méthode de Mamdani : $\hat{\mu}_{\text{conclusion}}(y) = \text{Min}(\mu_{\text{prémisse}}(x_0), \mu_{\text{conclusion}}(y))$
 - b. Méthode de Larsen : $\hat{\mu}_{\text{conclusion}}(y) = \mu_{\text{prémisse}}(x_0) * \mu_{\text{conclusion}}(y)$

3.4.3. L'agrégation et l'activation des règles

Une règle est activée dès qu'elle a une prémisse ayant une valeur de vérité non nulle. Plusieurs règles peuvent être activées simultanément et préconiser des actions avec différents degrés de validités ; ces actions peuvent être contradictoires. Il convient d'agréger (associer) des règles pour fournir une appartenance de la variable floue de sortie à une classe floue affirmée (confirmé)

(figure 3.18). Il existe dans plusieurs méthodes d'inférence floue qui dépendent des implications de la forme des fonctions. On considère que ces règles sont liées par un opérateur OU (on prend la valeur Max). Où : $\mu_B(y) = \text{Max} [\mu_{Bi}(y)]$ $i \in \{\text{indices des règles activées}\}$.

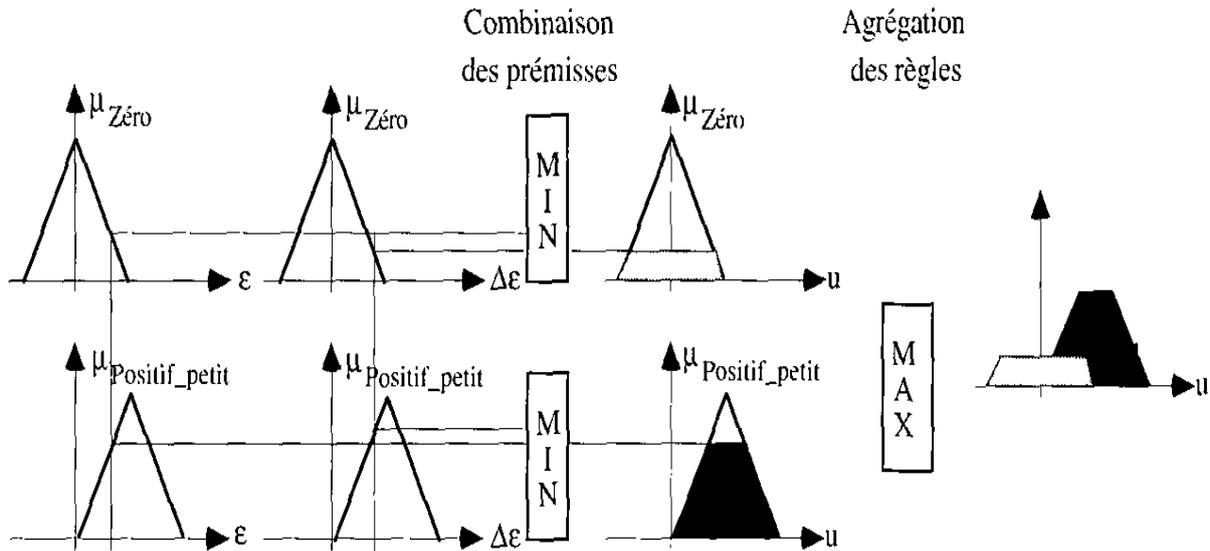


Figure 3.18. Agrégation et l'activation des règles.

Exemples :

1) **Pour un ensemble flou** : la conclusion d'une règle floue est l'appartenance d'une variable floue de sortie « Chauffer » à une classe floue « fort » (Figure 3.19).

$$\mu_{conclusion}(y) = \text{Min}(\mu_{prémisse}(x_0), \mu_{conclusion}(y))$$

Règle : SI la température est très basse ALORS puissance chauffe est Chauffer fort.

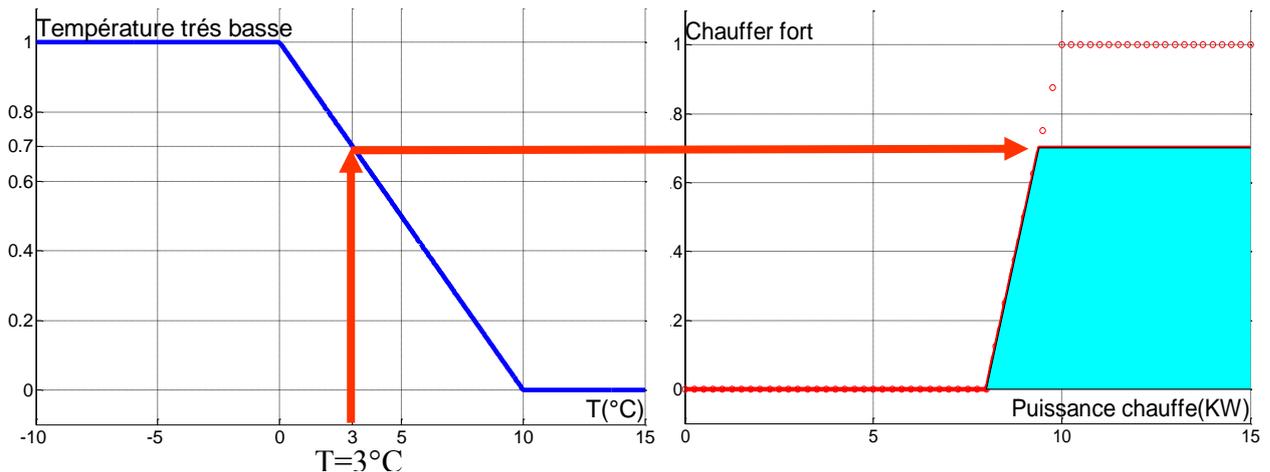
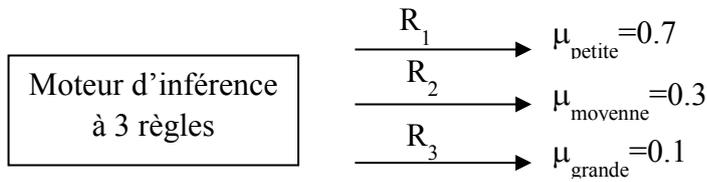


Figure 3.19. Agrégation d'une règle.

Selon la règle considérée, si $T=3\text{ }^{\circ}\text{C}$, son degré d'appartenance en entrée est **0.7**, alors la surface de décision de la sortie sera minimisée jusqu'au degré 0.7.

2) **Pour plusieurs ensembles flous** : On considère un moteur d'inférence à 3 règles qui fournit pour sa sortie S_1 les résultats suivants :



L'agrégation nous donne le résultat de la figure 3.20.

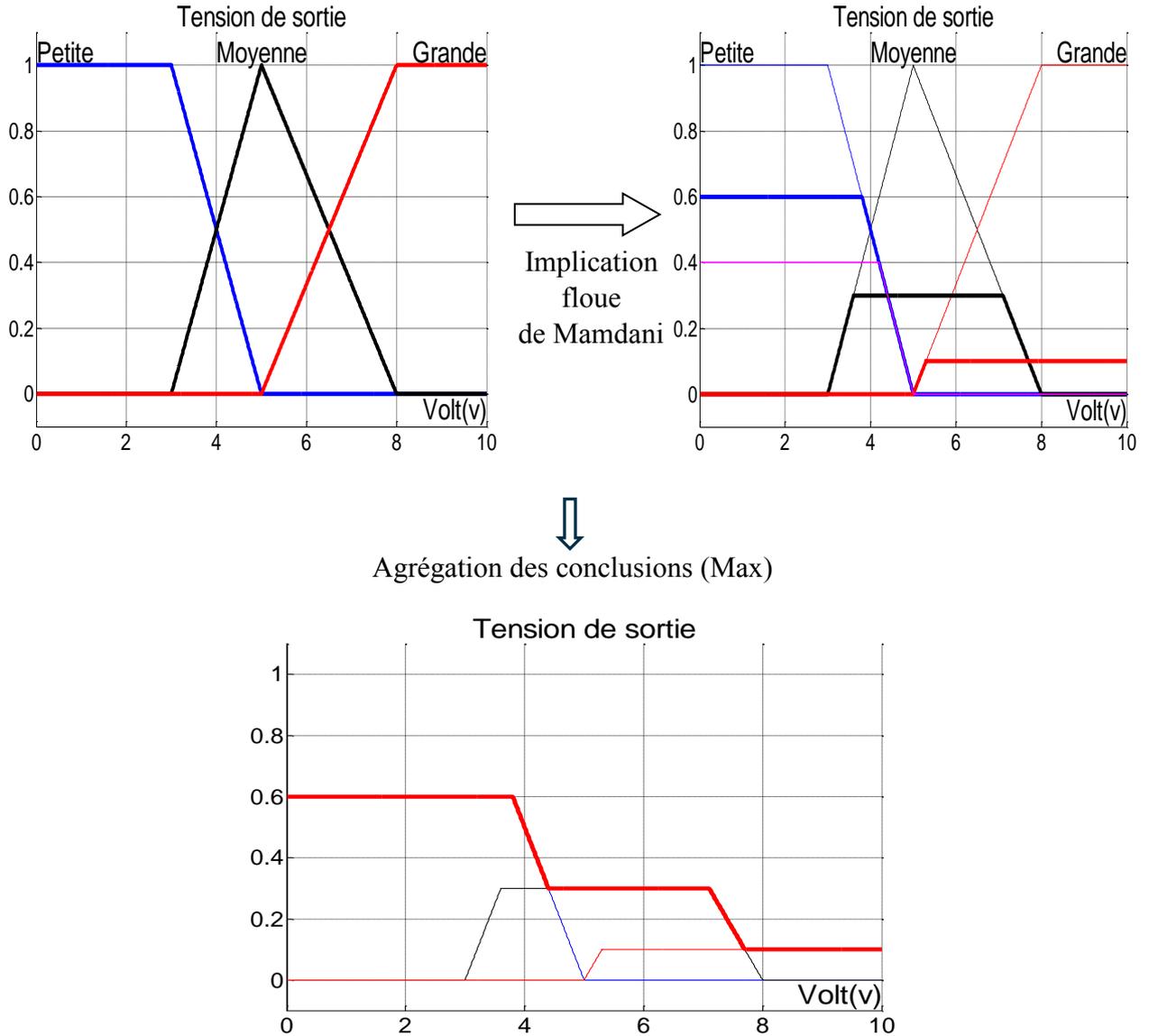


Figure 3.20. Agrégation d'un ensemble de règles.

A ce stade, on a la fonction d'appartenance d'un ensemble flou qui caractérise le résultat et qui est toute une surface de décision générée.

Comment décider quelle valeur de sortie à utiliser ?

Il faut **défuzzifier**, c'est-à-dire associer un nombre interprétable par l'utilisateur à cette surface floue.

3.4.4. La défuzzification

L'objectif de la défuzzification est de transformer un ensemble flou en une valeur de commande.

