MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université El-hadj Lakhdar –BATNA-
Faculté des Sciences de l'Ingénieur
Département de Génie industriel

MEMOIRE

PRESENTE AU
Laboratoire d'Automatique et Productique
En vue de l'obtention de
MAGISTERE

Spécialité : Génie Industriel
Option : Génie Industriel

Par

MAHADOUI RAFIK

Ingénieur d'état en Informatique
Université de Batna

Thème

DIAGNOSTIC INDUSTRIEL PAR NEURO-FLOU -APPLICATION A UN SYSTEME DE PRODUCTION

Directeur de mémoire : Dr. L.H. Mouss

Soutenu le : 02/03/2008 devant le jury :

N.K Mouss	MC	*Université de Batna*	*Président*
L.H. Mouss	MC	*Université de Batna*	***Rapporteur***
M.D. Mouss	CC	*Université de Batna*	***Co - Rapporteur***
M. Ben Mohamed	Pr	*université Constantine*	*Examinateur*
H. kalla	Dr	*Université de Batna*	*Examinateur*
El.K Merah	*CC*	*Centre Univ. Khenchela*	*Invité*

Année 2007

Sommaire

Liste des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Liste des algorithmes

Liste des tableaux

INTRODUCTION GENERALE

Le rôle premier de la surveillance industrielle est d'augmenter la disponibilité des installations industrielles afin de réduire les coûts directs et indirects de la maintenance des équipements de production. Les coûts directs de cette maintenance sont ceux relatifs aux diverses pièces de rechange, main d'œuvre, etc. Par contre, les coûts indirects sont essentiellement dus au manque à gagner engendré par un arrêt de production. On comprend alors que l'enjeu d'une bonne politique de surveillance est très important pour les entreprises soucieuses d'avoir une meilleure maîtrise des coûts de maintenance.

Le diagnostic est une composante principale du module de supervision. Il consiste à déterminer à chaque instant le mode de fonctionnement dans lequel le système se trouve. Pour cela, beaucoup de travaux ont été dédies à l'élaboration des systèmes d'aide au diagnostic. La majorité de ces travaux se base sur des outils informatiques qui couplent des méthodologies de l'intelligence artificielle comme la logique floue, les systèmes experts , et des outils inspirés des métaphores biologiques et physiologiques comme les réseaux de neurones et les algorithmes génétiques.

Bien que ces métaphores réalisent la tâche de diagnostic avec un certain succès, elles présentent aussi des inconvénients.

Pour pallier ces inconvénients, la tendance actuelle est d'intégrer ces outils dans des architectures hybrides. L'utilisation d'un réseau neuro-flou dans un module de diagnostic offre la possibilité de modéliser des connaissances à priori et des règles linguistique de décision obtenue par les experts du domaine, il profite des capacités et avantages de l'inférence floue modélisée par une architecture parallèle. Cette architecture arrive à pallier la dimension de boîte noire des réseaux neurones (non interprétabilité des neurones) par l'introduction de neurones spécialisés. Ainsi l'ajustement fin des paramètres du système flou se réalise par le biais de l'apprentissage neuronal.

L'objectif de ce travail est le développement d'un module de diagnostic industriel basé sur la reconnaissance de formes statistique neuro-flou qui s'appuie sur une représentation numérique et au même temps symbolique des formes. Le diagnostic de défaillances est essentiellement vu comme un problème de *classification*. Le but principal est de construire un bloc de correspondance tel qu'à partir d'un ensemble d'informations décrivant la situation courante de processus, il est possible d'obtenir les causes probables de situations anormales.

Quand le diagnostic est basé sur des observations multiples, ces observations sont regroupées pour former des classes qui définissent une situation ou un mode de fonctionnement du processus, auxquelles une nouvelle observation sera comparée pour être identifiée. En d'autres termes, le diagnostic a pour mission d'identifier le mode de fonctionnement d'un système à partir d'observations sur celui-ci.

La réalisation pratique de notre travail concerne le domaine de l'industrie du ciment et, plus précisément, le processus de clinkérisation.

Le mémoire est organisé en cinq chapitres qui peuvent être résumés comme suite :

Le premier chapitre est dédié aux différentes approches utilisées pour la conception d'un module de surveillance et de diagnostic. Les méthodologies de surveillance sont généralement divisées en deux groupes : méthodologies de surveillance *avec et sans modèle.* Les premiers se basent su l'existence d'un modèle formel de l'équipement et utilisent généralement les techniques de l'automatique. La deuxième catégorie de méthodologies est plus intéressante dès lors qu'un modèle de l'équipement est inexistant ou difficile à obtenir. Dans ce cas, on utilise les outils de la statistique et de l'Intelligence Artificielle. La fonction surveillance est alors vue comme une application de reconnaissance des formes. Les formes représentent le vecteur d'entrée composé par les différentes données de l'équipement (données mesurables et qualifiables) et les classes correspondent aux différents modes de fonctionnement.

Le deuxième chapitre quand a lui, est consacré à la présentation d'un état de l'art sur l'application des réseaux neurones artificiels dans le domaine d'industrie et plus précisément dans la supervision et diagnostic des systèmes de production.

Les avantages les plus importants que l'on peut attribuer à une application de diagnostic par réseaux de neurones sont : la modélisation et l'estimation de fonctions non linéaire par apprentissage, la fusion de données, la généralisation et la reconstruction des signaux capteurs. Deux architectures neuronals sont généralement utilisées pour les taches de diagnostic : le Perceptron Muti-Couches (PCM) et les Réseaux à base de Fonctions Radiales (RFR).

Le troisième chapitre est réservé a la logique floue fait l'objet, sa définition, ses caractéristiques essentielles de raisonnement approximatif, et le système d'inférence floue. La logique floue permet en effet de faire le lien entre modélisation numérique et modélisation symbolique, ce qui permet des développements industriels spectaculaires à partir d'algorithmes très simples de traduction de connaissances symboliques en entité numérique et inversement. Des applications de la logique floue dans perspective industrielle sont présentées.

Le quatrième chapitre est consacré à une présentation de la contribution des différents modèles de réseaux neuro-flous.

Les systemes neuro-flous nés de l'association des réseaux de neurones avec la logique floue, de manière à tirer profits des avantages de chaque une de ces deux techniques,les principales propriétés des systèmes neuro-flous et leur capacité à traiter dans un même outil des connaissances numérique et symboliques d'un système. Ces derniers permettent donc d'exploiter les capacités d'apprentissage des réseaux de neurones d'un part et les capacités de raisonnement de la logique floue d'autre part. ainsi les architectures les plus connues et les plus utilisés dans l'industrie et dans les domaines de classification sont présentés.

Le dernier chapitre est dédié à la présentation d'une exploitation industrielle innovante sur laquelle a débouché notre étude. En effet, nous proposons une stratégie pour le suivi du comportement d'un processus et la détection des défaillances. Une approche de diagnostic industriel basé sur la reconnaissance de formes statistique neuro-floue qui s'appuie sur une représentation numérique et au même temps symbolique des formes est alors mise au point.

Afin d'implanter cette approche et l'exploiter, pour diagnostiquer un système de production, nous avons développer un *pro- logiciel* informatique que nous avons nommé NEFDIAG (NEuro Fuzzy DIAGnosis).

Le NEFDIAG 1.00 (NEuro Fuzzy DIAGnosis) est un logiciel informatique de simulation interactive réalisé au sein de LAP, écrit sous DEPHI, pour le développement, l'apprentissage et le test d'un système neuro-flou de classification des pannes d'un procédé industriel.

NEFDIAG peut être représenté comme un type spécial de perceptron floue, à trois couches utilisé pour classifier des formes et des défaillances.

Enfin la dernière partie conclue ce mémoire. Nous y dresserons un bilan final de notre travail et nous y donnerons quelques perspectives de recherche prometteuses.

CHAPITRE 1

SURVEILLANCE ET DIAGNOSTIC INDUSTRIEL

Le diagnostic est une composante principale du module de supervision. Il consiste à déterminer à chaque instant le mode de fonctionnement dans lequel le système se trouve. Il s'appuie sur une connaissance à priori des modes de fonctionnement et sur une connaissance instantanée matériellement par une nouvelle observation de l'état du système. Il existe plusieurs approches pour réaliser le diagnostic, le choix d'une approche est lié au mode de représentation de la connaissance.

Dans ce chapitre, nous exposons le rôle crucial de la surveillance dans un système de production, nous situons ensuite le diagnostic des défaillances dans le cadre général de la supervision. Les différentes approches de diagnostic font l'objet de la deuxième partie de ce chapitre. La dernière partie de ce chapitre est consacrée à la présentation des méthodes de diagnostic par reconnaissance de formes.

1.1. Introduction

Les activités industrielles et humaines font presque les grands titres des actualités avec leurs incidents, accidents ou événements catastrophiques. En effet, le zéro défaut ou le risque zéro n'existe pas pour les activités industrielles à cause de l'occurrence de défaillance humaine ou matérielle.

Ce chapitre, vise à rappeler dans un premier temps la terminologie utilisée dans la littérature scientifique et celle que nous avons adoptée dans ce mémoire. Les principales approches de surveillance ont énoncées. Conduisant au principe fondamental sur lequel repose toute procédure de diagnostic n'utilisant pas des modèles, la problématique inhérente à ce type d'approche est alors présentée.

Les méthodologies de surveillance généralement divisées en deux grandes catégories : méthodologies de surveillance avec modèle et sans modèle. Les premières se basent sur l'existence d'un modèle formel de l'équipement et utilisent généralement les techniques de l'automatique. La deuxième catégorie est plus intéressante, dès lors qu'un modèle de l'équipement est inexistant ou difficile à obtenir. Dans ce cas, on utilise les outils de la statistique et de l'Intelligence Artificielle. La fonction surveillance est alors vue comme une application de reconnaissance des formes. Les formes représentent le vecteur d'entrée composé par les différentes données de l'équipement (données mesurables et qualifiables) et les classes représentent les différents modes de fonctionnement.

Nous allons décrire dans ce chapitre les méthodologies de surveillance industrielle et de diagnostic. Plusieurs travaux se sont intéressés à la supervision et la surveillance industrielle [32] [39] [40] [42] [45] [47].

1.2 Les méthodes de surveillance industrielle

Définition 1 :

La surveillance est une tâche continue en temps réel dont le but est de caractériser le mode de fonctionnement du système physique, en enregistrant des informations, en reconnaissant et en indiquant les anomalies de comportement [35].

Definition2 :

La surveillance est un dispositif passif, informationnel qui analyse l'état du système et fournit des indicateurs. La surveillance consiste notamment à détecter et classer les défaillances en observant l'évolution du système puis a le diagnostiquer en localisant les éléments défaillants et en identifiant les causes premières [47].

La surveillance se compose donc de deux fonctions principales qui sont la détection et le diagnostic. Les principales raisons qui conduisent à surveiller un système sont la conduite et la maintenance.

L'architecture d'un système de surveillance est illustrée sur la figure1.1.

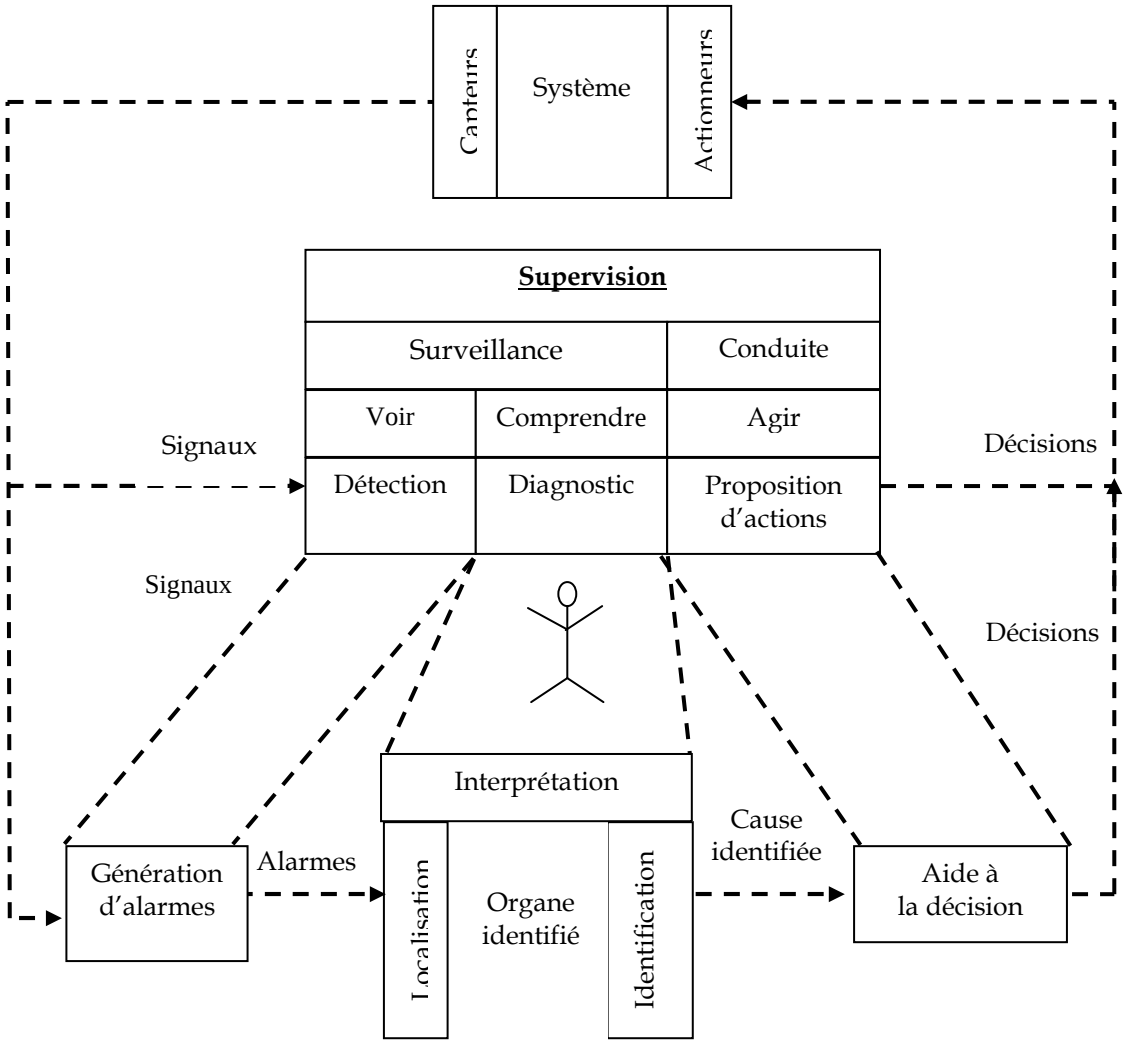


Figure 1.1. *Architecture générale d'un système de Surveillance en ligne* [47]

Aussi la surveillance consiste à :

- Recueillir en permanence tous les signaux en provenance du procédé et de la commande.
- Reconstituer l'état réel du système commandé.
- Faire toutes les inférences nécessaires pour produire les données utilisées pour dresser des historiques de fonctionnement.

Le cas échéant, pour mettre en œuvre un processus de traitement de défaillances.

Les méthodes de surveillance industrielle sont illustrées sur la figure 1.2. L'existence d'un modèle formel ou mathématique de l'équipement détermine la méthode de surveillance utilisée. La surveillance avec modèle se compose essentiellement de deux techniques : méthodes de redondance physique et analytique et méthodes d'estimation paramétrique.

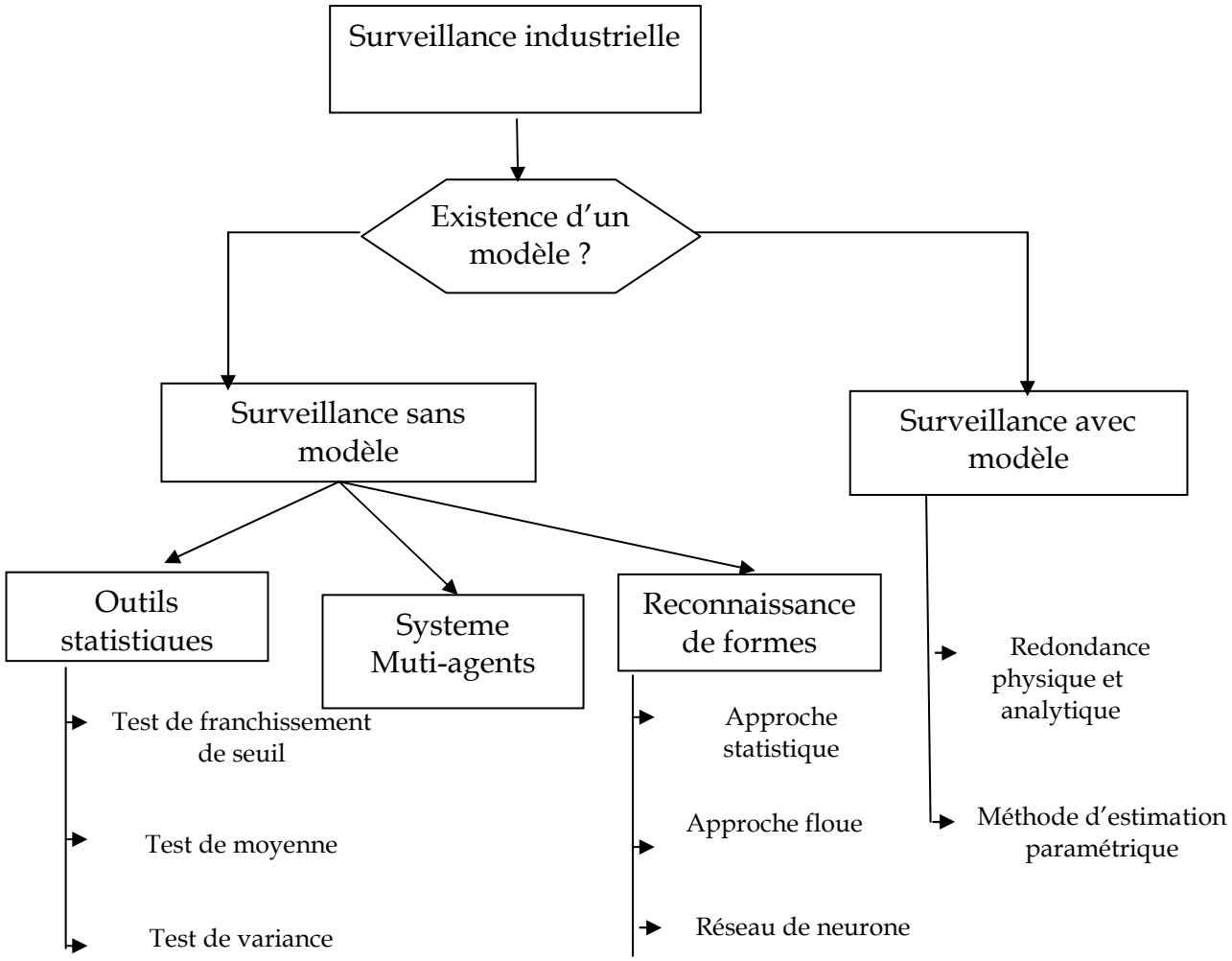


Figure 1.2. *Classification des méthodologies de surveillance industrielle.*

D'un autre côté, les méthodes qui ne se basent pas sur l'existence du modèle se divisent en deux catégories : méthodes utilisant des outils statistiques et celles utilisant la reconnaissance de formes. Les outils statistiques établissent des tests sur les signaux d'acquisition capables d'assurer que la fonction détection de défaillances. Par contre, les techniques de surveillance par reconnaissance de formes sont plus élaborées par rapport aux simples tests statistiques et sont capables de détecter et de diagnostiquer les défaillances [47].

1.2.1 Méthodes de surveillance avec modèles

Les méthodes de surveillance avec modèle ont pour principe de comparer les mesures effectuées sur le système aux informations fournies par le modèle [47]. Tout écart est synonyme de défaillance. Ces méthodes peuvent être séparées en deux techniques : techniques d'estimation paramétrique et techniques de redondance physiques et analytiques. Ces deux techniques sont présentées brièvement.

1.2.1.1 Méthodes d'estimation paramétrique

L'approche d'estimation paramétrique mesure l'influence des défauts sur les paramètres. Le principe consiste à estimer en continu des paramètres du procédé en utilisant les mesures d'entrée/sortie et en l'évaluation de la distance qui les sépare des valeurs de référence de l'état normal du procédé.

Pour détecter l'apparition de défaillances dans le système, il faut effectuer la comparaison entre les paramètres estimés et les paramètres théoriques.

Il existe principalement deux approches de l'estimation paramétrique:

- **l'estimation du maximum de vraisemblance (EMV)**

  Les paramètres sont supposés fixes, il s'agit de trouver les paramètres qui maximisent la probabilité d'observer les échantillons d'apprentissage.

- **l'estimation Bayesienne** (EB) : ou apprentissage Bayesien

  Les paramètres sont considérés comme des variables aléatoires pour lesquelles on suppose avoir des connaissances a priori. Les échantillons d'apprentissage permettent d'augmenter la précision des paramètres.

I.2.1.2. Redondances physiques et analytiques

a. ***Redondances analytiques***

Le principe consiste à établir un modèle du système à surveiller qui comprend un certain nombre de paramètres qui sont supposés connus lors du fonctionnement nominal.

Le but des méthodes de redondance analytique est d'estimer l'état du système afin de le comparer à son état réel (figure 1.3). L'estimation de l'état du système peut être réalisée soit à l'aide de techniques d'estimation d'état, soit par obtention de relations de redondance analytique.

Parmi les différentes méthodes de détection utilisant des modèles mathématiques, nous trouverons principalement l'espace de parité, les Observateurs.

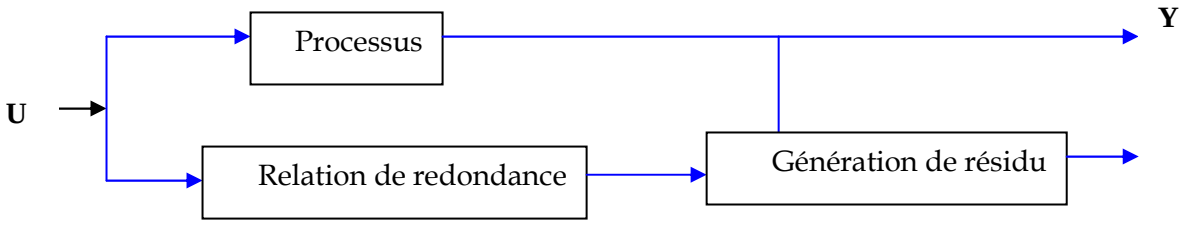


Figure 1.3. *Principe de méthodes redondance analytique.*

b. Redondances physiques

Cette méthode consiste à multiplier physiquement les capteurs critiques d'une installation. Un traitement des signaux issus des éléments redondants effectue des comparaisons et distingue l'élément défectueux en cas d'incohérence. Cette méthode est pénalisante en termes de volume du système et, coût (d'achat et de maintenance) et puissance consommée. Elle est donc essentiellement réservée aux cas où la continuité de

service est obligatoire (exp. l'aérospatiale, le nucléaire). En effet, elle apporte l'avantage, une fois la défaillance détectée et localisée, de pouvoir utiliser la partie de l'équipement encore saine mais elle ne s'applique généralement que sur des capteurs.

1.2.2 Méthodes de surveillance sans modèles

Nombreuses sont les applications industrielles dont le modèle est difficile, voire impossible à obtenir suite à une complexité accrue ou à de nombreuses reconfigurations intervenants durant le processus de production. Pour ce type d'applications industrielles, les seules méthodes de surveillance opérationnelles sont celles sans modèle. Deux techniques existent dans ce cas : surveillance par tests statistiques et surveillance par reconnaissance de formes. Nous nous intéressons dans le cadre de notre projet seulement aux méthodes par reconnaissances de formes, dont trois approches sont utilisées : approche probabiliste, approche floue et approche neuronale.

1.2.2.1 Surveillances avec outils statistiques

Les outils statistiques de détection de défaillances consistent à supposer que les signaux fournis par les capteurs possèdent certaines propriétés statistiques. On effectue alors quelques tests qui permettent de vérifier si ces propriétés sont présentes dans un échantillon des signaux mesurés [47].

a. Test de franchissement de seuils

Son principe consiste à comparer ponctuellement les signaux avec des seuils préétablis, le franchissement de ce seul par un des signaux capteurs génère une alarme. Ce type de méthode est très simple à mettre en œuvre mais ne permet pas d'établir un diagnostic de défaillance, cette méthode aussi et très sensible aux fausse alarmes [27].

b. Test de variance

On peut également calculer la variance d'un signal .Tant que cette variance se situe dans une bande situé autour de sa valeur nominale, l'évolution du système est supposée normale.

c. Test de moyenne

Contrairement à la méthode de « Test de franchissement de seuil », le test de comparaison est effectué sur la moyenne du signal contenu dans une fenêtre d'un ensemble de valeurs que sur une valeur ponctuelle.

1.2.2.2 Surveillance par reconnaissance de formes

L'approche de surveillance par reconnaissance de formes permet d'associer un ensemble de mesures effectuées sur le système à des états de fonctionnement (figure 1.4). Cette fonction permet d'avoir une relation d'un espace caractéristique vers un espace de

décision, de façon à minimiser le risque de mauvaise classification. Trois techniques de reconnaissance de formes sont présentées. La première est une technique classique de discrimination basée sur les outils de probabilité. Elle peut se montrer insuffisante car elle suppose une connaissance a priori de tous les états de fonctionnement et ne prend pas en compte l'évolution du système. Les deux autres techniques de discrimination reposent sur la théorie de l' IA. Elles ont l'avantage de ne pas se baser sur les connaissances a priori des états de fonctionnement mais plutôt sur une phase d'apprentissage. Ces deux techniques sont la reconnaissance de formes par la logique floue et la reconnaissance de formes par réseaux de neurones.

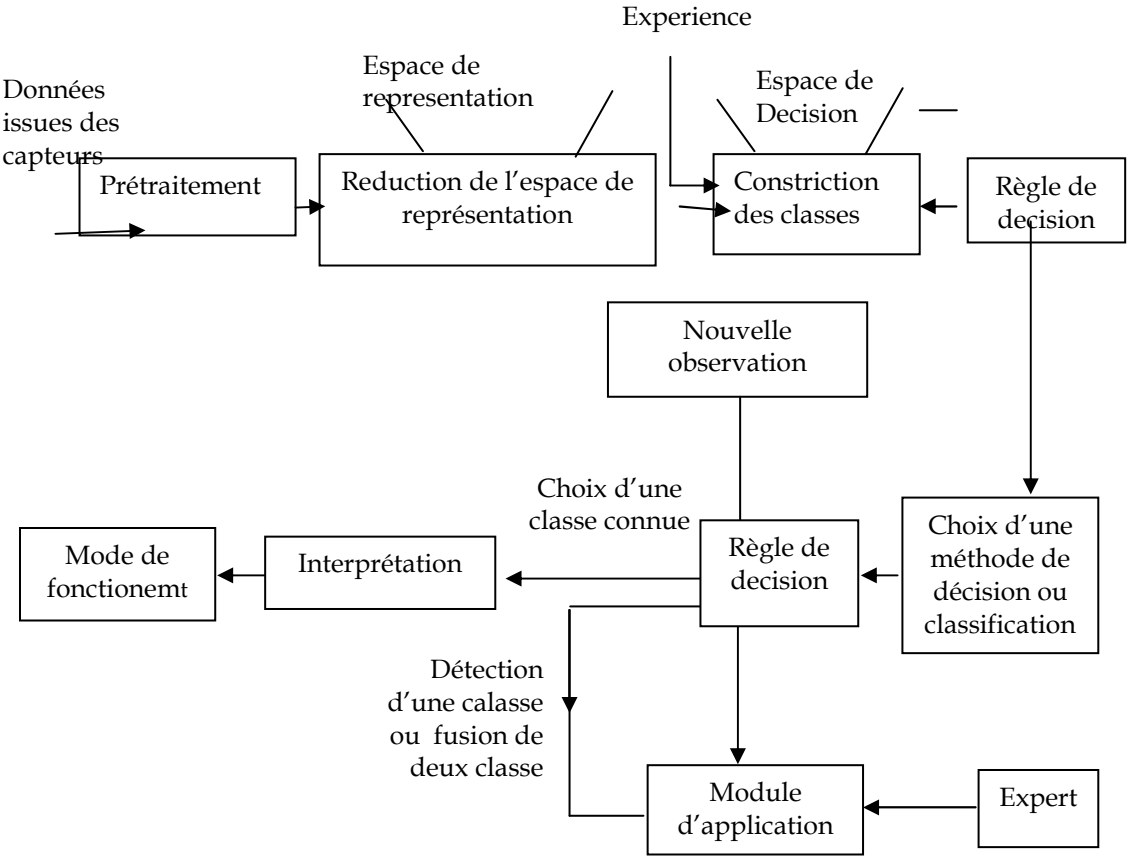


Figure 1.4. S*tructure d'un système de diagnostic par RdF* [39].

a. Reconnaissance de formes par une approche floue

Durant les vingt dernières années, les systèmes d'inférence floue (SIF) sont devenus très populaires. Les applications dans le traitement du signal, la modélisation, la commande, la supervision de procédés et la prise de décision sont en effet autant d'applications qui démontrent la capacité des SIF à traiter des problèmes non linéaires grâce à l'utilisation de connaissances expertes.

En reconnaissance de formes par approche floue, les classes sont représentées par des sous-ensembles flous. Une fonction d'appartenance quantifie le degré d'appartenance $\lambda_i(x)$

de chaque vecteur x à la classe α_i . Généralement, on donne pour chaque vecteur x l'ensemble des degrés d'appartenance à toutes les classes ($\lambda 1$(x), λ_i (x) … λn(x)). La mise en oeuvre d'une méthode de classification floue implique deux étapes : la construction des fonctions d'appartenance et la définition des règles de décision.

Nous présentons la structure générale d'un système de reconnaissance de forme floue (ou système d'inférence floue).

La structure de base d'un SIF est constituée de :

- Un univers de discours qui contient les fonctions d'appartenance des variables d'entrée et de sortie à des classes. Ces fonctions peuvent avoir différentes formes, les plus usuelles étant les formes triangulaires, trapézoïdales, et gaussiennes,
- Une base de connaissance qui regroupe les règles liant les variables d'entrées et de sorties sous la forme « Si…Alors… »,
- Un mécanisme de raisonnement qui base son fonctionnement sur la logique du « *modus ponens* » généralisé.

Définition des fonctions d'appartenance « fuzzification »

Le *fuzzy-k-means* est l'un des premiers algorithmes proposés pour construire automatiquement des fonctions d'appartenance dites aussi partition floue. Cet algorithme non supervisé consiste à minimiser itérativement un critère en fonction d'une matrice de partition floue :

$U=[\lambda_k(x_i)]_{k=1,m\ ;i=1,n}$ *et* $V=(\mu_1 .... \mu_m)$ de la forme :

$$j\ (U,V) = \sum_{i=1}^{n}\sum_{k=1}^{m} \lambda_k(x_i)^m d_k(x_i)^2 \qquad (1.1)$$

avec les conditions suivantes :

$$\lambda_k(x_i) \in [0,1] \quad \forall i,k \qquad (1.2)$$

$$\sum_{k=1}^{m} \lambda_k(x_i) = 1 \qquad \forall i \qquad (1.3)$$

$$0 < \sum_{i=1}^{n} \lambda_i(x_i) < N \quad \forall k \qquad (1.4)$$

Où $d_k(x_i)^2 = \|x_i - \mu_k\|^2$ représente la distance euclidienne entre le vecteur d'entrée x_i et le prototype μ_k (ou noyau) de la classe α_k , ***m*** est un paramètre appelée fuzzifier (m≥1). L'ensemble des vecteurs d'apprentissage est constitué de ***N*** vecteurs $\{x_1, x_2, ..., x_N\}$ susceptibles d'appartenir à M classes $\{\alpha_1, \alpha_2, ..., \alpha_M\}$

La solution qui minimise J_m est donnée par les deux conditions suivantes :

$$\mu_k = \frac{\sum_{i=1}^{n} \lambda_k(x_i)^m x_i}{\sum_{i=1}^{n} \lambda_k(x_i)^m} \qquad \forall k \qquad (1.5)$$

$$\lambda_k(x_i) = \frac{1}{\sum_{j=1}^{m} (d_k(x_i)/d_j(x_i))^{2(m-1)}} \qquad (1.6)$$

Les prototypes ainsi que les fonctions d'appartenance sont calculés d'une manière itérative par l'algorithme 1.1.

- ✓ Initialisation de la matrice de partition floue U_0 , $t = 0$;
- ✓ Faire
  - $t \leftarrow t+1$
  - Calcul de la matrice des prototypes V^t *avec (1.5).*
  - Mise à jour de la matrice de partition floue *U(par 1.6)* .
- ✓ jusqu'à $\left\| u^t - u^{t-1} \right\| \le \varepsilon$

Algorithme 1.1. *Calcul de prototype et fonction d'appartenance.*

Après la convergence de l'algorithme, pour chaque vecteur d'entrée x, on calcule la matrice de partition floue $U=[\lambda_k(x_i)]_{k=1,m}$ avec l'équation *(1.6).*

Décision à partir des degrés d'appartenance (defuzzification)

Apres avoir calculé la matrice de partition floue $U=[\lambda_k(x)]_{(k=1,M)}$ qui représente donc les degrés d'appartenance du vecteur x aux différentes classes, il reste à en déduire le choix d'une action $\gamma(x)$. Le cas le plus simple serait d'avoir chaque action $\gamma_k(x)$ qui représente l'affectation du vecteur x à la classe α_k Dans ce cas, on pourrait appliquer le principe du maximum d'appartenance, qui consiste à choisir la classe ayant le plus haut degré d'appartenance :

$$\gamma(x) = \gamma_k(x) \quad \text{Si} \quad \lambda_k(x) = \max \lambda_j(x) \qquad (1.7)$$

Afin de prendre en compte les notions de rejet en distance et d'ambiguïté présentées Précédemment, il est possible d'utiliser un seuil θ_k , pour chaque classe. Ce seuil est soit défini à priori, soit déterminé à partir de l'ensemble d'apprentissage :

$$\theta_k = \min_{x_i \in \alpha_k} \lambda_k(x_i) \qquad (1.8)$$

Si l'on considère donc { γ_0 γ_d γ_1 γ_m }l'ensemble des actions possibles incluant l'affectation à un rejet d'ambiguïté et l'action de rejet en distance, pour chaque vecteur d'entrée x, on obtient un ensemble des résultats des actions obtenues :

$$\Re(x) = \{k \in \{1,....M \} | \lambda_k(x) > \theta_k \qquad (1.9)$$

Le résultat est exprimé de la manière suivante :

Si	$\Re(x) = \{k\}$	alors	$\gamma(x) = \gamma_k$	
Si	$\Re(x) = \phi$	alors	$\gamma(x) = \gamma_d$	(1.10)
Si	$\mid \Re(x) \mid > 1$	alors	$\gamma(x) = \gamma 0$	

b. Reconnaissance de formes par réseaux de neurones

Les Réseaux de neurones artificiels sont en fait un système informatique constitué d'un nombre de processeurs élémentaires (ou noeuds) interconnectés entre eux qui traite de façon dynamique l'information qui lui arrive à partir des signaux extérieurs. Leur principal avantage par rapport aux autres outils est leur capacité d'apprentissage et de généralisation de leurs connaissances à des entrées inconnues.

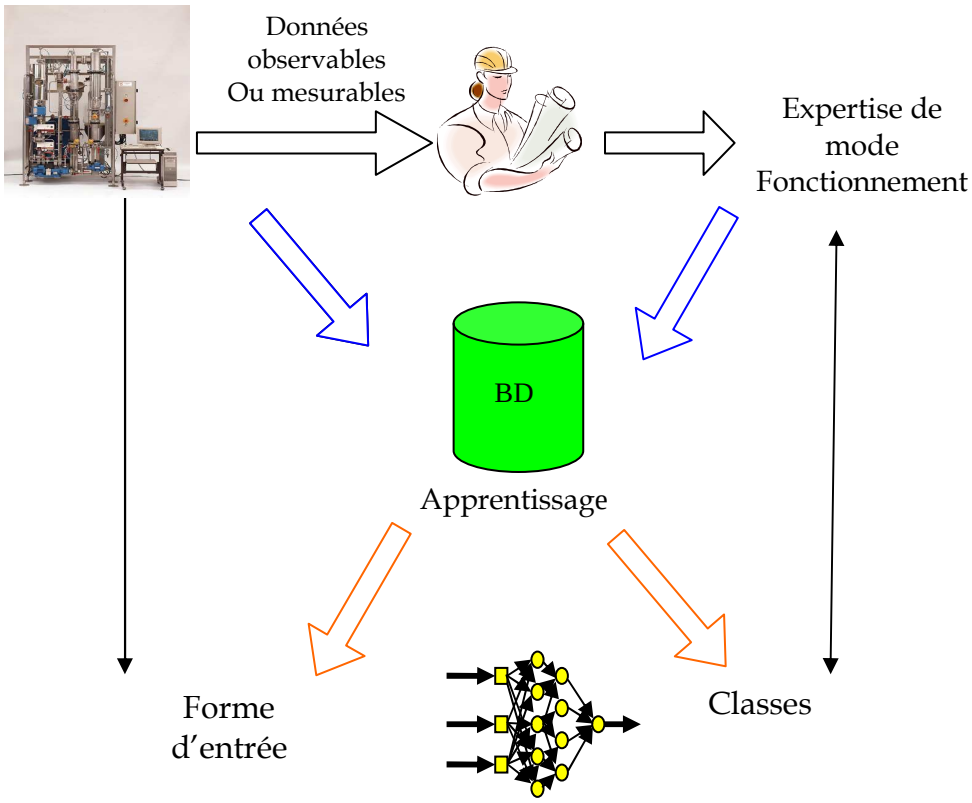


Figure 1.5. *Reconnaissance des formes par RNA*

La Figure 1.5 montre l'architecture générale d'une application de surveillance par reconnaissance de formes avec réseaux de neurones. L'expert humain joue un rôle très important dans ce type d'application. Toute la phase d'apprentissage supervisé du réseau de neurones dépend de son analyse des modes de fonctionnement du système.