Soit A un ensemble flou, et $defuzz$ l'opérateur de défuzzification, $z_u = defuzz(A)$, est une valeur précise. Il existe plusieurs méthodes de défuzzification, dont la méthode du Centre de gravité (COG) et la méthode de Moyenne des Maximum (MOM).

a) Centre de gravité : En commande floue, la défuzzification **COG** (Figure 3.21) est presque toujours utilisée. Elle prend en compte l'influence de l'ensemble des valeurs proposées par la solution floue. Elle est présentée par l'abscisse du centre de gravité de la surface sous la courbe résultante :

$$CG = \frac{\int_a^b \mu_A(x) \cdot x}{\int_a^b \mu_A(x)}$$

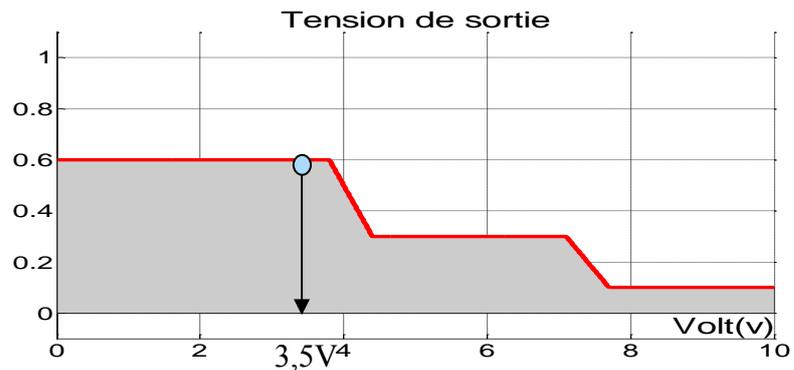


Figure 3.21. Centre de Gravité.

b) Moyenne des Maximum : C'est la moyenne des valeurs de sorties les plus vraisemblables (Figure 3.22). La défuzzification MOM est plutôt utilisée lorsqu'il s'agit de discriminer une valeur de sortie (Ex : reconnaissance de formes).

$$sortie = \frac{\int_S y \cdot dy}{\int_S dy}$$

$$où S = \left\{ y_0 \in U / \mu(y_0) = SUP_{y \in U} (\mu(y)) \right\}$$

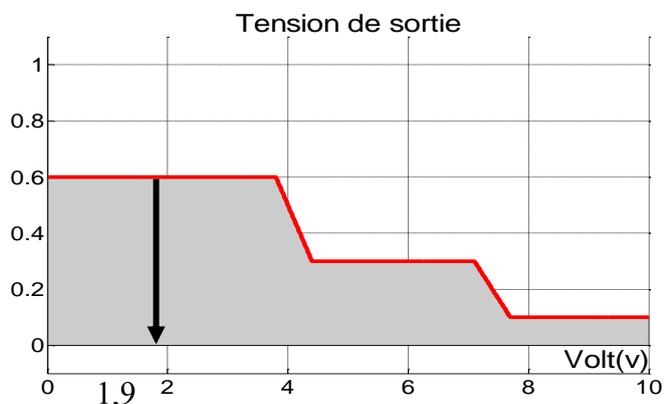
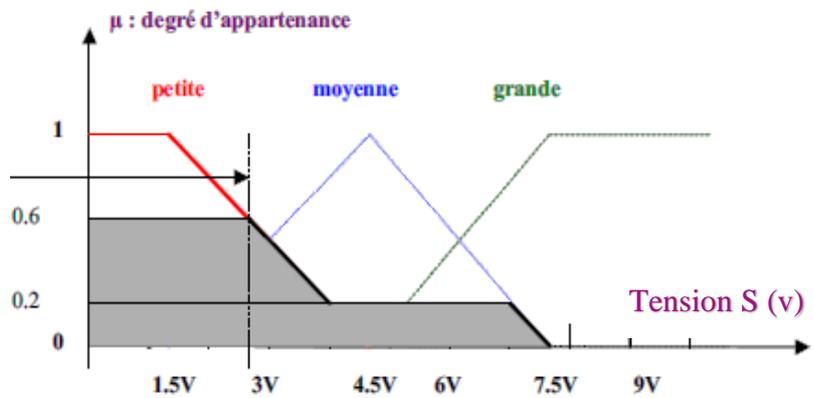


Figure 3.22. Moyenne des Maximum.

3.5. Exercices corrigés

Exercice 1 : sur la figure ci-contre :

1. Calculez le centre gravité
2. Calculez la moyenne des maximums
3. Déterminez la variable linguistique, les valeurs linguistiques et l'univers de discours.



Solution :

Centre de gravité : $((0*0.6)+(3*0.6)+(4*0.2)+(6*0.2))/(0.6+0.6+0.2+0.2) = 2.375$.

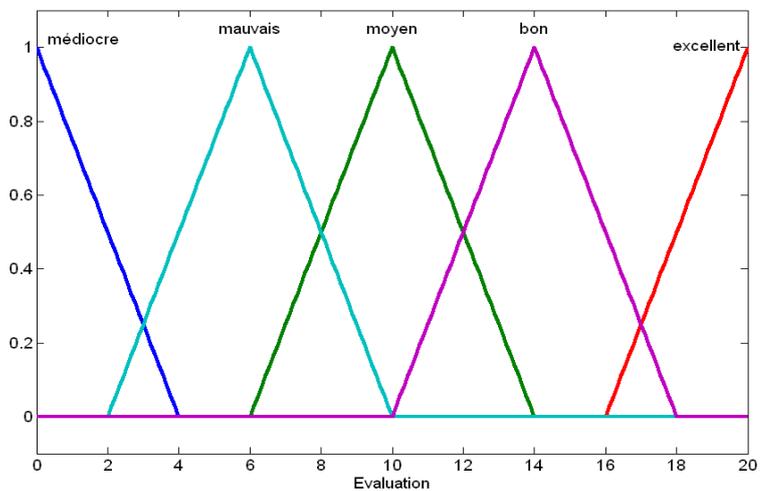
Moyenne des Maximums (on prend la droite des valeurs supérieures et on calcule la moyenne) : $(0+3)/2 = 1.5$.

La variable linguistique : S (Tension), Les valeurs linguistiques : petite, moyenne et grande, L'univers de discours : $[0-10]$ v.

Exercice 2 :

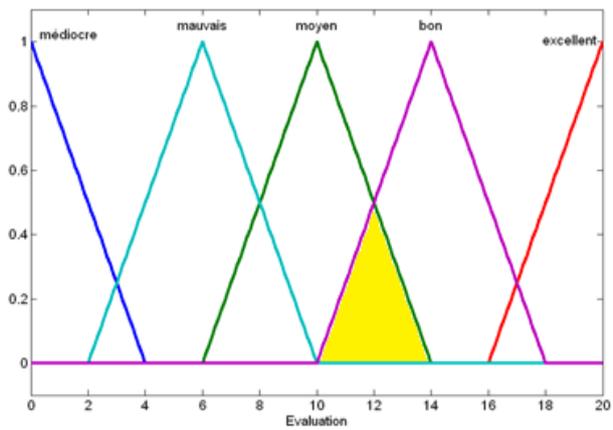
Dans le schéma ci-contre :

1. Schématisez l'ensemble : intersection des deux ensembles moyen et bon.
2. Schématisez l'ensemble : complément de l'ensemble bon.
3. Schématisez l'ensemble : union de tous les ensembles dans le schéma.

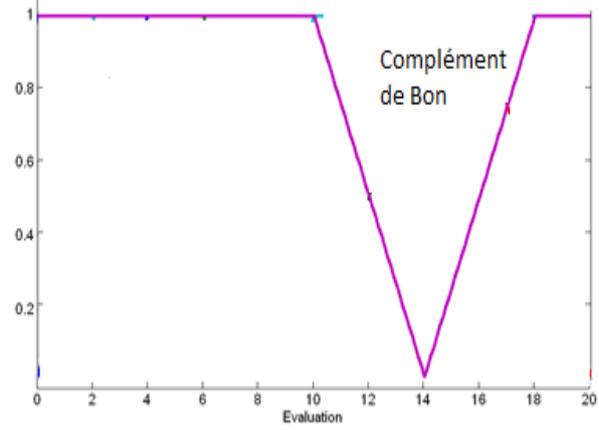


Solution :

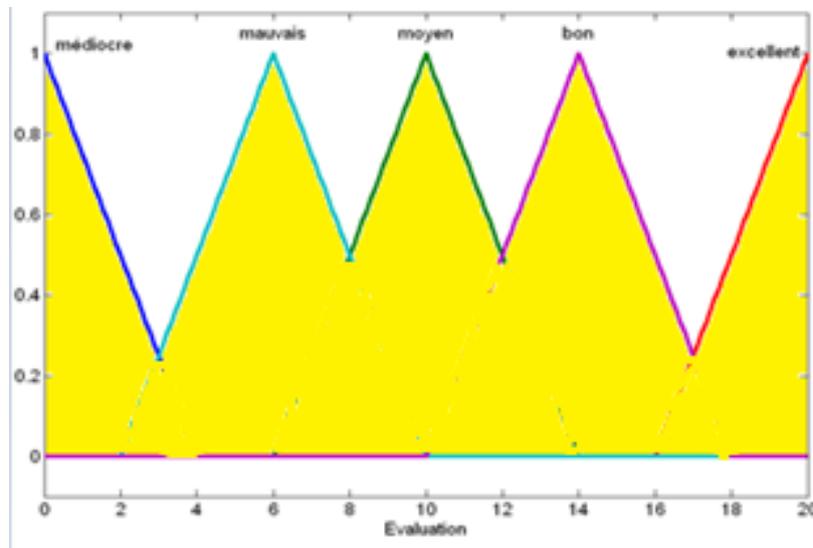
Les schémas sont représentés respectivement sur les figures suivantes :



Intersection des deux ensembles *moyen* et *bon*



Complément de l'ensemble *bon*



Union de tous les ensembles en couleur jaune

Exercice 3 : Contrôle de la puissance d'un chauffage central

1- Description du problème

On souhaite commander un chauffage central d'un immeuble à l'aide d'un contrôleur flou. On dispose de deux sondes de température : l'une à l'extérieur de l'immeuble, l'autre à l'intérieur. Sur la base de ces deux mesures et en faisant appel aux règles d'inférence, le contrôleur flou doit régler la puissance du chauffage.

2- Fuzzification de la température externe

On choisit deux intervalles flous et des fonctions d'appartenance de type trapézoïdales en définissant le «froid» comme correspondant à une température inférieure à 5 °C et le « chaud » comme étant une température supérieure à 20 °C.

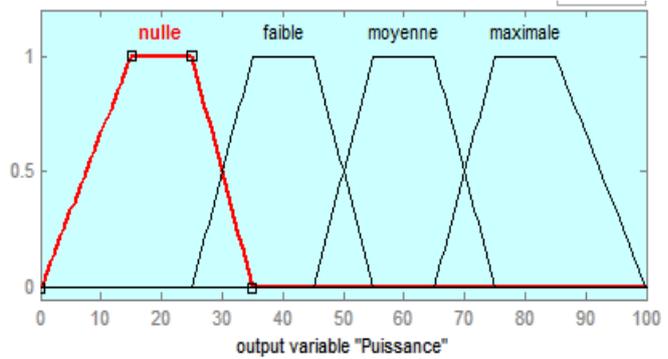
3- Fuzzification de la température interne

On choisit trois intervalles flous et des fonctions d'appartenance de type trapézoïdales en définissant le «froid» comme correspondant à une température inférieure à 15°C, le «bon» comme étant une température comprise entre 19°C et 21°C et le «chaud» comme étant une température supérieure à 25 °C.

4- Fuzzification de la puissance

On choisit quatre intervalles flous pour définir la puissance de l'installation avec des fonctions d'appartenance trapézoïdales. On définit les intervalles suivants :

Puissance	Valeur en %
nulle	0-15-25-35
faible	25 35 45 55
moyenne	45-55-65-75
maximale	65-75-85-100



5- Choix du type de défuzzification

La défuzzification se fait par le calcul du centre de gravité.

6- La Base de règles :

1. **Si** la température extérieure est « froide » **et** la température intérieure est « froide » **alors** mettre la puissance au « maximum »
2. **Si** la température extérieure est « froide » **et** la température intérieure est « bonne » **alors** mettre une puissance « moyenne »
3. **Si** la température extérieure est « froide » **et** la température intérieure est « chaude » **alors** mettre une puissance « faible »
4. **Si** la température extérieure est « chaude » **et** la température intérieure est « froide » **alors** mettre une puissance « moyenne »
5. **Si** la température extérieure est « chaude » **et** la température intérieure est « bonne » **alors** mettre une puissance « faible »
6. **Si** la température extérieure est « chaude » **et** la température intérieure est « chaude » **alors** mettre une puissance « nulle »

Hypothèse : la température extérieure est 10 °C et la température intérieure est 22 °C, quelle sera la puissance (%).

Solution :

1. Fuzzification :

Les valeurs floues de la température extérieure sont donc :

« froide » avec un degré de vérité de 0,67

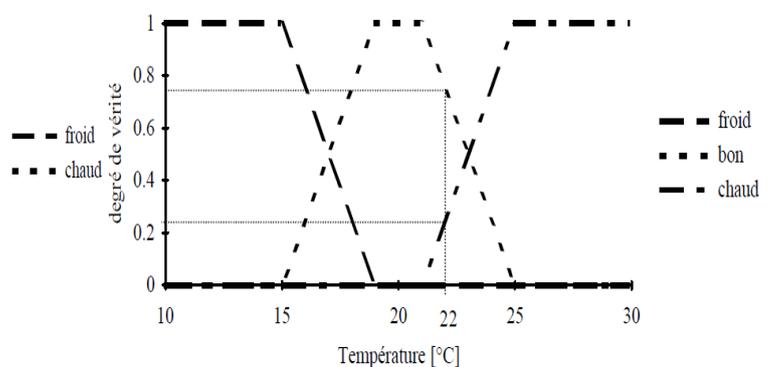
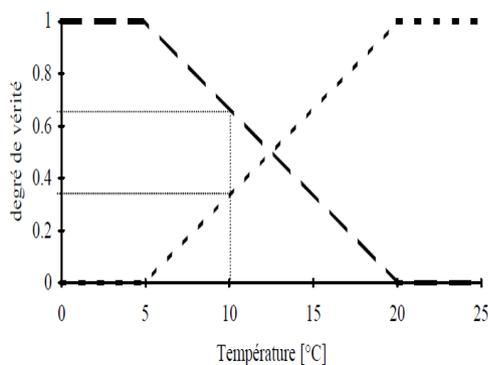
« chaude » avec un degré de vérité de 0,33

Les variables floues de la température intérieure sont donc :

« froide » avec un degré de vérité de 0

« bonne » avec un degré de vérité de 0,75

« chaude » avec un degré de vérité de 0,25



A- Inférence floue : en utilisant la base de règles floues, les règles d'inférences donnent les valeurs suivantes pour les valeurs floues de sortie :

1. « maximum » avec un degré de vérité de 0 (minimum entre 0,67 et 0)
2. « moyenne » avec un degré de vérité de 0,67
3. « faible » avec un degré de vérité de 0,25
4. « moyenne » avec un degré de vérité de 0
5. « faible » avec un degré de vérité de 0,33
6. « nulle » avec un degré de vérité de 0,25

Pour l'agrégation, l'opérateur « Max » appliqué sur les règles qui donnent les mêmes valeurs floues donnent :

1. « maximum » avec un degré de vérité de 0
2. « moyenne » avec un degré de vérité de 0,67 (max entre 0,67 et 0)
3. « faible » avec un degré de vérité de 0,33 (max entre 0,25 et 0,33)
4. « nulle » avec un degré de vérité de 0,25

B- Défuzzification : Le calcul du centre de gravité se calcule à l'aide de :

$$P = \frac{\sum_{i=1}^4 \mu_i \cdot P_i}{\sum_{i=1}^4 \mu_i} = \frac{0 \cdot 100 + 0,67 \cdot 67 + 0,33 \cdot 33 + 0 \cdot 0,25}{0 + 0,67 + 0,33 + 0,25} = 44,6$$

avec P : Puissance du chauffage en %

μ_i : degré de vérité de la variable floue « puissance » indice i

P_i : valeur de la variable floue « puissance » indice i

Le contrôleur flou impose donc une puissance de 44,6 % sur l'installation de chauffage pour une température extérieure de 10 °C et une température intérieure de 22 °C.

3.6. Conclusion

La logique floue offre une meilleure programmation, des possibilités de raffinement extrêmes, et une programmation beaucoup plus instinctive. Cependant, son utilisation demande une certaine expérience.

Cette technique, n'est pas restreinte aux systèmes dont la modélisation est difficile, qui sont contrôlés par des experts humains, ou ceux qui ont plusieurs entrées-sorties et des réponses non-linéaires. Elle est intéressante aussi dans tous les domaines où un « flou » persiste, notamment dans les domaines de l'économie et de la science.

Plusieurs outils et logiciels peuvent nous aider à réaliser des systèmes flous comme :

- Matlab (Mathworks) : en utilisant Toolbox Fuzzy logic, par la commande fuzzy.
- FuzzyControl++ (Siemens) : est un outil spécialement pour la création des systèmes flous. Ces derniers sont chargés dans des automates programmables Siemens.
- LABVIEW : FuzzyToolkit (Automatisation).
- Fuzzyclips : extension de CLIPS (C Language Integrated Production System). Il peut traiter le raisonnement exact, flou (ou inexact), permettant des termes flous et normales à être hybridés dans des règles et des faits d'un système expert.
-

Chapitre 4 : Les approches d'apprentissage

4.1. Introduction

L'un des objectifs majeurs de l'IA est la résolution de problèmes complexes et issus de domaines variés. Pour résoudre un problème donné, nous avons souvent besoin de classer et de manipuler des connaissances du domaine concerné. Parmi les méthodes ou les outils de classification intelligents existes K-means, K-PPV, les RNA et SVM.

La classification est la catégorisation algorithmique d'objets. Elle consiste à attribuer une classe ou catégorie à chaque objet (ou individu) à classer, en se basant sur des données statistiques. Elle fait couramment appel à l'apprentissage automatique et est largement utilisée en reconnaissance de formes. Parmi les techniques de classification, existent celles qui sont basées sur l'apprentissage supervisé et celles basées sur l'apprentissage non supervisé. Quelques techniques sont représentées sur la figure 4.1.

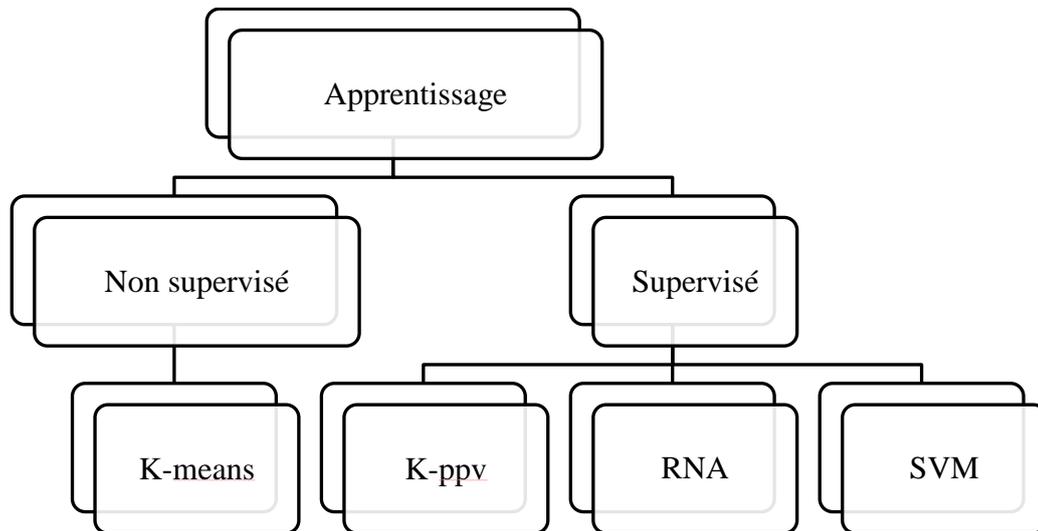


Figure 4.1. Techniques d'apprentissage.

K-means (K-moyenne) est basée sur le principe de classification par centre mobiles, plus connue sous le nom de K-means. Chaque centre mobile indique le représentant de la classe, il peut être calculé avec une fonction mathématique comme le centre de gravité, la distance euclidienne, ou autre fonction. A l'arrivée d'une nouvelle observation, elle sera affectée à la classe qui a une distance minimale entre son représentant et cette observation. La particularité de la méthode des "k-means", c'est que le nombre de classes doit être spécifié préalablement. Cependant, deux cas particuliers se présentent dans cette méthode. Le premier est dans le cas où les deux distances entre l'observation et les représentants des classes sont égales, ou connus sous le nom de rejet d'ambiguïté. Le deuxième apparaît si la nouvelle observation est jugée comme très lointaine des représentants des classes et s'appelle un rejet à distance. Une des solutions proposées dans ce cas

dans la littérature, est de proposer d'autres nouvelles classes. Ceci nécessite de refaire la classification. Par conséquent, cette solution peut prendre plus de temps. Ainsi, comme cette méthode appartient aux méthodes de classification non supervisée, l'inconvénient majeur est que le taux d'erreur de classification des nouvelles observations est maximisé.

K-PPV (K-Plus Proche Voisin) est une méthode de discrimination non paramétrique, aucune estimation des paramètres n'est nécessaire à son exécution. Elle s'emploie sur les données continues. L'idée principale est d'observer les K les plus proches d'une nouvelle observation afin de décider de l'appartenance de cette observation. Pour classer une observation, nous devons faire un calcul de la distance entre la nouvelle observation et chaque point dans la base d'apprentissage. Après, les K voisins ayant la distance la plus faible avec la nouvelle observation seront sélectionnés. Au vu des classes d'appartenance des K plus proche voisins, on décide sur la classe d'appartenance de cette nouvelle observation.

4.2. Historique des réseaux de neurones artificiels

L'évolution des réseaux de neurones artificiels jusqu'à nos jours, a passé par plusieurs étapes :

1890 : W. James, célèbre psychologue américain introduit le concept de mémoire associative, et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb.

1943 : J. Mc Culloch et W. Pitts laissent leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ce sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique).

1949 : D. Hebb, physiologiste américain explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement de type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraîne chez cet animal la sécrétion de salive à cette heure précise même en l'absence de nourriture. La loi de modification des propriétés des connexions entre neurones qu'il propose explique en partie ce type de résultats expérimentaux.

➤ Les premiers succès :

1957 : F. Rosenblatt développe le modèle du Perceptron. Il construit le premier neuro ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance de formes. Notons qu'à cette époque les moyens à sa disposition sont limités et c'est une prouesse technologique que de réussir à faire fonctionner correctement cette machine plus de quelques minutes.

1960 : B. Widrow, un automaticien, développe le modèle Adaline (Adaptative Linear Element). Dans sa structure, le modèle ressemble au Perceptron, cependant la loi d'apprentissage est

différente. Celle-ci est à l'origine de l'algorithme de rétropropagation de gradient très utilisé aujourd'hui avec les Perceptrons multicouches. Les réseaux de type Adaline restent utilisés de nos jours pour certaines applications particulières. B. Widrow a créé dès cette époque une des premières firmes proposant neuro-ordinateurs et neuro-composants, la "Memistor Corporation".

1969 : M. Minsky et S. Papert publient un ouvrage qui met en exergue les limitations théoriques du perceptron. Limitations alors connues, notamment concernant l'impossibilité de traiter par ce modèle des problèmes non linéaires. Ils étendent implicitement ces limitations à tous modèles de réseaux de neurones artificiels. Leur objectif est atteint, il y a abandon financier des recherches dans le domaine (surtout aux U.S.A), les chercheurs se tournent principalement vers l'IA et les systèmes à bases de règles.

➤ **L'ombre**

1969-1982 : Toutes les recherches ne sont, bien sûr, pas interrompues. Elles se poursuivent, mais déguisées, sous le couvert de divers domaines comme : le traitement adaptatif du signal, la reconnaissance de formes, la modélisation en neurobiologie, etc. De grands noms travaillent durant cette période telle : S. Grossberg, T. Kohonen, ...

➤ **Le renouveau**

1982 : J. J. Hopfield, à travers un article court, clair et bien écrit, a présenté une théorie du fonctionnement et des possibilités des réseaux de neurones. J. J. Hopfield fixe préalablement le comportement à atteindre pour son modèle et construit à partir de là, la structure et la loi d'apprentissage correspondant au résultat escompté. Ce modèle est aujourd'hui encore très utilisé pour des problèmes d'optimisation. Notons qu'à cette date, l'IA est l'objet d'une certaine désillusion, elle n'a pas répondu à toutes les attentes et s'est même heurtée à de sérieuses limitations. Aussi, bien que les limitations du Perceptron mise en avant par M. Minsky ne soient pas levées par le modèle d'Hopfield, les recherches sont relancées.

➤ **La levée des limitations**

1983 : La Machine de Boltzmann est le premier modèle connu apte à traiter de manière satisfaisante les limitations recensées dans le cas du perceptron. Mais l'utilisation pratique s'avère difficile, la convergence de l'algorithme étant extrêmement longue (les temps de calcul sont considérables).

1985 : La rétropropagation de gradient apparaît. C'est un algorithme d'apprentissage adapté aux réseaux de neurones multicouches (Perceptrons multicouches). Sa découverte réalisée par trois groupes de chercheurs indépendants indique que "la chose était dans l'air". Dès cette découverte, nous avons la possibilité de réaliser une fonction non linéaire d'entrée/sortie sur un réseau en décomposant cette fonction en une suite d'étapes linéairement séparables. De nos jours, les réseaux multicouches et la rétropropagation de gradient reste le modèle le plus étudié et le plus productif au niveau des applications.