Chaque mode est caractérisé par un ensemble de données recueillies sur le système. A chaque mode, on associe une expertise faite par l'expert.

1.2.2.3 Surveillance par système Multi Agent

On parle dans ce cas, de différentes architectures qui distribuent les tâches de la surveillance industrielle par des agents (unité de calcul), et aussi la phase de diagnostic. Ce type d'approche peut être classé dans la surveillance sans modèle.

Un agent est une entité autonome, réelle ou abstraite, capable d'agir sur elle-même et sur son environnement, qui, dans un univers Multi Agent, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, ses connaissances et les interactions avec les autres agents.

Les agents exécutent eux mêmes les tâches du processus de diagnostic comme l'acquisition et la gestion des données, le choix des outils diagnostics pour la détection, la gestion des conflits et la prise de décision, ainsi que l'information et l'aide à l'opérateur. Les «agents » conduisent à des composants logiciels, qui peuvent s'exécuter sur différents environnements. Chaque agent coopère avec ses voisins et a une ou plusieurs responsabilités spécifiques. Il forme une architecture spatialement distribuée avec des éléments modulaires et extensibles qui sont fonctionnellement et sémantiquement bien identifiés.

1.3 Le diagnostic industriel

1.3.1 Organisation générale d'un système de surveillance

La surveillance se compose de deux fonctions principales qui sont la détection et le diagnostic.

Le diagnostic des pannes dans les installations industrielles est le processus qui, à partir des symptômes observés, permet l'identification des causes à l'origine des dysfonctionnements et d'incrémenter le (ou les) composant (s) en panne du système [35].

Le diagnostic d'un système est l'identification du mode de fonctionnement, à chaque instant, par ses manifestations extérieures. Son principe général consiste à confronter les données relevées au cours du fonctionnement réel du système avec la connaissance que l'on a de son fonctionnement normal ou défaillant. Si le mode de fonctionnement identifié est un mode défaillant, le système de diagnostic peut localiser sa cause.

Ce sous système peut se décomposer en trois fonctions [35] :

- la ***localisation*** détermine le sous-ensemble fonctionnel défaillant est progressivement affine cette détermination pour désigner l'organe ou dispositif élémentaire défectueux.

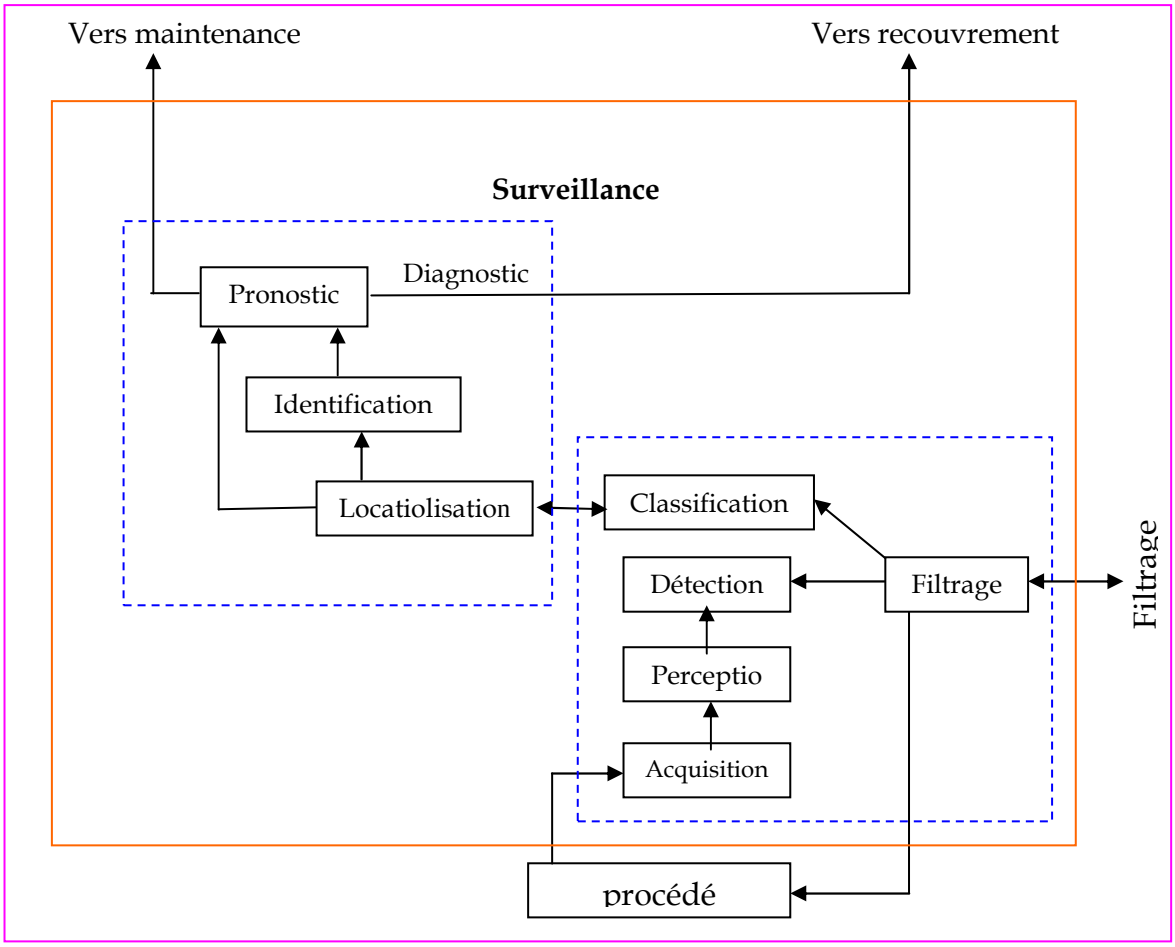


Figure 1.6. *Domaine de la surveillance.*

- L'***identification*** donne les causes qui ont engendré la ou les défaillances constatées.
- Le **pronostic** s'intéresse la propagation des défaillances. Il indique les conséquences inéluctable et mesure celles qui peuvent être anticipés de façon à ne pas solliciter les sous systèmes de la ressource défaillante .le pronostic es une étape sans laquelle la prose décision n'est pas faisable.

Donc, la fonction d'une opération de diagnostic est de déterminer les composants ou organes défaillants d'un système physique. Elle peut intervenir à plusieurs stades :

- Les contrôles "qualité": Il s'agit de tester des produits afin de garantir que leurs caractéristiques sont conformes à des spécifications.
- La supervision : Il s'agit de doter les systèmes physiques d'une intelligence en les équipant de dispositifs étudiant en temps réel leur comportement pour produire automatiquement un diagnostic qui sera fourni et exploité par l'opérateur.
- La maintenance prédictive : Il s'agit de déceler des dérives de comportements d'un système physique avant qu'une fonction ne soit altérée afin de remplacer les organes dégradés avant qu'ils ne tombent en panne.

- L'aide au diagnostic : Il s'agit d'aider un opérateur à remonter aux organes défectueux.

On peut schématiser le processus de diagnostic sur la figure 1.7 :

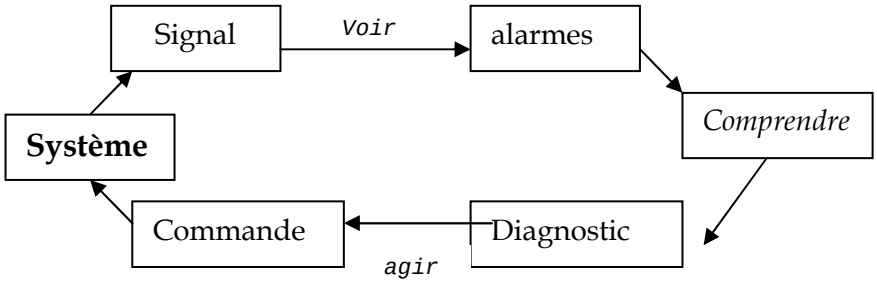


Figure 1.7. *Processus de diagnostic*

1.3.2 Les stratégies du diagnostic industriel

Dans le cas de systèmes physiques complexes, le nombre de tests de détection peut devenir très important. Il devient alors nécessaire de n'effectuer que certains tests dits initiaux qui, si un symptôme apparaît, ils déclencheront, selon les résultats obtenus, d'autres tests. Cette organisation relève de la stratégie de diagnostic. Les tests effectués lors de la phase initiale sont en principe des tests orientés bon fonctionnement (BF) tandis que durant la deuxième phase, des tests orientés bon ou mauvais fonctionnement (MF) peuvent être déclenchés.

Développer une stratégie pour un système de diagnostic consiste principalement à concevoir une politique de déclenchement des tests de détection. La décomposition d'une stratégie de diagnostic en deux étapes est naturelle à l'instar du diagnostic médical. La première étape correspond à l'analyse des symptômes fournis par les tests exclusivement basés sur des connaissances de comportement normal. La deuxième étape se base sur les tests de détection qui supposent des comportements anormaux spécifiques. En effet, il est plus rentable de commencer par déterminer ce qui se comporte normalement avant de chercher des défauts spécifiques, parce que le nombre de défauts envisageable peut-être très grand (MF) alors que le nombre de composants est finis (BF). Le MF n'apporte que peu d'informations car on ne peut que rejeter certaines hypothèses de défaut de manière sûr : impossible de prouver avec le MF que l'on est en présence d'un défaut spécifique.

1.3.2.1 Stratégies orientées bon fonctionnement

Durant la procédure de diagnostic orientée bon fonctionnement (comportement normal). Deux étapes sont introduits La première étape est basée sur des tests permanents qui représentent les tests de détection de niveau « 0 » et la deuxième étape est composée de tests de niveau « 1 » déclenchés par des inconsistances détectées par les tests permanents. Des niveaux plus hauts peuvent exister.

Le nombre possible de tests de détection orientés bon fonctionnement peut être très grand. Il est intéressant de ne déclencher initialement que certains tests de détection. C'est là où la stratégie de diagnostic doit jouer son rôle.

La stratégie de diagnostic doit être définie séparément du raisonnement diagnostic parce que différentes façons de raisonner peuvent être implémentées par le même système de diagnostic. Par conséquent, les outils utilisés, pour la conception d'une stratégie de diagnostic, doivent être indépendants de l'analyse des symptômes.

1.3.2.2 Stratégie orientée mauvais fonctionnement

Dans un système de diagnostic, les tests de détection qui intègrent dans leur construction des modèles de mauvais fonctionnement sont généralement disponibles. Tous ces tests devraient être déclenchés car quelques-uns d'entre eux peuvent être incompatibles avec les résultats des tests orientés bon fonctionnement. Pour répondre à la problématique, il est important de définir des règles de déclenchement des tests.
Mais en générale un test orienté mauvais fonctionnement est compatible avec un test orienté bon fonctionnement. Et Tout test compatible avec au moins un test faux orienté bon fonctionnement devrait être déclenché parce qu'il peut apporter plus d'information sur l'état réel du système [35].

1.4 Conclusion

L'objectif de ce chapitre a été de donner un aperçu des techniques généralement utilisées pour résoudre des problématiques de surveillance. Ainsi nous avons présenté le classement des techniques de surveillance qui dépend de l'existence ou non d'un modèle formel de l'équipement à surveiller et nous avons concentré notre étude sur les méthodes qui ne se basent pas sur un modèle de l'équipement .Les principales approches de détection sont alors présentées.

La surveillance d'un équipement industriel se fait au travers de deux fonctions : la détection et le diagnostic des défaillances. La surveillance à base de modèles est souvent opérée hors ligne, empêchant aussi des traitements temps réel. En revanche, les techniques de l'intelligence artificielles offre des outils totalement découplés de la structure des systèmes, permettant un suivi on temps réel de l'évolution de celui-ci.

Les systèmes de diagnostic par outils de l'Intelligence Artificielle peuvent donc représenter d'excellents systèmes d'aide à la décision pour l'expert humain. L'une des méthodes de l'intelligence artificielle la plus utilisée est celle des réseaux de neurones artificiels. Leur principe de fonctionnement est inspiré du cerveau humain.

En ce sens, le chapitre suivant sera consacré à la présentation des notions de base des réseaux de neurones, et de leur application en diagnostic.

CHAPITRE 2

LES RESEAUX DE NEURONES APPLIQUES AU DIAGNOSTIC DES SYSTEMES DE PRODUCTION

Les réseaux de neurones artificiels (RNA) sont des systèmes parallèles et distribués de traitement de l'information inspirés par le fonctionnement du cerveau humain. Le présent chapitre décrit le principe de fonctionnement des RNA, leurs différentes architectures, les familles d'algorithme d'apprentissage disponibles ainsi qu'un résumé des principaux avantages des réseaux de neurones en tant que système de traitement de l'information. La deuxième partie introduit les principaux domaines d'applications et secteurs d'activités de RNA. Dans la troisième partie du chapitre, nous verrons comment les RNA sont appliqués en surveillance et en diagnostic ?

2.1Introduction

Les RNA constituent une nouvelle approche de traitement de l'information. Ils offrent des solutions compactes et rapides pour une large gamme de problèmes, en particulier ceux ayant des contraintes en temps réel tel le cas de la plupart des applications spatiales actuelles. Ceci est davantage vrai avec l'utilisation des émulations et des implantations matérielles. Ils peuvent fournir une solution intéressante pour des problématiques de surveillance d'équipements industriels.

Parmi les propriétés importantes des réseaux de neurones, on peut citer leur tolérance aux fautes qui mesure leur aptitude à exécuter la tâche qui leur est demandée en présence d'informations erronées et de maintenir leur capacité de calcul même si une partie du réseau est endommagée.

La première partie est consacrée à la présentation des RNA et leurs propriétés les plus importantes. La deuxième partie est réservée aux architectures les plus utilisées en surveillance et par voie de conséquence en diagnostic et particulièrement le Perceptron Multi-Couches et Réseaux de Fonctions Radiales. Dans la troisième partie, nous verrons comment les RNA sont appliqués en diagnostic. Une application est alors présentée : la reconnaissance de formes.

2.2 Les concepts de bases des réseaux de neurones

Les premiers travaux sur les RNA ont débutés au début des années quarantaine et on été menés par J-Mc-Culloche et W-Pits. Ils décrivent les propriétés des systèmes neuronaux à partir de neurones idéalisés. Ce sont des neurones logiques (0 ou 1). Dix ans plus tard, on a constitué le premier modèle réel d'un réseau de neurone.

2.2.1 Le model mathématique

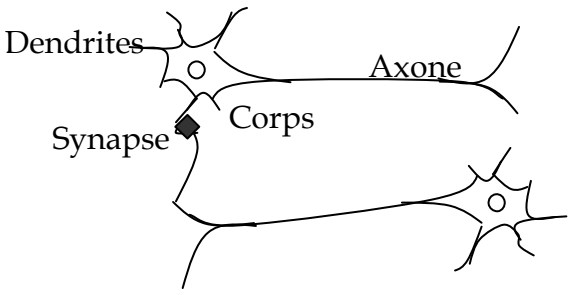


Figure 2.1.a *un neurone biologique*

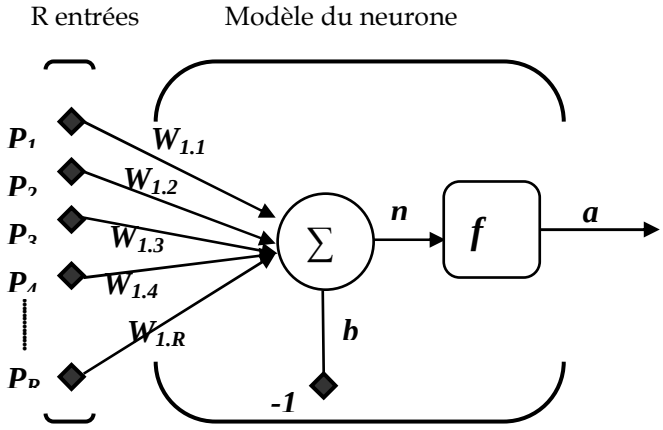


Figure 2.1.b *Modèle d'un neurone artificiel.*

Le modèle mathématique d'un neurone artificiel est illustré à la figure 2.1.b Un neurone est essentiellement constitué d'un intégrateur qui effectue la somme pondérée de ses entrées. Le

résultat ***n*** de cette somme est ensuite transformé par une fonction de transfert ***f*** qui produit la sortie à du neurone. Les *R* entrées du neurone correspondent au vecteur
$P = [P_1\ P_2\ \cdots\ P_R]$, alors que $w = [w_{1,1}\ w_{1,2}\ \cdots\ w_{1,R}]$ représente le vecteur des poids du neurone. La sortie ***n*** de l'intégrateur est donnée par l'équation suivante :

$$n = \sum_{j=1}^{R} w_{1,j} p_j - b \qquad (2.1)$$

Cette sortie correspond à une somme pondérée des poids et des entrées moins ce qu'on nomme le biais ***b*** du neurone. Le résultat n de la somme pondérée s'appelle le niveau d'activation du neurone. Le biais ***b*** s'appelle aussi le seuil d'activation du neurone. Lorsque le niveau d'activation atteint ou dépasse le seuil ***b***, alors l'argument de ***f*** devient positif (ou nul). Sinon, il est négatif.
On peut faire un parallèle entre ce modèle mathématique et certaines informations que l'on connait à propos du neurone biologique. Ce dernier possède trois principales composantes: les dendrites, le corps cellulaire et l'axone (figure 2.1.a). Les dendrites forment un maillage de récepteurs nerveux qui permettent d'acheminer vers le corps du neurone des signaux électriques en provenance d'autres neurones. Celui-ci agit comme une espèce d'intégrateur en accumulant des charges électriques. Lorsque le neurone devient suffisamment excité (lorsque la charge accumulée dépasse un certain seuil), par un processus électrochimique, il engendre un potentiel électrique qui se propage à travers son axone pour éventuellement venir exciter d'autres neurones. Le point de contact entre l'axone d'un neurone et la dendrite d'un autre neurone s'appelle le synapse. Il semble que c'est l'arrangement spatial des neurones et de leur axone, ainsi que la qualité des connexions synaptiques individuelles qui détermine la fonction précise d'un réseau de neurones biologique. C'est en se basant sur ces connaissances que le modèle mathématique décrit ci-dessus a été défini.

Un poids d'un neurone artificiel représente donc l'efficacité d'une connexion synaptique. Un poids négatif vient inhiber une entrée, alors qu'un poids positif vient l'accentuer. Il importe de retenir que ceci est une grossière approximation d'une véritable synapse qui résulte en fait d'un processus chimique très complexe et dépendant de nombreux facteurs extérieurs encore mal connus.
Il faut bien comprendre que notre neurone artificiel est un modèle pragmatique qui, comme nous le verrons plus loin, nous permettra d'accomplir des tâches intéressantes. La vraisemblance biologique de ce modèle ne nous importe peu. Ce qui compte est le résultat que ce modèle nous permettrons d'atteindre [60].

2.2.2 Réseau de neurone artificiel

Définition : Kohonen propose la définition suivante: "Les RNA Sont des réseaux massivement connectés en parallèle d'éléments simples (habituellement Adaptatifs) et leur organisation hiérarchique. Ils sont sensés interagir avec les objets du monde Réel de la même manière que les systèmes nerveux biologiques."

D'après cette définition on peut dire qu'un RNA réalise une ou plusieurs fonctions algébriques de ses entrées. Par la composition des fonctions réalisées par chacune des neurones on peut modéliser un RNA à l'aide d'un graphe orienté par l'interconnexion d'éléments simples (les neurones) et l'échange d'information via les connexions. Tel que le calcul sera fait d'une manière *distribué*, parallèle et *coopératif.*
La figure 2.2 illustre un RNA avec deux entrées est une sortie.

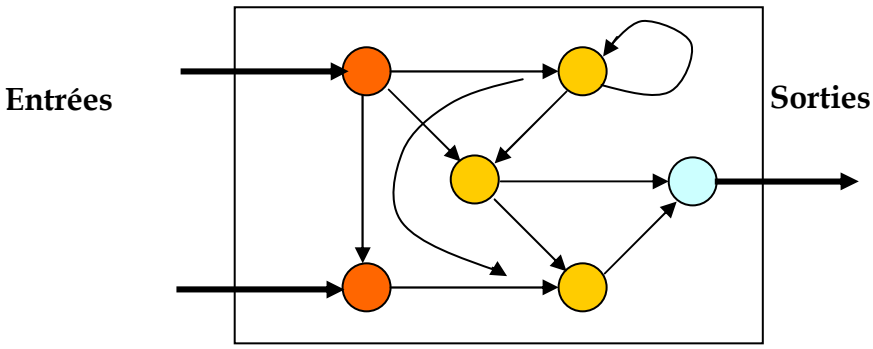


Figure 2.2. *Exemple d'un réseau de neurones artificiel*

Les caractéristiques essentielles d'un réseau de neurones sont : s*on architecture* (topologie) : type d'interconnexion, choix de fonction de transfert et *son mode d'apprentissage*, c'est à dire comment estimer ou apprendre les poids et surtout l'outil de représentation de connaissances. C'est une représentation distribuée, ou chaque neurone participe, ce qui nous amène à observer que les connexions entre les neurones qui composent le réseau décrivent la « topologie » du modèle. Le plus souvent, cette topologie fait apparaître une certaine régularité de l'arrangement des neurones, cependant celui-ci peut être quelconque.

2.3 Les architectures neuronales

2.3.1 Les réseaux de neurones non bouclés « feed forward »

Un réseau de neurone non bouclé est présenté par un ensemble de neurones connectés entre eux telle que l'information circulant des entrées vers les sorties sans retour en arrière.
Le calcul de Y *(sortie)* se fait en propageant les calculs de la gauche vers la droite, avec éventuellement des connexions directes linaires : $y = a*x+fw(x)$.

Ce type de réseau comprend deux groupes d'architectures: les réseaux *Mono-Couches* et les réseaux *Multi-Couches*. La figure 2.3 illustre un RNA Multi-Couche non bouclé.

Ils différent par l'existence ou non de neurones intermédiaires appelées neurones cachés entre les unités d'entrées et les unités de sorties appelées noeuds sources ou nœuds d'entrée et nœuds de sortie respectivement.

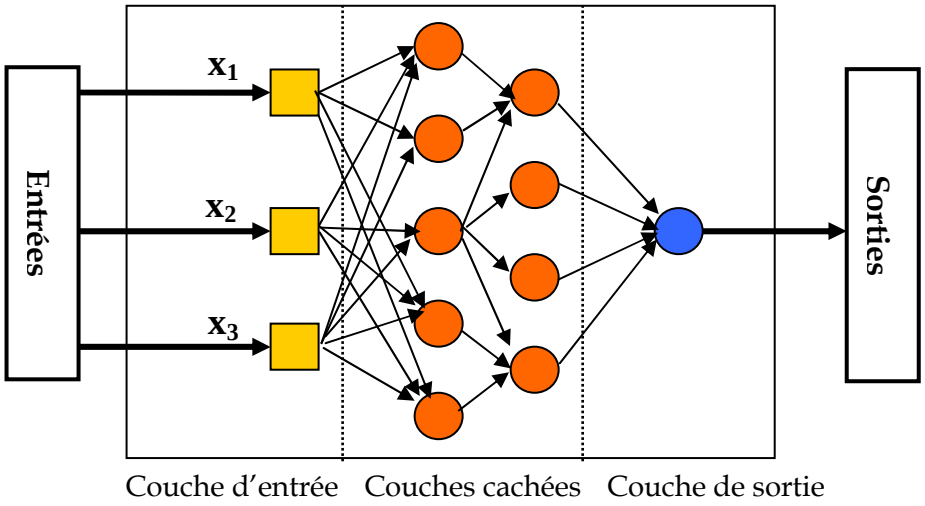


Figure 2.3. *Réseau de neurone non bouclé.*

2.3.1.1 Réseaux non bouclés Mono-Couche

Ce type de réseau possède une couche d'entrée recevant les stimuli à traiter par l'intermédiaire des noeuds sources. Cette couche se projette en une couche de sortie composée de neurones (noeuds de calcul) transmettant les résultats du traitement au milieu extérieur.

La figure 2.4 présente un réseau proactif Mono-Couche à 4 nœuds d'entrée et 3 noeuds de sortie. La désignation Mono-Couche est attribuée à la couche de sortie (noeuds de calcul).

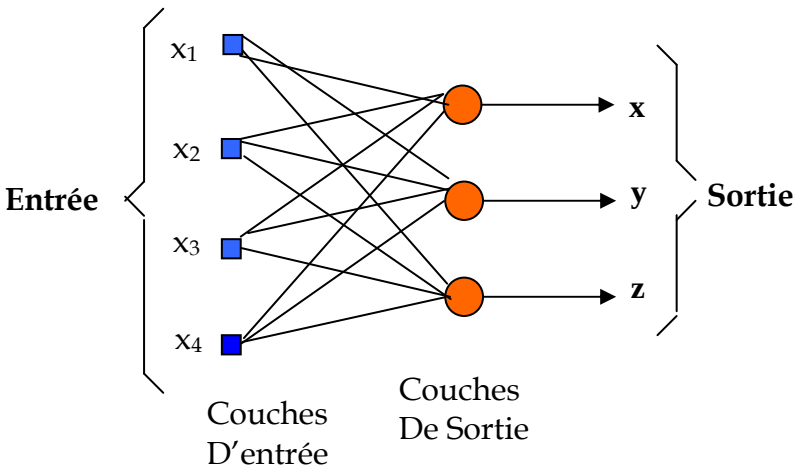


Figure 2.4. *Réseau non bouclé Mono-Couche*

2.3.1.2 Les réseaux non bouclés Multi-Couches

Ce type de réseaux proactifs se caractérise par la présence d'une ou de plusieurs couches cachées, dont les noeuds de calcul correspondants s'appellent neurones cachés ou unités cachées. Les couches cachées s'interposent entre l'entrée du réseau et sa sortie. Leur

rôle est d'effectuer un prétraitement des signaux d'entrée, reçus par la couche d'entrée en provenance du milieu extérieur, et de transmettre les résultats correspondants à la couche de sortie ou sera déterminée la réponse finale du réseau avant qu'elle soit transmise au milieu extérieur.

Ce rôle de prétraitement fait que, en ajoutant une ou plusieurs couches cachées, le réseau est capable d'extraire plus de propriétés statistiques que celles extraites d'un réseau similaire ayant moins de couches cachées. Ceci est utile pour réaliser des fonctions plus complexes que de simples séparations linéaires.

Dans ce type de réseaux, les entrées des neurones d'une couche particulière proviennent uniquement des sorties de la couche adjacente précédente. Les réseaux les plus fréquemment utilisés de cette catégorie sont les perceptrons Multi-Couches (Multilayered Perceptrons MLP)

La figure 2.5 illustre un réseau à une seule couche cachée comportant 7 unités d'entrée, 4 unités cachées et 3 unités de sortie (réseau 7-4-3).

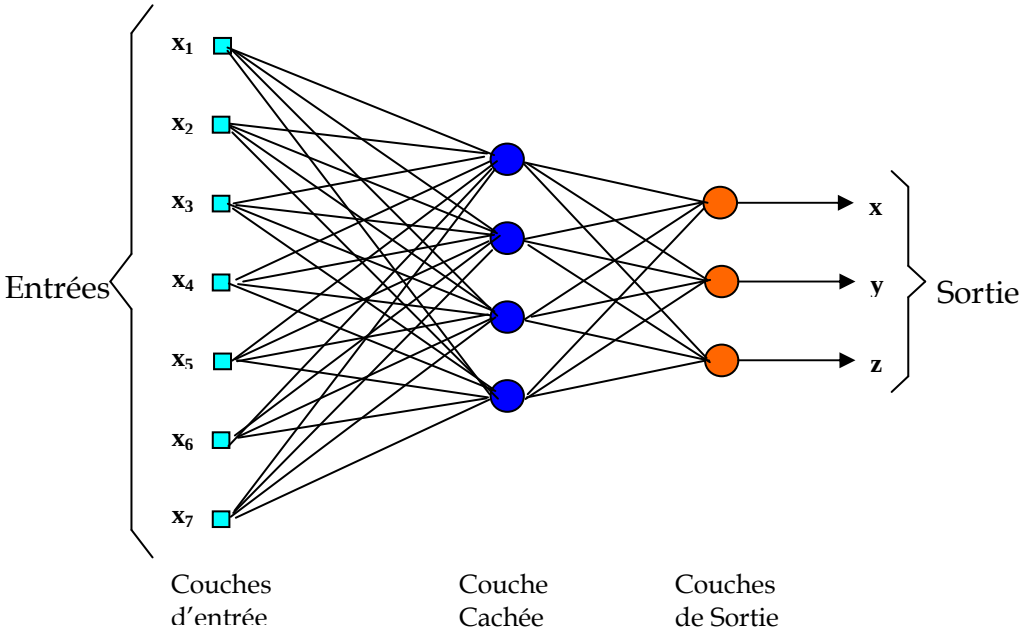


Figure 2.5. *Réseau non bouclé complètement connecté avec un seul couche cachée.*

Ce réseau est dit complètement connecté dans le sens que chaque noeud d'une couche est connecté à tous les noeuds de la couche adjacente suivante. Si éventuellement, des connexions manquaient entre des neurones de deux couches voisines, le réseau serait dit partiellement connecté.

2.3.2 Les réseaux de neurones bouclés (récurrents)

Un réseau de neurone bouclé à temps discret réalise une ou plusieurs équations aux différences non linaires, par composition des fonctions réalisées par chacun des neurones et des retards associés a chacune des connexions [27].

Ces réseaux caractérisent par la présence d'au moins une *boucle* de rétroaction au niveau des neurones ou entre les couches, et la prise en compte de l'aspect temporel du phénomène (figure 2.6). Mais ce sont des modèles plus durs à mètre en œuvre.

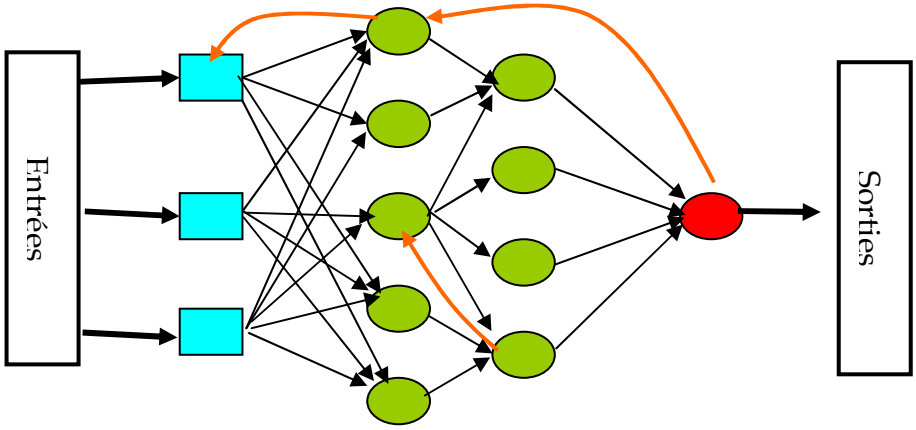


Figure 2.6. *Réseau de neurones bouclé.*

2.4 Les types d'apprentissage des réseaux de neurones

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux Neuronaux. Elle ne concerne cependant pas tous les modèles, mais les plus utilisés. L'apprentissage est une phase du développement du réseau de neurones durant laquelle on calcule les poids des neurones de telle manière que les sorties du réseau soient aussi proche que possible des sorties désirées [52].

L'apprentissage RNA est une phase qui permet de déterminer ou de modifier les paramètres du réseau, afin d'adopter un comportement désiré.

Les procédures d'apprentissage peuvent se subdiviser, en trois grandes catégories : apprentissage supervisé, non supervisé et semi supervisé (renforcement).

Cette distinction repose sur la forme des exemples d'apprentissage. Dans le cas de l'apprentissage supervisé, les exemples sont des couples (entrées, sorties associées) alors que l'on ne dispose que de valeurs (entrées) pour l'apprentissage supervisé.

L'exemple de la figure 2.7 est une partie de l'application de reconnaissance du caractère manuscrit. Il nous donne une idée sur la forme générale d'un apprentissage de réseau de neurones.

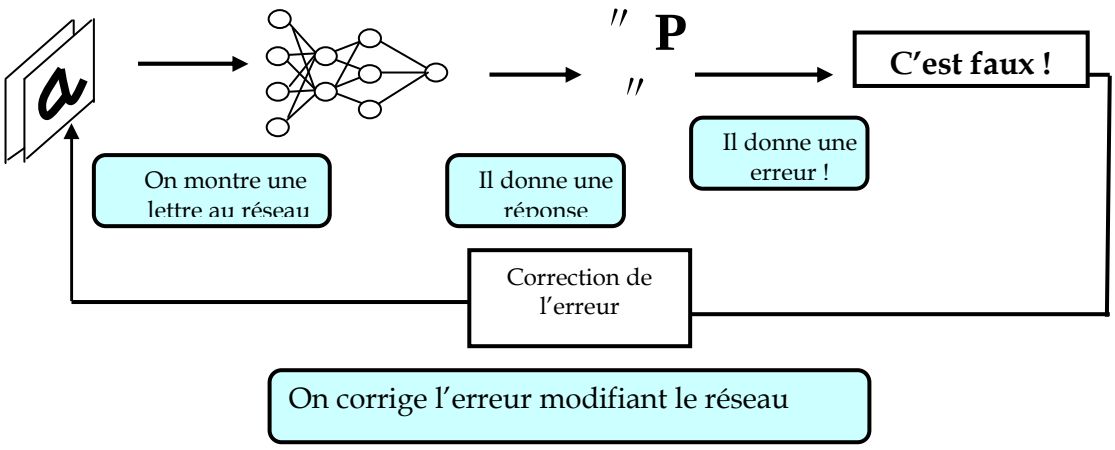


Figure 2.7. *Exemple d'apprentissage*

2.4.1 Apprentissage non supervisé

Dans l'apprentissage non supervisé, les données ne contiennent pas d'informations sur une sortie désirée, il n'y a pas de superviseur « ou expert humain ». Il s'agit de déterminer les paramètres du réseau de neurones suivant une critère à définir.

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. La figure 2.8.a illustre, un exemple d'apprentissage non supervisé.

Les RNAs qui utilisent ce type d'apprentissage sont appelés « auto organisatrice » où ce type d'apprentissage possède souvent une moindre complexité dans le calcul par rapport à l'apprentissage supervisé.

En résumé on peut dire :

- On fournit seulement des exemples X à l'algorithme.
- Il doit trouver W « les poids »tel que les X soient correctement groupés selon $F_{W.}$ (avec une bonne généralisation).

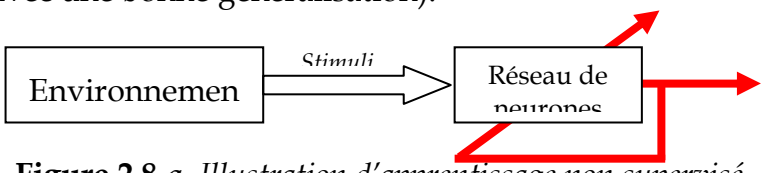


Figure 2.8.*a.* *Illustration d'apprentissage non supervisé*

La loi de Hebb , un exemple d'apprentissage non supervisé

La loi de Hebb (1949) s'applique aux connexions entre neurones, elle s'exprime de la façon suivante :

"*Si deux cellules sont activées en même temps alors la force de la connexion augmente*".

La modification de poids dépend de la co-activation des neurones présynoptique et post synaptique, ainsi que le montre la table 2.1 x*i* et *xj* sont respectivement les valeurs d'activation des neurones i et j, ∂w_{ij} (dérivée partielle du poids) correspond à la modification de poids réalisée.

xi	**xj**	∂w_{ij}
0 « *inactif* »	**0**« *inactif* »	**0** « *pas de modification* »
0 « *inactif* »	**1**« *actif* »	**0** « *pas de modification* »
1 « *actif* »	**0**« *inactif* »	**0** « *pas de modification* »
1 « *actif* »	**1**« *actif* »	**1** « *modification* »

Tab. 2.1. *Simple exemple de la loi de Hebb*

La loi de Hebb peut être modélisée par les équations w (t+1) (nouveau poids) et, wij(t) (ancien poids) :

$$W_{ij}(t+1) = W_{ij}(t) + \partial W_{ij}(t) \qquad (2.2)$$

$\partial W_{ij}(t) = x_i \,.\, x_j$ (la co-activité est modélisée comme le produit des deux valeurs d'activation) L'algorithme d'apprentissage modifie de façon itérative (petit à petit) les poids pour adapter la réponse obtenue à la réponse désirée. Il s'agit en fait de modifier les poids lorsqu'il y a erreur seulement.