4.3. Le neurone formel

Le premier neurone formel a été modélisé par W. Mc Culloch et W. Pitts, en 1943. Un neurone formel ou artificiel est un opérateur mathématique très simple. Un neurone possède des entrées qui peuvent être les sorties d'autres neurones, ou des entrées de signaux extérieures, et une sortie. La valeur de la sortie résulte du calcul de la somme des entrées, pondérées par des coefficients (dits poids de connexions ou poids synaptiques) et du calcul d'une fonction non linéaire (dite fonction d'activation) de cette somme pondérée. L'état du neurone, appelé aussi activité, est défini comme la somme pondérée de ses entrées. Son schéma de fonctionnement est donné en Figure 4.2. L'information est ainsi transmise de manière unidirectionnelle. Un neurone se caractérise par trois concepts : son *Etat*, ses *Connexions* avec d'autres neurones et sa *Fonction d'activation*.

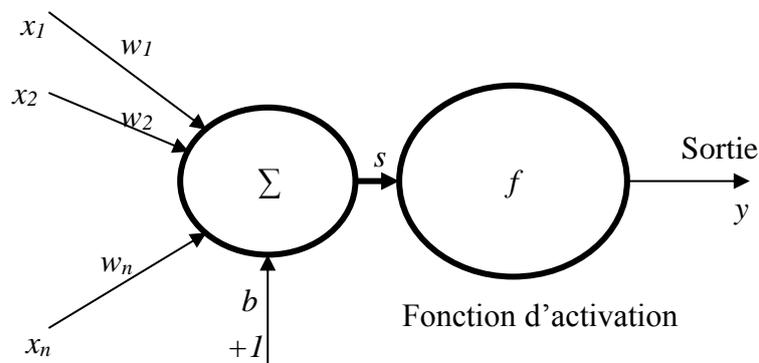


Figure 4.2. Neurone formel.

Où :

- x_i : les entrées du neurone.
- w_i : les poids de connexion entre les neurones (coefficients de pondération). Si w_i est positif, l'entrée x_i est excitatrice alors que si w_i est négatif, elle est inhibitrice.
- b : le poids de la connexion entre le neurone biais (+1) et les neurones i .
- f : la fonction d'activation associée au neurone.
- y : la sortie du neurone.

La fonction d'activation f est du type tout ou rien à seuil prenant les valeurs 0 et 1. Le seuil de déclenchement est en général provoqué par une entrée b appelée biais. Les fonctions d'activation les plus utilisées sont présentées par la figure 4.3.

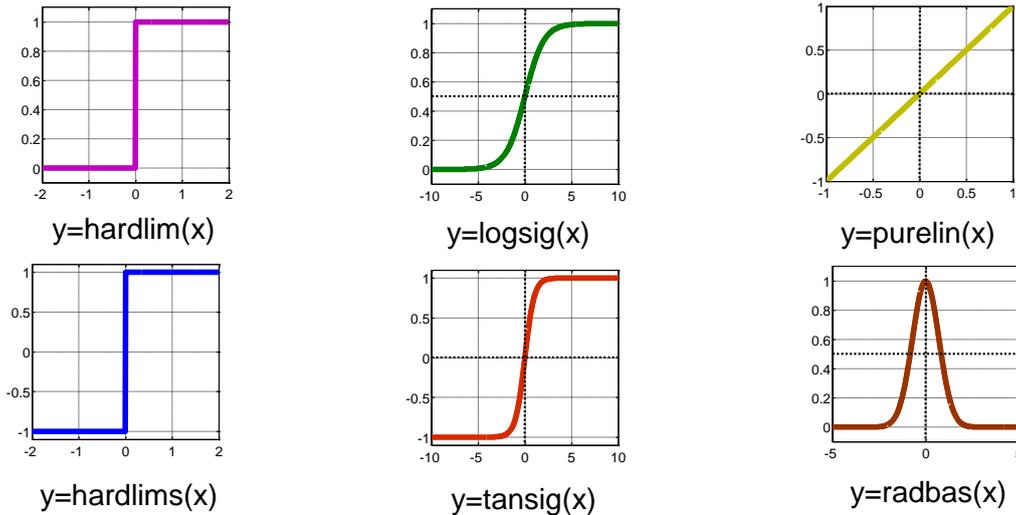


Figure 4.3. Les fonctions d'activation les plus utilisées.

4.4. Perceptron (premier réseau de neurones)

Le réseau le plus simple pour classer des données en deux classes est constitué d'un seul neurone binaire. Introduit par Frank Rosenblatt en 1958, qui l'a nommé perceptron (Figure 4.4). La sortie du perceptron dépend de la somme des composantes x_i du vecteur d'entrée, pondérées par des poids $w_i \in R$. Cette somme pondérée est appelée potentiel.

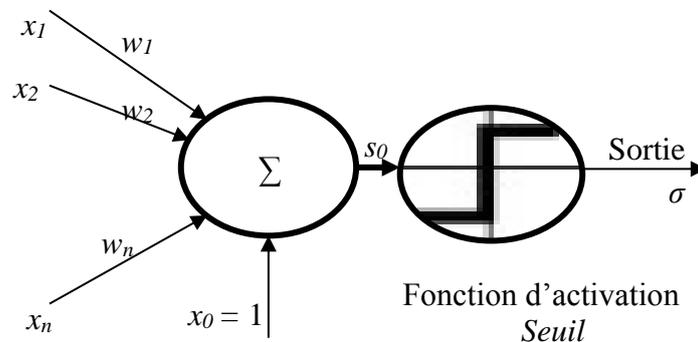


Figure 4.4. Schéma du perceptron.

Le potentiel étant une fonction affine des entrées, on appelle aussi le perceptron « séparateur linéaire ». Si le potentiel dépasse le seuil du neurone, s_0 , la sortie du perceptron est $\sigma = +1$. Autrement, elle est $\sigma = -1$. Le perceptron est donc un neurone dont la fonction d'activation est une fonction à seuil. Une entrée constante $x_0 = 1$ est incluse dans l'ensemble des entrées du perceptron affectée d'un poids w_0 . Celui-ci a pour effet de déplacer la valeur du seuil de la fonction d'activation. Chaque entrée ayant une composante supplémentaire $x_0 = 1$, cela équivaut à considérer un espace des entrées élargi, de dimension $N+1$. Le potentiel se note par la formule:

$$s_0 = \sum_{i=0}^N w_i x_i = w \cdot x. \quad (1)$$

$$\text{La sortie est donnée par : } \sigma = \text{signe}(s_0) \text{ (Drefus 2002).} \quad (2)$$

4.4.1. Types d'apprentissage

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Par définition l'apprentissage est une phase de développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage a comme objectif l'amélioration des performances futures du réseau, sur la base d'une connaissance acquise au fur et à mesure des expériences passées. Il existe principalement deux types d'apprentissage : l'apprentissage supervisé et l'apprentissage non supervisé.

1) Apprentissage supervisé

L'apprentissage est dit supervisé (Figure 4.5) lorsque les exemples sont constitués de couples de valeurs de type (valeur d'entrée, valeur de sortie désirée).

Étant donné un vecteur d'apprentissage de n couples (x_i, y_i) , $i=1,2,\dots, n$. Il est à déterminer les vecteurs des poids des neurones capables de donner le même vecteur de sortie à partir des mêmes vecteurs d'entrées. Ce type d'apprentissage nécessite la présence d'un superviseur qui présente au réseau ces entrées et leurs sorties désirées. Il est nécessaire de déterminer, dans ce genre d'apprentissage, la fonction qui mesure l'écart entre les sorties désirées et celles fournies par le réseau. Cette fonction appelée **fonction objectif**, constitue le critère à minimiser en agissant sur les poids synaptiques. Ce critère est en général une fonction de l'erreur entre la sortie désirée et la réponse du réseau : $E = \psi(t - \sigma)$. (3)

Où :

- ↪ E : l'erreur entre la sortie désirée et la réponse du réseau ;
- ↪ Ψ : la fonction d'activation ;
- ↪ t : la sortie désirée ;
- ↪ σ : représente l'estimation du réseau.

Ainsi l'apprentissage devient une procédure qui cherche à minimiser ce critère à travers la recherche de la séquence de poids synaptiques favorables.

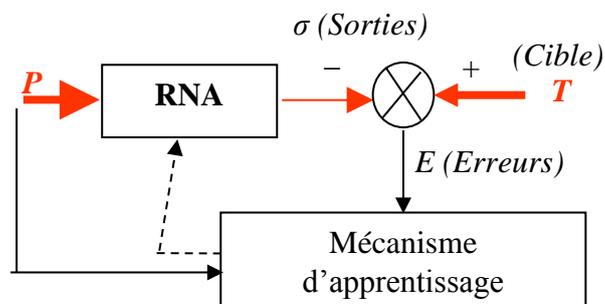


Figure 4.5. Apprentissage supervisé.

2) Apprentissage non supervisé :

L'apprentissage est qualifié de non supervisé (Figure 4.6) lorsque seules les valeurs d'entrée sont disponibles. Dans ce cas, les exemples présentés à l'entrée provoquent une auto adaptation du réseau afin de produire des valeurs de sortie qui soient proche en réponse pour des valeurs d'entrées similaires.

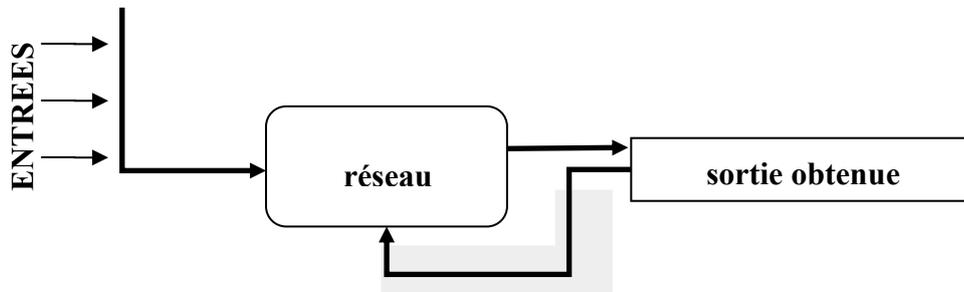


Figure 4.6. Apprentissage non supervisé.

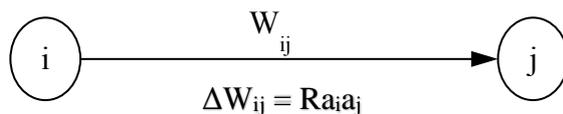
4.4.2. Règle d'apprentissage

L'apprentissage consiste à modifier le poids des connexions entre les neurones. Il existe plusieurs règles de modification :

- Loi de Hebb : $\Delta w_{ij}=R a_i a_j$
- Règle de Widrow-Hoff (delta rule) : $\Delta w_{ij}=R(d_i - a_i)a_j$
- Règle de Grossberg : $\Delta w_{ij}=R(a_j - w_{ij})a_i$

1- Loi de Hebb : Si deux unités connectées sont actives simultanément, le poids de leur connexion est augmenté ou diminué. R est une constante positive qui représente la force d'apprentissage (Learning rate).

	$a_i = -1$	$a_i = 1$
$a_j = -1$	$\Delta W_{ij} = R$	$\Delta W_{ij} = -R$
$a_j = 1$	$\Delta W_{ij} = -R$	$\Delta W_{ij} = R$

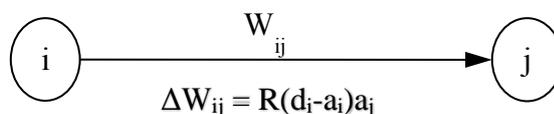


2- Loi de Widrow-Hoff (delta rule) :

Considérons : a_i activation produite par le réseau, et d_i réponse désirée par l'expert humain.

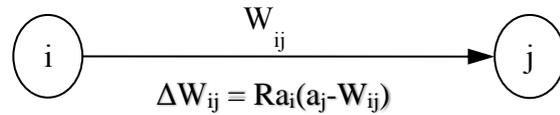
Par exemple si la sortie est inférieure à la réponse désirée, il va falloir augmenter le poids de la connexion à condition bien sûr que l'unité j soit excitatrice (égale à 1). On est dans l'hypothèse d'unités booléennes {0, 1}.

	$a_i = 0$	$a_i = 1$
$d_j = 0$	$\Delta W_{ij} = 0$	$\Delta W_{ij} = -R$
$d_j = 1$	$\Delta W_{ij} = -R$	$\Delta W_{ij} = 0$



3- Loi de Grossberg :

On augmente les poids qui entrent sur l'unité gagnante a_i s'ils sont trop faibles, pour les rapprocher du vecteur d'entrée a_j . C'est la règle d'apprentissage utilisée dans les cartes auto-organisatrices de Kohonen.



4.4.3. Exemple (ET, OU et XOR)

Le perceptron est un classifieur binaire. Nous voulons l'appliquer sur les opérations logiques ET, OU et OU exclusif.

Considérons un perceptron avec deux entrées x_1, x_2 et deux poids correspondants, w_1 et w_2 , et une fonction de Seuil θ ($y = 1$ si $s > \theta$ et $y = 0$ si $s \leq \theta$).

L'opération logique ET (AND) : l'opération ET est présentée par sa table de vérité.

x_1	x_2	$s = x_1 \text{ ET } x_2$
1	1	1
1	0	0
0	1	0
0	0	0

Dans le plan (x_1, x_2) , elle définit quatre points (Figure 4.7) relatifs aux quatre combinaisons possibles des valeurs des variables x_1 et x_2 . Pour ce perceptron, les deux représentations donnent deux classes linéairement séparables.

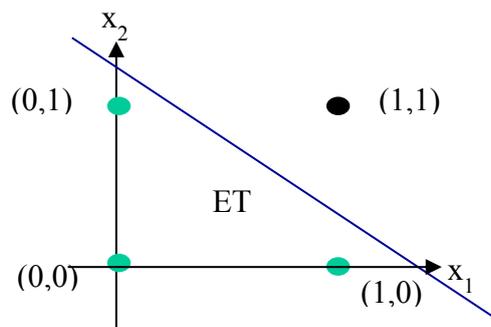


Figure 4.7. Représentation de l'opération logique ET.

L'opération logique OU (OR) : l'opération OU présentée par sa table de vérité.

x_1	x_2	$x_1 \text{ OU } x_2$
1	1	1
1	0	1
0	1	1
0	0	0

Dans le plan (x_1, x_2) , elle définit quatre points (Figure 4.8) relatifs aux quatre combinaisons possibles des valeurs des variables x_1 et x_2 . Pour ce perceptron, la représentation donne deux classes linéairement séparables.

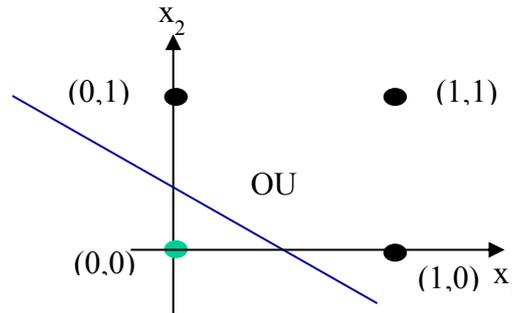


Figure 4.8. Représentation de l'opération logique OU.

Le « OU » Exclusif (XOR) : L'apprentissage de la fonction logique OU exclusif (XOR) entre deux variables est un problème classique de test du perceptron multicouches. Il s'agit d'un problème de classification, dont les deux classes «0 » ou «1 » ne sont pas linéairement séparables. La table de vérité relative à l'opération logique OU exclusif est la suivante :

x_1	x_2	x_1 OU x_2
1	1	0
1	0	1
0	1	1
0	0	0

Dans le plan (x_1, x_2) , elle définit quatre points (figure 4.8) relatifs aux quatre combinaisons possibles des valeurs des variables x_1 et x_2 .

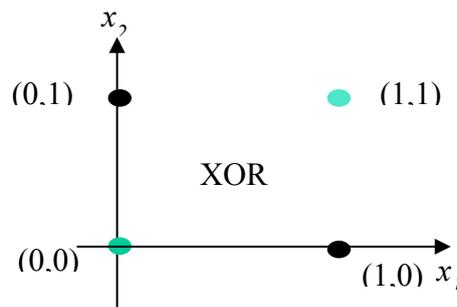


Figure 4.9. Représentation de l'opération logique OU exclusif (XOR).

4.4.4. Limitation du perceptron

A partir des exemples précédents, le perceptron ne peut pas apprendre le XOR. Il est incapable de distinguer les patterns non séparables linéairement [Minsky 69]. Ce problème peut être résolu à l'aide d'un perceptron multicouches.

Le problème d'apprendre une classification binaire d'un ensemble de données revient à les séparer en deux groupes. Dans l'espace à deux dimensions, chaque donnée correspond à un point dans l'espace 2D et il s'agit de trouver une ligne droite les séparant. Pour généraliser, dans l'espace à 3 dimensions, il s'agit de trouver un plan. Ainsi de suite, dans l'espace à n dimensions, un ensemble de données est linéairement séparable s'ils peuvent être séparés par un hyperplan de dimension $n-1$.

4.5. Réseau de neurones multicouches

Le neurone lui-même, en tant qu'unité autonome élémentaire n'a aucun pouvoir. La force et l'efficacité du cerveau résident en effet, dans le regroupement de ces neurones et le partage des tâches entre eux. Ce regroupement est appelé champs de neurones où chaque élément reçoit et envoie de l'information. L'organisation de plusieurs champs entre eux, constitue un réseau de neurones, dont les unités à l'intérieur doivent travailler ensemble pour remplir une certaine tâche bien déterminée. Ces neurones sont reliés entre eux par des connexions. Les mécanismes de cette organisation déterminent l'architecture du réseau.

Les neurones sont arrangés par couche. On place ensuite bout à bout plusieurs couches et l'on connecte les neurones de deux couches adjacentes. Les entrées des neurones de la deuxième couche sont en fait les sorties des neurones de la couche amont. Les neurones de la première couche sont reliés au monde extérieur et reçoivent le vecteur d'entrée. Ils calculent alors leurs sorties qui sont transmises aux neurones de la seconde couche qui calculent eux aussi leurs sorties et ainsi de suite de couche en couche jusqu'à celle de sortie. Il peut y avoir une ou plusieurs sorties à un réseau de neurones.

Dans un réseau multicouche classique (Figure 4.10), il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones de la couche aval. Tous les neurones de la couche amont sont connectés à tous les neurones de la couche aval, de prendre un nouveau départ. Les couches extérieures du réseau sont appelées respectivement couches d'entrée et de sortie ; les couches intermédiaires sont appelées couches cachées.

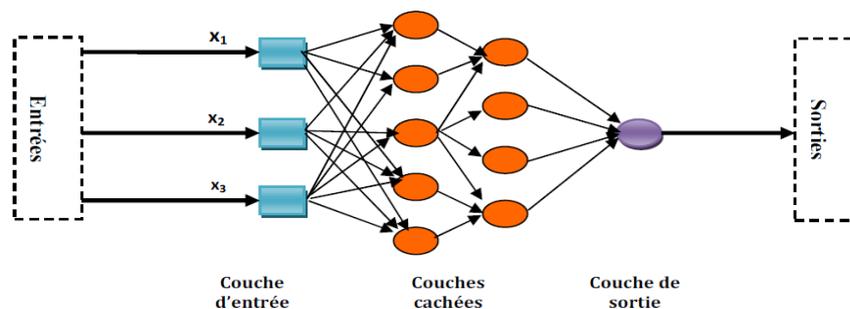


Figure 4.10. Réseau multicouche.

Il existe plusieurs architectures de réseaux de neurones multicouches, parmi elles :

A. Réseau à connexions locales

C'est aussi un réseau multicouche, mais tous les neurones d'une couche amont ne sont pas connectés à tous les neurones de la couche aval. Nous avons donc dans ce type de réseau de neurones un nombre de connexions moins important que dans le cas du réseau de neurones multicouche classique (Figure 4.11).

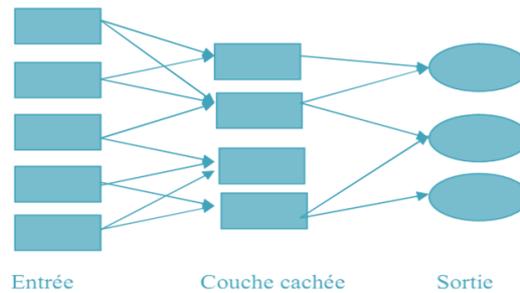


Figure 4.11. Réseau à connexions locales.

B. Réseau à connexions récurrentes

Un réseau de ce type signifie qu'une ou plusieurs sorties de neurones d'une couche aval sont connectées aux entrées des neurones de la couche amont ou de la même couche. Ces connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Les réseaux à connexions récurrentes sont des réseaux plus puissants car ils sont séquentiels plutôt que combinatoires comme l'étaient ceux décrits précédemment. La rétroaction de la sortie vers l'entrée permet à un réseau de ce type de présenter un comportement temporel (Figure 4.12).

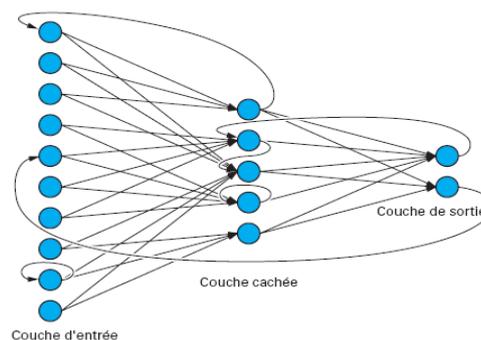


Figure 4.12. Réseau à connexions récurrentes.

C. Réseau à connexions complexes

Chaque neurone est connecté à tous les neurones du réseau y compris lui-même, c'est la structure d'interconnexion la plus générale (Figure 4.13).

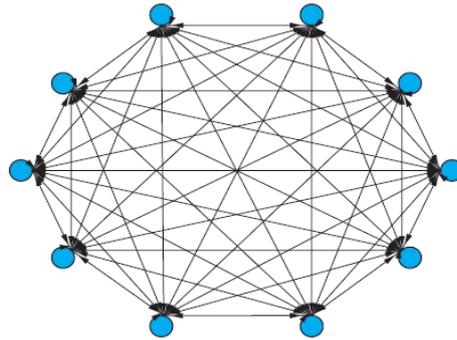


Figure 4.13. Réseau à connexions complexes.

Les réseaux de neurones les plus répandus sont les réseaux perceptrons multicouches MLP, et les réseaux de neurones à fonction radiales de bases RBF.

4.5.1. Perceptron multicouches (MLP)

Les réseaux de neurones artificiels perceptrons multicouches MLP consistent à cascader un certain nombre de perceptrons en plusieurs couches avec des fonctions de décision différentiables reliées entre elles par des coefficients synaptiques (poids) et sont organisés de la manière suivante : une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie (Figure 4.14).

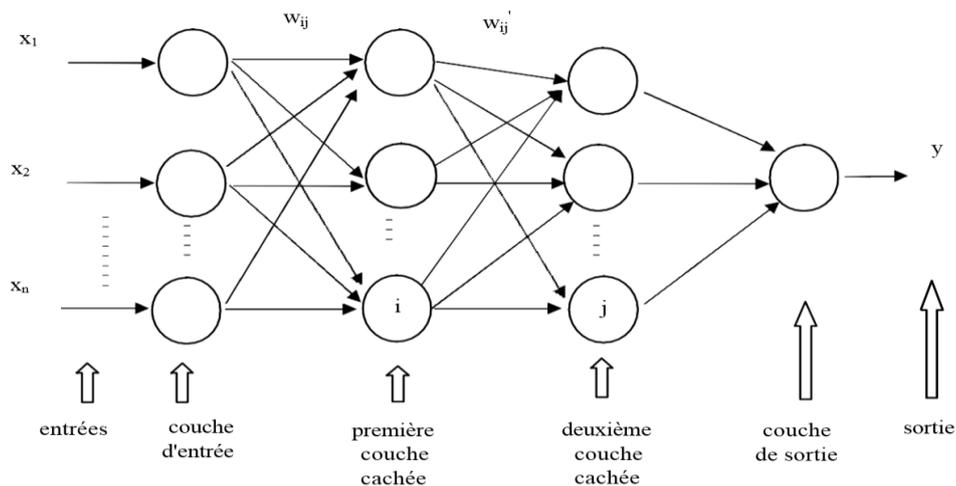


Figure 4.14. Perceptron multicouches (MLP) avec deux couches cachées.