Algorithme : *Loi de Hebb*

1. Initialisation des poids et du seuil *S* à des valeurs (petites) choisies au hasard.

2. Présentation d'une entrée $E_l = (e_1, \ldots e_n)$ de la base d'apprentissage.

3. Calcul de la sortie obtenue *x* pour cette entrée: **a = Σ(*wi* . *ei*) - S** (la valeur de seuil est introduite ici dans le calcul de la somme pondérée)

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

4. Si la sortie *x* est différente de la sortie désirée *dl* pour cet exemple d'entrée alors Modification des poids (μ est une constante positive, qui spécifie le pas de modification des poids) : *wij(t+1) = wij(t) + μ.(x i . xj)*

5 Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (i.e. Modification des poids), **retour à l'étape 2**.

Algorithme 2.1. *Apprentissage non supervisé « loi de Hebb »*

2.4.2 Apprentissage supervisé

Dans l'apprentissage supervisé, un superviseur fournit une valeur ou un vecteur de sortie (appelé cible ou sortie désirée) que le réseau de neurones doit associer au vecteur d'entrée, donc l'apprentissage supervisé implique l'existence d'un « professeur » qui a pour rôle d'évaluer le succès (ou l'échec) du réseau quand il lui est présenté un stimule connu.

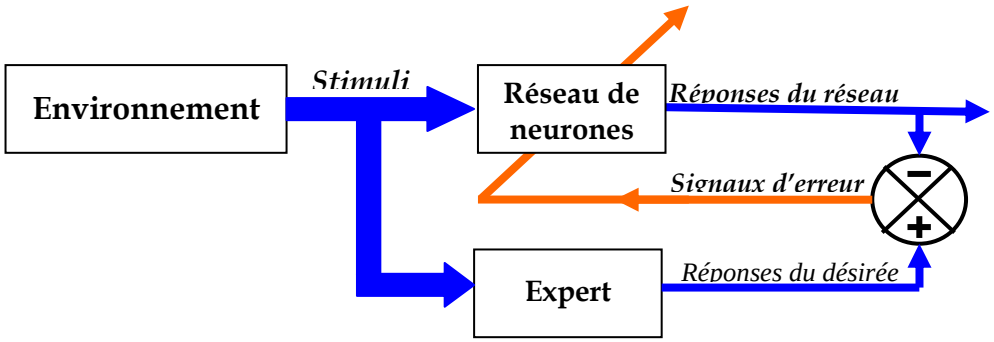


Figure 2.8.b. *Illustration d'apprentissage supervisé*

La règle de Widrow-Hoff « *règle de delta* » est une méthode de minimisation de l'erreur entre la sortie réelle et la sortie désirée. Cette règle est généralement utilisée dans le cas d'apprentissage supervisé. Donc les fonctions d'activations sont des neurones linéaires. Elle converge vers la solution des moindres carrés qui minimise la fonction d'erreur E. Son but est de faire évoluer le réseau vers le minimum de sa fonction d'erreur (erreur commise sur l'ensemble des exemples). Elle est utilisée dans le modèle de l'adaline (ADAptive LINear Element). L'apprentissage est réalisé par itération (les poids sont modifiés après chaque exemple présenté), et on obtient le poids à l'instant t+1 par la formule :

$$W(t+1) = W(t) + n.(T - O).E$$

avec W est le poids, T la sortie théorique et O la sortie réelle. E l'entrée et n un coefficient d'apprentissage (entre 0 et 1) que l'on peut diminuer au cours de l'apprentissage. C'est en fait un cas particulier de l'algorithme de rétro propagation du gradient.

2.5 Les réseaux de neurones les plus utilises

Aujourd'hui, le nombre de types de réseaux neuronaux possibles est assez élevé (figure 2.9). Nous présentons le tableau récapitulatif (tab.2.1) de ce qui se fait, sachant que

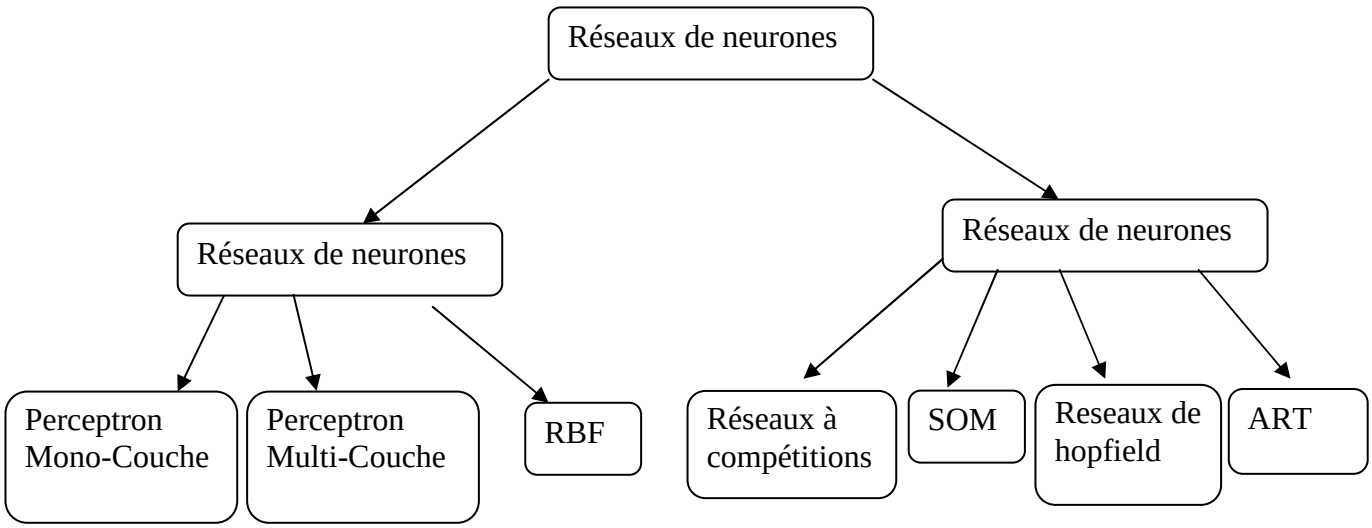


Figure 2.9. *Quelques RNAs usuels*

2.5.1 Perceptron simple

Le perceptron est le premier modèle de réseau de neurones inventé en 1957 par Frank Rosenblatt. Le but du perceptron est d'associer des formes en entrée à des réponses. Le perceptron se compose de deux couches : la rétine et la couche de sortie qui donne la réponse correspondant à la stimulation présente en entrée. Les cellules de la première couche répondent en oui/non. La réponse « oui » correspond à une valeur « 1 » et la réponse « non » correspond à une valeur « 0 » à la sortie du neurone. Les cellules d'entrée sont reliées aux cellules de sortie grâce à des synapses d'intensité variable. L'apprentissage du perceptron s'effectue en modifiant l'intensité de ces synapses. Les cellules de sortie évaluent l'intensité de la stimulation en provenance des cellules de la rétine en effectuant la somme des intensités des cellules actives.

Le perceptron doit trouver l'ensemble des valeurs à donner aux synapses pour que les configurations d'entrée se traduisent par des réponses voulues. Pour cela, on utilise la règle d'apprentissage de Windrow-Hoff.

Pour apprendre, le perceptron (figure 2.10) doit savoir qu'il a commis une erreur, et doit connaître la réponse qu'il aurait dû donner. De ce fait, on parle d'apprentissage supervisé.

La règle d'apprentissage est locale dans ce sens que chaque cellule de sortie apprend sans avoir besoin de connaître la réponse des autres cellules. La cellule ne modifie l'intensité de ses synapses (apprend) que lorsqu'elle se trompe.

Minsky a montre qu'une forme toute simple (le **xor**) ne peut être apprise par un neurone de type perceptron. Un neurone ne peut séparer que deux régions séparables par un hyper plan. Avec plusieurs neurones, ça va déjà mieux mais il est vite clair qu'une seule couche de perceptron ne peut pas apprendre des figures complexes.

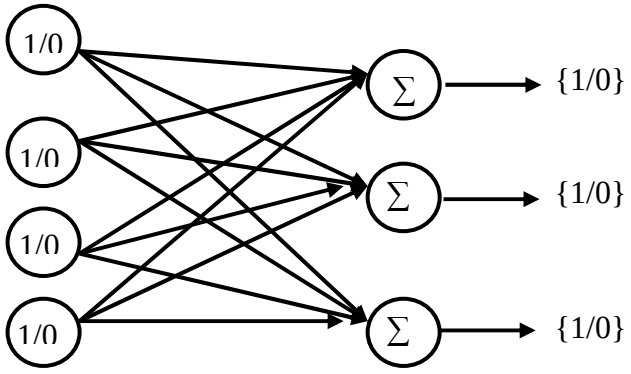


Figure 2.10. *Schéma général de perceptron simple.*

2.5.2 Perceprton Multi Couches (PMC)

Le perceptron Multi-Couche est un réseau orienté de neurones artificiels organisé en couches et où l'information voyage dans un seul sens, de la couche d'entrée vers la couche de sortie.

La figure 2.11 donne l'exemple d'un réseau contenant une couche d'entrée, deux couches cachées et une couche de sortie. La couche d'entrée représente toujours une couche virtuelle associée aux entrées du système. Elle ne contient aucun neurone. Les couches suivantes sont des couches de neurones. Dans l'exemple illustré, il y a 3 entrées, 4 neurones sur la première couche cachée, trois neurones sur les deuxièmes et quatre neurones sur la couche de sortie. Les sorties des neurones de la dernière couche correspondent toujours aux sorties du système.

Dans le cas général, un perceptron Multi-Couche peut posséder un nombre de couches quelconque et un nombre de neurones (ou d'entrées) par couche également quelconque.

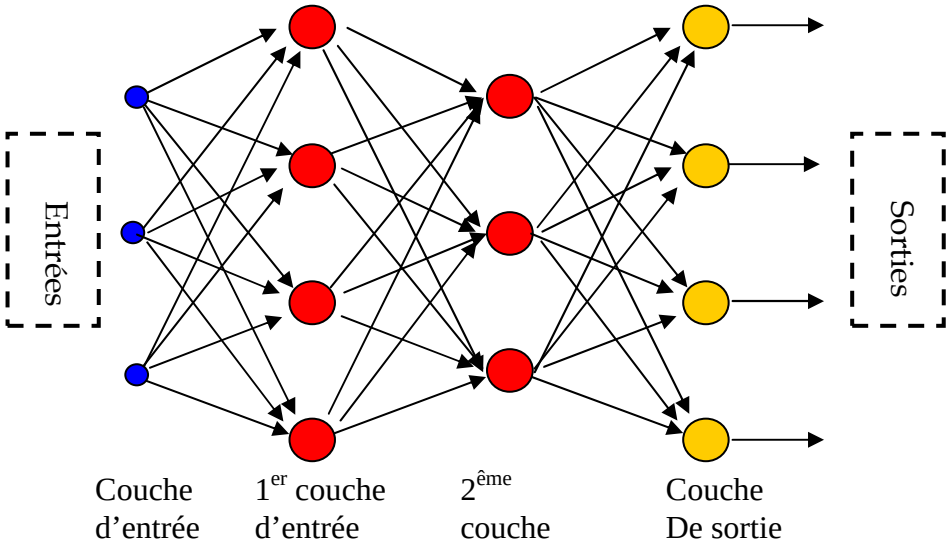


Figure 2.11. *Exemple de réseau de type perceptron Multi-Couche.*

La création d'un perceptron Multi-Couche pour résoudre un problème donné passe donc par l'inférence de la meilleure application possible telle que définie par un ensemble de données d'apprentissage constituées de paires de vecteurs d'entrées et de sorties désirées. Cette inférence peut se faire, entre autre, par l'algorithme dit de rétro propagation.

2.5.1.2 la rétro propagation

Un des inconvénients du Perceptron est qu'il minimise une erreur en tout ou rien à cause de sa fonction d'activation. Il ne prend donc pas en compte la notion de distance. De ce fait, il est très peu robuste. La règle d'apprentissage de Widrow-Hoff (*règle de Delta)* ne

travaille plus en tout ou rien mais minimise une fonction d'erreur quadratique, donc plus robuste. Malheureusement, cette règle ne peut s'appliquer que sur des réseaux à une seule couche de poids adaptatifs. C'est donc en étendant la règle de Widrow-Hoff que plusieurs équipes de chercheurs ont développé un algorithme d'apprentissage appelé rétropropagation du gradient de l'erreur, généralisé ensuite par l'équipe de Rummelhart en 1986. Cet algorithme fournit une façon de modifier les poids des connexions de toutes les couches d'un Perceptron Multi Couches (PMC) [47] (figure 2.3).

Cet algorithme est réalisé afin de répondre à la question : comment répercuter , sur chacune des connexions, le signal d'erreur qui ne peut être mesuré que sur la couche de sortie après avoir traversé plusieurs étapes non linéaires ?.

Pour cet algorithme de même que l'on est capable de propager un signal provenant des cellules d'entrées vers la couche de sortie, on peut en suivant le chemin inverse, rétro propager l'erreur calculée en sortie vers les couches internes.

L'algorithme de rétro propagation du gradient de l'erreur a permis de dépasser les limites du Perceptron simple. Il s'avère capable de résoudre un grand nombre de problèmes de classification et de reconnaissance de formes et a donné lieu à beaucoup d'applications. Cet Algorithme souffre néanmoins de nombreux défauts, parmi lesquels [27]:

- le temps de calcul : l'apprentissage est très long ;
- une grande sensibilité aux conditions initiales, c'est-à-dire à la manière dont sont initialisés les poids des connexions ;
- fe nombreux problèmes sont dus à la géométrie de la fonction d'erreur : minimums locaux. Ce problème est en partie résolu avec le gradient stochastique, mais il subsiste quand même ;
- le problème de dimensionnement du réseau : La rétro propagation apprend une base d'apprentissage sur un réseau dont la structure est fixée a priori. La structure est définie par le nombre de couches cachées, le nombre de neurones par couches et la topologie des connexions. Un mauvais choix de structure peut dégrader considérablement les performances du réseau.

2.5.2 Les Réseau de neurones à Fonctions de Base Radiales (RFB)

Les réseaux à fonctions de base radiales (RBF) ou plus simplement réseaux à bases radiales ont été proposés par J. Moody et C. Darken. On retrouve une organisation comportant une couche d'entrée, une couche cachée et une couche de sortie. Chaque neurone caché ne réagit ici qu'à une petite partie de l'espace d'entrée (sa zone d'influence).

Pour un réseau comportant *n* entrées et *m* unités cachées, l'activation des neurones cachés est donnée par une fonction de type gaussienne (formule 2.3) (les fonctions d'entrée et d'activation sont confondues):

$$a_i = \exp\left(-\frac{1}{2} \sum_{k=1}^{n} \frac{\left(e_k - c_{k,i}\right)^2}{\sigma_{k,i}^2} \right) = \prod_{k=1}^{n} \exp\left(-\frac{1}{2} \frac{(e_k - c_{k,i})^2}{\sigma_{k,i}^2} \right) \qquad (2.3)$$

Où *i* désigne l'indice du neurone, *k* parcourt l'ensemble des entrées notées e_k, et c_{ki} et $\sigma_{k,i}^2$ sont des paramètres appelés respectivement centres et variances des gaussiennes. La figure 2.12 présente la forme de cette fonction d'activation pour un neurone possède une seule entrée.

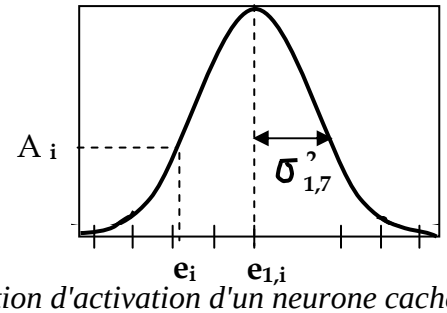


Figure 2.12. *Fonction d'activation d'un neurone caché possédant une seule entrée*

Chacun de ces neurones ne s'active donc de manière significative que pour des valeurs d'entrée relativement proches des centres des gaussiennes. Les connexions provenant des neurones d'entrée ne sont pas pondérées. L'activation d'un neurone de sortie d'indice i est donnée par :

$$\sigma_i = \frac{\sum_{j=1}^{m} w_{ij} a_j}{\sum_{j=1}^{m} a_j} \qquad (2.4)$$

Où *j* parcourt l'ensemble des indices des neurones cachés. Les neurones de ce type réalisent donc une somme pondérée des valeurs d'activation des neurones cachés. Le terme $\sum_{i=1}^{m} a_i$ appelé facteur de normalisation n'est pas obligatoire. On parle de réseau normalisé lorsqu'il est employé.

L'apprentissage se fait dans ces réseaux par modification des poids des connexions entre les neurones cachés, les neurones de sortie, les centres et les variances des gaussiennes. On réalise comme précédemment une descente de gradient ayant pour but de minimiser l'erreur quadratique, dont l'expression est donnée par l'équation (2.5).

$$q = \frac{1}{2} \sum_i [a_i - s_i]^2, \qquad (2.5)$$

Ce modèle souffre cependant d'un inconvénient par rapport aux réseaux Multi-Couches puisque contrairement à ceux-ci, son domaine d'approximation (i.e. domaine dans lequel il réalise une approximation satisfaisante) est strictement borné. Ce dernier se limite en effet aux zones d'influence des neurones cachés, en dehors desquelles le réseau est incapable d'extrapoler. Lorsque la dimension ou la taille du domaine d'entrée est très importante, le nombre de neurones nécessaires peut devenir considérable et l'emploi de réseaux Multi-Couches peut êtres plus approprié.

2.5.3 Réseaux auto-oganisateur (réseau de kohonen)

Les cartes topologiques ou cartes auto organisatrices ont été introduites pour la première fois par T. Kohonen en 1981. Les premiers modèles cherchaient tout particulièrement à représenter des données multidimensionnelles. La particularité la plus importante des cartes auto-roganisatrices est qu'elles rendent possible la comparaison des groupements qui ont réalisés directement à partir des données. Une observation est affectée à un groupe qui est projeté en un nœud de la carte. La comparaison des projection liées à deux observation distinctes permet d'apprécier la proximité des groupes dont elles sont issues [27].

Dans le plus part des applications, les neurones d'une carte de Kohonen sont disposés sur une grille 2D (figure 2.13). Chaque neurone i de la carte effectue un calcul de la distance euclidienne entre le vecteur d'entrée et le vecteur poids W_i.

Dans les réseaux de kohonen, la mise à jour des paramètres des neurones s'effectue sur tout un voisinage d'un neurone i ,Un rayon de voisinage r représente donc la longueur du voisinage d'un neurone i en terme de nombre de neurones. On définit alors une fonction $\hbar(i,k)$ égale à 1 pour tous les neurones **k** voisins du neurone i compris dans le rayon r et égale à zéro pour tous les autres neurones.

L'algorithme d'apprentissage de la carte de Kohonen se présente comme suit :

Après un long temps de convergence, le réseau évolue de manière à représenter au mieux la topologie de l'espace de départ. Il faut noter que la notion de conservation de la topologie est en fait abusive puisqu'en général, la taille du vecteur d'entrée est bien supérieure à la dimension de la carte (souvent égale à 2) et il est donc impossible de conserver parfaitement la topologie.

Algorithme. Appr*entissage de réseau de* ***Kohonen***

- ✓ Initialiser aléatoirement les vecteurs ***Wi***. On donne une valeur initiale au rayon r et au taux d'apprentissage $\boldsymbol{\eta}$.
- ✓ Calculer la distance euclidienne entre le vecteur présenté ξ et le vecteur de poids de chaque neurone
  - Choisir le neurone ***k*** ayant la distance la plus petite,
  - Les vecteurs de pondération de tous les neurones i de la carte de Kohonen sont alors mis à jour selon l'équation :

$$w_i \leftarrow w_i + \eta h(i,k)(\xi - w_i))$$

  - Réduire la taille du voisinage et reprendre l'apprentissage du vecteur des pondérations.

Algorithme 2.2. *Apprentissage de la carte de Kohonen*

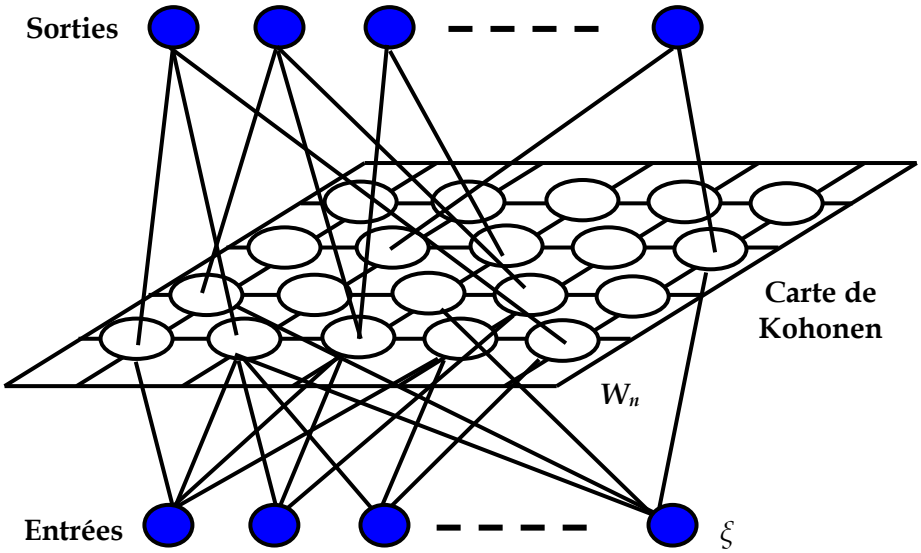


Figure 2.13. *Carte topologique auto-adaptative de Kohonen*

Les avantages de la carte auto organisatrice sont :

- L'espace de sortie est un espace de représentation donc on peut visualiser les sorties de la carte.
- représentation des données de grande dimension.

L'un des inconvénients est le temps de convergence très long. Il n'y a pas de preuve de convergence en multidimensionnel et d'unicité de la représentation.

2.6 Les différentes applications de RNAs

Les réseaux de neurones sont utilisés dans de plus en plus de domaines, citons la classification, la reconnaissance de formes, l'identification et la commande de processus. Le choix d'utiliser tel ou tel type de réseau de neurones dépend de l'application mais aussi des capacités de traitement du processeur sur lequel ils s'exécutent.

2.6.1 Reconnaissance de formes

C'est un domaine privilégié d'application pour les RNAs et c'est lui qui a marqué leur début. Le terme de reconnaissance est un terme général qui désigne les processus traitant des données pour en extraire des informations afin de parvenir à leur classification. Une partie non négligeable des applications neuronales actuelles appartiennent à cette catégorie, parmi lesquelles nous pouvons citer [18] :

- Reconnaissance de caractères Latins, Chinois, Coréens, Japonais, Russes ;
- Reconnaissance de visages ;
- Classification d'images et de documents ;
- Diagnostic des maladies ;
- Diagnostic des pannes dans les installations électriques, nucléaires, mécanique, ainsi que dans les systèmes de télécommunications.

2.6.2 Traitement de la parole

Le traitement des signaux stochastiques (filtrage, classification) provenant des systèmes non linières est une tache difficile avec les méthodes classiques de traitement.

En effet, la résolution de ce type de problèmes implique la capacité de modéliser d'une façon adéquate, le système inconnu afin d'estimer sa sortie, ce qui n'est possible que pour des cas simples. La capacité d'apprentissage des réseaux de neurones leur permet d'extraire les propriétés statistiques des systèmes non linéaires, ce qui les rend aptes à traiter ce type de problèmes. Parmi les applications réalisées dans ce domaine nous pouvons citer [18] :

- Filtrage et suppression de bruits : amélioration du rapport signal/bruit.
- Analyse de signaux;
- Diagnostic de pannes;
- Démodulation Multi-Utilisateurs;
- Classification des cibles radars dans les avions ;
- Classification des bruits;
- Compression [Schmidhuber96] et fusion [Wol_93] des données.

2.6.3 Détection d'anomalies

Ceci est une dérivation des reconnaissances de formes. On apprend à un réseau une image du fonctionnement « normal » d'un système et celui-ci sera ainsi capable d'indiquer tout état de dysfonctionnement quand certains paramètres engendrent une image « anormale ». On trouve ce genre de système pour la surveillance de disjoncteurs à très haute tension (dans ce cas précis on écoute les bruits émis par le disjoncteur) ou encore dans la détection de colis piégés (examen du rayonnement gamma induit dans divers matériaux).

2.6.4 Traitements dépendant du temps

Dans ce cas-ci on se retrouve devant des problèmes de prédiction et, d'identification et de commande de processus.

- *Prédiction :* Ici on va surtout utiliser des réseaux récurrents. Beaucoup de travaux sont consacrés à la prédiction de données financières et boursières et quelques résultats semblent devoir entretenir un optimisme mesuré.
- *Identification et commande de processus. :* On retrouve ce genre d'applications dans l'aide au pilotage de systèmes spécifiques tel que les réacteurs chimiques, une automobile sans pilote etc. Ces applications demandent peu de ressources car les temps caractéristiques sont longs.

Nous pouvons citer également les applications bancaires et financières, le traitement de la parole et la robotique

2.7 Les RNAs appliqués en diagnostic Industriel

Nous présentons les deux architectures les plus utilisées en surveillance industrielle, à savoir le Perceptron Multi Couches (PMC) et les Réseaux à Fonctions de base Radiales (RFR).

Toute la phase d'apprentissage supervisé du réseau de neurones dépend de son analyse des modes de fonctionnement du système. Chaque mode est caractérisé par un ensemble de données recueillies sur le système. A chaque mode, on associe une expertise faite par l'expert.

Cette association (ensemble de données - modes de fonctionnement) sera apprise par le réseau de neurones. Après cette phase d'apprentissage, le réseau de neurones associera les classes représentant les modes de fonctionnement aux formes d'entrée caractérisées par les données du système le principe de cette approche est illustré sur la (figure 1.5, chapitre 1).

On peut donc dire que les RNAs sont utilisées en surveillance industrielle soit comme outil

- secondaire pour la surveillance, c'est-à-dire comme un approximateur de fonctions pour l'identification des systèmes dynamiques grâce à une boite neuronale,
- principal de détection et diagnostic, en l'occurrence tous les travaux de classification.

Les RNAs peuvent fournir, dans certains cas, des solutions plus intéressantes que les autres outils de surveillance à condition de choisir judicieusement le type de l'architecture neuronale et surtout de bien mener le processus d'apprentissage.

2.7.1. Application réalisées par RNAs

Une application présentée par [47] [39] pour la détection et localisation des défauts capteurs d'une centrale d'épuration hydraulique par la reconstruction des signaux capteurs avec une comparaison de deux architectures neuronales: le Perceptron Multi Couches comparé à la carte de Kohonen. Le PMC est constitué de cinq couches, six neurones d'entrée et six neurones de sortie. Son objectif est de reconstituer six mesures de six sorties capteurs après un apprentissage par rétro propagation. Dans ce cas, ce réseau peut donc être considéré comme une mémoire auto associative. La détection ainsi que la localisation sont effectuées après une phase de comparaison avec seuillage de la sortie estimée avec la sortie réelle du capteur. Cette mémoire auto associative est donc comparée à la carte de Kohonen appelée carte topologique auto adaptative. Cette carte contient 15*15 neurones avec un vecteur d'entrée de 6 neurones (dimension de l'ensemble des sorties capteurs). Chaque neurone de la carte est caractérisé par un prototype et un paramètre définissant le rayon d'influence, déterminés par le processus d'apprentissage non supervisé. Pour chaque vecteur d'entrée, la réponse est donnée par un neurone gagnant qui correspond à celui dont la réponse de la fonction gaussienne est la plus importante. D'après la conclusion des auteurs, les deux techniques ont des performances similaires. Ces réseaux ont été testés sur deux types de fautes isolées et une succession de deux fautes. La carte de Kohonen se montre plus rapide

pour la détection et l'identification du capteur défaillant. La technique devient insignifiante si plus de 50% des variables d'entrée sont erronées [47].

Mais il existe un autre grand problème qui nous donne des idées sur les capacités d'apprentissage des réseaux de neurones : c'est le problème de la classification des fautes et pannes et celui les défaillances. Lors de la mise en place d'un système de diagnostic par reconnaissance de formes, l'expert est censé connaître les modes de bon fonctionnement et certains des modes de défaillances. Une grande partie des modes de bon fonctionnement est généralement fournie par les données du constructeur de l'équipement. Par contre, les informations concernant les modes de défaillance peuvent provenir de deux origines différentes : soit fournies par le constructeur ou par le bureau des études (provenance de haut), soit collectées en cours de fonctionnement de l'équipement (provenance de bas). Ces connaissances sont emmagasinées dans un historique de fonctionnement (base de données). Celui-ci contient les différentes relations de "causes à effets" des situations de dysfonctionnement d'un équipement. L'opération de diagnostic menée par l'expert est souvent très complexe et demande des connaissances ainsi qu'un raisonnement, généralement difficiles à formaliser. Les informations contenues dans l'historique de fonctionnement, représentent la base d'apprentissage supervisé du réseau de neurones (figure 2.14) La réussite d'une telle application est donc tributaire de la qualité des informations contenues dans l'historique de fonctionnement[27].

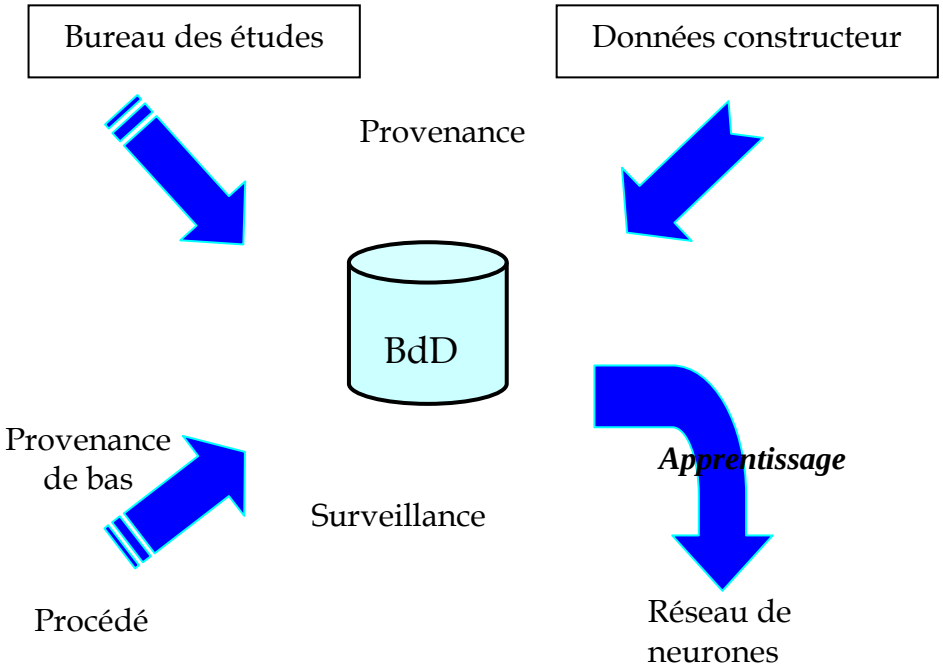


Figure 2.14. *Diagnostic par RNA.*

J.Hines et Miller [47] ont publié un travail sur la surveillance des équipements d'énergie nucléaire. Leur méthode est basée sur une technique hybride qui combine méthode analytique à base de modèle de l'équipement et une technique neuronale qui permet d'identifier la défaillance à partir des résidus obtenus par les techniques classiques de

générations de résidus. Le réseau de neurones utilisé est un réseau à trois couches. Le nombre des neurones en entrée du réseau est donc égal au nombre de résidus. Chaque neurone de sortie représente une classe de fonctionnement (mode de fonctionnement de l'équipement). [47].

Les auteurs semblent hésiter entre deux types d'architectures neuronales : le Perceptron Multi Couches et le Réseau à Fonctions de bases Radiales. Le PMC est considéré comme une méthode de discrimination non linéaire, par contre, le RFR ne couvre qu'une partie de son espace de données en fonction de ce qu'il a appris. Ce réseau est donc capable de dire « je ne sais pas », contrairement au PMC qui peut mal classer un mode nouvellement rencontré et induire ainsi l'expert en erreur. Les auteurs ont tout de même choisi le PMC, malgré les arguments précédemment évoqués. La raison essentielle de leur choix est que leur base de connaissance (base d'apprentissage) était assez exhaustive pour couvrir la quasi-totalité de l'espace des données d'entrée. Cette technique hybride semble donner des résultats assez satisfaisants. L'intérêt majeur des réseaux de neurones dans cette application par rapport aux autres méthodes classiques de détection des défaillances, concerne la résolution du problème de la redondance des alarmes.

2.7.2 Exemple d'application

Dans l'article proposé par [61] intitulé « Diagnostic de défauts du rotor de l'hélicoptère par SOM supervisées », une application des RNA appliquée au problème de la classification supervisée est proposée. Cette application entre dans le cadre du diagnostic de défauts des rotors d'hélicoptères par analyse de signatures vibratoires. La première phase expérimentale, eut pour objectif de proposer un nouvel algorithme pour la supervision de cartes auto-organisatrices de Kohonen (Self Organizing Maps : SOM). Les performances de ce nouvel algorithme furent analysées à l'aide de signatures vibratoires générées à l'aide d'un modèle paramétrique de l'hélicoptère. La seconde phase expérimentale, faisant l'objet de cet article, valide le réseau de neurones proposé et son architecture à l'aide de signatures vibratoires mesurées en vol dédiées au réglage de la voilure de l'hélicoptère. Cette phase précède la phase de validation en vol qui consistera à l'analyse des performances du classifieur neuronal retenu pour le diagnostic du rotor de l'hélicoptère.

Le processus complet de diagnostic du rotor, décrit figure 2.15, se décompose en deux étapes :

La mesure des signaux vibratoires en vol, actuellement utilisée pour le réglage du rotor, qui comprend l'échantillonnage et le traitement harmonique des accélérations mesurées par analyse de Fourrier,

Le diagnostic qui consiste à détecter puis localiser les déréglages et (ou) éléments mécaniques défectueux du rotor.

Actuellement, aucune méthode scientifique ne contrebalance la connaissance des experts (pilotes, ingénieurs et mécaniciens navigants) sur la caractérisation des défauts du rotor.
La méthodologie présentée dans cet article a pour objectif la détection et la localisation de défauts et de déréglages des rotors d'hélicoptères. Elle consiste, à partir de représentations de chacun des états du système à travers des signatures vibratoires, à appliquer une discrimination entre elles à l'image des méthodes de reconnaissance des formes, ou plus généralement de classification de données, nécessitant une phase d'apprentissage amont de chacun des états dynamiques de l'hélicoptère.

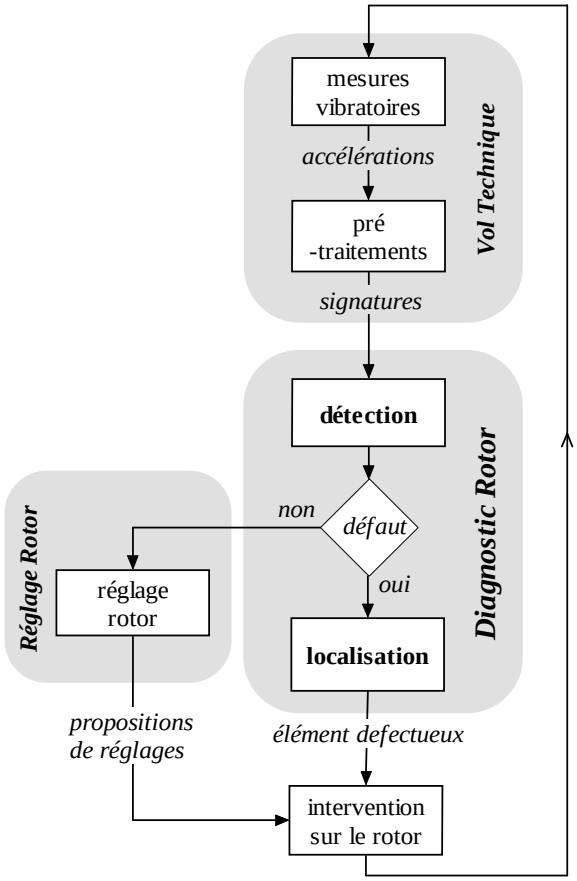


Figure 2.15 *processus de réglage et de diagnostic du rotor principal de l'hélicoptère.*

Ce contexte est idéal et rarement rencontré en pratique, car les systèmes à surveiller sont généralement trop coûteux et (ou) trop critiques pour que l'on puisse envisager d'y injecter des défauts [61].

L'algorithme d'apprentissage de SOM utilisé se décompose en deux étapes. La première consiste à initialiser et organiser de l'espace caractéristique de la SOM en k sous-espaces disjoints (figure 2.16) de dimensions identiques, correspondants au nombre de classes : ils seront les k espaces de référence pour la supervision des k classes lors de l'apprentissage. Les k espaces sont organisés autour de l'espace de référence C_{ref} correspondant à la classe des signatures du rotor réglé sans défaut. La seconde étape est celle

de l'apprentissage qui consiste au calcul itératif des poids synaptiques, pour N_{iter} itérations, vis à vis de l'échantillon de vecteurs d'apprentissage choisi et de leur classe d'appartenance (supervision).