Les neurones de deux couches consécutives sont entièrement connectés. Le nombre total de couches d'un réseau est en général donné par celui des couches cachées plus la couche de sortie. L'apprentissage de ces réseaux consiste à trouver les valeurs optimales des différents vecteurs de poids. L'approche classique la plus utilisée est la rétro-propagation (back-propagation). Mathématiquement, cette méthode est basée sur l'algorithme de la descente du gradient et utilise les règles de dérivation des fonctions dérivables. Dans cette méthode, l'erreur commise en sortie du réseau est rétro-propagée vers les couches cachées, d'où le nom de rétro-propagation. En effet, la convergence de l'algorithme est basée sur l'ajustement de multiples variables, citons par exemple

: la structure du réseau, le nombre maximum d'itérations d'apprentissage, le coefficient d'apprentissage, nature de la fonction d'activation, nombre de couches cachées, la taille de la couche cachée.

Néanmoins, ces réseaux sont connus pour avoir un certain nombre d'inconvénients (la sensibilité aux minimums locaux lors de l'étape d'optimisation des poids qui empêche la convergence et cause l'oscillation de l'erreur, et l'absence de résultats théoriques satisfaisantes qui permettent de dimensionner correctement un réseau (combien de couches cachées faut-il utiliser? et combien doit-il y avoir de neurones dans chacune des couches cachées ?), seule l'expérience permet de donner une idée sur ce choix. La sortie de l'unité j par exemple de la couche q est donnée par :

$$Y_j^q = f\left(\sum_{i=1}^{N_{q-1}} W_{ij}^q Y_i^{q-1}\right) \quad (4)$$

où W_{ij}^q représente le poids entre le neurone i de la couche $q-1$ et le neurone j de la couche q , Y_i^{q-1} est la sortie de l'unité i dans la couche $q-1$, et f représente la fonction d'activation ou de décision pour toutes les unités des couches cachées. C'est une fonction différentiable non linéaire. Le plus souvent, la fonction d'activation utilisée est la fonction sigmoïde définie pour tout réel x

$$\text{par [30] : } f(x) = \frac{1}{1+e^{-x}} \quad (5)$$

4.5.2. Réseaux à fonction radiales de bases (RBF)

Parmi les méthodes alternatives permettant de tenir compte de la non linéarité, mais conceptuellement plus simples sont les réseaux de neurones artificiels à fonctions radiales de base RBF (Figure 4.15). La construction d'un réseau de neurones artificiel RBF est rapide et facile, et c'est là le principal avantage de ce type de réseau de neurones (Douha 2013).

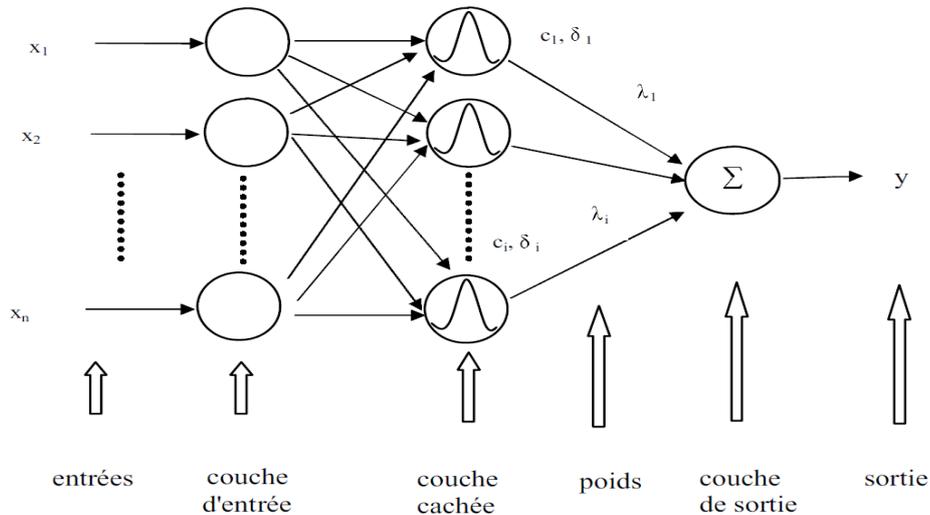


Figure 4.15. Réseau de neurones à fonctions radiales de base (RBF).

Un réseau de neurones artificiel RBF comporte deux couches de neurones ; une couche cachée dont les neurones sont connectés à ceux de la couche d'entrée par des connexions non pondérées, et une couche de sortie dont les neurones sont connectés à ceux de la couche cachée par des connexions pondérées. Dans la couche de sortie, la fonction d'activation est linéaire. La sortie réalisée par le réseau est exprimée sous la forme d'une combinaison linéaire de fonctions radiales [30] :

$$y(x) = \sum_{j=1}^k \lambda_j \Phi(\|x - c_j\|) \quad (6)$$

où k représente le nombre de neurones dans la couche cachée, λ_j est le poids associé à la connexion entre le neurone j de la couche cachée et la couche de sortie et c_j est la valeur des centres. Dans la couche cachée, la fonction radiale la plus couramment utilisée est le noyau gaussien [30] :

$$\Phi(\|x - c_j\|) = \exp\left(-\frac{(\|x - c_j\|)^2}{2\delta_j^2}\right) \quad (7)$$

où δ_j est la largeur associée au noyau j , $x=(x_1, \dots, x_n)$ et $c_i=(c_{i1}, \dots, c_{in})$ représentent le vecteur d'entrée et le vecteur "centre" propre à chaque neurone respectivement. Dans un réseau de neurones artificiel RBF il y a quatre paramètres principaux à régler.

1. Le nombre de neurones k dans la couche cachée.
2. La position des centres des gaussiennes c de chacun des neurones.
3. La largeur de ces gaussiennes δ_j (généralement l'écart-types des noyaux gaussiens)
4. Les poids des connexions λ_i entre les neurones de la couche cachée et les neurones de la couche de sortie.

La procédure d'apprentissage dans ce cas est divisée en deux étapes, un apprentissage non supervisé dans la couche cachée suivi par un apprentissage supervisé dans la couche de sortie. L'algorithme des K-means et la méthode des moindres carrés sont souvent utilisées pour la sélection des centres "c" et l'adaptation des poids de connexions λ respectivement. Comparés aux réseaux multicouches, l'apprentissage des réseaux RBF est plus rapide. En effet cette rapidité vient du fait que ce type de réseaux possède deux couches seulement ($c+\delta$ et λ) (Douha 2013).

4.6. SVM (Machine à Vecteur de Support)

4.6.1. Définition

Les machines à vecteurs de support SVM (Support Vector Machine) développées au cours des années 90 par Vapnik, sont une famille d'algorithmes d'apprentissage destinées à résoudre les problèmes de classification et de régression et la détection des valeurs aberrantes qui sont aujourd'hui considérées comme une des méthodes les plus performantes sur de nombreux

problèmes réels, notamment pour les problèmes en grande dimension. L'efficacité de SVM est due à sa base théorique solide qui constitue l'un de ses atouts (Douha 2013).

Les SVM possèdent plusieurs avantages, parmi eux :

- Efficace dans les espaces de grande dimension.
- Toujours efficace dans les cas où le nombre de dimensions est supérieur au nombre d'échantillons.
- Utilise un sous-ensemble de points d'apprentissage dans la fonction de décision (appelés vecteurs de support), donc il est également efficace en mémoire.
- Polyvalent : différentes fonctions du noyau peuvent être spécifiées pour la fonction de décision. Des noyaux communs sont fournis, mais il est également possible de spécifier des noyaux personnalisés.

Les SVM possèdent aussi des inconvénients, comme :

- Leurs besoins de calcul et de stockage augmentent rapidement avec le nombre de vecteurs de formation.
- Si le nombre de fonctionnalités est beaucoup plus grand que le nombre d'échantillons, évitez de sur-ajuster dans le choix des fonctions du noyau et le terme de régularisation est crucial.
- Ne fournissent pas directement d'estimations de probabilité, celles-ci sont calculées à l'aide d'une validation croisée coûteuse.

4.6.2. Principe

L'algorithme des machines à vecteurs de support a initialement été développé comme un algorithme de classification binaire par apprentissage supervisé. L'algorithme sous sa forme initiale revient à chercher une frontière de décision linéaire appelé hyperplan qui permet de regrouper les données similaires dans une même classe et de séparer les données hétérogènes, en garantissant que la marge entre l'hyperplan et les points les plus proches de chaque classe soit maximale.

Le problème de recherche de l'hyperplan séparateur optimal possède une formulation duale. Ceci est particulièrement intéressant car, sous cette formulation duale, le problème peut être résolu au moyen de méthodes d'optimisation quadratique standard. L'intérêt de cette méthode est la sélection de vecteurs supports qui représentent les vecteurs discriminants grâce auxquels est déterminé l'hyperplan. Les exemples utilisés lors de la recherche de l'hyperplan ne sont alors plus utiles et seuls ces vecteurs supports sont utilisés pour classer un nouveau cas, ce qui peut être considéré comme un avantage pour cette méthode.

Une machine à vecteurs de support construit un hyper-plan ou un ensemble d'hyperplans dans un espace de dimension élevée ou infinie, qui peut être utilisé pour la classification, la régression ou d'autres tâches. Intuitivement, une bonne séparation est obtenue par l'hyper-plan qui a la plus grande distance des points de formation les plus proches de n'importe quelle classe (marge dite fonctionnelle), car plus la marge est grande, plus l'erreur de généralisation du classificateur est faible (Douha 2013).

4.6.3. Fonction du noyau (kernel)

Séparer n'importe quel jeu de données par un simple hyperplan est un problème qui n'est pas évident. Les SVM sont forts dans le cas où les données sont linéairement séparables. C'est une limitation sévère qui condamne à ne pouvoir résoudre que des problèmes particuliers. Cependant, dans la plupart des problèmes réels, ce n'est pas toujours le cas et il est donc nécessaire de contourner ce problème. Dans ce but, l'idée clé de la méthode SVM est de changer l'espace des données.

Le principe consiste à projeter les données d'apprentissage x_i dans un espace de dimension (d'), plus élevée que celle de l'espace d'origine (d) grâce à une fonction non linéaire $\Phi(x)$ qu'on appelle fonction noyau, choisie a priori et de réaliser une séparation linéaire dans le nouvel espace. L'espace ainsi obtenu est appelé espace des caractéristiques ou aussi espace transformé ou espace d'attributs ou encore espace de redescription [45]. La figure 4.16 donne une illustration de ce principe.

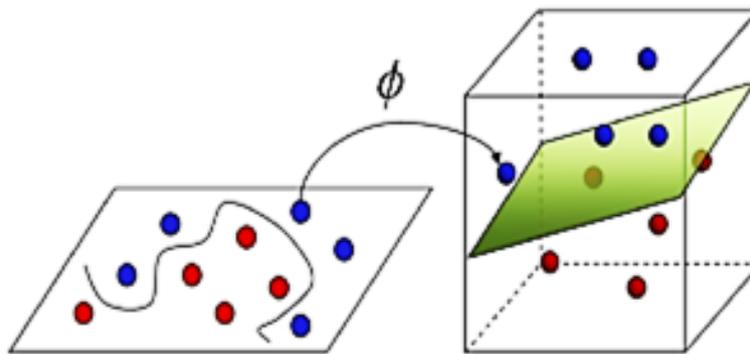


Figure 4.16 . Formes du séparateur hyperplan de SVM.

En d'autre terme, L'idée est de transformer un problème de séparation non linéaire dans l'espace original en un problème de séparation linéaire dans l'espace de redescription de plus grande dimension. En effet, intuitivement, plus la dimension de l'espace de redescription est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée (Douha 2013).

4.6.4. Extensions aux cas multi-classe

Dans le cas où la variable de sortie y compte plus de deux classes, il existe plusieurs façons d'étendre directement les méthodes du cas binaire.