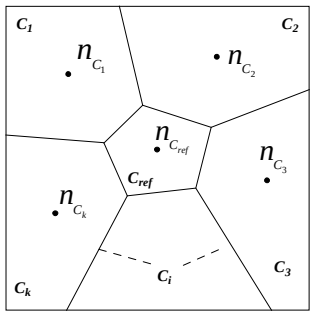


Figure 2.16 *Organisation de l'espace caractéristique de la SOM en k sous-espaces.*

La phase de détection, présentée (figure 2.16) dans le processus de réglage du rotor, consiste à attribuer une classe d'appartenance aux signatures vibratoires dans ***Ec.***

La figure 2.17 présente la classification dans l'espace caractéristique de la SOM supervisée des signatures moyennées.

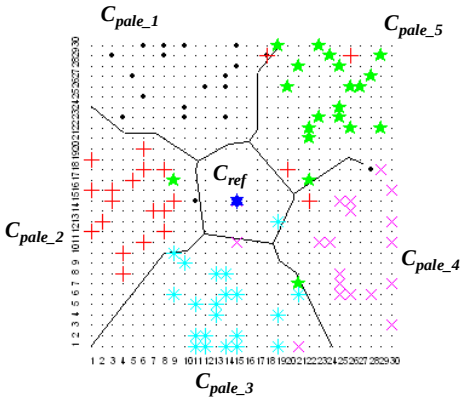


Figure 2.17 Class*ification des signatures moyennées*

L'analyse des performances de détection obtenues (classification des signatures vibratoires dans E_c) présente un taux de performances de 90.16% pour le cas des signatures moyennées et de 88.87% pour la classification des signatures « bruitées ». Les erreurs de classification sont essentiellement dues au positionnement de signatures représentatives de déréglages en masse, dans l'espace caractéristique de la SOM supervisée de la pale opposée. Dans le cas d'un rotor à nombre de pales pair, il ne s'agit plus d'une erreur de classification [61]. L'effet du bruit de mesure a pour impact une réduction des performances égale à 1.29%, qui est jugé faible vis à vis du milieu ambiant de mesure.

De même, l'analyse des performances de localisation obtenues (classification des signatures vibratoires dans C_p) présente un taux de performances de 86.38% pour le cas des signatures moyennées et de 86.07% pour la classification des signatures « bruitées ». Les erreurs de classification sont essentiellement dues au positionnement de signatures représentatives de déréglages en t_{abs}, ayant une faible influence sur la dynamique vibratoire

de l'hélicoptère. La figure 4 montre une répartition des erreurs de classification homogène des signatures C_{tabs} par rapport aux signatures des classes C_{masse} et C_{bielle}. Par contre, il n'y a pas de confusion entre les signatures appartenant aux classes C_{masse} , C_{bielle} et C_{ref}. Dans ce cas, la faible convergence de γ et Φ a un impact négligeable pour la classification des signatures vibratoires dans C_i.

2.8 Conclusion

Les avantages les plus importants que l'on peut attribuer à une application de diagnostic par réseaux de neurones sont :

- la modélisation et estimation de fonctions non linéaires par apprentissage,
- la fusion de données et la généralisation et reconstruction des signaux capteurs.

Deux architectures neuronales sont généralement utilisées pour des tâches de diagnostic : le PMC et RFB.

Des différences majeures existent entres ces deux architectures qui ont une représentation globale pour le PMC et locale pour les RFR. La représentation locale est plus avantageuse pour la surveillance que la représentation globale. L'un des avantages est que contrairement au PMC, les RFR sont capables de dire « je ne sais pas ».

Cette caractéristique propre aux RFR est très importante en diagnostic industriel. Les bases de connaissances du fonctionnement des équipements industriels sont rarement exhaustives. Le système de surveillance se doit donc de pouvoir reconnaître les nouveaux modes susceptibles d'être rencontrés tout au long du fonctionnement de l'équipement. Les RFR attirent ainsi l'attention de l'expert humain sur un nouveau mode rencontré alors que le PMC a tendance à donner une mauvaise réponse et l'expert ne saura jamais qu'un nouveau mode a été rencontré.

Il existe évidement de nombreuses autres variantes de réseaux de neurones, mais dans cette études nous nous sommes limités aux réseaux de neurones de type RFR , PMC car ils s'y prêtent le mieux a notre application .

En fin il est intéressant de remarquer qu'il existe une équivalence fonctionnelle entre un réseau de neurones RBF et un système d'inférence floue bien que provenant d'origines complètement différentes (les RBF de la physiologie et l'inférence floue des sciences cognitives), pour bien comprendre les principes des systèmes de la logique floue, le chapitre suivant est consacré à la logique floue.

CHAPITRE 3

LA LOGIQUE FLOUE ET LE DIAGNOSTIC INDUSTRIEL

La logique floue permet de faire le lien entre modélisation numérique et modélisation symbolique, ce qui permet des développements industriels spectaculaires à partir d'algorithmes très simples de traduction de connaissances symboliques en entité numérique et inversement .

Dans ce chapitre nous commençons par définir et expliquer la terminologie utilisée en logique floue, la théorie des ensembles flous et ainsi que le mode de raisonnement propre aux variables floues. Enfin les domaines d'application de ce dernier et particulièrement en diagnostic industriel.

3.1 Introduction

La théorie des ensembles flous a également donné naissance à un traitement original de l'incertitude, fondée sur l'idée d'ordre, et qui permet de formaliser le traitement de l'ignorance, et permet de formaliser les systèmes d'informations avancés. Les ensembles flous ont également un impact sur les techniques de classification automatique, et ont contribué à un certain renouvellement des approches existantes de l'aide à la décision.

Dans ce chapitre, nous allons présenter brièvement les concepts fondamentaux de la logique floue dans une perspective industrielle, les différentes applications de cette dernière et particulièrement en diagnostic industriel.

3.2 Les opérations des ensembles flous

La logique floue a été formulée par Lotfi A. Zadeh dans le milieu des années soixante. Elle constitue une généralisation de la logique booléenne classique, et ajoute cependant une fonctionnalité déterminante : la possibilité de calculer un paramètre en disant simplement dans quelle mesure il doit se trouver dans un intervalle de valeurs.

3.2.1 Notions d'ensemble flou

On considère que X est un ensemble non nul. Un ensemble flou A dans X est caractérisé par son fonction d'appartenance $\mu_A : X \to [0,1]$. $\mu_A(x)$ est interprété comme le *degré d'appartenance* d'élément X dans l'ensemble flou A car chaque $x \in X$.

On peut donc dire qu'un élément peut appartenir à un ensemble flou (figure 3.1) de manière graduelle, ce qui rompt avec le tout ou rien de la théorie ensembliste classique.

À l'inverse de la logique booléenne, la logique floue permet à une condition d'être en un autre état que *vrai* ou *faux*. Il y a des degrés dans la vérification d'une condition.

Considérons par exemple la vitesse d'un véhicule sur une route nationale. La vitesse normale est de 90 km/h. Une vitesse peut être considérée comme élevée au-dessus de 100 km/h, et comme plus du tout élevée en dessous de 80 km/h.

La logique booléenne envisagerait les choses de la manière suivante (figure 2.1.a) :

- La vitesse est considérée à 100% comme élevée à partir de 100 km/h, et à 0% en dessous.

La logique floue, à l'inverse, permet des degrés de vérification de la condition « La vitesse est-elle élevée ? »

- La vitesse est considérée comme pas du tout élevée en dessous de 80 km/h. On peut donc dire qu'en dessous de 80 km/h, la vitesse est élevée à 0%.
- La vitesse est considérée comme élevée au-dessus de 100 km/h. La vitesse est donc élevée à 100% au-dessus de 100 km/h.
- La vitesse est donc élevée à 50% à 90 km/h, et à 25 % à 85 km/h.

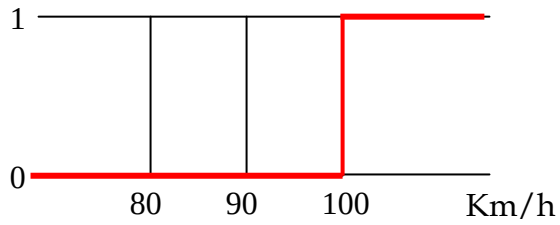


Figure 3.1.a *la vitesse par logique classique*

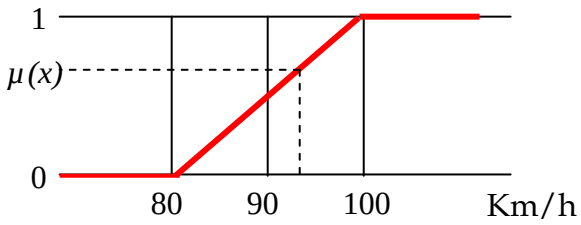


Figure 3.1.b *la vitesse par logique floue*

3.2.2 Les opérations et les normes

Comme dans la théorie des ensembles classiques, on définit l'intersection, l'union des ensembles flous ainsi que le complémentaire d'un ensemble flou. Ces relations sont traduites par les opérateurs « *et* », « *ou* », « *non* ». De nouvelles fonctions d'appartenance liées à ces opérateurs sont établies :

- ***Et* :** opérateur de conjonction : les plus employés sont le « *minimum* »et le « *produit* ». x appartient à A et B $\Leftrightarrow x \in A \cap B \Leftrightarrow \mu_{A\cap B}(x)$. L'opérateur « et » se défini par une norme triangulaire (t-norme) :
  $T : [0,1]\times[0,1] \rightarrow [0,1]$ $(x, y) \mapsto z = xTy$
- ***Ou :*** opérateur de disjonction, les plus employés sont le « *maximum* » et la « *somme* ». x appartient à A ou B $\Leftrightarrow x \in A \cup B \Leftrightarrow \mu_{A\cup B}(x)$. L'opérateur « ou » se défini par une co-norme triangulaire (T*) qu'on appelle aussi S-norme(S):
  S: $[0,1]\times[0,1] \rightarrow [0,1]$ $(x,y) \mapsto z=xSy$.
- ***Non :*** opérateur qui désigne le complémentaire d'un ensemble flou: x appartient au complément de $A \Leftrightarrow x \in \overline{A} \Leftrightarrow \mu_{\overline{A}} - (x)$.

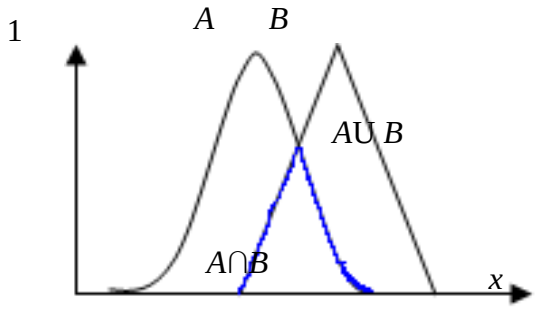


Figure 3.1.c *intersection et union de deux ensembles flous*

3.2.3 Les propositions floues et les variables linguistiques

L'association de sous ensembles flous à des termes linguistique définis sur un univers de discours quelconque, autorise la représentation d'information plus au moins spécifique et précise.

Une variable linguistique peut être définie comme l'association d'une variable classique et de plusieurs sous-ensembles flous caractérisant les valeurs possibles de celle-ci. On appelle une proposition floue élémentaire, une proposition du type **X** *est* **A**, où :

X est une variable linguistique et ***A*** un sous ensemble flou.

Une telle proposition possède un degré de vérité $\mu_A(X)$, compris entre 0 et 1.

Implication floue **(règle IF THEN)**:

L'opérateur d'implication permet d'introduire la notion de règle floue qui caractérise les relations de dépendance entre plusieurs propositions floues quelconques : ***X1*** est *A1* et ***X2*** est *A2* implique ***Y*** est *B*.

Cette règle peut également être exprimée sous forme plus classique :

Si(X1 est A1) ***et***(X2 est A2) ***alors*** (Y est B).

Le tableau 3.1 regroupe les types d'implications les plus utilisées

N°	Nom	Implication I(a,*b*)
01	Kleene-dienes (binaire)	Max {1-a,b}
02	Lukasiewiez	Min {1,1-a-b}
03	Reichenbach	1-a+a*b
04	Fodor	$\begin{cases} 1 & si \quad a \le b \\ \max\{1-a,b\} & si \quad a > b \end{cases}$
05	yager	$\begin{cases} 1 \ si & a = 0 \\ b^a \quad si & a > 0 \end{cases}$
06	Zadeh	Max {min{a,b,1-a}}
07	Dubois-Prade	$\begin{cases} 1-a \quad si \quad b = 0 \\ b \quad si \quad a = 1 \\ 1 \quad si \ non \end{cases}$

Tableau 3.1. *Implications floues [61]*

3.3 Système d'inférence floue

La notion de règle floue permet de définir un système expert flou comme une extension d'un système expert classique, manipulant la proposition floue. Donc un système d'inférence floue (SIF) est formé de trois blocs comme l'indique la figure 3.2. Le premier, bloc de fuzzification transforme les valeurs numériques en degrés d'appartenance aux différents ensembles flous de la partition. Le second bloc est le moteur d'inférence, constitué de l'ensemble de règles. Enfin, le bloc de defuzzification permet, si nécessaire, d'inférer une valeur nette, à partir du résultat de l'agrégation des règles.

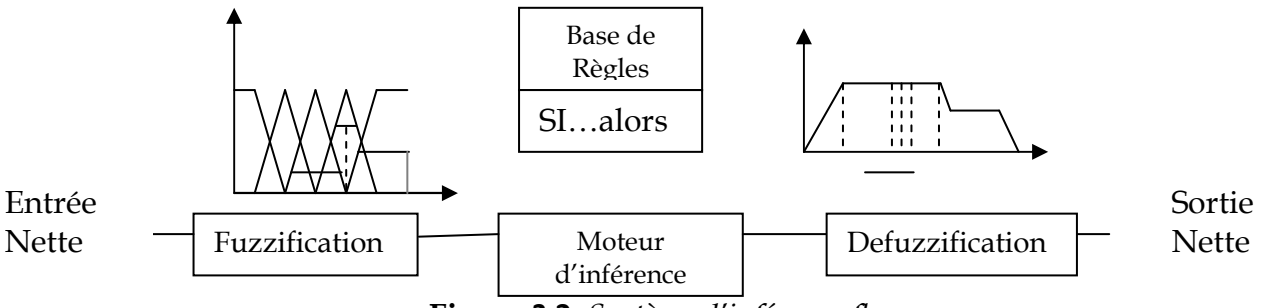


Figure 3.2. *Système d'inférence floue*

3.3.1 Fuzzyfication (quantification floue)

Elle consiste à associer à chaque valeur d'entrée un ou plusieurs sous-ensembles flous ainsi que les degrés d'appartenance associés. Cette étape réalise la transformation de valeurs numériques en informations symboliques floues.

Le choix du nombre d'ensembles flous, de la forme des fonctions d'appartenance, du recouvrement de ces fonctions et de leur réparation dur l'univers de discours n'est pas évident. Il y a cependant des facteurs qui sont plus important que d'autre.

3.3.1.1 Comment fuzzifier?

Pour fuzzifier, il faut donner:

1. L'univers du discours, i.e : Plage de variations possibles de l'entrée considérée.
2. Une partition en classe floue de cet univers.
3. Les fonctions d'appartenances de chacune de ces classes.

3.3.1.2 L'Algorithme fuzzy k-means

L'un des premiers algorithmes, introduit par Bazdek 1974, proposés pour construire automatiquement des fonctions d'appartenance est l'algorithme « fuzzy k-means » ou algorithme des centres mobiles flou. Cet algorithme non supervisé consiste à minimiser itérativement un critère en fonction d'une matrice de partition flou

$U = [\lambda_k(x_i)]_{(k=1,m;i=1,n)}$

où $\lambda_i(x)$ le degré d'appartenance de vecteur x a la classe i.

et $V = (\mu_1 .....\mu_M)$ de la forme :

$$J_m(U,V) = \sum_{i=1}^{N}\sum_{k=1}^{M} \lambda_k(x_i)^m d_k(x_i)^2 \qquad (3.1)$$

Sous les contraintes :

- $\lambda_k(x_i) \in [0,1] \qquad \forall i,k$ (3.2)
- $\sum_{k=1}^{M} \lambda_k(x_i) = 1$ (3.3)

Cette condition traduit le concept de totale appartenance.

$$0 \langle \sum_{i=1}^{N} \lambda_k(x_i) \langle 1 \qquad (3.4)$$

L'ensemble des vecteurs d'apprentissage est constitué de N vecteurs $\{x_1,x_2,\ldots,x_n\}$ susceptibles d'appartenir à M classes $\{\alpha_1, \alpha_2, ..., \alpha_M\}$.

La solution qui minimise la formule (3.1) est donnée par les deux conditions suivantes :

$$\mu_k = \frac{\sum_{i=1}^{N} \lambda_k(x_i)^m x_i}{\sum_{i=1}^{N} \lambda_k(x_i)^m} \qquad (3.5)$$

$$\lambda_k(x_i) = \frac{1}{\sum_{j=1}^{M} (d_k(x_i) \times d_j(x_i))^{-2\times(1-m)}} \qquad (3.6)$$

Les caractéristiques essentielles de cet algorithme sont : Partitionnement, non exclusivité, supervisé.

Algorithme. *Fuzzy k-means*

- initialisation de la matrice de partition floue U_0, t=*0.*
- **Faire**
  - $t \leftarrow t+1$
  - calcul de la matrice des prototypes V^t avec l'équation (3.5).
  - Mise a jour de la matrice de partition floue U^t avec l'équation (3.6).
- **Jusqu'à** $\left\| U^t - U^{t-1} \right\| \leq \varepsilon$

Algorithme 3.1. *Fuzzy k-means*

Apres la convergence de l'algorithme, pour chaque vecteur d'entrée *x,* on calcul la matrice de partition floue avec (3.6)[61].

3.3.2 Inférence floue

La phase d'inférence consiste à calculer le degré de vérité des différentes règles du système, en utilisant les formules données dans la phase de fuzzification, et à associer à chaque règle une valeur de sortie. Cette valeur de sortie dépend de la partie conclusion des règles qui peut prendre plusieurs formes. Donc on peut dire que l'inférence floue est l'opération d'agrégation des règles floues.

Il peut s'agir d'une proposition floue, et l'on parlera dans ce cas de règle de *type mamdani:* SI alors **Y** est **B.**

Il peut également s'agir d'une fonction réelle des entrées, et l'on parlera dans ce cas de règle de *type sugéno* : Si Alors **Y** est $f(x_1, x_2, \ldots, x_n)$.où $x_1, \ldots x_n$ sont des valeurs réelles des variables d'entrée.

Une règle est une combinaison de variables d'entrées, combinaison qui utilise les opérateurs flous, tel que le "ET" et le "OU" (min et max).

Ainsi, pour toutes les combinaisons possibles des entrées, nous allons définir une règle. Cette règle nous indique à quel sous-ensemble de sortie, le coefficient de modification va être attribué.

Les règles d'inférences peuvent être décrites de plusieurs façons [20]

a- *linguistiquement :* on écrit les règles d'une façon explicite.

b- *symboliquement* : il s'agit en fait d'une description linguistique où l'on remplace la désignation des ensembles flous par des observations.

c- *Par matrice d'inférence* : elle rassemble toutes les règles d'inférence sous forme de tableau à deux dimension : Les entrées du tableau représentent les ensembles flous des variables d'entrée. L'intersection d'une colonne et d'une ligne donne l'ensemble flou de la variable de sortie définie par la règle. Il y a autant de cases que de règles.

Il existe plusieurs méthodes pour calculer la valeur représentative d'un ensemble de sortie, dont les principales sont : **MIN / MAX, MAX / PROD, SOM / PROD.**

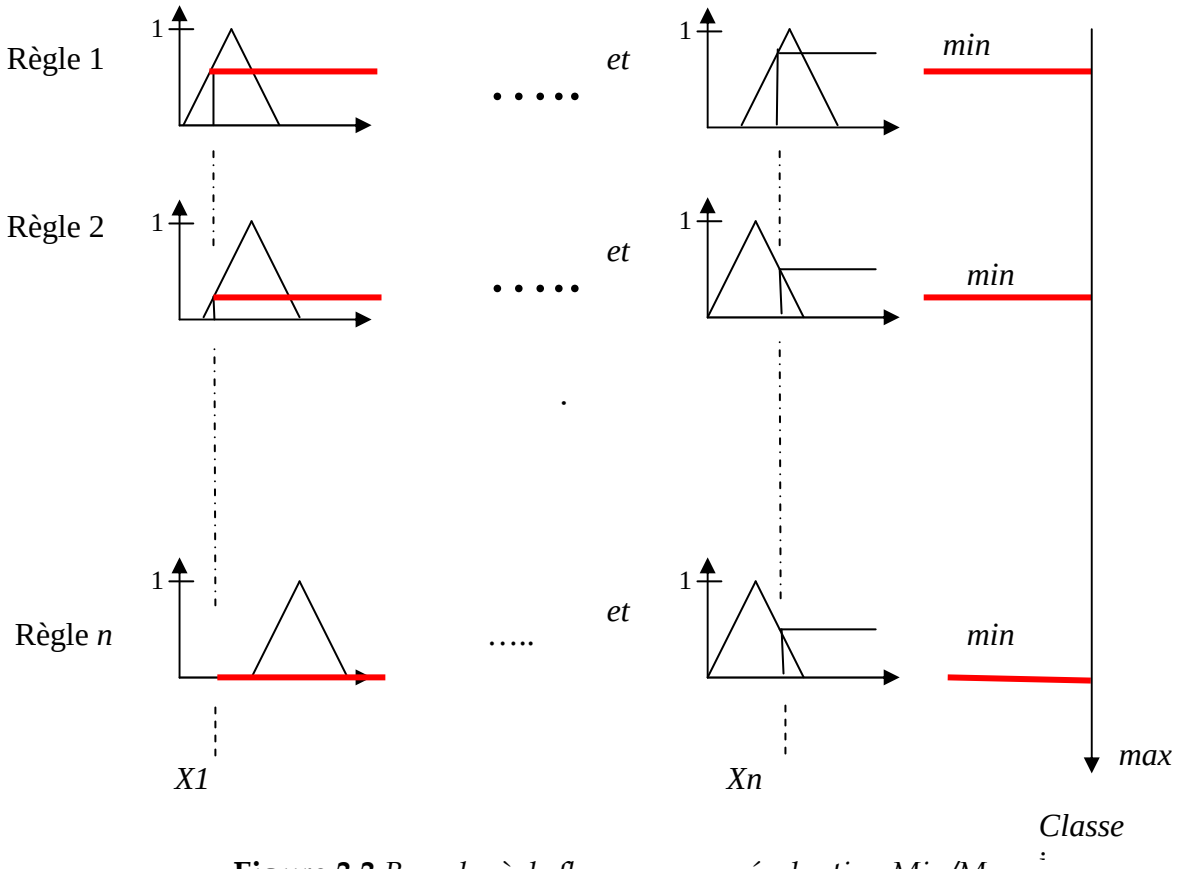


Figure 3.3 *Base de règle floue avec une évaluation Min/Max*

	MIN/MAX	MAX / PROD	SOM / PROD
ET	Min	Min	Prod
OU	Max	Max	Som
ALORS	Min	Prod	Prod
Combinaison	Max	Max	Som
Combinaison des règles activées (OU)			

Tableau 3.2. *Méthodes d'inférence floues.*

3.3.3 Défuzzification ou concrétisation

Le moteur d'inférence fournit une fonction d'appartenance résultante pour la variable de sortie. Il s'agit donc d'une information floue.

Par cette étape se fait le retour aux grandeurs de sortie réelles. Il s'agit de calculer, à partir des degrés d'appartenance à tous les ensembles flous de la variable de sortie, l'abscisse correspondant à la valeur de cette sortie.
Il existe plusieurs méthodes pour calculer la valeur représentative d'un ensemble de sortie, dont les principales sont : Défuzzifications basées sur le centre de gravité des ensembles ; le MIN / MAX ; le SOM / PROD.

3.3.3.1 Défuzzification par le centre de gravité

La méthode de défuzzification la plus utilisée est celle de la détermination du centre de gravité de la fonction d'appartenance résultante $\mu_{RES}(xR)$[52]. Dans ce contexte il suffit de calculer l'abscisse z*. La figure 3.4 montre le principe de défuzzification.

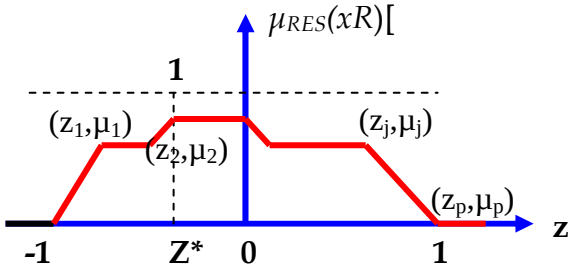


Figure 3.4 *Défuzzification par centre de gravité.*

L'abscisse du centre de gravité peut être déterminée à l'aide de la relation générale suivante :

$$CdG = X^*_R = \frac{\int_{-1}^{1} x_R \mu_{RES}(x_R) dx_R}{\int_{-1}^{1} \mu_{RES}(x_R) dx_R} \qquad (3.7)$$

Lorsque la fonction d'appartenance est composée par morceaux de droits, il est possible de faire les intégrations analytiquement. Avec les coordonnées z_j, μ_j des points d'intersections des p segments de droites.

L'abscisse du centre de gravité peut être calculé par la relation (3.8)

$$z^* = \frac{\sum_{j=1}^{p} (z_{j+1} - z_j)\left[(2z_{j+1} + z_j)\mu_{j+1} + (z_{j+1} + 2z_j)\mu_j\right]}{3\sum_{j=1}^{p} (z_{j+1} - z_j)(\mu_{j+1} + \mu_j)} \qquad (3.8)$$

3.3.3.2 Calcul du centre de gravité lors de la méthode d'inférence SOM / PROD

Cette méthode est la plus utilisée vu son temps de calcul court. Les coefficients issus du moteur d'inférence sont utilisés pour multiplier les fonctions d'appartenance des sous-ensembles de sortie. La valeur de la sortie correspondra au centre de gravité de tous ces ensembles pris individuellement.

Remarque : La position du centre de gravité de chaque sous-ensemble n'a pas été modifiée par le produit, d'où l'avantage d'un calcul simple du centre de gravité global.
Le calcul du centre de gravité peut être ramené au calcul suivant:

$$X^*_R = \frac{\sum_{i=1}^{m} \mu_{Ci} x^*_i S_i}{\sum \mu_{Ci} Si} \qquad (3.8)$$

où μ_{Ci} : Cœfficient de modification à appliquer au sous-ensemble
Si : surface du sous-ensemble i
*X*i* : centre de gravité du sous-ensemble de sortie *i*.

3.3.3.3 Méthode par valeur maximum

Cette méthode est beaucoup plus simple, a valeur de sortie est choisie comme l'abscisse de la valeur maximale de la fonction d'appartenance (figure 3.5).

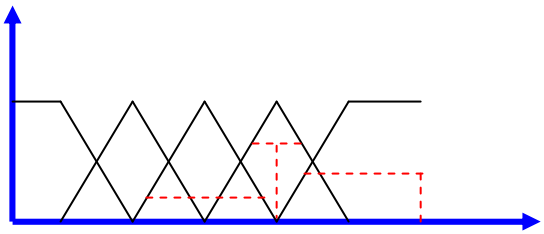
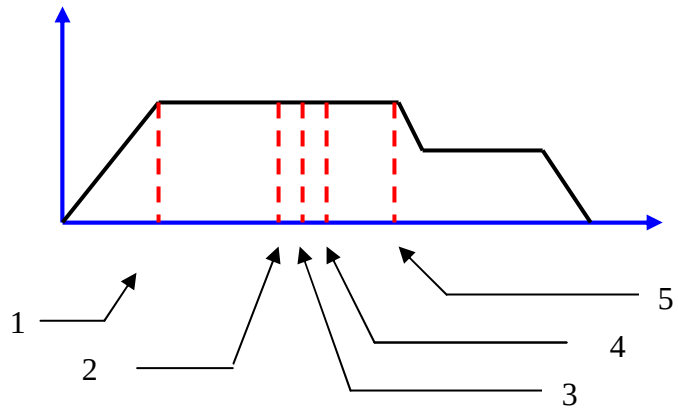


Figure 3.5. *Defuzzification par valeur maximum*

La figure 3.6 illustre la plus part des stratégies de défuzzification.



1-plus petit maximum.
2- moyenne de maxima.
3- centre de gravité.
4- bissectrice de la zone.
5-plus grand maximum.

Figure 3.6. *Stratégies de défuzzification à partir de l'union de plusieurs sous ensembles flous*

3.4 Les applications de la logique floue

Les domaines d'applications de la logique floue concernant principalement les problèmes où les données ne peuvent être formulées de manière explicite, ainsi que des techniques de contrôle et de réglages, lorsque les moyens classiques atteignent leurs limites

(exemples caméra, systèmes non linéaires, etc.).et aussi pour les systèmes contrôlés par des experts humains.

3.4.1 Commande floue

La Commande floue est l'application la plus utilisée de la logique floue. Elle consiste à remplacer les algorithmes de réglage conventionnels par des règles linguistiques. Ainsi, on obtient un algorithme linguistique s'y prête mieux que les méthodes traditionnelles à la commande d'un processus. Le schéma général dune commande floue est illustré sur la figure 3.7 :

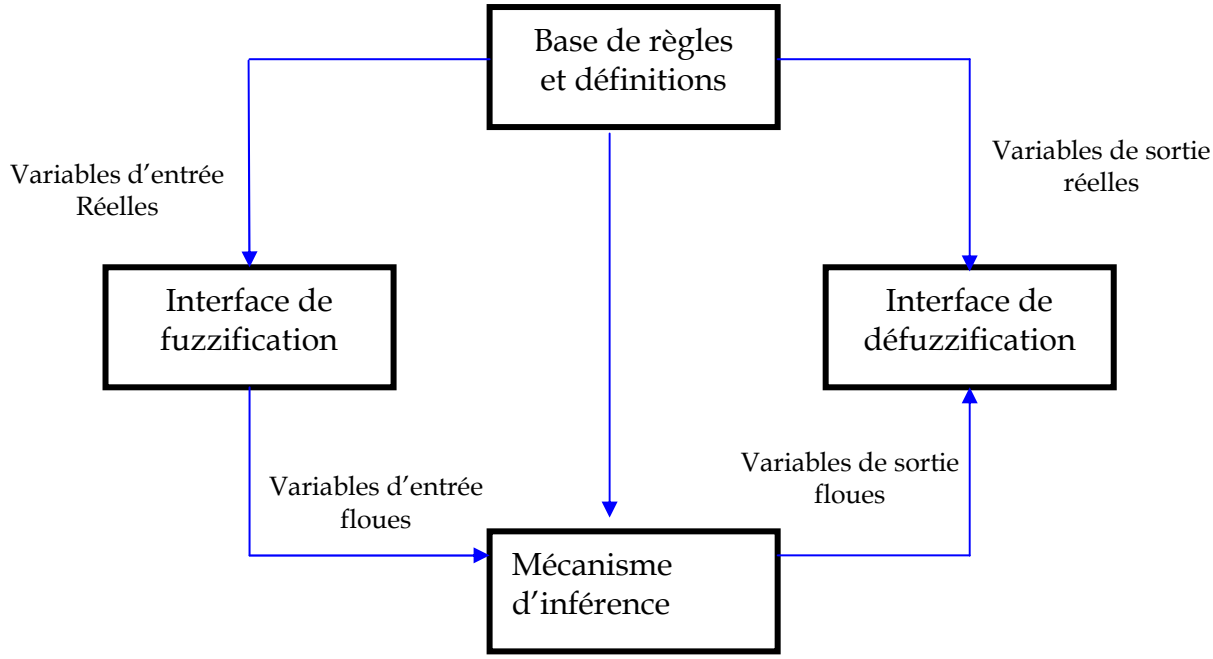


Figure 3.7. *Schéma général d'une commande floue*

Nous procédons tout d'abord à la partition en sous-ensembles flous des différents univers de discours (ou référentiels) que le système impose. Nous déterminons ensuite la base de règles qui va caractériser le fonctionnement désiré du système. Puis il faut transformer les variables réelles, c'est à dire celles qui ont une réalité physique, en variables floues. On appelle cette étape la « fuzzification ». Nous utilisons alors ces variables floues dans un mécanisme d'inférence qui crée et détermine les variables floues de sortie en utilisant les opérations sur les fonctions d'appartenance. Enfin, nous opérons à la defuzzification qui consiste à extraire une valeur réelle de sortie à partir de la fonction d'appartenance du sous-ensemble flou de sortie établi par le mécanisme d'inférence.

3.4.2 Diagnostic industriel

Les applications de la logique floue sont extrêmement nombreuses et variées. Les plus courantes sont les systèmes experts flous, le raisonnement à partir de cas est la reconnaissance floue de formes. Dans le cadre de la surveillance et du diagnostic, on trouve

principalement les systèmes experts. Dans ces différents contextes (aide au diagnostic, aide à la décision), l'expert humain exprime des connaissances ou des données dans un langage naturel fondamentalement imprécis; la logique floue permet donc d'une part de prendre en compte les imprécisions inhérentes aux données et d'autre part de rendre compte de l'expression des règles qui permettent de formuler un diagnostic ou de déterminer une action. On trouve par exemple dans [52] l'architecture d'un outil de détection/diagnostic de station d'épuration, dans lequel la logique floue intervient sous forme d'un système expert flou et dans les étapes de classification (figure 3.8).

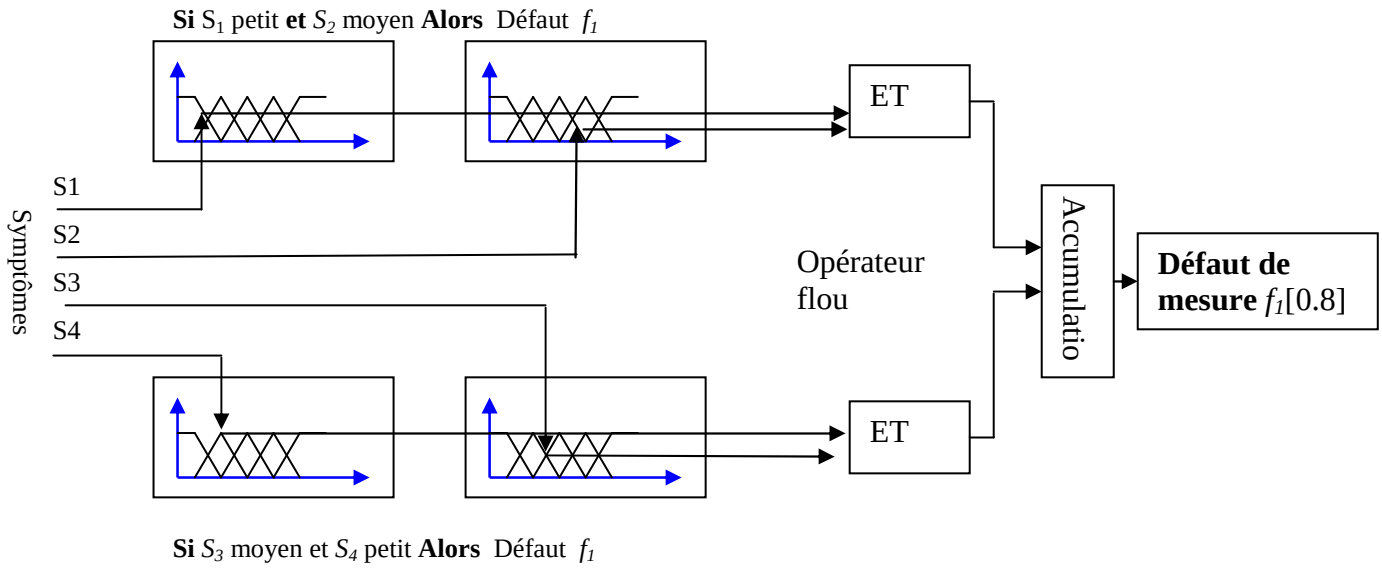


Figure 3.8. *Exemple de diagnostic par logique floue.*

Dans le cadre de la surveillance des systèmes industriels, la logique floue se trouve également associée à d'autres outils et techniques d'analyses. A titre d'exemple, la logique floue a trouvé des applications en combinaison avec les arbres de défaillances afin d'évaluer l'apparition de l'événement sommet [58].

Dans ces diverses applications, l'utilisation de la logique floue est assez naturelle dans la mesure où elle permet de traiter l'imprécision, l'incertitude et l'incomplétude liées aux connaissances du domaine. En plus, la logique floue leur confère une capacité d'utilisation en pronostic. Cependant, même si la logique floue fournit des résultats satisfaisants, on ne peut considérer ces applications comme de véritables applications de la logique floue pour le diagnostic dans la mesure où ces différents outils ne s'appliquent pas à la localisation et à l'identification des causes expliquant un défaut. Utilisée avec les arbres de défaillances, cette dernière devrait fournir une évaluation sur l'occurrence ou la présence des événements de base de l'arbre de défaillances qui sont eux à l'origine de l'événement sommet. On obtiendrait ainsi l'évaluation des causes à l'origine d'un dysfonctionnement.