- **Un contre un** : Dans le cas où l'on cherche à prédire un label pouvant $K \geq 3$ classes, on peut considérer toutes les paires de labels (k, l) possibles, $1 \leq k < l \leq K$ (il y en a C_K^2) et ajuster un classifieur $C_{k,l}(x)$ pour chacune d'entre elles. La prédiction correspond alors au label qui a gagné le plus de "duels".
- **Un contre tous** : Pour chaque classe k , on apprend un classifieur permettant de discriminer entre les populations $Y = k$ et $Y \neq k$. A partir des estimations des probabilités a posteriori, on affecte le label estimé le plus probable.

4.6.5. Exemple de classification

Si les données sont sous forme vectorielle, on peut utiliser un noyau classique. On peut aussi construire un noyau spécifique qui travaille plus directement sur la représentation initiale (structure).

➤ **Images 2D** : choix d'une résolution et d'une discrétisation des couleurs.

- Vecteur des valeurs des pixels,
- Coefficients d'une transformée en ondelettes,
- Histogramme des couleurs.

➤ **Textes** : choix d'un dictionnaire (suppression des mots simples).

- Sac de mots : vecteur des occurrences,
- Autres attributs (liens, position des mots...).

➤ **Séquences d'ADN**

- Fenêtres de taille fixe (sous-séquence) et représentation binaire (un bit par valeur possible).

```

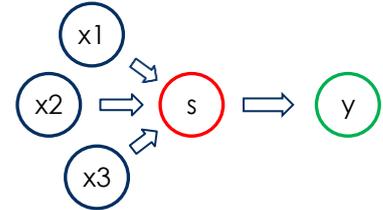
A T A G C A
1 0 1 0 0 1
0 1 0 0 0 0   ⇒ (1,0,0,0, 0,1,0,0, 1,0,0,0, 0,0,1,0, 0,0,0,1, 1,0,0,0)
0 0 0 1 0 0
0 0 0 0 1 0
    
```

- Coefficients d'un modèle probabiliste générateur.

4.7. Exercices corrigés

Exercice 1 : Soit le neurone artificiel avec 3 entrées et 1 sortie :

- Une fonction de combinaison : produit scalaire.
- Une fonction d'activation : linéaire.



Calculez la réponse y à l'excitation $x = (5 \ 2 \ 1)$ et les poids $w = (1 \ -5 \ 2)$.

Solution:

```
x=[0 1 5]
w=[1 -2 6]
s=x*w
s=28
y=s=28
```

Exercice 2 : Programmez un neurone artificiel à 3 entrées et 1 sortie, avec :

- Une fonction de combinaison : distance euclidienne.
- Une fonction d'activation : radiale de base, avec $\gamma=1$.

Calculez la réponse y en utilisant un script Matlab à l'excitation : $x=(0 \ 1 \ 5)$ et les poids $w=(1 \ -2 \ 6)$.

Solution :

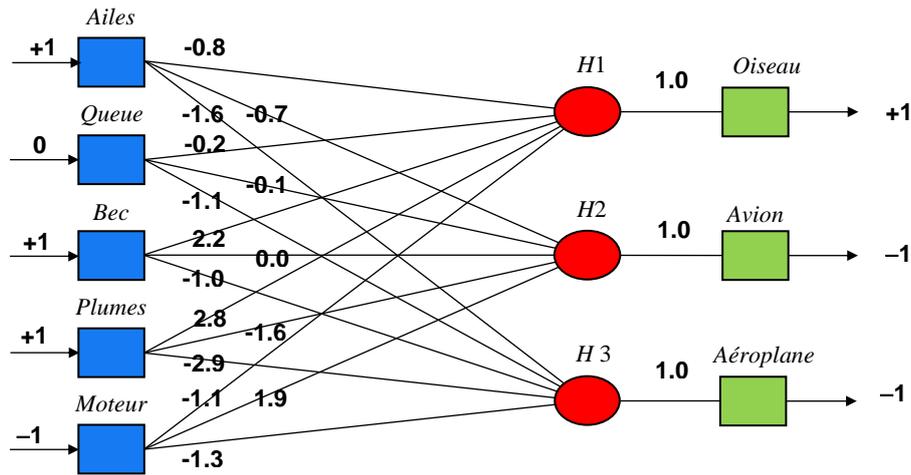
```
x=[0 1 5]
w=[1 -2 6]
s=abs(x-w)
s=sqrt(sum(s.*s))
s =3.3166
y=exp(-s*s)
y = 1.6702e-005
```

Exercice 3 :

Sur le réseau de neurones suivant, en utilisant l'application d'entrée par sommation des pondérations et la fonction d'activation Threshold (Seuil), Comment conclure que :

- L'objet de sortie est un Oiseau.
- Il ne s'agit pas d'un Avion.
- Il ne s'agit pas d'un Aéroplane.

N.B : Les valeurs d'entrée = +1 (vrai), -1 (faux), et 0 (indéterminé).



Solution :

En utilisant la fonction Seuil, avec les valeurs d'entrée +1 (vrai), -1 (faux), ou 0 (indéterminé), il est possible de donner une interprétation sémantique à l'activation de tout neurone de sortie.

$$X_{H1} = 1 * (-0.8) + 0 * (-0.2) + 1 * 2.2 + 1 * 2.8 + (-1) * (-1.1) = 5.3 > 0$$

$$Y_{H1} = Y_{Oiseau} = +1$$

$$X_{H2} = 1 * (-0.7) + 0 * (-0.1) + 1 * 0.0 + 1 * (-1.6) + (-1) * 1.9 = -4.2 < 0$$

$$Y_{H2} = Y_{Avion} = -1$$

$$X_{H3} = 1 * (-0.6) + 0 * (-1.1) + 1 * (-1.0) + 1 * (-2.9) + (-1) * (-1.3) = -4.2 < 0$$

$$Y_{H3} = Y_{Aeroplane} = -1$$

Si l'objet d'entrée possède des ailes (+1), un bec (+1) et des plumes (+1), mais pas de moteur (-1), nous pouvons conclure qu'il s'agit d'un oiseau (+1) :

Exercice 4 : Rétro-propagation du gradient

On a un réseau 3-2-3,

- Première couche (entrée) = 3 neurones d'entrée,
- Deuxième couche (cachée) = 2 neurones cachés,
- Troisième couche (sortie) = 3 neurones de sortie,
- $X1 = 0.9, X2 = 0.1, X3 = 0.9$ doivent correspondre aux sorties : $Y1 = 0.1, Y2 = 0.9, Y3 = 0.9$

Étape 1 : Initialisation du réseau

Les poids sont initialisés aléatoirement sur $[-m, +m]$, On augmente les corrections apportées lors de la rétro-propagation.

Étape 2 : Propagation avant

1- Calculez les potentiels pour chaque neurone de forme : $P_i = \sum w_{ji} * s_j$

- 2- Pour chaque neurone de la couche cachée calculez son signal utilisant la fonction sigmoïde

$$\psi(s) = \frac{1}{1 + \exp(-2s)}$$

$$d\psi/ds = 2\psi(s)(1 - \psi(s))$$

- 3- On propage les résultats de la couche cachée vers la couche de sortie, calculez les potentiels des neurones de sortie.
4- Calculez le signal des neurones de sortie.

Étape 3 : calcul de l'erreur

Calculez l'erreur afin de voir si le réseau a convergé en utilisant la formule : $E = \frac{1}{2} \sum (d_o - s_o)^2$

Étape 4 : calcul du signal d'erreur sur la couche de sortie

Si le réseau n'a pas convergé, on doit calculer le signal d'erreur sur les neurones de sortie

$$\delta_o = (d_o - s_o) * \varphi(p_o)$$

- 1- Calculez la dérivé, sachant que pour la fonction logistique : $\varphi(p_o) = s_o * (1 - s_o)$
2- Calculez les signaux d'erreur.

(On a besoin des poids initiaux pour rétro-propager l'erreur de la sortie sur les neurones de la couche cachée.)

Étape 5 : calcul du signal d'erreur sur la couche cachée

$$\delta_h = \varphi(p_h) * \sum_{o=1}^o \delta_o * w_{ho}$$

Le calcul de la dérivée est identique à la couche de sortie.

Pour chaque neurone de la couche cachée, on doit cumuler les signaux d'erreur des neurones de sortie qui leur sont liés, pondérés par les coefficients synaptiques : H_1 et H_2

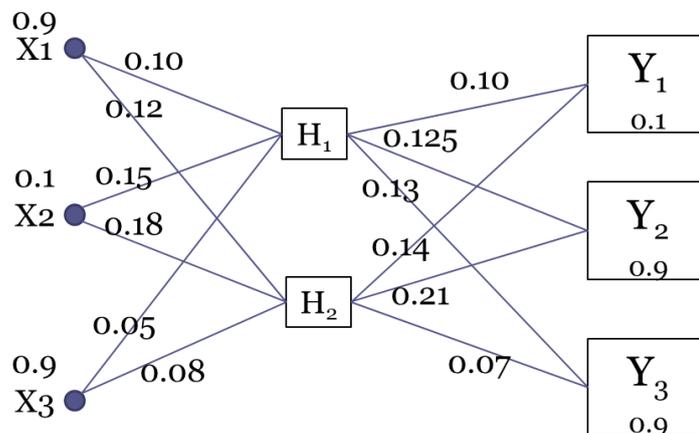
- 1- Calculez les signaux d'erreur : δ_{h1} et δ_{h2}

Étape 6 : Correction des poids synaptiques de la couche de sortie $W_{hi,yj}$ et d'entrée $W_{xi,hj}$

La correction utilise les fonctions : $w_{ho}(t+1) = w_{ho}(t) + \Delta w_{ho}$, $\Delta w_{ho} = \eta * \delta_o * s_h$ Avec $\eta = 1$

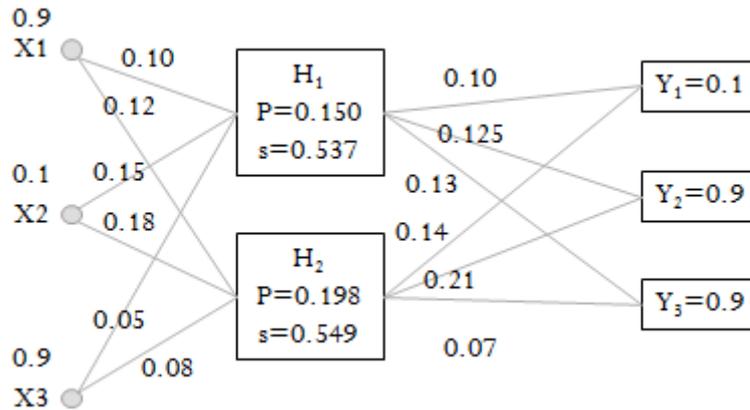
Solution :

Étape 1 : Initialisation du réseau

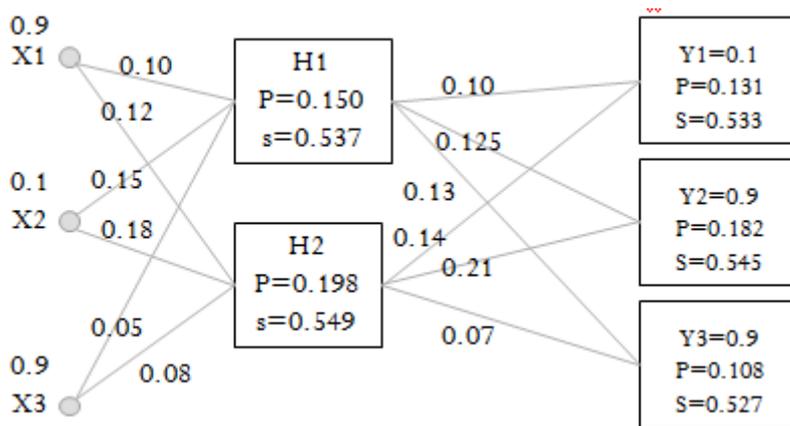


Étape 2 : Propagation avant

- Calculons les potentiels pour chaque neurone de forme : $P_i = \sum w_{ji} * s_j$
 - $P_{h1} = 0.9*0.10+0.1*0.15+0.9*0.05=0.150$
 - $P_{h2} = 0.9*0.12+0.1*0.18+0.9*0.08=0.198$
- Chaque neurone de la couche cachée calcul son signal utilisant la fonction sigmoïde :
 - $S_{h1} = 1/(1+e^{-0.150}) = 0.537$,
 - $S_{h2} = 1/(1+e^{-0.198}) = 0.549$



- On propage les résultats de la couche cachée vers la couche de sortie, on calcul les potentiels des neurones de sortie :
 - $P_{y1} = 0.537*0.10+0.549*0.14=0.131$
 - $P_{y2} = 0.537*0.125+0.549*0.21=0.182$
 - $P_{y3} = 0.537*0.130+0.549*0.07=0.108$
- On calcul le signal des neurones de sortie :
 - $S_{y1} = 1/(1+e^{-0.131}) = 0.533$,
 - $S_{y2} = 1/(1+e^{-0.182}) = 0.545$,
 - $S_{y3} = 1/(1+e^{-0.130}) = 0.527$,



Étape 3 : calcul de l'erreur

Afin de voir si le réseau a convergé : $E = \frac{1}{2} \sum (d_o - s_o)^2$

- $Y_1 = 0.1$ (sortie désiré) - 0.533 (sortie actuelle) = -0.433
- $Y_2 = 0.9 - 0.545 = 0.355$
- $Y_3 = 0.9 - 0.527 = 0.373$

$\Rightarrow E = 0.5 * (-0.433^2 + 0.355^2 + 0.373^2) = 0.226$

Si l'erreur est jugée suffisamment faible, on considère que l'apprentissage est terminé.

Étape 4 : calcul du signal d'erreur sur la couche de sortie

Si le réseau n'a pas convergé, on doit calculer le signal d'erreur sur les neurones de sortie

$\delta_o = (d_o - s_o) * \varphi'(p_o)$, on calcul la dérivé, sachant que pour la fonction logistique : $\varphi'(P_o) = s_o * (1 - s_o)$

$Y_1 = 0.533 * (1 - 0.533) = 0.249$

$Y_2 = 0.545 * (1 - 0.545) = 0.248$

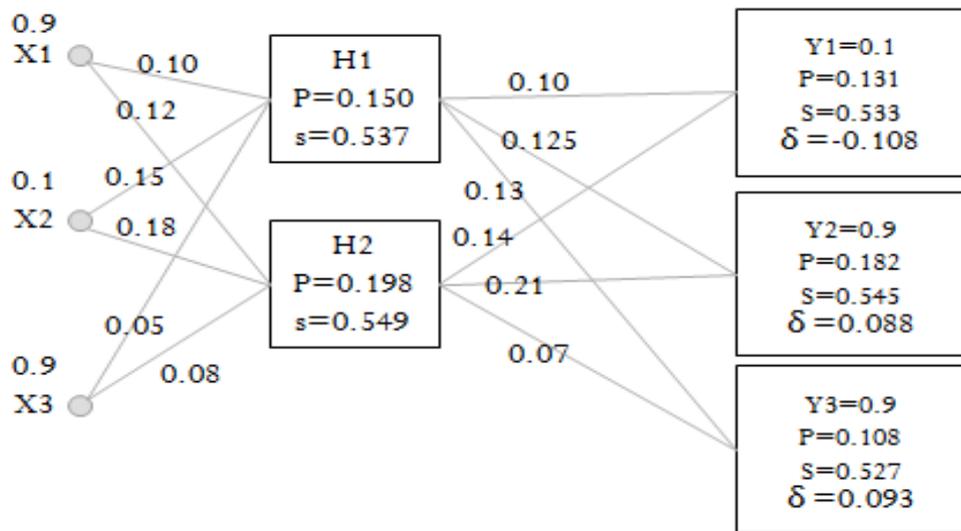
$Y_3 = 0.527 * (1 - 0.527) = 0.249$

Les signaux d'erreur :

$\delta_{y1} = -0.433 * 0.249 = - 0.108$

$\delta_{y2} = 0.355 * 0.248 = 0.088$

$\delta_{y3} = 0.373 * 0.249 = 0.093$



On a besoin des poids initiaux pour rétro-propager l'erreur de la sortie sur les neurones de la couche cachée.

Étape 5 : calcul du signal d'erreur sur la couche cachée

$$\delta_h = \varphi'(P_h) \cdot \sum_{o=1}^O \delta_o \cdot w_{ho}$$

Le calcul de la dérivée est identique à la couche de sortie.

- $H_1 = 0.537 * (1 - 0.537) = 0.249$
- $H_2 = 0.549 * (1 - 0.549) = 0.247$

Pour chaque neurone de la couche cachée, on doit cumuler les signaux d'erreur des neurones de sortie qui leur sont liés, pondérés par les coefficients synaptiques :

$H_1 = -0.108 * 0.10 + 0.088 * 0.125 + 0.093 * 0.13 = 0.0123$

$H_2 = -0.108 * 0.14 + 0.088 * 0.21 + 0.093 * 0.07 = 0.00987$

Ce qui ne donne les signaux d'erreur suivants :

$\delta_{h1} = 0.249 * 0.0123 = 0.0031$ et

$\delta_{h2} = 0.247 * 0.00987 = 0.0024$

Étape 6 : Correction des poids synaptiques de la couche de sortie

La correction utilise les fonctions : $w_{ho}(t+1) = w_{ho}(t) + \Delta w_{ho}$, $\Delta w_{ho} = \eta \cdot \delta_o \cdot s_h$ Avec $\eta = 1$

$w_{h1,y1} = 0.10 + (-0.108 * 0.537) = 0.042$

$w_{h1,y2} = 0.125 + (0.088 * 0.537) = 0.172$

$$W_{h1,y3} = 0.130 + (0.093 * 0.537) = 0.080$$

$$W_{h2,y1} = 0.140 + (-0.108 * 0.549) = 0.081$$

$$W_{h2,y2} = 0.210 + (0.088 * 0.549) = 0.258$$

$$W_{h2,y3} = 0.070 + (0.093 * 0.549) = 0.121$$

Étape 7 : Correction des poids synaptiques de la couche d'entrée

La correction utilise les fonctions : $w_{ih}(t+1) = w_{ih}(t) + \Delta w_{ih}$, $\Delta w_{ih} = \eta \cdot \delta_{oh} \cdot s_i$ Avec $\eta = 1$

$$W_{x1,h1} = 0.10 + (0.0031 * 0.9) = 0.103$$

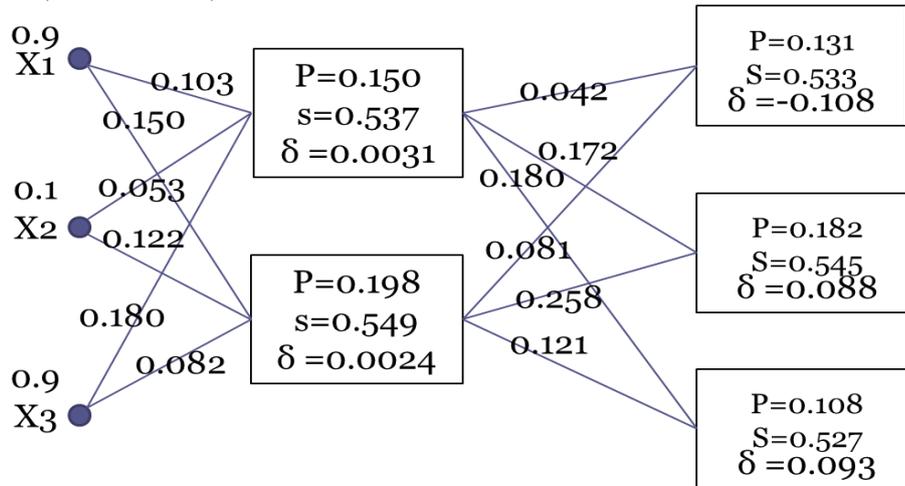
$$W_{x2,h1} = 0.15 + (0.0031 * 0.1) = 0.150$$

$$W_{x3,h1} = 0.05 + (0.0031 * 0.9) = 0.053$$

$$W_{x1,h2} = 0.12 + (0.0024 * 0.9) = 0.122$$

$$W_{x2,h2} = 0.18 + (0.0024 * 0.1) = 0.180$$

$$W_{x3,h2} = 0.08 + (0.0024 * 0.9) = 0.082$$



- Les nouveaux poids sont presque les mêmes poids initiaux,
- On peut juger que l'erreur est suffisante et le réseau converge,
- On peut faire une deuxième itération pour avoir les poids exacts de ce réseau de neurone.

4.8. Conclusion

L'apprentissage est l'opération qui consiste à adapter les poids des connexions en fonction des stimuli présentés à la couche d'entrée. Les règles d'apprentissage indiquent comment les poids évoluent en réponse à un exemple. Pendant la phase d'exploitation ou de rappel du réseau, on présente à l'entrée du réseau un nouveau vecteur et on obtient après la réponse apprise par le réseau.

Les mécanismes d'apprentissage sont les propriétés les plus intéressantes des réseaux de neurones car certains d'entre eux tentent de copier le processus de mémorisation des connaissances du cerveau humain. Cette mémorisation est essentiellement assurée par les poids synaptiques qui sont ajoutés, supprimés ou au contraire atténués lors de la phase d'apprentissage.

Références Bibliographiques

- O. Boisard, Cours d'Intelligence Artificielle, Cours, Ecole Centrale de Lille, France, 2014.
- P. J. Denning, Towards a science of expert systems, IEEE Expert, 1, No. 2, pp. 80–83 (Summer, 1986).
- F. Denis, Cours d'intelligence artificielle, Systèmes experts, Lille, France, 2002.
- H. Farreny, Les Systèmes Experts, principes et exemples, Techniques Avancées de l'Informatique. Paris. Edition : Berti, 1989.
- B. Meunier, La logique floue, Édition : Presses Universitaires de France – PUF, France, 2007.
- L. Gacogne, Logique floue et application, Editions : CANM, Institut d'Informatique d'Entreprise d'Evry, France, 2003.
- H. Zermane, Télécontrôle et Techniques d'IA pour un Système de Conduite à Distance, Thèse de doctorat, Département Génie Industrie, Université Batna 2, 2017.
- G. Dreyfus J.-M. Martinez M. Samuelides M. B. Gordon F. Badran S. Yhiria L. Hérault, Réseaux de neurones: méthodologie et applications, Eyrolles 2002.
- L. Douha, Analyse des données en grande dimension en utilisant des modèles linéaires et non linéaires, Thèse de Doctorat, Département Electronique, Université Batna 2, 2013.

Références bibliographiques proposées

1. S. Russell, P. Norvig, Intelligence artificielle: avec plus de 500 exercices, livre, Pearson Education, France, 2010.
2. L. Frécon, O. Kazar, Manuel d'intelligence artificielle, PPUR Presses Polytechniques, 2010.
3. B. Faltings, M. Schumacher, L'intelligence artificielle par la pratique, Edition : PPUR Presses Polytechniques, 2009.
4. P. Habermehl, Intelligence Artificielle, Cours Master Informatique, Université Paris 7, 2006.
5. Meghyn, Introduction à l'Intelligence Artificielle, Cours L3, Université Aix-Marseille 1, 2008.
6. P. Marquis, O. Papini and H. Prade, Panorama de l'Intelligence Artificielle- Ses bases méthodologiques, ses développements, vol. 3, Edition : Cépaduès, France, 2014.
7. J. L. Laurière, Intelligence Artificielle, Résolution de problèmes par l'homme et la machine, Editions : Eyrolles, France, 2006.
8. J.C. Pomerol, Les Systèmes Experts, Technologies de Pointe. Paris. Edition : Hermes, 1988.
9. K. Arnold, Le pramétrage des moteurs d'inférence, Traité des nouvelles technologies, Série Mathématiques Appliquées. Edition : Hermes, France, 1988.

10. L. Gacogne, Logique floue et application, Editions : CANM, Institut d'Informatique d'Entreprise d'Evry, France, 2003.
11. C. Earl, La Logique Floue Pour les affaires et l'industrie. [trad.] Maurice Clerc. Edition : International Thomson Publishing, France, 1997.
12. F. Chevrie, F. Guély, La logique Floue, Extrait du Cahier Technique Schneider n° 191. Edition : Schneider Electric, 1998.
13. F. Chevalier et J. Le Bellac, La classification, Cours, Faculté des sciences économiques, Université de Rennes 1, France, 2013.