3.5 Conclusion

Les outils fournis par la logique floue permettent une modélisation des phénomènes pouvant en un certain sens s'approcher du raisonnement humain. Le fait de transcender le tout ou rien des ordinateurs introduit une souplesse faisant la puissance des outils flous dans de nombreux domaines.

Nous avons présenté dans ce chapitre les concepts de base de SIF, le concept de variable linguistique. Nous avons constater que les SIF peuvent en effet raisonner avec l'information imprécise,et expliquer leurs décisions mais ne peuvent pas automatiquement acquérir les règles qu'ils l'utilisent pour prendre ces décisions.

Par ailleurs, les réseaux neuronaux sont bons à reconnaître des modèles mais ne sont pas intéressants pour expliquer comment ils atteignent leurs décisions.

Ces limites ont été une raison derrière la création de systèmes hybrides intelligents où ; deux ou plus de techniques sont combinées dans une manière à vaincre les limitations d'une seule technique.

Les applications montrent les avantages de la logique floue quand le modèle des systèmes est difficile a implémenté. Malheureusement, avec l'augmentation dans la complexité du modèle de l'existence du processus, nous avons rencontré une difficulté pour développer des règles floues et des fonctions d'appartenance.

Ceci nous a conduit au développement d'une autre approche qui est principalement connu comme approche neuro-floue

Dans le chapitre suivant nous faisons une adaptation entre les avantages du modèle de SIFs et les RNA et extraire une typologie (modèle) qui sera compatible avec une application industrielle du module de diagnostic.

CHAPITRE 4

LES SYSTEMES NEURO-FLOUS

Les systèmes neuro-flous sont nés de l'association des réseaux de neurones et de la logique floue, de manière à tirer profits des avantages de chaque une de ces deux techniques . Cependant,la logique floue permet une spécification rapide des tâches à accomplir à partir de la connaissance symbolique disponible. le réglage précis du système obtenu et l'optimisation de ses différents paramètres reste néanmoins beaucoup plus difficile dans de nombreux cas. Par contre les modèle les plus courant de RNA ,n'autorisent pas l'incorporation de connaissance a priori mais permet de régler par apprentissage le comportement précis du système.

Donc, la principale propriété des systèmes neuro-flous est leur capacité à traiter dans une même outil des connaissances numérique et symboliques d'un système. Il permettent donc d'exploiter les capacités d'apprentissage des réseaux de neurones d'un part et les capacités de raisonnement de la logique floue d'autres part.

L'intégration des réseaux de neurones et les systèmes d'inférence flous peut être formulée en trois principales catégories: coopérative , concurrentes et hybride.

Dans ce chapitre nous allons proposer une définition des systèmes neuro-flous, les différents types avec le principe de fonctionnement de chaque catégorie. Nous donnons ensuite les avantages de cette approche qui sera suivi par quelques modèles proposés et réalisés dans différents secteurs.

4.1 Introduction

Les Systèmes hybrides qui combinent la logique floue, les réseaux neuronaux , les algorithmes génétiques, et les systèmes experts prouvent leur efficacité dans une variété de problèmes de monde réel et dans l'industrie.

Par exemple, pendant que les réseaux neuronaux sont intéressants pour reconnaître des modèles, ils ne peuvent pas expliquer comme ils atteignent leurs décisions. De même pour les systèmes de la logique floue qui peuvent raisonner avec l'information imprécise, sont intéressants pour expliquer leurs décisions mais ne peuvent pas automatiquement acquérir les règles qu'ils utilisent pour prendre ces décisions.

Ces limites ont été une raison derrière la création de systèmes hybrides intelligents où deux ou plus de techniques sont combinées dans une manière à vaincre les limitations d'une seule technique.

Dans le chapitre précédent, quelques unes des applications de l'approche de la logique floue ont été présentées. Elles montrent les avantages de la logique floue quand le modèle des systèmes est difficile à implémenter. Malheureusement, avec l'augmentation de la complexité du modèle de l'existence du processus, nous avons rencontré des difficultés pour développer des règles floues et des fonctions d'appartenance.

Cela nous a mené au développement d'une autre approche qui est principalement connu comme approche neuro-floue.

L'intégration des réseaux de neurones et les systèmes d'inférence floues peut être formulé en trois principales catégories: coopérative , concurrente et hybride.

[47] a montre que les réseaux de neurones peuvent approximer n'importe quel système de règles floue et inversement les réseaux de neurones peuvent être approximés avec un système d'inférence de règles floue.. Différents travaux se sont intéressés à l'intégration des RNA et les SIF [3],[7],[11], [12] [14],[22], ,[29], [37].

4.2 Définitions

Les systèmes neuro-flous sont des systèmes flous formés par un algorithme d'apprentissage inspiré de la théorie des réseaux de neurones. La technique d'apprentissage opère en fonction de l'information local et produit uniquement des changements locaux dans le système flou d'origine[10].

Les règles floues codées dans le système neuro-flou représentent les échantillons imprécis et peuvent être vues en tant que prototypes imprécis des données d'apprentissage.

Un système neuro-flou ne devrait par contre pas être vu comme un système expert (flou). Il n'a rien à voir avec la logique floue dans le sens stricte du terme. On peut aussi noter que les systèmes neuro-flous peuvent être utilisés comme des approximateurs universels.

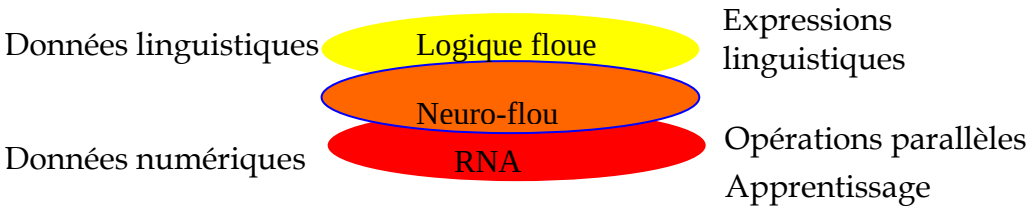


Figure 4.1. *le système neuro-flou*

Afin de résumer l'apport du neuro-flou, le tableau 4.1 regroupe les avantages et les inconvénients des réseaux de neurones et de la logique floue.

Réseaux de neurones	Logique floue
Avantages	
▪ Le modèle mathématique non requis ▪ Aucune connaissance basé sur les règles ▪ Plusieurs algorithmes d'apprentissage sont disponibles	▪ Le modèle mathématique non requis ▪ La connaissance antérieure sur les règles peut être utilisée ▪ Une interprétation et implémentation simple
Inconvénients	
▪ Boite noire(manque de traçabilité) ▪ L'adaptation aux environnements différents est difficile et le réapprentissage est souvent obligatoire(sauf pour RBF) ▪ La connaissance antérieure ne peut pas être employée (apprentissage à partir de zéro) ▪ Aucune garantie sur la convergence de l'apprentissage.	▪ Les règles doivent être disponibles ▪ Ne peut pas apprendre Adaptation difficile au changement de l'environnement ▪ Aucune méthode formelle pour l'ajustement

Tableau 4.1. *comparaison entre la logique floue et les réseaux de neurones.*

Diverses associations de ces deux approche ont été développées depuis 1988 et sont le plus souvent orientées vers la commande de systèmes complexe et les problèmes de classification. Il existe ainsi trois méthodes neuro-floues :

4.3 Le système neuro-flou coopératif

Il utilise des réseaux de neurones et des systèmes flous associés en série ou en parallèle. Plusieurs variantes d'utilisation sont ainsi possibles :

Le réseau de neurones fonctionne en amont du système flou (figure 4.3). Les variantes d'entrées du système flou sont déterminées à partir des sorties du réseau de neurones (dans le cas où elles ne sont pas mesurables directement) ou encore un réseau de neurones effectue une tâche de classification ou de reconnaissance de formes, suivie d'un système flou d'aide à la décision. Un réseau de neurones qui fonctionne en aval du système flou, dans le but d'ajuster les sorties d'un système de commande floue à de nouvelles connaissances obtenues, les variables de sorties étant les erreurs sur les variables de sortie du système flou.

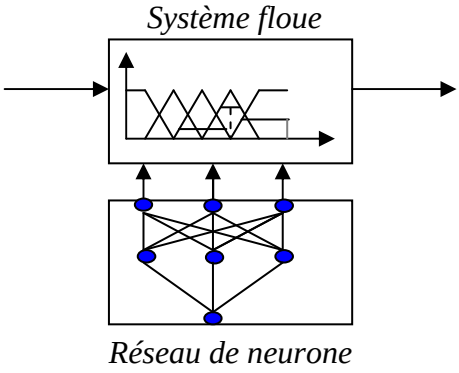


Figure 4.2. *Système Neuro-flou pour régler une fonction d'appartenance*

Deuxièmement le système en série, qui serait utilisé si la sortie n'est pas convenable pour une relation direct à l'entrée du système flou. Les systèmes post-traitement existent aussi et dans lesquels la sortie d'un système flou n'est pas convenable par rapport direct aux systèmes externes, et par conséquent un réseau neurone fournit une interface qui exécute une projection topographique qui ne pourrait pas être porté facilement (figure 4.3).

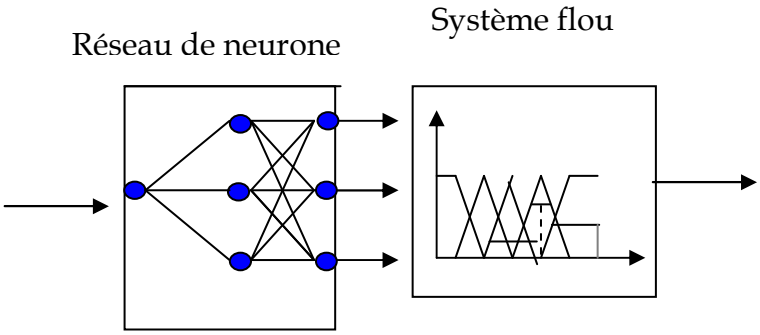


Figure 4.3. *Système neuro-flou en série avec prétraitement neuronal*

On trouve aussi un système neuro-flou parallèle qui fait une coopération entre les réseaux de neurones et les systèmes flous en parallèle en même temps. (figure 4.4).

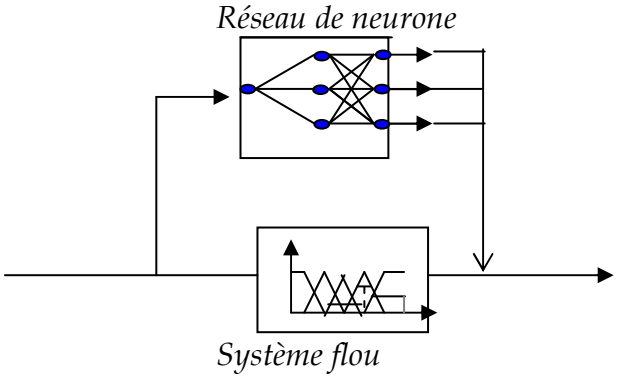


Figure 4.4. *Système neuro-flou en parallèle.*

Ces deux combinaisons des réseaux de neurones et de la logique floue sont des paradigmes qui ont besoin d'une plateforme mathématique très forte, et aussi ne peuvent être implémenté que d'une manière informatisée vu les gigantesques probabilités de calcul .

Le modèle du système neuro-flou coopératif peut être considéré comme un pré-processeur avec des mécanismes d'apprentissage des RNA détermines les fonctions d'appartenance du système d'inférence floue ou les règles floues quand les paramètres de SIF(système d'inférence floue) sont déterminées . .

4.4 Le système de neuro-flou concurrent

Il utilise des réseaux de neurones et des systèmes flous associés en série ou en parallèle. Plusieurs variantes d'utilisation sont ainsi possibles :

Le réseau de neurones fonctionne en amont du système flou. Les variantes d'entrées du système flou sont déterminées à partir des sorties du réseau de neurones (dans le cas où elles ne sont pas mesurables directement) ou encore un réseau de neurones effectue une tâche de classification ou de reconnaissance de formes, suivie d'un système flou d'aide à la décision.

Un réseau de neurones qui fonctionne en aval du système flou, dans le but d'ajuster les sorties d'un système de commande floue à de nouvelles connaissances obtenues, les variables de sorties étant les erreurs sur les variables de sortie du système flou.

4.5 Le système neuro-flou hybride

4.5.1 définition

Les approches neuro-floues modernes sont de cette forme :. Un réseau neuronal et un système flou combinés dans une architecture homogène. Le système peut être interprété comme un réseau neuronal spécial avec des paramètres flous ou comme un système flou mis en application sous une forme distribuée parallèle (figure 4.5).

Dans ce modèle, les algorithmes d'apprentissage des système de neurones sont utilisés pour déterminer les paramètres du système d'inférence floue. Le système neuro-flou partage la structure des données et la représentation de connaissance .Le SIF peut utiliser l'expertise humaine pour implémenter les composantes essentielles dans la base de connaissance et la base de données et exécute le raisonnement flou pour inférer la valeur de sortie totale.

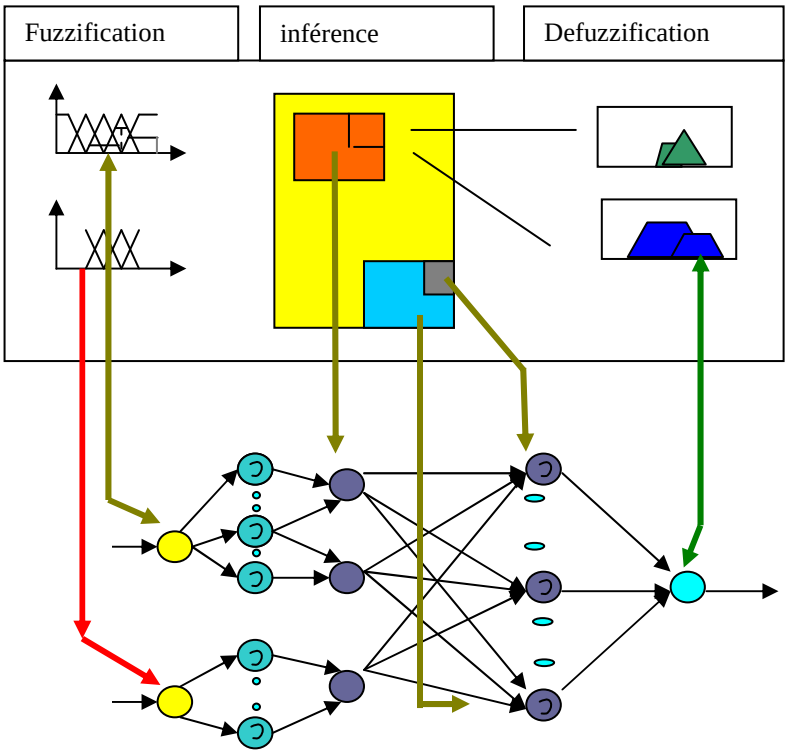


Figure 4.5. *Principe de fonctionnement du neuro-flou hybride.*

4.5.2 Le système neuro-flou hybride de type Mamdani

Le système neuro-flou hybride de Mamdani utilise la technique d'apprentissage supervisé (rétro propagation) pour faire une apprentissage des paramètres de fonctions d'appartenance .

L'architecture du système neuro-flou hybride de Mamdani est illustré sur la figure 4.6.[55] Les détails de chaque couche sont ainsi présentés :

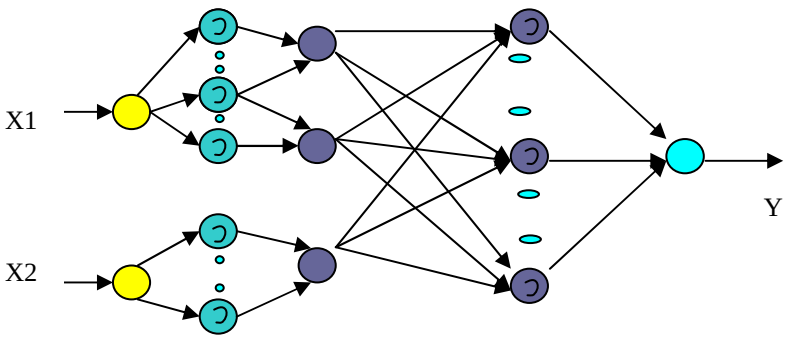


Figure 4.6. *Le système neuro-flou de Mamdani.*

- *la couche 1 (la couche d'entrée) :*

Aucun calcul n'est fait dans cette couche. Chaque noeud dans cette couche qui correspond à une variable d'entrée transmet seulement et directement des valeurs d'entrée à la prochaine couche. Le poids du lien dans la couche 1 est de 1.

- *La couche -2 (la couche de fuzzification)* :

Chaque noeud dans cette couche correspond à un terme linguistique (excellent, bon, etc.) à une des variables d'entrée dans la couche 1. En d'autres termes. Le lien de la production représente la valeur d'appartenance qui spécifie le degré auquel une valeur d'entrée appartient à un ensemble flou est calculé dans couche 2.
Un algorithme de regroupement décidera du nombre initial et du type de fonctions d'appartenance qui doit être alloué à chacun de la variable d'entrée. Les dernières formes du MFs seront réglé pendant l'apprentissage du système.

- *La couche -3 (la couche de permise des règles):*

Un noeud dans cette couche représente la partie antérieure d'une règle. Habituellement un opérateur t-norme est utilisé dans ce noeud. La sortie d'une noeud de la couche 3 représente la « *force du tir* »de la règle floue correspondante.

- *La couche -4(la couche du partie conclusion des règles) :*

Ce noeud a deux tâches fondamentalement : combiner les nouveaux antécédents des règles ,et déterminer le degré auquel ils appartiennent à la variable linguistique de sorties (haut, moyen, bas, etc.).

Le nombre de noeuds dans cette couche sera égal au nombre de règles.

- *La couche -5 (la couche de combinaison et de défuzzification) :*

Ce noeud fait la combinaison de tous le conséquents des règles qui utilise un opérateur t-conorme et finalement calcule la sortie après défuzzification.

4.5.3 le système neuro-flou de type takagi-sugeno

les systèmes neuro-flous de type Takagi-Sugeno font usage d'un mélange d'algorithme de rétro propagation pour faire un apprentissage des fonctions d'appartenance et la méthode de moindre carré pour déterminer les coefficients des combinaisons linéaires dans les conclusions de la règles.

Un pas dans la procédure d'apprentissage a obtenu deux parties: dans la première partie, les modèles de l'entrée sont propagées, et les paramètres de la sortie optimaux sont estimés par la procédure du moindre carré, pendant que les paramètres antérieurs sont supposés être fixés pour le cycle courant à travers l'ensemble de l'apprentissage.

Dans la deuxième partie, les modèles sont encore propagés, et dans cette tâche, la rétro propagation est utilisée pour modifier les paramètres antérieurs, pendant que les paramètres de la conclusion « conséquence » restent fixes. Cette procédure sera alors répétée.

L'architecture du système neuro-flou hybride de TS est illustré sur la figure 4.7. Les détails de chaque couche sont ainsi présentés:

- *les couches 1,2 et 3* sont les même que celle du modèle de Mamdani.

- *la couche -4 (normalisation des poids de la règle):*

Chaque noeud dans cette couche calcule le coefficient de la l'i^{eme} règle à la somme de toutes les règles .

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}, i = 1,2... \qquad (4.1)$$

- *la couche -5 :* chaque nœud i de cette couche est une noeud fonctionnel

$$\overline{w_i} f_i = \overline{W_i}(p_i x_1 + q_i x_2 + r_i \qquad (4.2)$$

tel que $\overline{w_i}$ est une sortie de la couche 4, et $\{p_i, q_i, r_i\}$ sont des paramètres des ensembles.
Un chemin bien établi est de déterminer les paramètres conséquents qui utilisent l'algorithme des moindres carrés.

- *La couche -6 (la couche d'inférence des règles) :* Le seul nœud de cette couche calcule la somme total des sorties de touts les signaux.
-

$$la\ somme\ total = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \qquad (4.3)$$

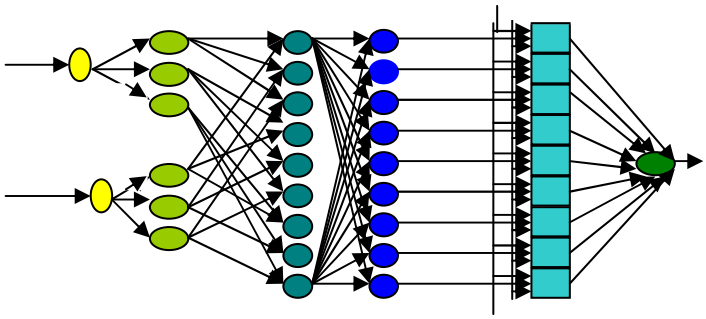


Figure 4.7. L*e réseau de neuro-fiou de takagi-sugeno*

4.6 Les avantages des systèmes neuro-flous

Les réseaux de neurones artificiels	Système d'inférence flou	Système Neuro-flou
La base de règle ne peut être utilisée	La base de règle peut être utilisée	La base de règle peut être utilisée.
L'apprentissage peut être démarré de zéro	Pas d'apprentissage (utilise la connaissance linguistique).	L'apprentissage peut être démarré de zéro ou avec une base de règles initiale
Boite noire	Interprétable(la règle IF-THEN)	Une interprétation et implémentation simples
Complexité des algorithmes d'apprentissage	Universel et implémentation simple	Le modèle mathématique non requis.
Difficulté pour extraire les connaissances	Les connaissances doivent être disponible	Simple pour extraire de connaissances a partir des données (qualitatif, quantitatif).

Tableau 4.2. *Les avantages des systèmes neuro-flous.*

4.6.1 La rapidité de calcul

Les systèmes neuro-flous sont très rapides parce que l'évaluation d'un système compétent implique typiquement des opérations de la comparaison simples et un nombre limité de calculs de la surface linéaire. Cette accélération est analogue à l'algorithme de l'alpha bêta de la théorie des jeux. Les arbres de décision sont très rapides, et la comparaison des paramètres est faite d'une manière floue pas comme la méthode classique et enfin la base des règles floue sera repartie et que chaque règle fonctionne d'une manière autonome et en même temps collectif .

4.6.2 La Flexibilité

Un système neuro-flou peut traiter des problèmes complexes avec beaucoup de variables de l'entrée. Au lieu d'adapter une architecture interne fixe, l'architecture d'un Système neuro-flou peut grandir dynamiquement et efficacement en réponse à la complexité des données d'apprentissage. La structure d'un système neuro-floue représente efficacement le rapport entre les entrées de votre problème et les sorites.

4.6.3 Généralisation des connaissances

Les systèmes d'apprentissage ont souvent des difficultés quand il y a un manque de données historiques pour former, ou bien les données contiennent trop de bruit. Les

Systèmes de neuro-flous peuvent compenser ces problèmes pendant la phase de l'apprentissage en changent(adapter) leur structure interne. Souvent les règles sont de la forme: plus des entrée , plus des résultas satisfaisante .

4.7 Modèles neuro-flous

Dans les sections suivantes, nous discutons brièvement les différents modèles neuro-flous qui font usage des complémentarités de réseaux neuronaux et systèmes d'inférence flou rendent effectif un SIF de type Mamdani ou Takagi Sugeno .

Nous citons quelques travaux dans ce domaine comm *GARIC, FALCON, ANFIS, NEFCON, NEFCLASS, NEFPROX, FUN, SONFIN, FIEST, EFUNN, DMEFUNN, EVONF* [29][55][54][59][60].

4.7.1 ANFIS (Adaptative-Network-based Fuzzy Inference System)

ANFIS représente un système à inférence flou mis en application dans le cadre des réseaux adaptatifs. Il utilise la procédure d'apprentissage Hybride.

Cette architecture (Figure 4.8) affine les règles floues obtenues par des experts humains pour décrire le comportement d'entrée-sortie d'un système complexe. Il est implémenté dans la boite à outils « Neuro-Fuzzy » de MATLAB. Ce modèle donne de très bons résultats en poursuite de trajectoire, approximation non linéaire, commande dynamique et traitement du signal.

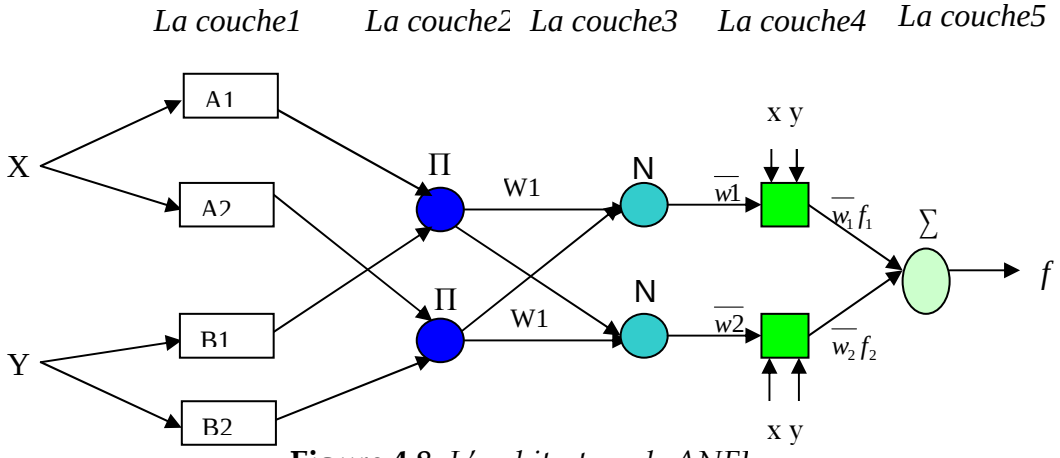


Figure 4.8. *L'architecture de ANFIS*

4.7.2 NEFCLASS (Neuro-Fuzzy CLASSification)

Modèle utilisé généralement en classification. Il est constitué de 3 couches : une couche d'entrée avec les fonctions d'appartenance, une couche cachée représentée par des règles et une couche de sortie définissant les classes [14] (Figure 4.9)

Ce modèle est facile à mettre en application, il évite l'étape de défuzzyfication, tout en étant précis dans le résultat final, avec une rapidité bien supérieure aux autres architectures.

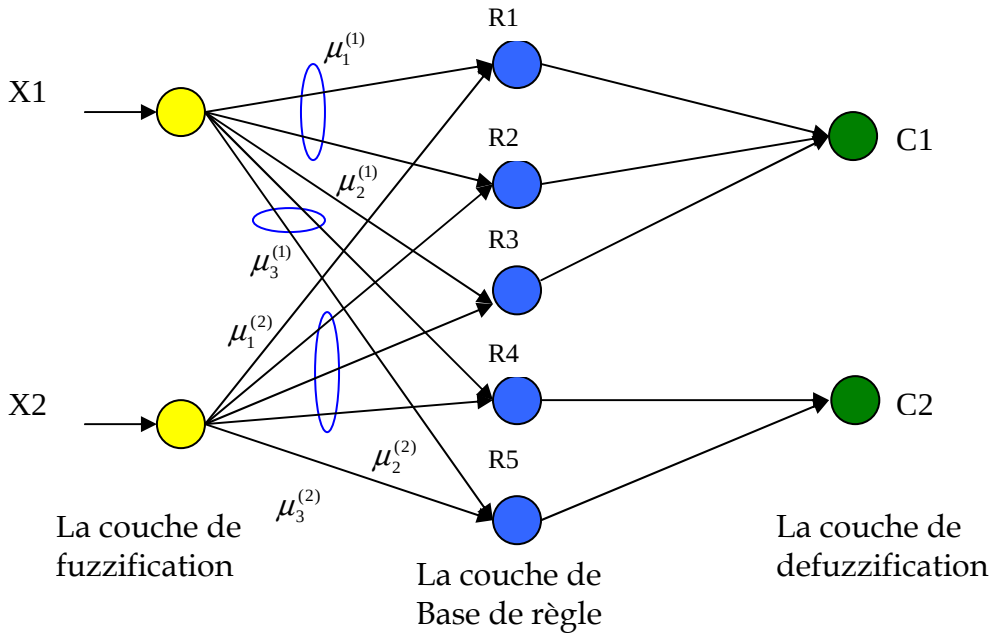


Figure 4.9. A*rchitecture de* NEFCLASS

4.7.3 NEFCON (Neuro-Fuzzy Controller)

Modèle formé de 3 couches. Une couche cachée formée par des règles, une couche d'entrée incluant les noeuds d'entrée avec les sous-ensembles flous d'antécédences et une couche de sortie avec un noeud de sortie et les sous-ensembles des conséquences.

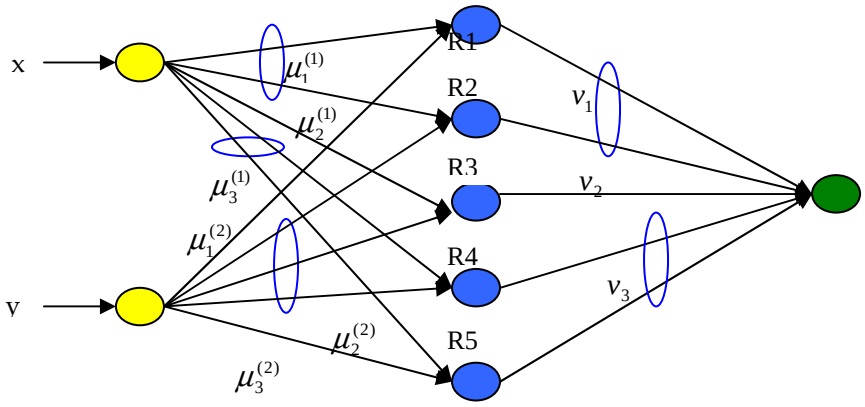


Figure 4.10. A*rchitecture de* NEFCON

L'élaboration des règles est similaire à l'architecture NEFCLASS, avec une différence en sortie [4]. Cette architecture est généralement utilisée en approximation de fonctions et en contrôle flou (Figure 4.10).

Le processus d'apprentissage du NEFCON peut être divisé en deux phases. La première phase consiste à trouver les règles de base initiale. Si les connaissances antérieures ne sont pas disponibles, les règles de base seront apprises avec difficulté. Et si cette règle est définie par un expert l'algorithme les complète. Dans la seconde phase, les règles de base sont optimisées par modification des sous-ensembles flous des règles. Les deux phases

utilisent l'erreur floue, cette erreur peut être trouvée avec la différence entre la sortie désirée et celle obtenue.

4.7.4 NEFPROX (Neuro Fuzzy function apPROXimator)

Modèle obtenu par l'association des deux architectures: NEFCLASS et NEFCON, il est utilisé dans différentes applications comme la classification et l'approximation de fonctions [10]. Le NEFCLASS utilise un algorithme supervisé pour définir les règles floues, le NEFCON utilise un algorithme d'apprentissage non supervisé avec le calcule de l'erreur de sortie. Les deux modèles emploient la rétro propagation afin de définir les sous-ensembles flous.

Comparé au modèle ANFIS, NEFPROX (figure 4.11) est beaucoup plus rapide, mais ANFIS donne de meilleurs résultats en approximation.

Le NEFPROX est le premier système interprétable et lisible, dédié à l'approximation de fonction. Néanmoins, ses résultats en classification reste moins bons que ceux donnés par le NEFCLASS.

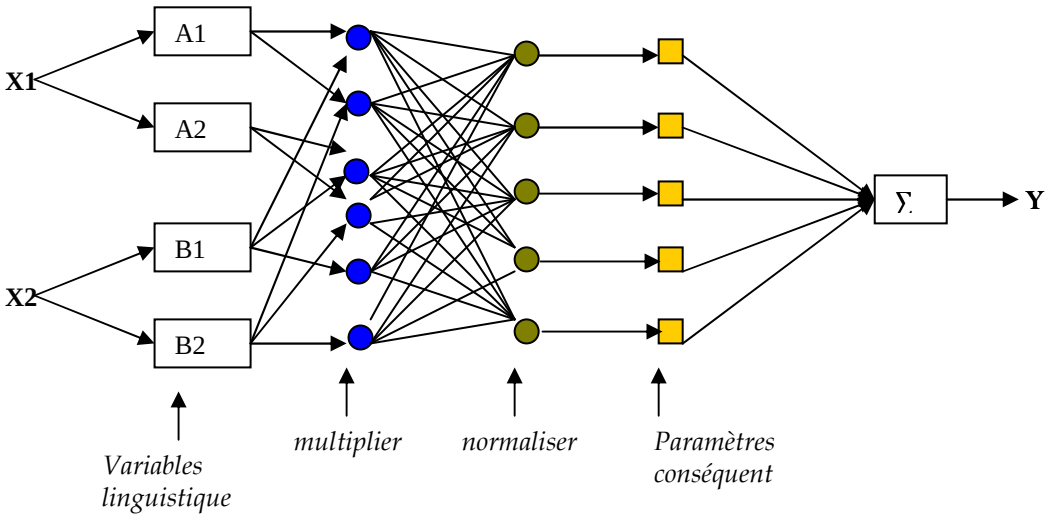


Figure 4.11. *Architecture de NEFPROX*

Nous pouvons initialiser le système NEFPROX si nous savons déjà des règles convenables ou autrement le système est capable à d'apprendre d'une manière incrémentale toutes les règles.

4.7.5 Système neuro-flou hybride (HyFIS)

Il à été introduit par (Kim et Kasabov) en 1999 [16]. Il est constitué de par deux parties :

- Un module d'analyse flou pour l'extraction des règles floues par les données des entées par l'utilisation de la méthode de Wang 1994.

- Un module connexionniste qui implémente et règle les règles floues à travers l'application de l'algorithme du rétro propagation.

Structure d'apprentissage

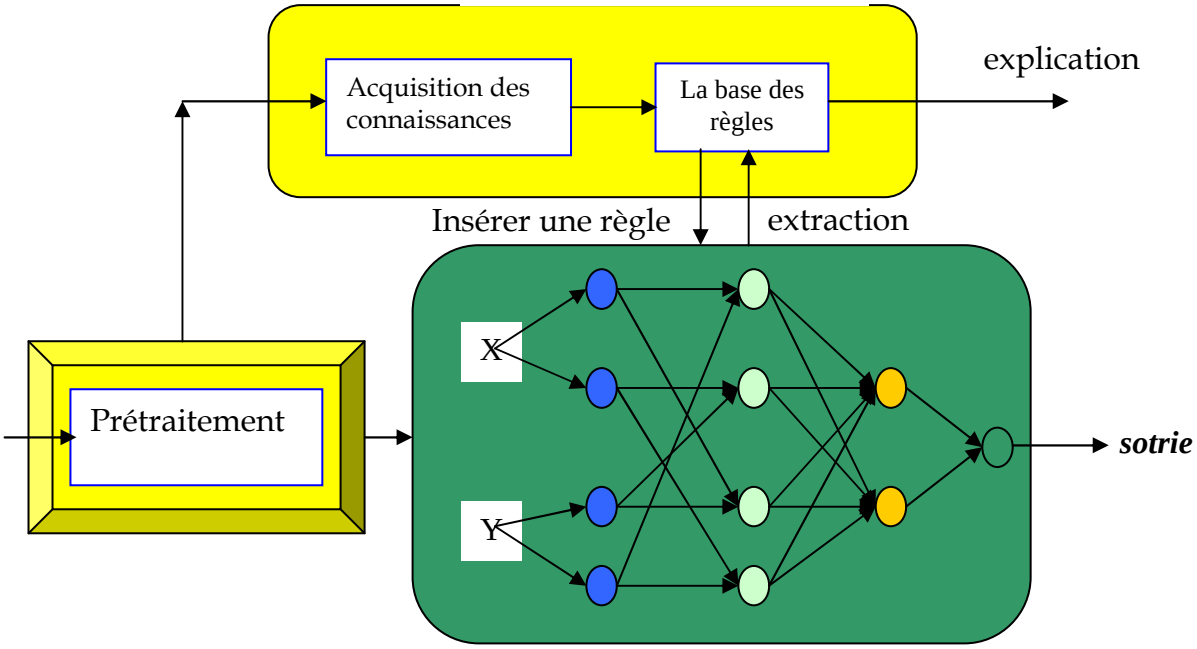


figure 4.12. *Le schéma bloc du diagramme de HyFIS*

4.7.6 Les Systèmes neuro-flous à Fonctions de Base Radiales

L'équivalence entre un système d'inférence ou utilisant des règles de type Sugeno et un système de type RBF est assez intuitive [7]. Les fonctions gaussiennes définissant les activations des unités d'entrée du système peuvent également être utilisées pour représenter les sous-ensembles flous. En utilisant la logique probabiliste, le degré de vérité de la prémisse d'une règle ne contenant que l'opérateur conjonction peut être calculé par un neurone caché d'un réseau RBF .La prémisse Si (X est A) et (Y est B) et (Z est C), est dans ce cas représentée par un neurone possèdent la fonction d'activation suivante :

$$a(x,y,z) = \exp(-\frac{(x-c_a)^2}{\sigma_a^2}) \times \exp(-\frac{(y-c_b)^2}{\sigma_b^2}) \times \exp(-\frac{(z-c_a)^2}{\sigma_c^2}) \qquad (4.4)$$

où x, y et z sont respectivement les valeurs des variables X, Y et Z, et $c_a, c_b, c_c, \sigma_a, \sigma_b, \sigma_c$ sont les paramètres des fonctions d'appartenance des sous-ensembles fous A;B et C.

Chaque sous-ensemble OU est ainsi codé par une dimension de l'entrée du neurone. Si les règles utilisées sont de type Sugeno d'ordre 01, leurs conclusions sont tout simplement représentées par les poids des connexions vers la couche de sortie. Les unités de cette couche concordent avec les sorties du système OU et réalisent par leur fonction d'activation une somme normalisée qui correspond à la défuzzification :

$$sortie = \frac{\sum_j w_j a_j}{\sum_j a_j} \qquad (4.5)$$

où j parcourt toutes les règles dont a_j représente le degré de vérité et wj la sortie. Cette équivalence parfaite permet d'étendre l'utilisation des algorithmes d'apprentissage définis pour les réseaux de type RBF aux systèmes d'inférence fous.
Cette approche impose cependant plusieurs restrictions. Si le même sous-ensemble flou apparaît dans les prémisses de deux règles différentes, il est représentée par deux neurones distincts dont les paramètres peuvent évoluer différemment lors de l'apprentissage. De plus l'utilisation de règles plus complexes comportant notamment des disjonctions ou des négations est impossible avec ce formalisme. Leur représentation demande la modification de la structure du réseau .

4.7.7 Les Systèmes neuro-flous Multi-Couches

Ce choix consiste à utiliser un réseau Multi-Couches pour lequel chaque couche correspond à la réalisation d'une étape d'un système d'inférence floue.

- la couche d'entrée reçoit les valeurs des variables d'entrée ;
- les unités de la ou des premières couches cachées calculent par leur fonction d'activation le degré de vérité des différents sous-ensembles fous (fuzzification) ;
- la couche suivante correspond au calcul du degré de vérité des prémisses des différentes règles ;
- la ou les dernières couches du réseau réalisent la phase de dèfuzzification et fournissent les valeurs de sortie.

Plusieurs réalisations de ces différentes étapes sont possibles. Il est important de noter que l'utilisation d'un algorithme d'apprentissage de type descente de gradient (e.g. la rètropropagation du gradient) impose l'emploi de fonctions d'activation dérivables pour l'ensemble des unités du réseau.

4.7.7.1 Codage des sous-ensembles flous

La première couche d'une architecture de ce type comporte autant de neurones qu'il y a de sous-ensembles flous dans le système d'inférence représenté. Chaque unité calcule le degré de vérité d'un sous-ensemble particulier par sa fonction de transfert. La seule restriction sur le choix de cette fonction concerne sa dérivabilité. On retrouve généralement dans la littérature, l'utilisation de fonctions gaussiennes similaires a celles des réseaux RBF. Afin de conserver le formalisme des RNA, les paramètres modifiables (i.e. centre et pente de la gaussienne) sont codés par les poids de connexions provenant d'un neurone dont la sortie reste fixée à 1.

4.7.8.2 Calcul du degré d'activation des prémisses

Les neurones de la deuxième couche cachée représentent chacun la prémisse d'une règle. Ils reçoivent en entrée le degré de vérité des différents sous-ensembles fous composant cette prémisse et ont en charge le calcul de son propre degré de vérité. Les fonctions d'activation

utilisées pour ces neurones dépendent des opérateurs présents dans les règles (conjonction ou disjonction) et de la logique utilisée :

- *logique de Zadeh :* les opérateurs minimum et maximum ne sont pas dérivables. Ils peuvent cependant être approchés par des fonctions dérivables, appelées *Softmin* et *Softmax*[6].

$$SOFTMIN(x,y) = \frac{x\exp(-kx) + y\exp(-ky)}{\exp(-kx) + \exp(-ky)} \qquad (4.6)$$

$$SOFTMAX(x,y) = \frac{x\exp(kx) + y\exp(ky)}{\exp(kx) + \exp(ky)} \qquad (4.7)$$

Dans ces deux équations, *K* est une constante qui doit être choisie la plus grande possible

- *logique de Lukasiewicz* : l'opérateur de conjonction peut être obtenu à l'aide de l'approximation de la fonction maximum(0; x) par une sigmoïde[6]

$$Maximum(0,x) \approx \frac{1}{1+\exp(\frac{-(x-0.5)}{0.227}} \qquad (4.8)$$

4.7.8.3 Inférence et défuzzification

Ces deux phases sont généralement réalisées par les unités de la ou des dernières couches du réseau. Les fonctions de transfert utilisées dépendent du type de règles choisi :

- *règles de type Mamdani :* les nombreuses méthodes de defuzzification proposées pour les règles de ce type ne peuvent généralement pas être exprimées par une expression dérivable. H. R. Berenji et P. Khedar [20] fournissent une solution particulière dans le cas où les fonctions d'appartenance des sous-ensembles fous apparaissant dans les conclusions sont des fonctions triangulaires. Une première couche de neurones calcule une valeur numérique pour chaque règle en utilisant l'expression suivante :

$$sortie = s + \frac{1}{2}(L_d - L_g)(1 - \mu_{prémisse} \qquad (4.9)$$

où *S* est l'abscisse du sommet de la fonction d'appartenance triangulaire, *Ld* et *Lg* sont respectivement les longueurs des parties du sous-ensemble situées à droite et à gauche de *S,* et $\mu_{premisse}$ est le degré de vérité de la prémisse de la règle.

Cette valeur correspond à la moyenne de la zone pour laquelle le sous-ensemble ou possède une valeur maximum .

Une dernière couche de neurones (la couche de sortie) réalise une somme normalisée des valeurs ainsi obtenues pour les différentes règles. Cette méthode de défuzzification est baptisée moyenne locale des maxima (LMOM).

- *règles de type Sugeno d'ordre 0 :* ces règles sont les plus fréquemment rencontrées dans les applications neuro-flous .Puisque la sortie est uniquement une constante réelle, ilsuffit de coder sa valeur par le poids de la liaison entre le neurone calculant le degré

de vérité de la prémisse et un neurone calculant la valeur de sortie. Ce dernier calcule encore une fois une somme normalisée des valeurs des différentes règles .

- *règles de type Sugeno d'ordre supérieur* : lorsque la sortie des règles utilisées est une fonction des entrées, il suffit de rajouter une couche de neurones calculant cette fonction

- *règles de type Tsukamoto:* ces règles sont un cas particulier des règles de type Mamdani pour lesquelles les sous-ensembles flous utilisés en conclusion utilisent des fonctions d'appartenance monotones. Il suit ici d'utiliser des neurones codant l'inverse de ces fonctions et recevant en entrée le degré de vérité des prémisses pour obtenir une valeur numérique pour chaque règle. Encore une fois les neurones de sortie réalisent une somme normalisée des valeurs des différentes règles. On retrouve notamment ce type de règle dans le système NEFCON de D. Nauck et R. Kruse [10].

4.7.8.4 Apprentissage

Dans la majorité des cas rencontrés la stratégie d'apprentissage repose sur une adaptation de l'algorithme de rétro propagation du gradient. On trouve cependant quelques autres exemples [13] utilisant une méthode de perturbation aléatoire des poids du réseau et ne gardent que les modifications qui améliorent les performances. Si une telle approche ne garantit absolument pas la convergence vers un système optimal, elle possède l'avantage de n'imposer aucune contrainte sur la structure du réseau.

4.8 Les applications des systèmes neuro-flou

Les premières applications de RNA flous à produits du consommateur sont parus sur le marché (Japonais et Coréen) en 1991. Quelques exemples incluent des appareils à conditionner de l'air, électrique, moquettes, ventilateurs électriques, thermo pots électriques, appareils de chauffage du ventilateur du kérosène, fours à micro-ondes, réfrigérateurs, cuisinières du riz, le nettoyeur, machines à laver, en des machines photocopieurs , et traitements de texte.

4.9 Conclusion

Dans ce chapitre, nous avons présenté les différents systèmes neuro-flous et les différent modèles de ces systèmes et aussi les différents type d'apprentissage associés. Quelques uns de ces modèles sont des approximateurs universels, d'autres sont des contrôleurs et d'autres des classificateurs .

L'acquisition de la donnée et traitement qui forment les données est aussi assez importante pour le succès de systèmes neuro-flou . Beaucoup de modèles neuro-flou utilisent des techniques supervisées et non supervisées pour apprendre les paramètres de système d'inférence. Le succès du processus d'apprentissage n'est pas garanti, comme le modèle conçu ne peut pas être optimal.

Dans le chapitre suivant, nous utilisons ces connaissances pour obtenir un module de diagnostic flexible qui utilise le système neuro-flou hybride comme un outils de détection et d'isolation des défauts dans un système de production choisi, et utilise les techniques des reconnaissance de formes floue qui sont compatibles avec les systèmes neuro-flou hybride .

CHAPITRE 5

NEFDIAG :

UNE APPROCHE NEURO-FLOUE PROPOSEE POUR LE DIAGNOSTIC INDUSTRIEL

Dans ce chapitre nous proposons une stratégie pour le suivi du comportement d'un processus et détection de défaillances.

Une approche de diagnostic industriel basé sur la reconnaissance de formes statistique neuro-flou s'appuyant sur une représentation numérique et symbolique des formes est mise en œuvre.

Dans ce cadre, un logiciel informatique de simulation interactive baptisé NEFDIAG (Neuro Fuzzy DIAGnosis) version 1.0 est développé. Ce pro-logiciel écrit sous DEPHI consacré essentiellement à la création, l'apprentissage et au test d'un système neuro-flou de classification des pannes d'un procédé industriel. NEFDIAG peut être représenté comme un type spécial de perceptron flou, à trois couches utilisé pour classifier des formes et des défaillances.

5.1 Introduction

Ce chapitre est exclusivement réservé à la stratégie dédiée au suivi du comportement d'un processus et à la détection des défaillances. Cette stratégie s'appuie sur les données historiques et les données en ligne. Nous abordons les différentes étapes à suivre pour l'élaboration du système de diagnostic à partir de méthodes de classification et de reconnaissance floue de formes. Nous discuterons plus précisément les méthodes floues de la classification supervisée et ferons ensuite une « projection » de ces méthodes au plan d'un système neuro-flou et l'appliquerons à un système industriel.

Notre étude se propose de développer des outils d'aide au diagnostic basés sur des techniques situées à l'intersection des techniques neuronales et de la logique floue. Pour une meilleure exploitation des caractéristiques des systèmes neuro-flous, nous proposons une méthode de diagnostic industriel basée sur la *Reconnaissance des Formes Statistique Neuro-floue (RdFSNF)* qui s'appuie sur une représentation numérique et en même temps symbolique des formes.

Nous avons choisi la méthode AMDEC (Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité) pour représenter toutes les informations sur le système à étudier. AMDEC est une technique déductive et qualitative avec laquelle les effets (conséquences) des défaillances des composantes élémentaire sont systématiquement identifiés.

5.2 Diagnostic par reconnaissance des formes statistique neuro-floue

Le fonctionnement d'un système de diagnostic par RdFSNF se déroule en trois phases : une phase d'analyse, une phase de choix d'une méthode de décision floue (inférence floue) et une phase d'exploitation (figure 5.1).

Le diagnostic de défaillances dans ce cadre est essentiellement vu comme un problème de *classification*. Le but principal est de construire un bloc de correspondance tel qu'à partir d'un ensemble d'informations décrivant la situation courante de processus, il est possible d'obtenir les causes probables de situations anormales.

Quand le diagnostic est basé sur des observations multiples, elles sont regroupées pour former des classes qui définissent une situation ou un mode de fonctionnement du processus, auquel une nouvelle observation sera comparée pour être identifiée. En d'autres termes, le diagnostic a pour mission d'identifier le mode de fonctionnement d'un système à partir d'observations sur celui-ci. La figure 5.1 illustre ses étapes et le diagnostic par RdFSNF.

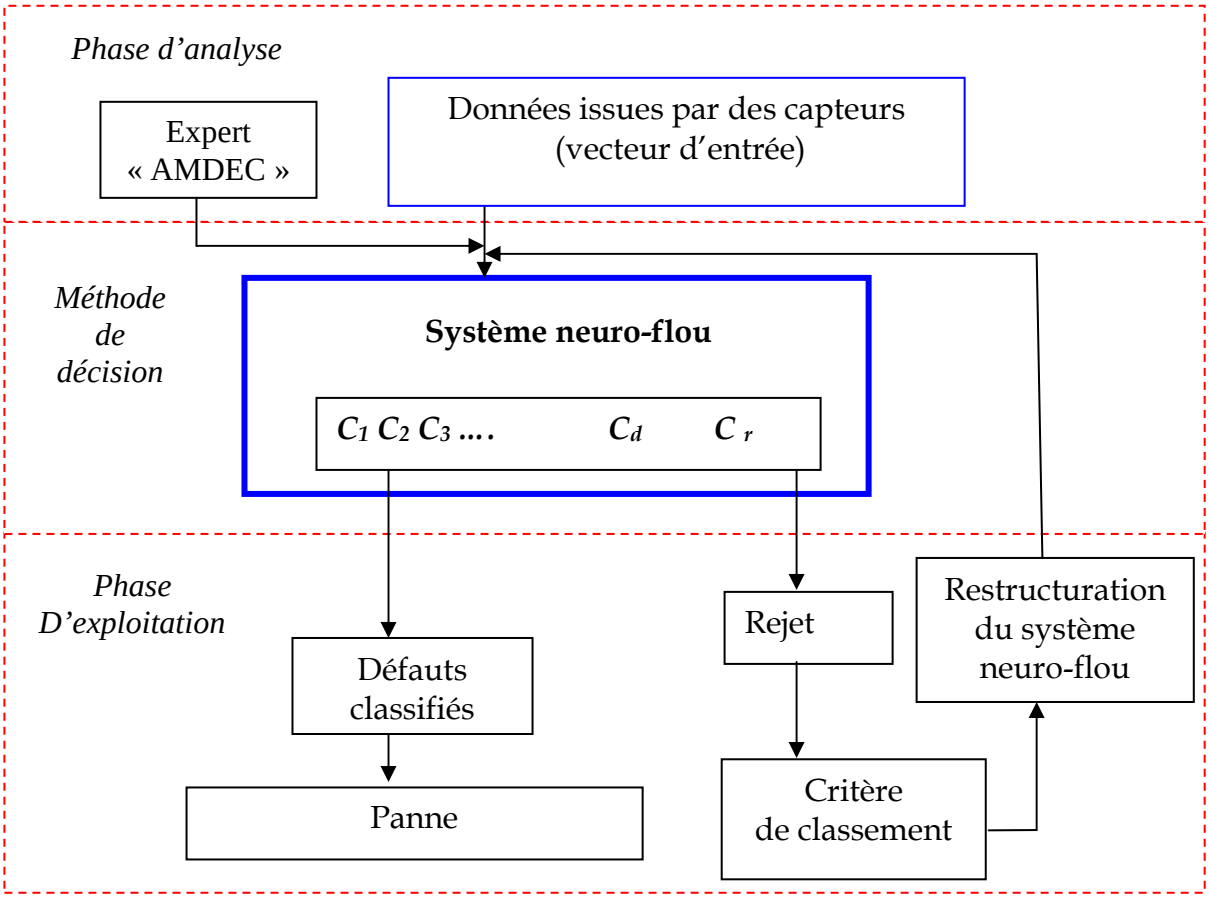


Figure 5.1. *Le diagnostic par RDFSNF.*

L'objectif de ce système RdFSNF est de savoir associer toute nouvelle observation X_i à une classe de l'espace de décision. L'affectation d'une observation X_i à une des K classes notées $C_1,\ldots C_K$ indique une opération de classement ou de discrimination.

Sur la figure 5.2, on retrouve un petit classificateur simple à l'aide d'un perceptron simple qui nous donne trois zones de classe « des régions » où x_1, x_2 sont les variables de forme à classer et R_1, R_2, R_3 sont des régions ou « cluster ».

Les paramètres caractérisant le vecteur d'observation, et par conséquent l'espace de représentation, représente les mesures disponibles issues des capteurs et actionneurs du processus ou bien des informations extraites de ceux-ci. Les classes peuvent être assimilées aux différents modes de fonctionnement du processus ou modes de défaillances.

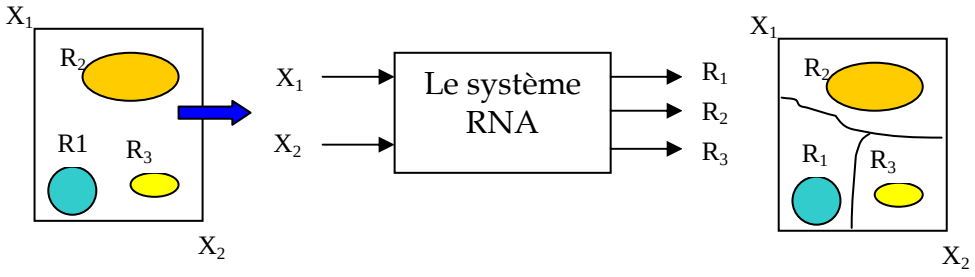


Figure 5.2. *Méthode de classification par le système neuro-flou.*

5.2.1 Phase d'analyse

Pour surveiller un système, il est nécessaire de disposer d'informations provenant de celui-ci, informations généralement délivrées par des capteurs. Il s'agit donc de définir à partir de données, prétraitées ou non, un nombre ***D*** de caractéristiques ou « d'observations ». Il n'existe pas de règles, il faut trouver un compromis entre la dimension ***D*** du vecteur et la pertinence des caractéristiques. Très souvent, ce choix est guidé par l'expertise sur le procédé.

Dans cette étape nous utilisons, l'AMDEC, comme méthode de représentation de toutes les informations. Elle permet une analyse systématique et très complète, composant par composant, de tous les modes de défaillance possibles et précise leurs effets sur le système global.

L'utilisation des tableaux d' AMDEC à des fins de diagnostic industriel conduit à utiliser une procédure déductive, c'est-à-dire à utiliser ces tableaux comme outil d'identification des causes de défaillances à partir des effets observés

La démarche consiste d'abord à définir le système, ses fonctions et ses composants. Ensuite, l'ensemble des modes à défaillances des composants doit être établi. Pour chaque mode de défaillance, sont recherchés ensuite les causes possibles de son apparition. Finalement, une étude des effets sur le système est faite pour chaque combinaison (cause, mode de défaillance). La criticité permet d'extraire les modes de défaillance les plus critiques.

5.2.2 Phase de choix d'une méthode de décision

Une phase de choix du système de défaillance qui consiste à construire une règle de décision qui établira des frontières entre les différentes classes. La règle de décision permettra d'affecter ou non une nouvelle observation à l'une des classes connues.

Les systèmes neuro-flous ont été choisis comme méthode de décision. Dans ce cadre, un pro logiciel baptisé NEFDIAG (NEuro-Fuzzy DIAGnosis) a été développé La section suivante explique les détails de ce choix et les étapes de ce développement.

5.2.2.1 Classificateur neuro-flou NEFDIAG

5.2.2.1.1 Le perceptron flou

L'architecture du perceptron flou est identique a celle de perceptron Multi-Couches usuel, mais les poids sont modelés par des ensembles flous. Les activations, les sorties, et les fonctions de propagation seront changées aussi. L'intention de ce modèle est qu'il est

interprétable par des règles linguistiques et peut utiliser des bases de connaissance, des règles à priori, donc l'apprentissage peut ne pas démarrer à zéro (la base de règles n'est pas vide).

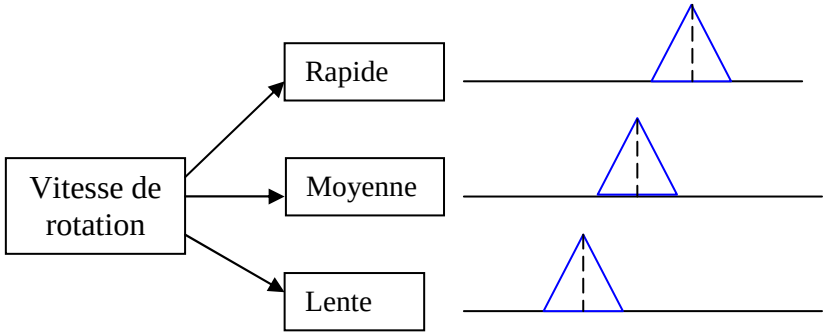


Figure 5.3. *Un terme linguistique avec trois fonctions d'appartenance*

La figure 5.4 illustre un perceptron flou présenté par NEFDIAG avec **N** entrées, **M** règles floues et **P** sorties. Les neurones de la première couche réalisent la première phase de l'inférence floue .encore appelée « fuzzification ».

A chaque observation correspond au moins deux neurones de la première couche d'entrée (figure 5.3), chargés de calculer les degrés d'appartenance des variables floues aux différents sous ensembles de termes linguistiques. La fonction d'appartenance utilisée est la fonction triangulaire symétrique.

Les neurones de la deuxième couche calculent le degré de vérité des antécédents des règles floues par l'intermédiaire de t-norme. Le nombre de neurones de cette couche est égal à la taille de base de règles. La connexion entre la première couche et la deuxième n'est pas totale car définie par la structure de la règle linguistique. Les valeurs de sortie de la troisième couche sont le maximum des valeurs d'activation de toutes les unités de règles qui sont associés à une classe prédéfinie.

5.2.2.1.2 Représentation de NEFDIAG

Un logiciel informatique de simulation interactive baptisé NEFDIAG (Neuro Fuzzy DIAGnosis) version 0.0 est développé au sein de LAP (Batna.). Ce logiciel écrit sous DEPHI consacré essentiellement au développement, à l'apprentissage et au test d'un système neuro-flou de classification des pannes d'un procédé industriel.

NEFDIAG peut être représenté comme un type spécial de perceptron flou, à trois couches utilisé pour classifier des défaillances.

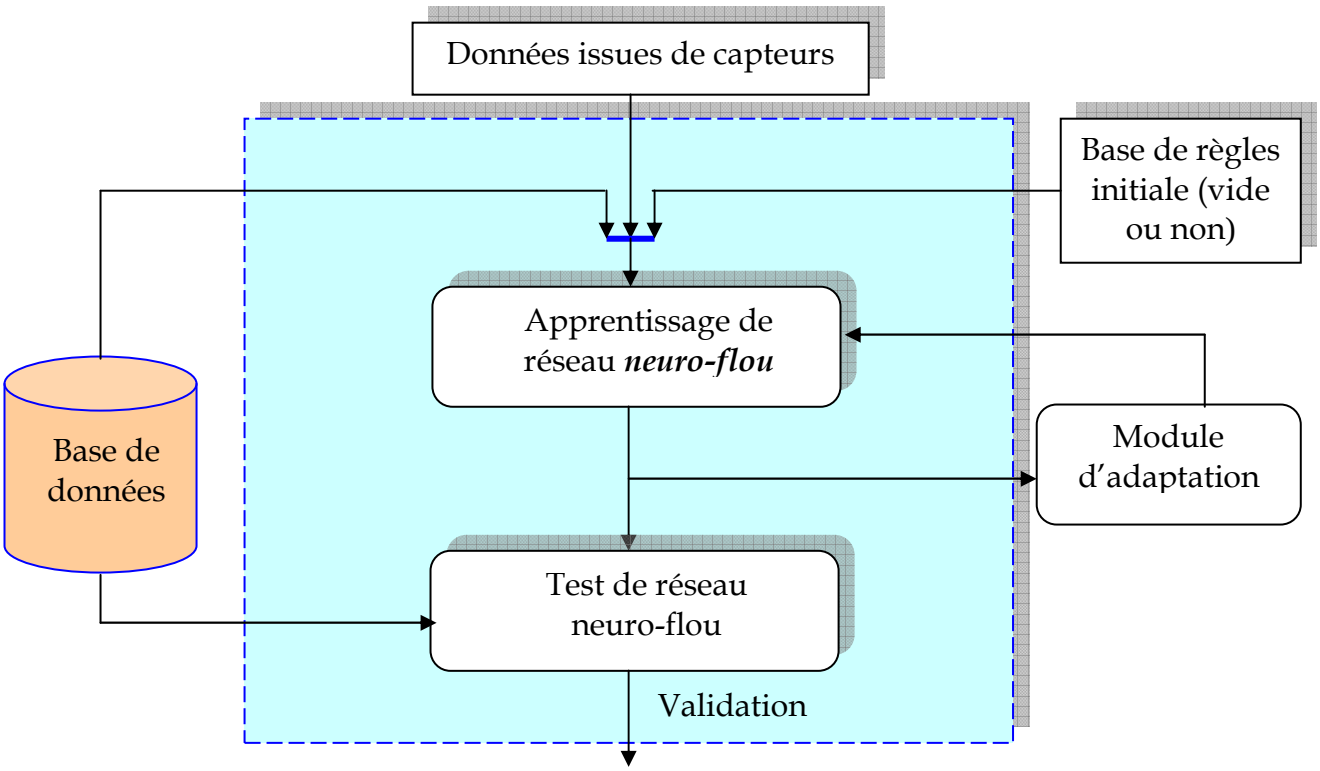


Figure 5.4. *Architecture générale du système NEFDIAG.*

NEFDIAG fait son apprentissage par un ensemble de formes, telle que chaque forme soit affectée (classée) vers une des classes prédéfinies (figure 5.4). NEFDIAG génère les règles floues par un parcours de données et optimise ensuite les règles par apprentissage des paramètres des sous ensembles flous qui sont utilisés pour partitionner les données « caractéristiques » des formes à classer et les paramètres des données. NEFDIAG représente un classificateur flou F_r avec un ensemble de classes $C=\{c_1,c_2\ldots\ldots c_m\}$.

Apres l'apprentissage, NEFDIAG peut être utilisé pour classifier une nouvelle observation, le système peut être représenté sous forme de règles floues :

Si symptôme$_1$ **est** A_1
symptôme$_2$ **est** A_2
symptôme$_3$ **est** A_3
symptôme$_n$ **est** A_n
Alors la forme $(x_1, x_2, x_3,\ldots, x_n)$ est à la classe ***mode de defaillance1.***

Telque A_1, A_2, A_3, A_n sont des termes linguistiques représentés par des ensembles flous. Cette caractéristique nous permet de connaître des analyses sur nos données, et utilise ces connaissances pour les classifier.

La figure 5.5 illustre un système neuro-flou représenté par NEFDIAG avec ***N*** entrées, ***M*** règles floues et ***P*** sorties tel que les entrées représentent les observations de chaque composant (sous système) du système à surveiller, donc pour chaque composant, il y une ou plusieurs observations.

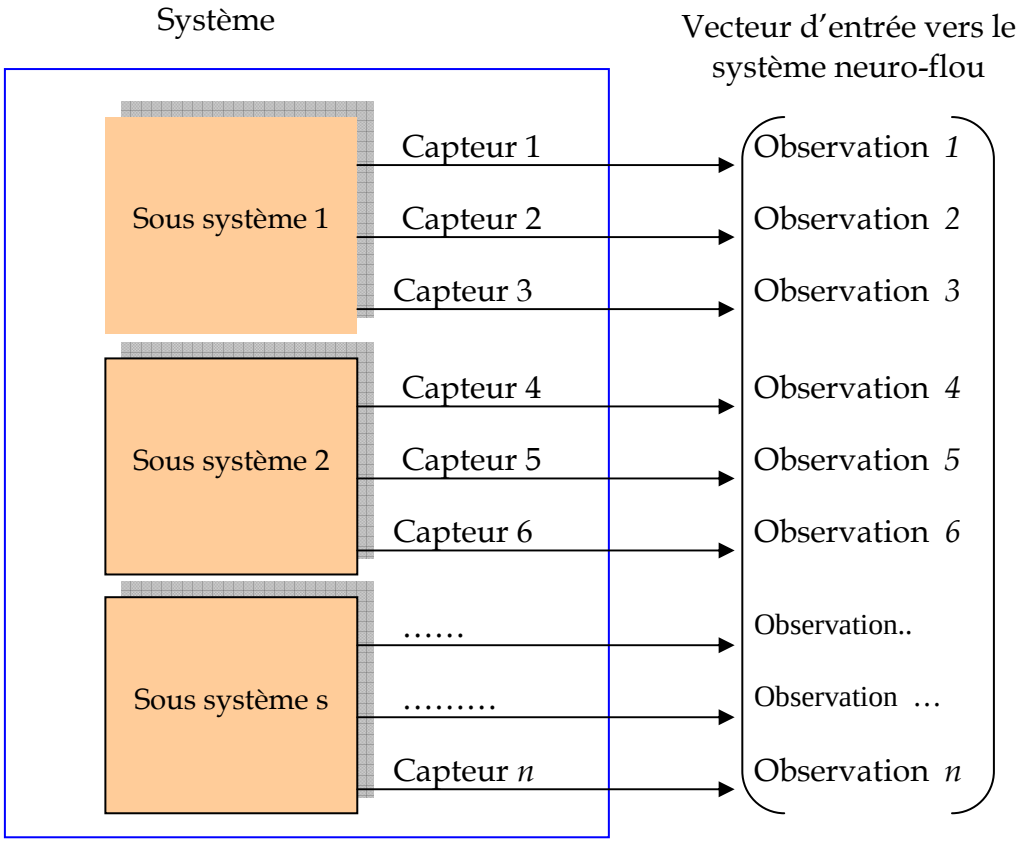


Figure 5.5. *Données issues par des capteurs (vecteur d'entrée)*

La tâche du système NEFDIAG est de découvrir « créer les règles par les ensemble de formes d'apprentissage » les règles et l'apprentissage des fonctions d'appartenance pour déterminer la mode de défaillance exact pour une observation quelconque.

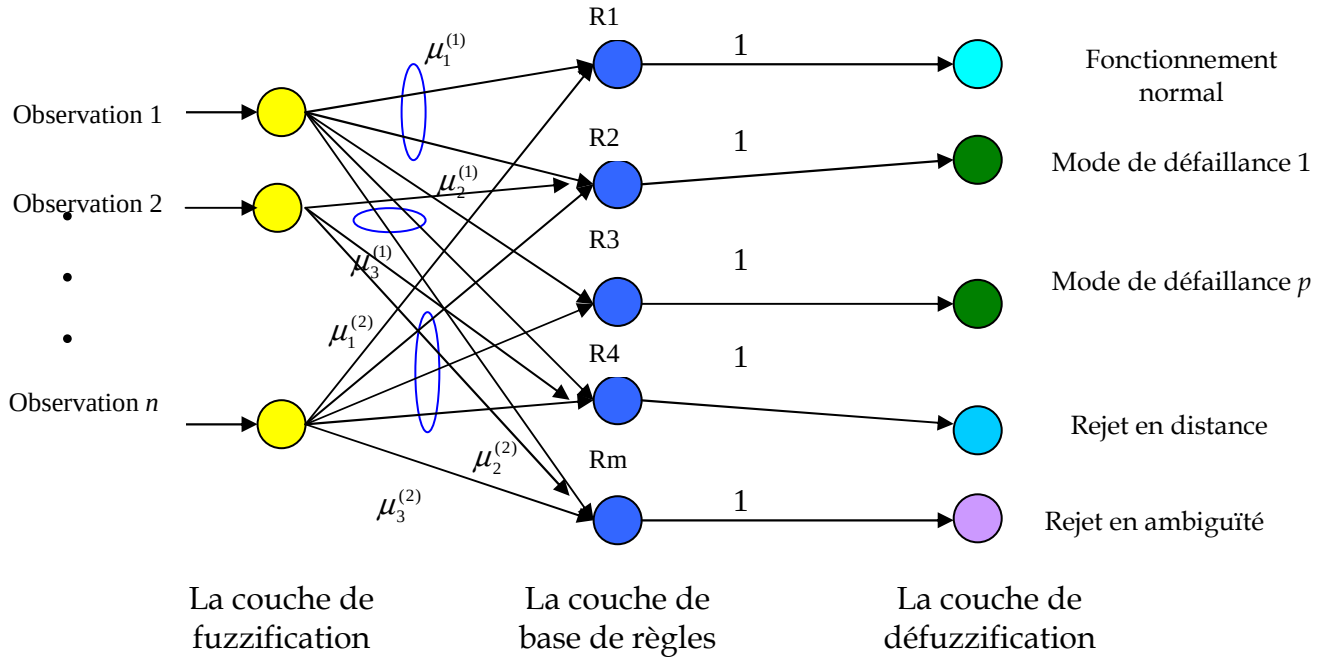


Figure5.6. *Structure interne du système NEFDIAG*

5.2.2.1.3 L'apprentissage

L'apprentissage des systèmes de neuro-flous artificiels est une phase qui permet de déterminer ou de modifier les paramètres du système, afin d'adopter un comportement désiré.

L'étape d'apprentissage est basée sur la descente de gradient d'erreur quadratique moyenne commise par le RNF. Après avoir calculé l'activation en avant des neurones des différentes couches, l'erreur est ensuite rétro propagée dans le sens inverse de l'activation pour pouvoir calculer, pour chaque neurone, sa contribution (5.1).

$$J_m(U,V) = \sum_{k=1}^{N}\sum_{i=1}^{m}(u_{ik})^p (d_{ik})^2 \qquad (5.1)$$

Des corrections sont ensuite apportées aux différents paramètres du système (les poids, les seuils et les paramètres des fonctions d'appartenance).

L'utilisateur définit le nombre initial des fonctions d'appartenance pour partitionner les domaines des données d'entrées et spécifie le nombre *k*, nombre maximum de neurones des règles qui seront créés dans la couche cachée.

Les principales étapes d'apprentissage sont ainsi présentées (figure 5.7) :

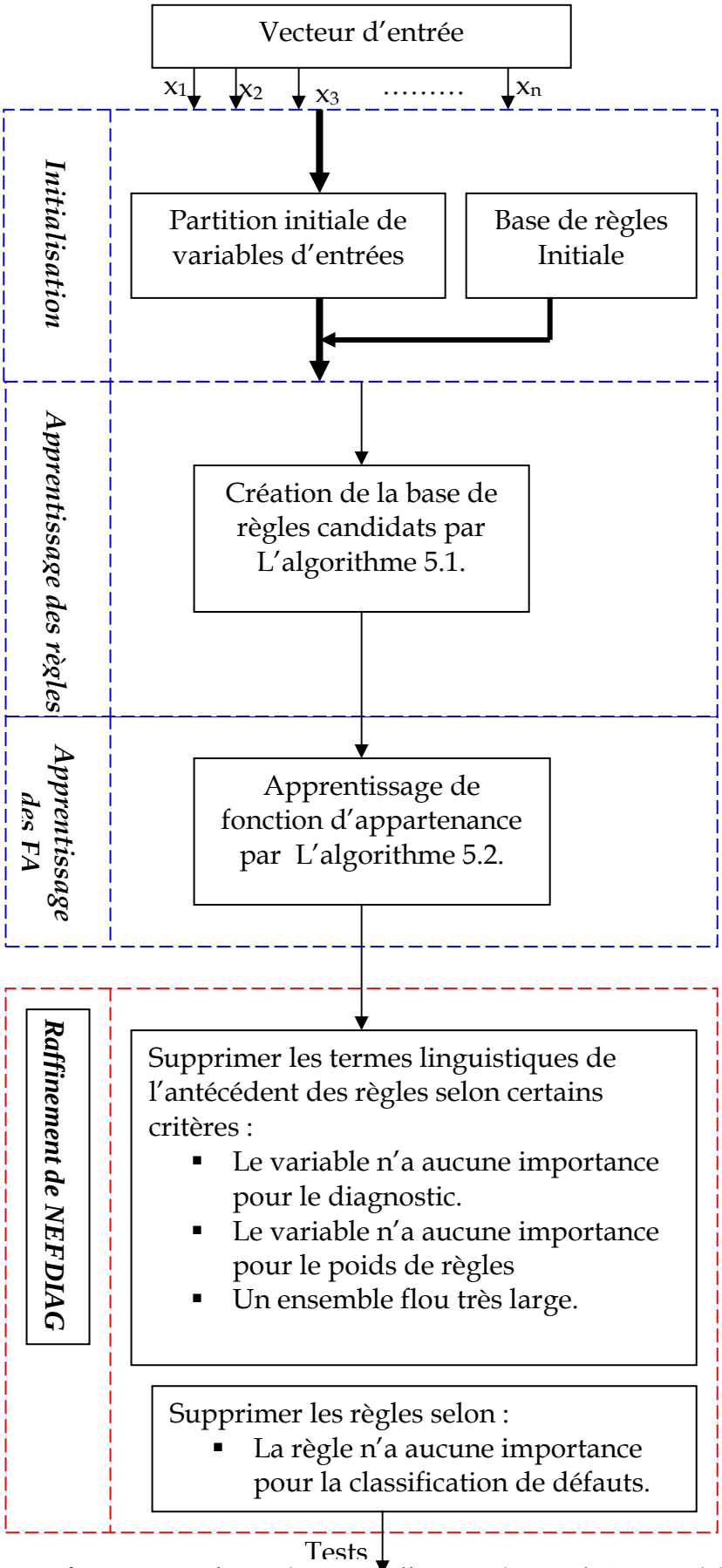


Figure 5.7. *L'organigramme d'apprentissage de NEFDIAG.*

- ***initialisation :*** pour chaque donnée issue des capteurs, il y a une unité d'entrée, et pour chaque mode de défaillance il y a une unité de sortie. Pour chaque unité d'entrée une partition floue initiale est spécifiée (exp. nombre des fonctions d'appartenance triangulaire).

- ***Apprentissage des règles :*** Le système NEFDIAG peut démarrer avec une base de connaissance partielle des formes, et raffiner durant l'apprentissage. La règle sera créée par la recherche (pour une forme donnée *f*) la combinaison des fonctions d'appartenance telle que chaque entrée produit la plus grande fonction d'appartenance. Si cette combinaison n'est pas identique pour les règles existantes dans la base des règles et le nombre de règles n'est pas maximum, alors une règle sera créée et ajoutée à la base des règles.

- ***Apprentissage des FA :*** pour l'apprentissage des fonctions d'appartenance, une simple rétro propagation sera utilisée. Une décision sera créée qui dépend de l'erreur de sortie pour chaque unité de règles. Chaque règle change ses fonctions d'appartenance par le changement de leurs supports.

Les algorithmes présentés dans cette section sont des extensions d'une approche proposée par [53]. Ils sont utilisés dans des logiciels « NEFCLASS, NEFCON, NEFPROX. ».

L'algorithme d'apprentissage des règles de l'approche proposée utilise les poids des règles et détermine les meilleurs règles par la « mesure de la performance » ou par une valeur qui indique l'efficacité d'une règle quelconque. En plus l'algorithme essaie de faire une réduction de la taille de la base de règles, par la sélection des règles en utilisant leurs performances en classification des modes de défaillances, ou par le déroulement des données d'apprentissage.

a. Apprentissage des règles

D'abord toutes les variables du système, (données issues par des capteurs), à diagnostiquer sont partitionnés par des fonctions d'appartenance avant que l'algorithme d'apprentissage ne démarre.

Si l'utilisateur ou l'expert du système détermine ces fonctions d'appartenance et les nomment lui-même, alors ces noms sont utilisés en tant que « vocabulaire » pour la description du problème qui est représenté par la base de règles. Les règles sont créées par le déroulement des données d'apprentissage. Au début, la base de règles est vide, ou contient quelques règles « définie par l'utilisateur ou l'expert de système » comme une connaissance à priori.

Le nombre maximum des règles sera défini par l'utilisateur, si notre système a trouvé des règles> k_{max} règles, alors il raffine la base des règles (candidats).

Pour le premier cycle, tous les antécédents sont crées (Annexe 2), pour chaque point des ensembles de données utilisées pour la création de règles, une combinaison des ensembles flous est sélectionnée. Pour chaque variable, sa fonction d'appartenance, qui a le maximum degré de ces FA pour la valeur d'entrée courante. Si l'antécédent combiné de ces ensembles flous n'existe pas dans la liste antécédents, additionner-le dans la liste des antécédents.

Durant la deuxième cycle, le meilleur conséquent pour chaque antécédent est sélectionné et les règles sont complétées. Le maximum de nombre de règles candidates est

$$\min \quad \left\{ s, \prod_{i}^{n} q_i \right\}$$

Telque ***s*** est la cardinalité d'ensemble des données d'apprentissage, et ***qi*** est le nombre des ensembles flous d'une variable d'entrée *xi*.

L'exemple de la figure 5.8 illustre trois classes de la donnée représentée par les règles suivantes :

Si *x est petit* ***alors*** *y est grand*
Si *x est moyen* ***alors*** *y est petit*
Si *x est grand* ***alors*** *y est moyen*

Nous considérons maintenant NEFDIAG (figure 5.6) comme **N** unités d'entrée x_1, $x2,\ldots\ldots, x_n$.

K<= k_{max} unités des règles $R_1, R_2,\ldots..R_k$

M unités de sorties (modes de défaillance).

La tâche d'apprentissage Γ est ainsi définie : $\Gamma = \{(f_1, t_1),\ldots...(f_s,t_s)\}$ de ***s*** formes.

Elle consiste en une forme d'entrée $f \in \Re^n$ et une forme cible $t \in \{0,1\}^m$

L'algorithme d'apprentissage suivant [53] est utilisé pour créer ***k*** unités de règles de NEFDIAG.

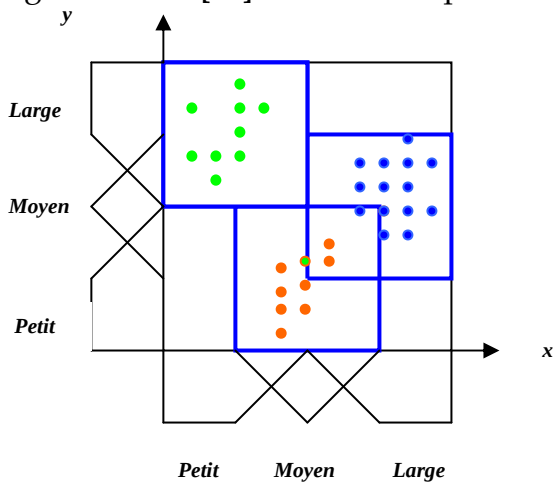


Figure 5.8. *Trois FA pour chaque variable d'entrée*

Algorithme: *Apprentissage des règles floues*

début

- ***(1)*** sélectionner la forme suivante (f,t) de Γ.
- ***(2)*** pour chaque unité d'entrée $x_i \in U_1$ chercher la fonction d'appartenance $\mu_{ii}^{(i)}$

$$\mu_{ji}^{(i)}(p_i) = \max_{j \in \{1,\ldots,qi\}} \{\mu_j^{(i)}(x_i)\}$$

- ***(3)*** si le nombre des règles <= k_{max}, et il n'existe pas de règle R avec :
  W(x1,r)= $\mu_{i1}^{(1)}$,……..,W(x_n,R)= $\mu_{in}^{(n)}$
  Alors créer un nœud et connecter la au nœud de sortie CI si ti=1.
- ***(4)*** s'il y a des formes non traitées dans Γ et k<= k_{max} alors aller a ***(1).*** Sinon

Fin.

Algorithme 5.1. *Apprentissage des règles floues.*

L'algorithme d'apprentissage détecte (calcule) tous les antécédents des règles et crée ensuite la liste des antécédents. Dans un premier temps, cette liste est vide, ou contient des antécédents de règles de connaissances à priori.

L'algorithme sélectionne un conséquent pour chercher l'antécédent ***A*** et crée la liste de base de règles candidates.

Pour chaque antécédent, une mesure de performances des règles p ϵ [-1 ,1] est calculée ;

Pour *p=1*, la règle est générale et classifie correctement.

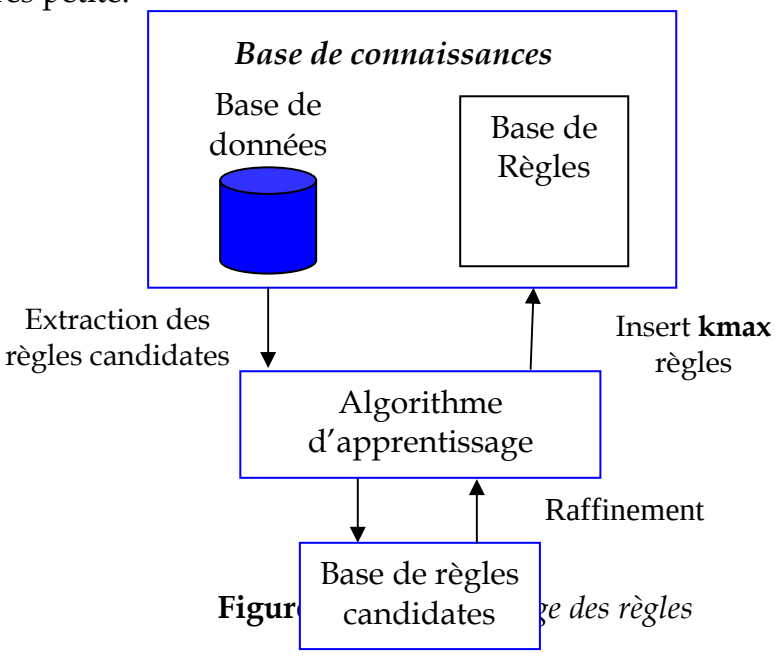
Pour *p=-1*, la règle classifie toutes les formes incorrectement.

Pour P=0 quelques formes sont classées correctement et autres non.

Seulement les règles avec p>0 sont considérées pour l'utilisation durant toute la deuxième phase d'apprentissage.

La figure 5.9 illustre l'algorithme d'apprentissage, et l'extraction de règles en utilisant les observations sur le système qui sont stockées dans une base de donnée.

Les meilleurs règles sont sélectionnées dans la base des règles candidates, en base de mesure de performance, dans ce cas quelques classes (modes de défaillance) ne seront pas représentés dans la base de règles, si les règles pour ce mode de défaillance à une valeur de performance très petite.



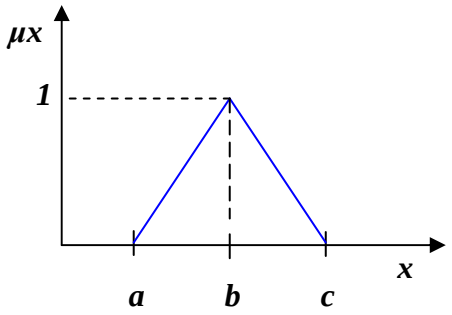
Figur *ge des règles*

b. apprentissage des fonctions d'appartenance

Quelques approches d'apprentissage des fonctions d'appartenance basées sur l'implémentation de rétro propagation du gradient ont été proposées [55].

Pour utiliser la technique de descente de gradient; il faut que la fonction d'appartenance et toutes les fonctions qui seront utilisées pour l'évaluation des règles floues seront différentiables. Aussi nous pouvons utiliser les fonctions d'appartenance triangulaire ou trapézoïdal (figure 5.10).

$$\mu_{a,b,c} : R \to [0,1], \quad \begin{cases} \dfrac{x-a}{b-a} & si \ \ x \in [a,b[\\ \dfrac{c-x}{c-a} & si \ \ x \in [b,c] \\ 0 & si \ \ non \end{cases} \qquad (5.1)$$



$$\mu_{a,b,c,d} : R \to [0,1], \quad \begin{cases} \dfrac{x-a}{b-a} & si \ \ x \in [a,b[\\ 1 & si \ \ x \in [b,c] \\ \dfrac{d-x}{d-c} & si \ \ x \in [b,c] \\ 0 & sin \ \ on \end{cases} \qquad (5.2)$$

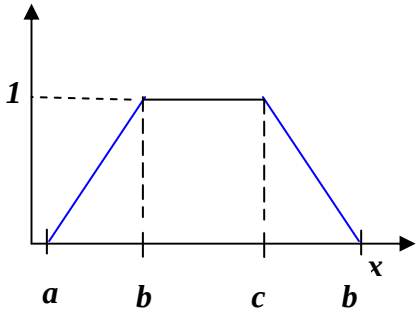


Figure 5.10. *FA triangulaire, trapézoïdal, et leur domaine de définition.*

[55] a proposé une procédure d'apprentissage pour un système neuro-flou simplifié du *sugeno* qui utilise seulement des constantes dans les conséquents des règles. Rappelons qu'il existe quatre formes des règles floues dans un système neuro-flou.

La mise à jour des paramètres d'apprentissage d'une procédure d'apprentissage de l'algorithme de descente de gradient peut être calculé par l'application de l'algorithme (Algorithme 5.2).

Nomura et al [56] modifient les ensembles flous pour chaque règle, indépendamment de toutes les autres règles. L'algorithme qui fait la mise à jour des fonctions d'appartenance dans les antécédents et les conséquences des règles utilise une simple heuristique. ***(Annexe 2).***

Algorithme : *Apprentissage des ensembles flous.*

Début

1- *sélectionner la forme suivante (p,t) de* Γ,propager la,par NEFDIAG et *détermine le vecteur de sortie C.*

2- *pour chaque unité de sortie* C_i *; déterminer la valeur de delta*

$\delta C_i = ti - Oc_i$

3- *pour chaque unité de règle R avec* $O_R > 0$

a- *déterminer la valeur de δ*

$$\delta_R = O_R(1 - O_R) \sum_{C \in U_3} W(R,C)\delta_C \ .$$

b- *trouver x' tel que :* $w(x', R)(o_{x'}) = \min_{x \in U_1} \{w(x, R)(\delta_C\}$

c -*pour l'ensemble flou w(x',R) déterminer les valeurs de δ pour les paramètres a,b,c utilisant le pas d'apprentissage σ>0.*

$\delta_b = \sigma . \delta_R .(c-a) sgn(O_{x'} - b)$,

$\delta a = -\sigma \delta_R ..(c-a) + \delta_b$,

$\delta c = \sigma \delta_R ..(c-a) + \delta_b$.

Appliquer le changement aux w (x', R).

4 –si *l'itération est terminée, ou les critères de fin sont validé ;* ***alors*** *stop*

Sinon *aller a* ***(1).***

Fin

Algorithme 5.2. *Apprentissage de fonctions d'appartenance*

La figure 5.11.a illustre la modification des paramètres de fonctions d'appartenance après la création de règles *(a)*, et après la compatibilité des ensembles floues *(b)*. L'adaptation des ensembles flous est fait par un simple changement de paramètres des fonctions d'appartenance tel que le degré d'appartenance pour la valeur de la variable courant est augmenté ou diminué. Figure 5.11.b (L'ensemble flou au centre : situation initiale, l'ensemble flou à droite : situation d'augmentation, l'ensemble flou à gauche : situation de diminution).

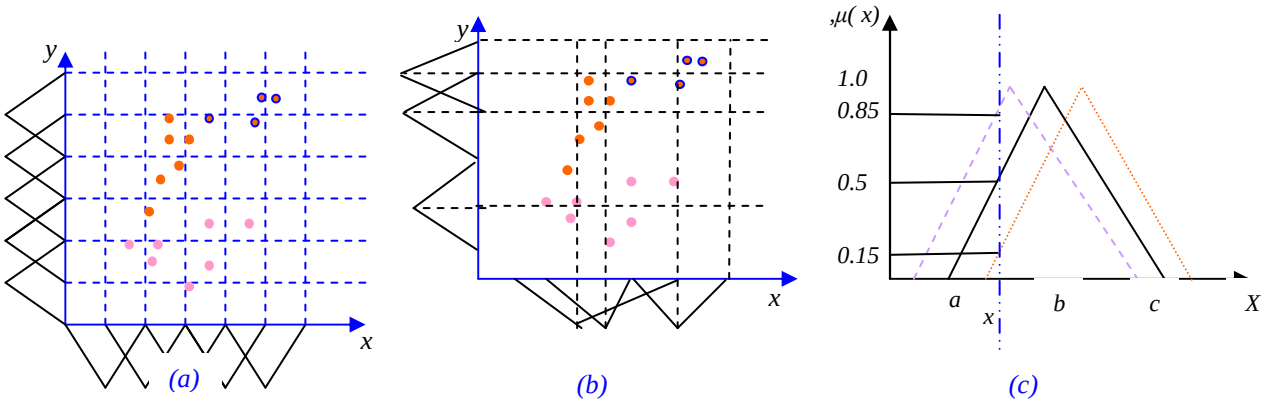


Figure5.11.a: *Modification des paramètres FA*

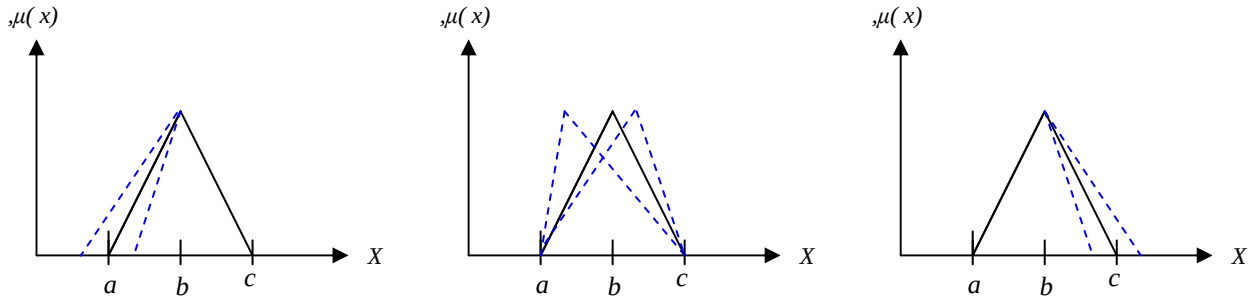


Figure 5.11.b : *Modification des paramètres de FA*

5.2.2.1.4 Aspects d'implémentation

L'un des plus importants aspects pour l'implémentation d'un système neuro-flou «dans notre cas NEFDIAG» pour la génération d'un système flou interprétable est capable de faire des apprentissages et que pour chaque terme linguistique est représenté par exactement un ensemble flou. Cela à conduit, par exemple à l'utilisation d'un langage de programmation orientée objet et spécifie les objets pour, les ensembles flous, les partitions floues, et les règles floues.

Chaque objet a des enregistrements (Record) pour stoker les données et des méthodes pour traiter les données et pour diagnostiquer le système.

Une liste des objets de partitions floues sera créée, un objet pour chaque variable. Chaque partition floue contient une liste de q_i objets des ensembles flous.

Un objet des règles floues contient une liste de référence «pointeurs» vers les ensembles flous à objectifs pour construire leurs antécédents.

Par l'utilisation des pointeurs «*listes binaire chaînées*», nous sommes sûr que les expressions linguistiques qui sont dans des différentes règles utilisent la même fonction d'appartenance (figure 5.12).

Mais dans la base de règle, nous avons utilisé une base de données qui implémente cette base de règles, et en mode d'exécution nous avons implémenté cette base de règles par une liste d'objets de la base de règles.

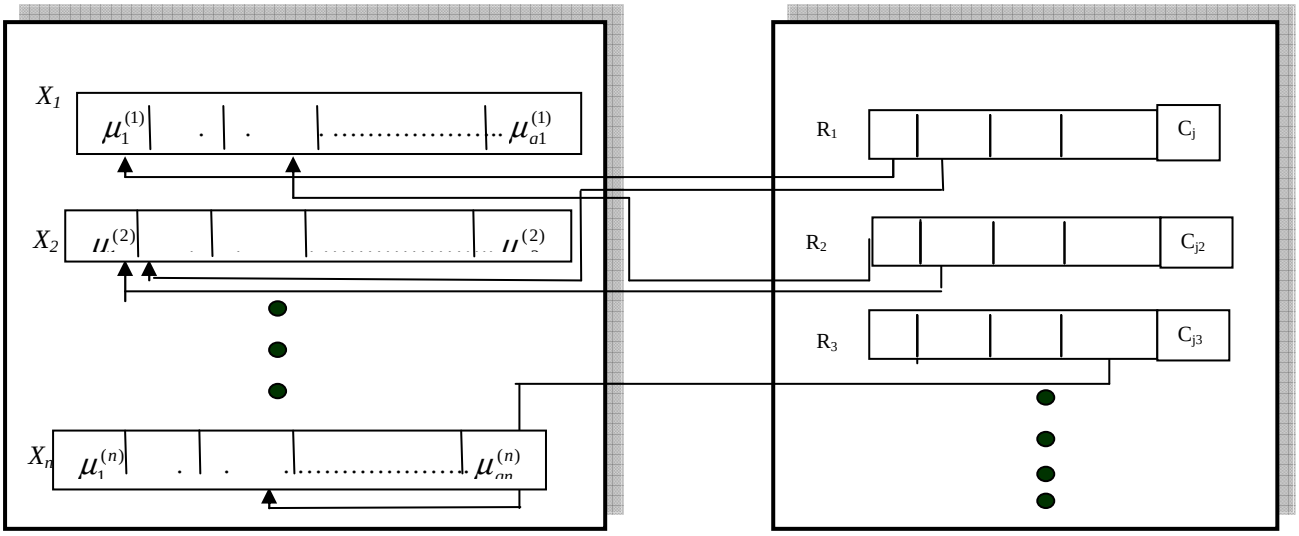


Figure 5.12. *Implémentation de la base de règles dans NEFDIAG*

Les règles ***R1*** et ***R2*** utilisent des expressions linguistiques « ***x2*** est $\mu_1^{(2)}$», et l'ensemble flou $\mu_1^{(2)}$ est partagé par plusieurs règles floues. Durant toute la phase d'apprentissage ***R1*** et ***R2*** utilisent et/ou modifient $\mu_1^{(2)}$ les règles utilisant la même version de $\mu_1^{(2)}$ pour calculer les paramètres du NEFDIAG qui utilisent cet ensemble flou. Les classes d'objets qui sont utilisées dans NEFDIAG sont visualisées sur la figure 5.12.

Class ***ensmble_floue***	Class ***regle_floue***	Calss ***partition_floue***
Champs : *-paramètres* *-M-a-jours* *-appartenance*	Champs : *-pointeur vers FA* *-erreur* -performance.	Champs : *Liste_des_FA*
Methodes : *-M-à-J* -apprentissage -graphe.	Methodes : *-erreur* *-M-à-j* *-performance*	Methodes : *-appartenace.* *-M-à-j* *-graphe.*

Figure 5.13. *Les principaux objets de NEFDIAG*

5.2.2.2 Notion de rejet

Le rejet en ambiguïté, introduit par Chow[39],affecte une observation, se situant approximativement à égale distance entre plusieurs classes, dans une nouvelle classe appelée classe de rejet en ambiguïté W_0. Cette classe rassemble les observations correspondantes au cas d'indécision.

Lorsqu'une observation à classer se trouve dans une zone de l'espace éloigné de toute classe, elle n'est représentée par aucune classe connue. Pour cela, Dubuisson [39] introduit

une règle de rejet en distance. Les points situés loin des classes connues sont affectés dans une classe appelée la classe de rejet en distance W_d. La règle de décision incluant à la fois le rejet en ambiguïté et le rejet en distance est donnée par l'expression suivante [39]:

$$x \text{ est classée dans} \begin{cases} w_d \ \ si \ \ f(x) < C_d \\ w_j \ \ si \ f(x) \geq C_d \ \ et \ p(w_j|x) = \max_{i=1..c} \ p(w_m|x) \geq 1 - C_r \\ w_o si \ \ f(x) \geq C_d et \max_{i=1..c} \ p(w_i|x) < 1 - C_r \end{cases}$$

Où C_r désigne le coût de rejet en ambiguïté et *Cd* le seuil de rejet en distance.

La notion de rejet est la base de tout procédure de décision adaptative. Les rejets en ambiguïté peuvent amener à fusionner deux classes et le rejet en distance peut engendrer la création de nouvelles classes sous le contrôle de module d'adaptation. La figure 5.13 montre la structure d'un module d'adaptation de NEUFDIAG.

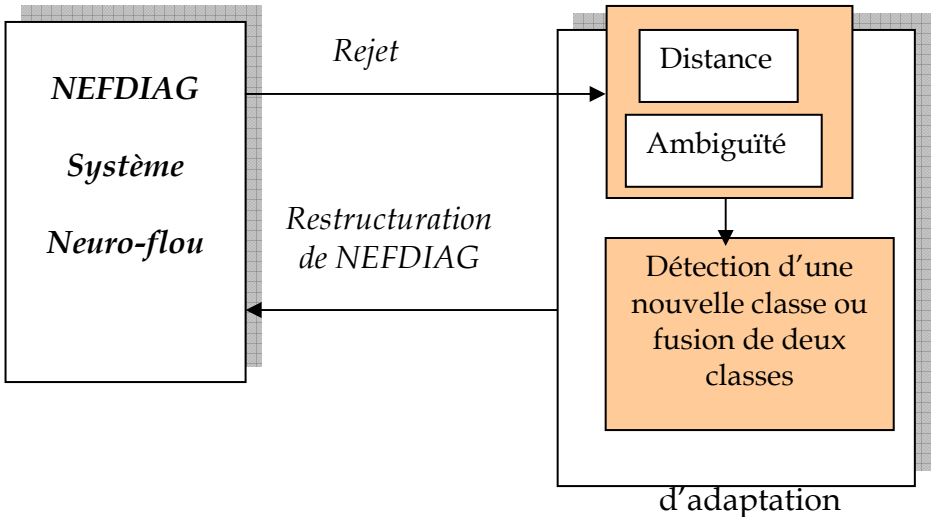


Figure 5.14. *Module d'adaptation*

Apres l'apparition d'un autre nouveau mode de défaillance dans la phase d'apprentissage, notre système neuro-flou va faire une adaptation ou une restructuration du système pour s'adapter à la nouvelle situation.

D'abord la couche des règles (ou la base de règles) ajoute toutes les règles du mode de défaillance détecté. Dans la couche des modes de défaillance, un autre nœud est connecté au système neuro-flou.

La figure 5.14 illustre une restructuration de notre système après l'apparition d'un autre mode de défaillance.

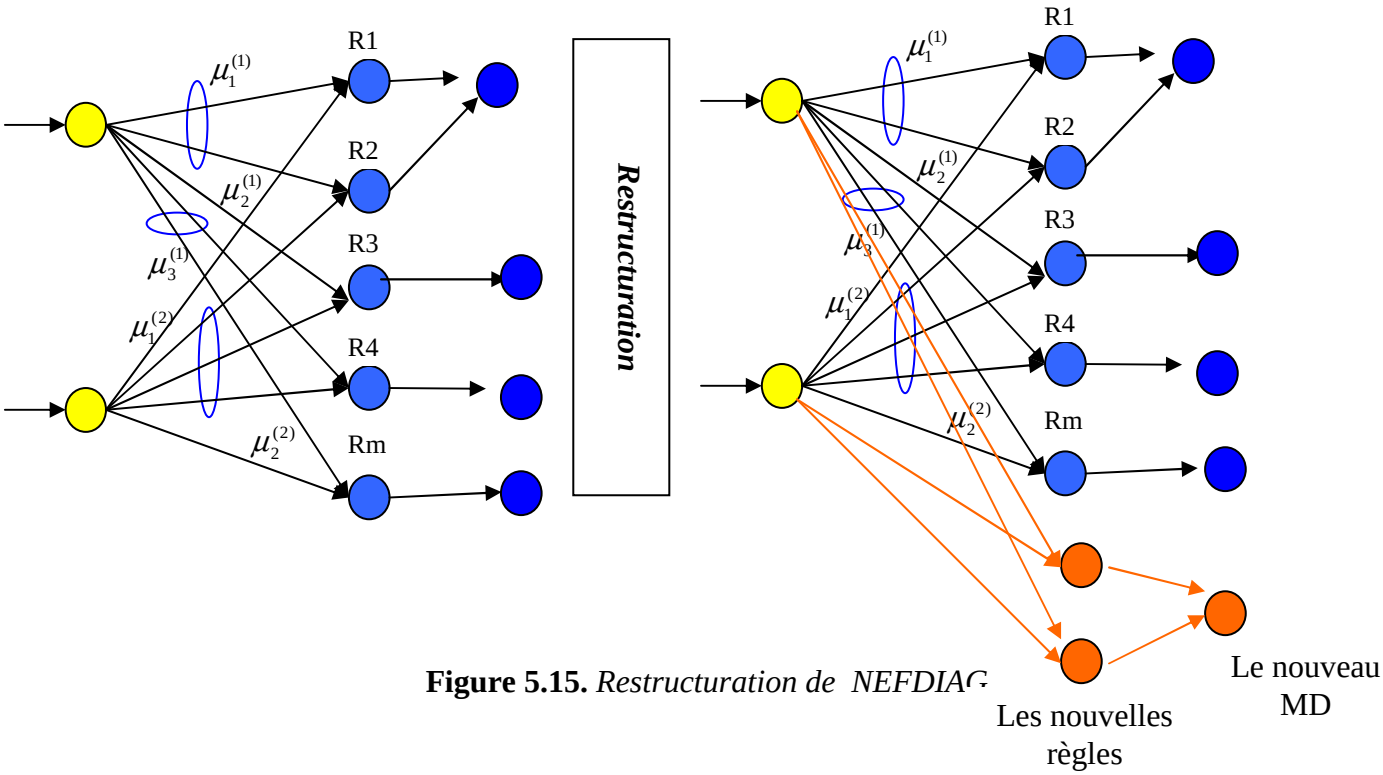


Figure 5.15. *Restructuration de NEFDIAG*

5.3 Application industrielle du NEFDIAG

5.3.1. Introduction

Notre application est illustrée sur un procédé industriel de fabrication du ciment. Cette installation fait partie de cimenterie de Ain-Touta (SCIMAT). En se basant sur l'étude réalisée par [35], [56], [15] sur l'atelier de cuisson, nous avons élaboré une AMDEC en ne considérant que les modes de défaillances les plus critiques (criticité>10) et ceci pour des raisons de simplicité

5.3.2. Brève présentation de l'entreprise

Nous allons exploiter la caractéristique au cours de développement de notre application sur le processus de clinkerisation objet de cette section.

Cette cimenterie d'une capacité de 2.500.000 t/an (2 fours) est composée de plusieurs unités qui déterminent la différente phase du processus de fabrication du ciment. La carrière d'où sont extraites les matières premières (argile et calcaire), les stations de concassage destinées à réduire la taille des blocs de matières (calcaire et argile 1 000 t/h, les ajouts 150 t/h).l'atelier de broyage de cru est composé de deux broyeurs de 140t/h chacun. L'atelier de cuisson regroupe deux fours dont le débit clinker est de 1560 t/h. le broyage de ciment comprend deux broyeurs de 100t/h chacun. Les expéditions du ciment s'effectuent à partir de deux stations, une pour les camions et une pour les wagons.

La description complète des différents aspects des procédés de transformation nécessite différents types d'informations représentant les éléments de base pour la modélisation et la simulation du procédé [35].

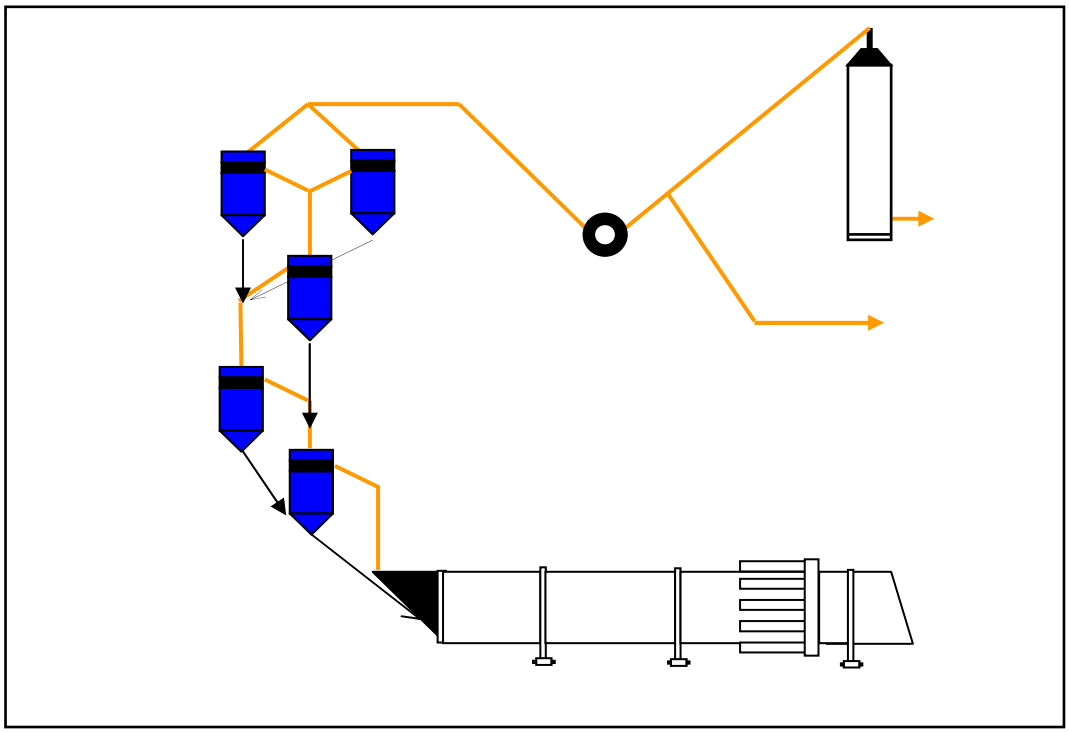


Figure 5.16. *Schéma synoptique de l'atelier de clinkérisation.*

5.3.3. Diagnostic de l'atelier de clinkérisation par neuro-flou

5.3.3.1 Analyse de dysfonctionnements

Cette étape à pour objectif l'identification des dysfonctionnements pouvant affecter la mission du système. Cette analyse est grandement facilitée par la reconnaissance des modèles structurels et fonctionnels de l'installation. L'AMDEC de l'atelier de clinkérisation est présenté en ***annexe 1.***

Les variables de notre système neuro-flou utilisées sont présentées sur le Tableau 6.1 Donc, nous avons un système neuro-flou de 27 entrées et 4 sorties qui sont été créées pour faire un diagnostic du système. L'Architecture du système neuro-flou est illustré sur la figure 5.16. Les règles créées avec le système sont des connaissances a priori, *c.à.d la base de règle à priori*. Chaque variable à une partition initiale est modifié au long de la phase d'apprentissage (nombre de ensembles flous pour chaque variable).

Les entrées de RNF (observations)	*La couche des règles*	*Les sorties de RNF (mode de défaillance)*
1. Vitesse de rotation four. 2. Débit entrée gaz four. 3. Débit sortie gaz four. 4. Température entrée gaz four. 5. Température sortie gaz four. 6. Débit entrée charge four. 7. Débit sortie charge four. 8. Température entrée charge four 9. Température sortie charge four. 10. débit entrée gaz cyclone 11. Débit sortie gaz cyclone 12. Température entrée gaz cyclone 13. Température sortie gaz cyclone 14. Débit entrée charge cyclone 15. Débit sortie charge cyclone 16.Température entrée charge cyclone 17. Température sortie charge cyclone 18. Débit sortie gaz brûleur 19. Température sortie gaz brûleur. 20. Débit sortie charge partie préparation. 21. température sortie charge partie préparation. 22. Débit entrée charge vanne 23. Débit sortie charge vanne. 24. Etat vanne. 25. Capacité silo. 26. Débit entrée charge silo. 27. Débit sortie charge silo.	➢ **Vide** Ou ➢ **Règles (connaissance A priori)**	1. Décarbonation perturbée. 2. Mauvaise clinkérisation. 3. Mauvais refroidissement. 4. Insuffisance ou absence de gaz.

Tableau 5.1. *Les variables d'états d'unité de clinkerisation.*

Le raisonnement pour le diagnostic est décrit sous forme de règles floues à l'intérieur du système neuro-flou. L'avantage principal d'utilisation de la base de règle floue réside dans sa modularité et facilité d'extension (suppression ou ajout d'autres règles).

La base de règles initiale pour établir le diagnostic des défaillances est construite en exploitant le modèle élaboré en phase de dysfonctionnement de notre système (AMDEC).

En effet, cette analyse permet d'établir les liens de causes à effets entre les composants défaillants et les symptômes observés. Ces liens sont représentés sous formes de règles floues construisant la base de connaissances (tableau 6.2) qui sera apprise plus tard et ensuite testée, pour effectuer les raisonnements flous nécessaires pour aboutir aux résultats exprimant la fonction diagnostic (figure 6.2).

La détection des anomalies est représentée sous forme de message d'alarme destiné à signaler à l'opérateur l'apparition d'une anomalie (ou des anomalies) et permet d'identifier le composant responsable à l'aide d'une base de donnée qui stocke toutes les informations de l'AMDEC. Après apparition du message, l'opérateur peut consulter le message pour plus d'information ou bien le supprimer.

La figure 5.17 illustre trois variables critiques de l'atelier étudié en fonctionnement normal.

5.3.3.2. L'apprentissage de NEFDIAG

Après avoir donné la structure du système nous, avons besoin d'associer à chaque variable d'entrée un nombre fini d'ensembles flous et utiliser le type de fonction d'appartenance associée, et aussi l'intervalle de définition des variables et ou ensembles flous. Toutes ces configurations, ont l'avantage d'accélérer la vitesse d'apprentissage et aussi l'orientation du système neuro-flou pour une meilleure exploitation des ressources.

étape.1 : après détermination de toutes les variables d'entrée et les paramètres de notre système, NEFDIAG donne d'abord les partitions de chaque variable d'entrée. La figure 5.16 illustre une partition initiale de fonction d'appartenance de la variable d'entrée « vitesse rotation four ». La figure 5.17 représente la partition des variables après la phase d'apprentissage.

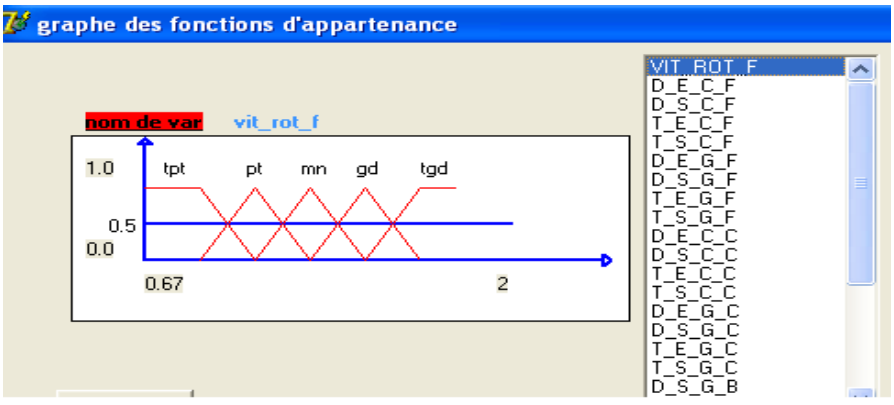


Figure 5.17.a *Partition initiale par des FA de « vitesse rotation four »*

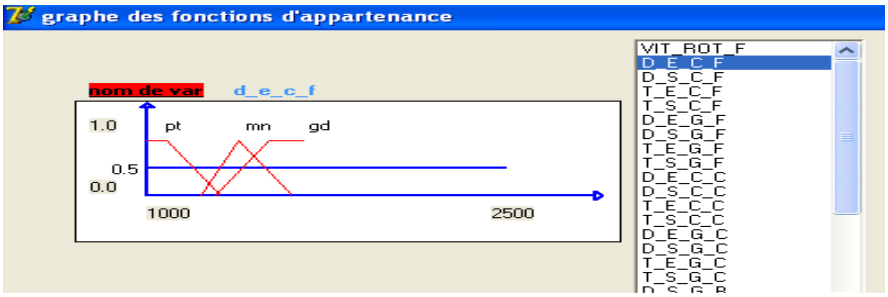


Figure 5.17.b *Les fonctions d'appartenance après apprentissage.*

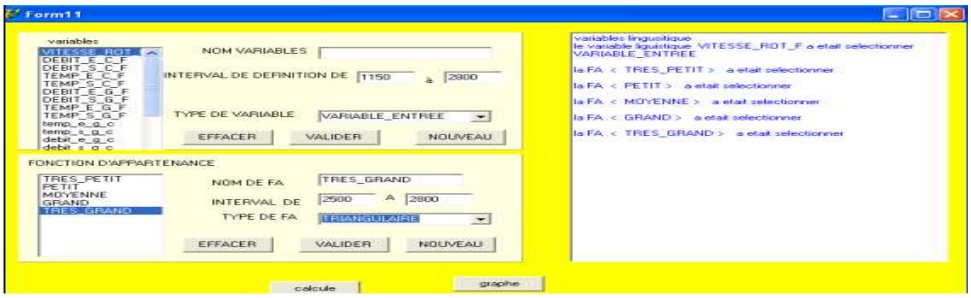


Figure 5.18.a *Définition des paramètres d'entrées.*

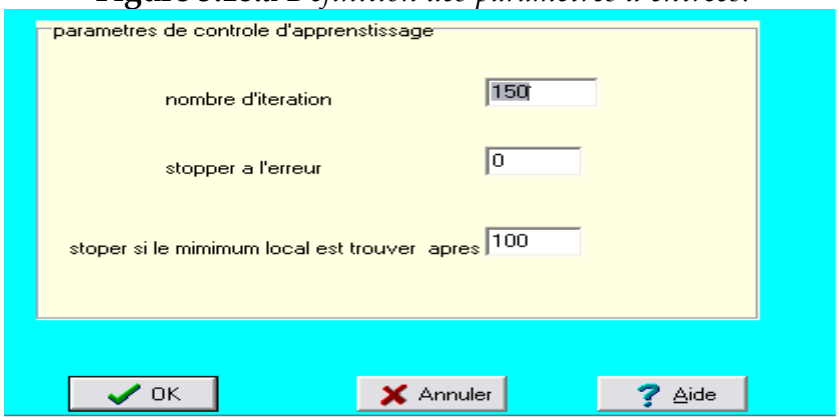


Figure 5.18.b *Les paramètres d'apprentissage.*

La figure 5.19 représente le changement des paramètres du SNF « Système Neuro-Flou » qui très important pour que notre système réalise sa phase d'apprentissage de meilleures conditions.

Cette phase d'initialisation est caractérisée aussi par une partition type de chaque variable d'entrée et non pas pour les sorties. La figure 5.16 illustre la vitesse de rotation du four, les profils de température des fumées et de la charge de l'entrée jusqu'à la sortie du four sans une défaillance

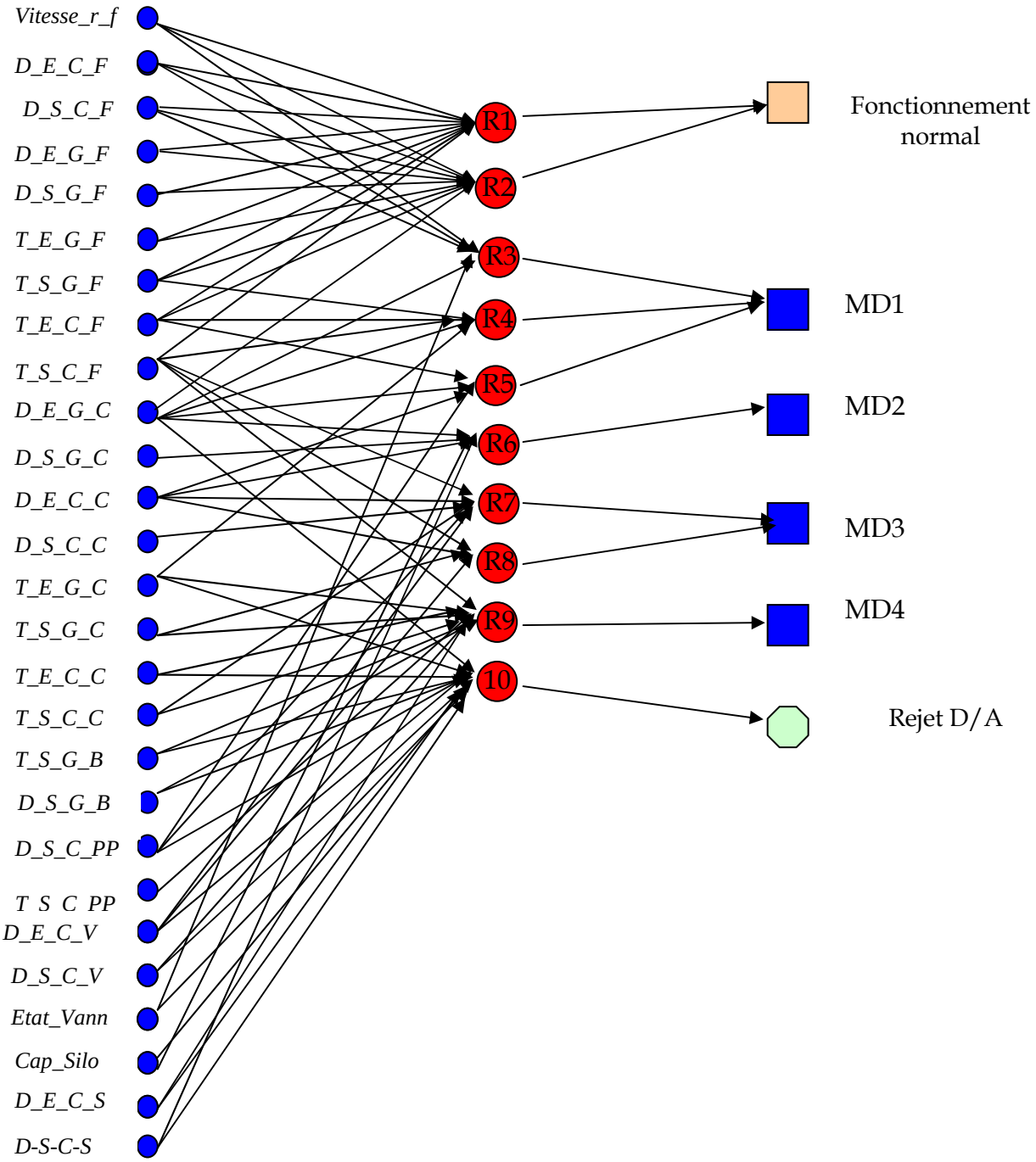
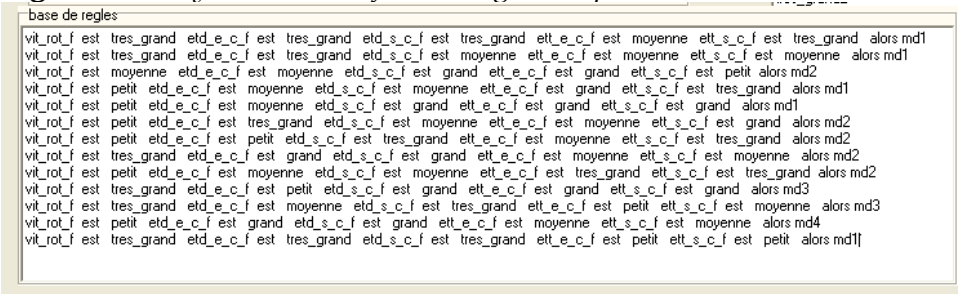


Figure 5.19. *Système neuro-flou à diagnostic pour unité de clinkérisation.*


base de regles

vit_rot_f est tres_grand etd_e_c_f est tres_grand etd_s_c_f est tres_grand ett_e_c_f est moyenne ett_s_c_f est tres_grand alors md1
vit_rot_f est tres_grand etd_e_c_f est tres_grand etd_s_c_f est moyenne ett_e_c_f est moyenne ett_s_c_f est moyenne alors md1
vit_rot_f est moyenne etd_e_c_f est moyenne etd_s_c_f est grand ett_e_c_f est grand ett_s_c_f est petit alors md2
vit_rot_f est petit etd_e_c_f est moyenne etd_s_c_f est moyenne ett_e_c_f est grand ett_s_c_f est tres_grand alors md1
vit_rot_f est petit etd_e_c_f est moyenne etd_s_c_f est grand ett_e_c_f est grand ett_s_c_f est grand alors md1
vit_rot_f est petit etd_e_c_f est tres_grand etd_s_c_f est moyenne ett_e_c_f est moyenne ett_s_c_f est grand alors md2
vit_rot_f est petit etd_e_c_f est petit etd_s_c_f est tres_grand ett_e_c_f est moyenne ett_s_c_f est tres_grand alors md2
vit_rot_f est tres_grand etd_e_c_f est grand etd_s_c_f est grand ett_e_c_f est moyenne ett_s_c_f est moyenne alors md2
vit_rot_f est petit etd_e_c_f est moyenne etd_s_c_f est moyenne ett_e_c_f est tres_grand ett_s_c_f est tres_grand alors md2
vit_rot_f est tres_grand etd_e_c_f est petit etd_s_c_f est grand ett_e_c_f est grand ett_s_c_f est grand alors md3
vit_rot_f est tres_grand etd_e_c_f est moyenne etd_s_c_f est tres_grand ett_e_c_f est petit ett_s_c_f est moyenne alors md3
vit_rot_f est petit etd_e_c_f est grand etd_s_c_f est grand ett_e_c_f est moyenne ett_s_c_f est moyenne alors md4
vit_rot_f est tres_grand etd_e_c_f est tres_grand etd_s_c_f est tres_grand ett_e_c_f est petit ett_s_c_f est petit alors md1

Tableau 5.2. *La base de règle associée*

Etape-2 : pour l'étape de simulation en fonctionnement normale la figure 5.20 illustre trois variables critique de l'unité étudié qui sont : la vitesse de rotation du four, le température entrée gaz et la température sortie charge du four dans un fonctionnement sans anomalies .

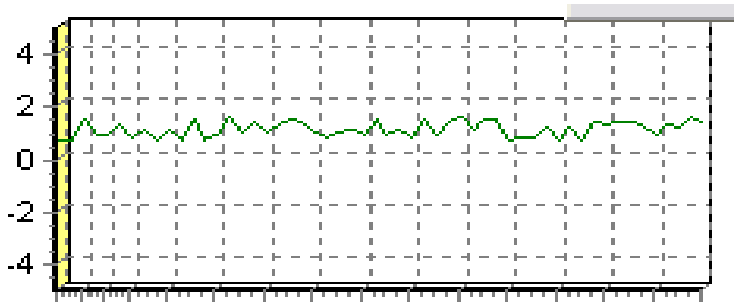


Figure 5.20.a. *Vitesse rotation four en fonctionnement normale.*

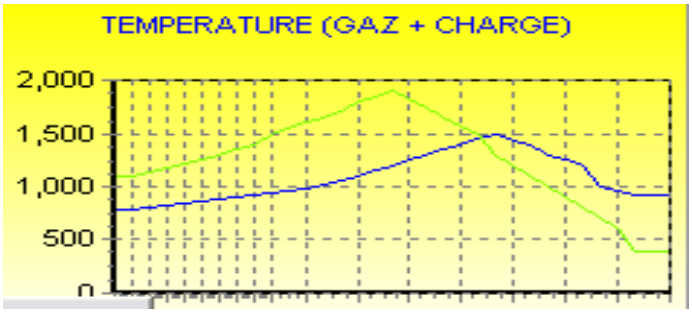


Figure 5.20.b. *température entrée gaz et charge en fonctionnement normale.*

Etape-3 : le système Neuro-Flou NEFDIAG, après son apprentissage fait une classification des modes de défaillance. La figure 5.21 illustre quatre modes de défaillance classés par NEFDIAG.

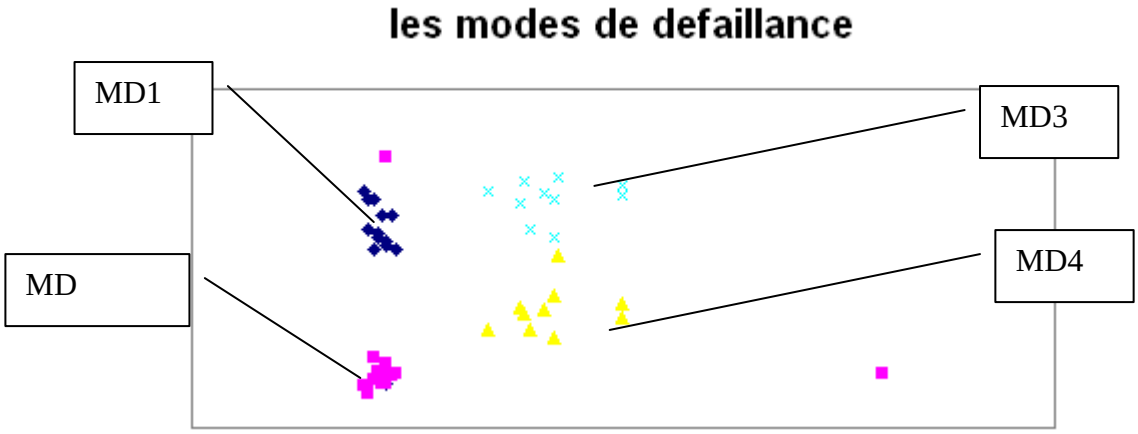


Figure 5.21. *Classification de modes de défaillances.*

Etape-4 : on provoque une défaillance au niveau de four, par la modification de vitesse de rotation à 1.8 tour/s. La température de sortie de charge tend vers un nouvel état permanent : 1600°C « très élevée ».

Le système NEFDIAG fait une intervention soit pour

- Faire diminuer la température du gaz d'entrée par le réglage de la température et/ou du débit de sortie de gaz de brûleur, ou
- Faire diminuer la vitesse de rotation de four.

La figure 5.22 illustre l'intervention de NEFDIAG après l'apparition d'une défaillance « température charge très élevée » qui a comme cause « vitesse de rotation très élevée» par le réglage de cette dernière.

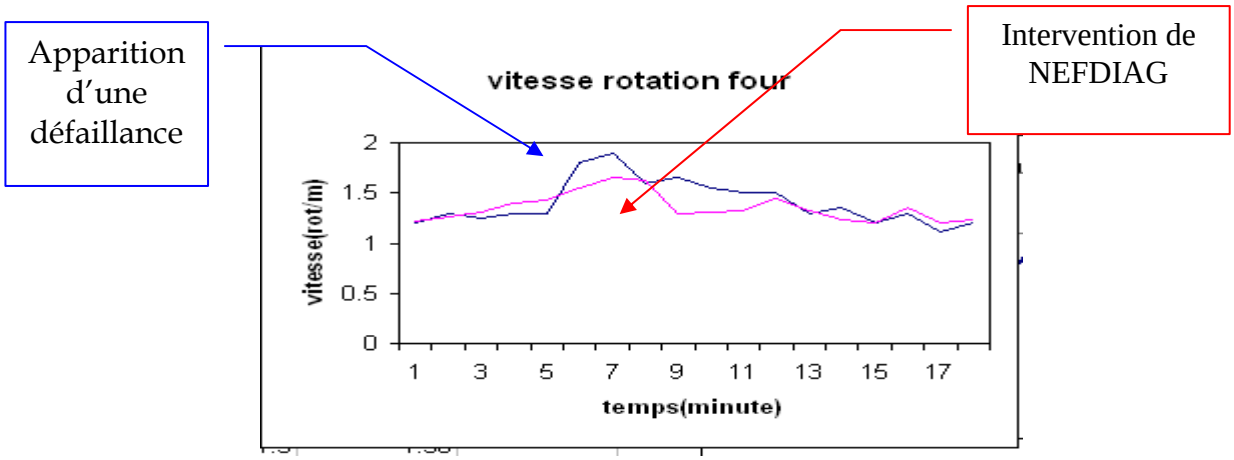


Figure 5.22. L'intervention de NEFDIAG après l'apparition d'une défaillance.

5.3.3.3 Présence d'un dysfonctionnement

Apres apparition d'une anomalie, un message d'alarme permet à l'opérateur de détecter le dysfonctionnement et aussi de localiser le composant responsable. La figure 5.19 illustre le système de cuisson avec la présence d'un dysfonctionnement

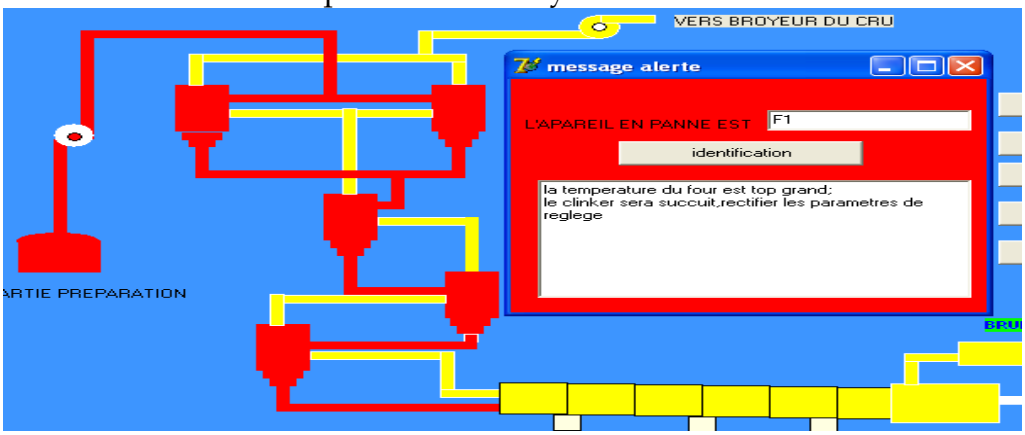


Figure 5.23. *Un message d'alerte signalant une panne.*

Sur le tableau 5.3 nous avons fait une description détaillée des variables d'entrée et sorties de notre système pour mieux exploiter les connaissances a priori pour aboutir un meilleur résultat d'apprentissage.

5.3.3 Résultats des données « Iris de FISHER »

Étape -1 : Les données Iris introduites par « Fisher » est un exemple concret pour l'explication des résultats obtenus.

Les trois type de fleurs Iris peuvent être classifiés par la longueur et la largeur de «sépale» et «pétale».

L'ensemble de données contient 3 classes « *setosa, versicolor, verginica* »de 50 instances pour chacun. L'ensemble d'apprentissage contient 150 cas, de 4 attributs et les classes sont codées par un vecteur binaire de trois composantes.

Pour l'implémentation dans NEFDIAG nous avons divisés ces données en deux sous ensembles de données, une pour l'apprentissage et l'autre pour le test.

Le tableau 5.4 illustre les rangs des 4 variables d'entrées.

	Min	***max***
« ***Sepal length*** »	4	8
« ***Sepal width*** »	2	5
« ***Petal length*** »	1	7
« ***Petal width*** »	0	3

Tableau 5.3. *Les rangs de variables d'entrées « Iris ».*

Les classes sont codées par « classe1, classe2, classe3 ».

Sur la figure 5.24 la partition initiale de « sepal length » est donnée par 4 fonctions d'appartenances. Sur la figure 5.24, nous illustrons cette partition après l'apprentissage.

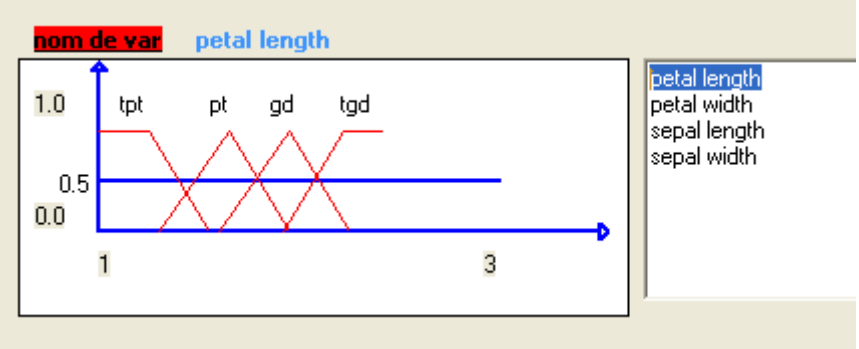


Figure 5.24. *Partition initiale de « petal length ».*

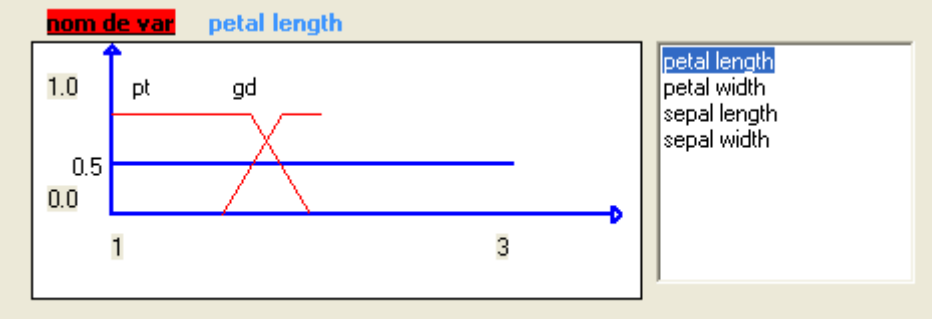


Figure 5.25. *Partition après apprentissage de « petal length ».*

Etape-2 : après la partition des variables et touts les paramètres de NEFDIAG, nous avons fait un apprentissage de NEFDIAG les 100 premières formes. Apres la phase d'apprentissage nous avons trouvé 5 règles floues qui sont listés sur le tableau 5.5.

règles	*Sepal length*	*Sepal whidth*	*Petal length*	*Petal width*	*classes*
R1	petit	moyenne	petit	moyenne	**setosa**
R2	moyenne	petit	moyenne	moyenne	**versicolor**
R3	moyenne	petit	grand	grand	**virginica**
R4	grand	moyenne	grand	grand	**Virginica**
R5	grand	petit	grand	grand	**virginica**

Tableau 5.4. *La base de règle pour 3 expériences d'apprentissage.*

Les résultats du test de 50 formes resté de données par NEFDIAG sont schématisés sur la figure 5.26.

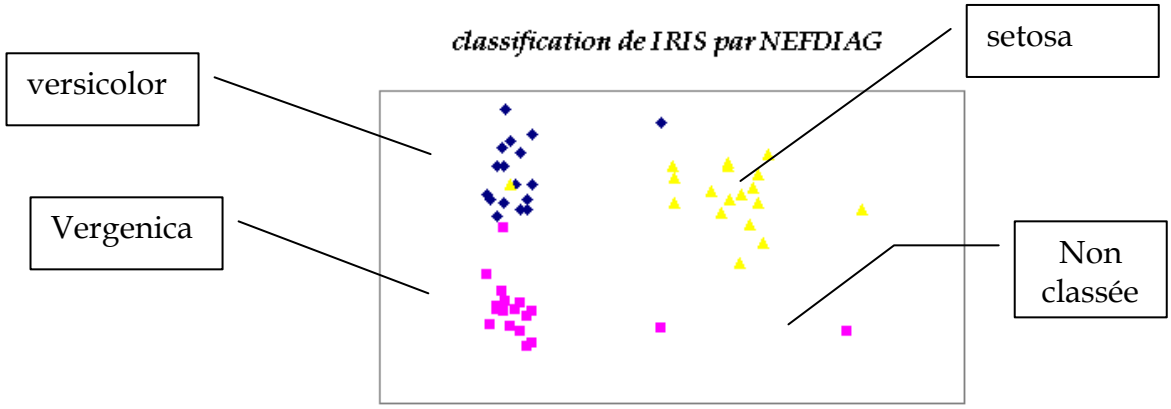


Figure 5.26. *Classification de données « Iris ».*

Nous observons qu'il y a 6 formes qui sont éliminées par NEFDIAG «mal classifiées». Le pourcentage de classification est ainsi donné :

$$\frac{nombre\ de\ formes\ correcteme\ nt\ classées\ \times 100}{nombre\ de\ formes\ total} = \frac{44 \times 100}{50} = 88\%$$

5.3.4 Comparaison avec autres systèmes

Nous avons fait une petite comparaison entre RNA, ANFIS et NEFDIAG.

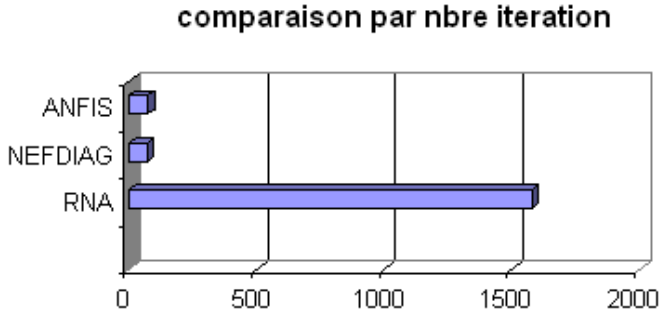


Figure 5.27. Comparaison entre ANFIS, RNA, NEFDIAG.

Nous remarquons que le nombre d'itération de RNA est très élevé et dépasse 1600 itérations. Par contre les deux autres systèmes Neuro-flous leur itération est environs 60 et 75 itération d'apprentissage.

5.4. Conclusion

Dans ce chapitre, nous avons présenté un nouvel outil de diagnostic neuro-flou, nous avons détaillé la mise en œuvre d'un exemple d'application industrielle par l'outil de développement NEFDIAG.

Nous avons illustré d'utilisation de cet outil d'aide au diagnostic sous forme d'un prototype « NEFDIAG » installé sur un PC.

Nous avons abordé les différentes étapes pour l'élaboration d'un système d'aide au diagnostic à partir de méthodes de classification et reconnaissances de formes floue.

Quand le diagnostic est basé sur des observations multiples, ces derniers sont regroupées pour former des classes (modes de défaillances), auxquelles une nouvelle observation sera comparée pour être identifiée.

NEFDIAG est un logiciel informatique de simulation interactive, réalisé au sein de LAP implémenté par DELPHI, pour le développement, l'apprentissage et le test d'un système Neuro-Flou de classification des pannes d'un procédé industriel.

NEFDIAG a été représenté comme un type spécial de Perceptron flou, à trois couches utilisé pour classifier des défaillances,en utilisant le système Neuro-Flou de type 3.

NEFDIAG fait un apprentissage à deux phases :

- ✓ apprentissage des règles, NEFDIAG génère les règles floues par le parcours des données ensuite optimise les règles par :
- ✓ apprentissage des paramètres des ensembles flous qui sont utilisés pour partitionner les données des formes à classer et les paramètres des donnes.

CONCLUSION GENERALE

Le travail mené au cours de ce mémoire nous a permis d'élaborer un module de diagnostic en ligne pour un système de production. Ce module est basé sur l'approche neuro-flou. Cette approche a été sélectionnée pour

- sa conception à base de règles floues,
- sa tolérance aux fautes,
- son calcul parallèle,
- et ses capacités d'apprentissage.
- L'application aux systèmes complexes et incertains.
- L'application aux control des systèmes industriels.

Les principales contributions de ce mémoire sont regroupées en trois parties. Une première partie regroupe l'état de l'art autour de trois points d'intérêts en étroite corrélation :

- Différentes méthodologies de surveillance des systèmes de production (Chapitre 1).
- Application des réseaux de neurones et de la logique floue au diagnostic industriel (Chapitre 2, Chapitre3).
- Différentes architectures de réseaux de neurones artificiels et une petite comparaison de ces architectures pour mettre en exigence les avantages et les inconvénients de ces derniers afin de sélectionner la meilleure architecture utilisée en diagnostic.

La deuxième partie de notre travail, articulée autour des deux chapitres suivants, synthétise l'essentiel de notre contribution scientifique. Nous avons ainsi proposé une nouvelle approche pour le diagnostic industriel d'un système de production par l'utilisation des systèmes neuro-flous .

La troisième partie concerne l'exploitation industrielle. Cette étude a été menée afin de diagnostiquer un système de production par l'utilisation de l'approche neuro-flou « Perceptron flou ».

Les méthodologies de surveillance peuvent être divisées en deux grandes catégories : avec et sans modèles. Les premiers se basent sur l'existence d'un modèle formel de l'équipement et utilisent généralement les techniques de l'automatique. La deuxième catégorie de méthodologies et plus intéressante des lors qu'un modèle de l'équipement est inexistant ou difficile à obtenir. Dans ce cas, on utilise les outils de la statistique et de l'intelligence artificielle. La fonction diagnostic est alors vue comme une application de reconnaissance de formes. Les formes représentent le vecteur d'entrée composé par les différentes données de l'équipement (données mesurables et qualifiables) et les classes représentent les différentes modes de fonctionnement/défaillances.

Les Systèmes d'intelligence artificiels hybrides qui combinent la logique floue, les réseaux neurones, les algorithmes génétiques, et les systèmes experts prouvent leur efficacité dans une variété de problèmes de monde réel et dans l'industrie.

Chaque technique intelligente à des propriétés particulières (par exemple capacité d'apprentissage, explication de décisions) cela le fait a convenu pour des problèmes particuliers et pas pour autres.

Ces limites ont été une raison derrière la création de systèmes hybrides intelligents où ; deux ou plus de techniques sont combinées dans une manière qui vaincre les limitations d'une seule technique.

Un résultat direct de cette hybridation dans le domaine de IA est : les systèmes neuro-flous. Ces systèmes sont nés de l'association des réseaux de neurones avec la logique floue, de manière à tirer profit des avantages de chaque une de ces deux techniques. Les principales propriétés des systèmes neuro-flous sont leur capacité à traiter dans un même outil des connaissances numérique et symboliques d'un système.

Ils permettent donc d'exploiter les capacités d'apprentissage des réseaux de neurones d'un part et les capacités de raisonnement de la logique floue d'autres part. Afin d'exploiter cette technique dans le diagnostic industriel, nous avons choisi un réseau neuro-flou spécial : le perceptron flou, et pour implémenter ce dernier, nous avons développé un pro-logiciel informatique que nous avons nommé NEFDIAG 1.00 .

Dans le dernier chapitre de ce travail, nous avons présenté un nouvel outil de diagnostic neuro-flou. Nous avons détaillé la mise en œuvre d'un exemple d'application industrielle par l'outil de développement NEFDIAG.

Nous avons illustré d'utilisation de ne notre outil d'aide au diagnostic sous forme d'un prototype « NEFDIAG » installer sur un PC.

Nous avons abordé les différentes étapes à suivre pour l'élaboration d'un système d'aide au diagnostic à partir des méthodes de classification et de reconnaissances de formes floue.

Quand le diagnostic est basé sur des observations multiples, ces dernières sont regroupées pour former des classes (modes de défaillance), auxquelles une nouvelle observation sera comparée pour être identifiée.

NEFDIAG est un logiciel informatique de simulation interactive implémenté par DELPHI, pour le développement, l'apprentissage et le test d'un système neuro-flou de classification des pannes d'un procédé industriel.

NEFDIAG à été représenté comme un type spécial de perceptron flou, à trois couche utiliser pour classifier des défaillances, en utilisant le système neuro-flou de type 3.

NEFDIAG fait son apprentissage par un ensemble des formes, tell que chaque forme sera affectée (classée) vers un des classes prédéfinies. NEFDIAG génère les règles floues par un parcoure des données ensuite optimise les règles par apprentissage les paramètres des sous ensembles floues qui sont utilisée pour partitionner les données « caractéristiques » des formes à classées.

Le système NEFDIAG peut démarre avec une base de connaissance partiel des formes, et peut ensuite la raffiner durant l'apprentissage, ou il peut démarrer avec une base de connaissance vide.

Ce travail a permis d'ouvrir les perspectives suivantes **:**

- un premier point concerne l'amélioration des performances de notre système neuro-flou, par l'ajout de concepts « temporels ». On peut imaginer alors un système neuro-floue temporel qui nous donne une solution distribuée, c'est-à dire un réseau de neurones dynamique avec sa couche neuro-floue temporelle pour chaque type de cause.

- D'une autre part on peut intégrer notre système neuro-flou dans un système expert « exemple G2 », parce que nous avons trouver que l'apprentissage de notre système neuro-floue tombe quelque fois dans un minimum local, et nous avons éliminé ce problème par l'utilisation de la méthode « recherche tabou ». Cependant, nous avons remarqué que nous pouvons faire l'apprentissage de notre RNF par les caractéristiques essentielles des Systèmes Experts « *chaînage avant, chaînage arrière* ». Aussi, nous pouvons faire une projection de l'algorithme de rétro propagation de gradient, par « la propagation des formes utilisant le chaînage avant, et le rétro propagation d'erreur par le chaînage arrière »

- Afin d'améliorer le temps de test nous proposons une intégration de notre système neuro-flou dans une circuit VLSI, par l'utilisation des spécifications d'une synthèse à haut niveau.

- Le développement actuel des Technologies de l'Information et de Communication « TIC », facilite la mise en place d'une vraie stratégie de e-maintenace. Ainsi les systèmes Multi Agents représentent des outils de l'IAD « intelligence artificiel distribuée » avec une capacité très intéressante concernant la prise de décision en fonction de certains critères imposés par l'expert. Pour cela nous proposons une architecture neuro-flou pour un agent mobile interactive utilisons la « TIC » pour la e-maintenace des systèmes industriels.

- Faire un étude de spécification pour créer une méthode ou paradigme qui intègre toutes les méthodes de IA « RNA, SMA, SE, AG, RNF, LF,SVM… » ou leurs combinaisons « deux ou plus », pour faire une décision multi critères et choisir la meilleure approche pour un module de diagnostic d'un système .

Sur un plan plus général, plusieurs prolongements à ce travail sont envisageables et permettent de valider et d'améliorer l'approche proposée.

Références bibliographies

[1] H. Hirota et W. Pedrycz, *OR/AND neuron in modeling fuzzy set connectives*, IEEE Transactions on Fuzzy Systems **2,** pp**.** 151–61. 1994.

[2] C. Moraga, P. Sincak et J. Vascak, *Neuro-fuzzy modeling of compensating systems, Quo vadis Computational Intelligence? ,* Springer Verlag, Heidelberg. 2000.

[3] H. Kwan et Y.Cai, *A fuzzy neural network and its application to pattern recognition*, IEEE Transactions on Fuzzy Systems, pp .185-193. 1994.

[4] D. Nauk, F. Klawonn et R. Kruse, *combing neural networks and fuzzy controller*s, dept .of Computer Science, Université de Braunschweig, Germany .1999.

[5] H.R. Berenji et P. Khedkar, *Learning and Tuning Fuzzy Logic Controllers through Reinforcements*, IEEE Transactions on Neural Networks, Vol (3), pp. 724-740.1992.

[6] Y. El-shayeb, *apport de la logique floue à l'évaluation de l'aléa « mouvement de terrain des sites géométrique » : proposition d'une méthodologie général*, thèse de doctorat, institut national polytechniques de Loraine. 1999.

[7] J. Roger Jang, *ANFIS: Adaptive-network-based fuzzy inference system*, IEEE Trans. Syst., Man, and Cybernetics, pp. 665-685. 1993.

[8] J. Keller et H. Tahani, *Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks*, *International Journal of Approximate Reasoning*, pp. 221-240. 1992.

[9] H. Berenji et P. Khedkar, *Fuzzy Rules for Guiding Reinforcement Learning, In International*. Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pp. 511-514.1992.

[10] D. Nauck et R. Kruse, *Neuro-Fuzzy Systems for Function Approximation, Fuzzy Sets and Systems*, pp. 261-271.1999.

[11] S. Tano, T. Oyama et T. Arnould, *Deep combination of Fuzzy Inference and Neural Network in Fuzzy Inference, Fuzzy Sets and Systems*, pp. 151-160. 1996.

[12] J. Feng et L. Teng, *An Online Self Constructing Neural Fuzzy Inference Network and its Applications*, IEEE Transactions on Fuzzy Systems, Vol(6),pp. 12-32.1998.

[13] S. Sulzberger, N. Tschicholg-Gurman et S. Vestli, *FUN: Optimization of Fuzzy Rule Based Systems Using Neural Networks*, In Proceedings of IEEE Conference on Neural Networks, San Francisco, pp. 312-316.1993.

[14] N. Kasabov, *Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation*, Methodologies for the Conception, Design and Application of Soft Computing, World Scientific ,pp. 271-274.1998.

[15] **R.Mahdaoui**, H. Houassi, Pilotage d'un système de production par approche base de connaissance. Memoire d'Ingénieur, Université de Batna,2001.

[16] N. Kasabov, *Evolving Fuzzy Neural Networks for supervised/Unsupervised On-line*, Knowledge-Based Learning, IEEE Transactions of Systems, Man and Cybernetics, Part B – Cybernetics , vol31,No.6, December,2001.

[17] James J. Buckley;Leonard J. Jowers, *Simulating Continuous Fuzzy Systems, edition springer. 2006.*

[18] A. Assoum, *étude de la tolérance aux aléas logiques des réseaux de neurones artificiels*, thèse de doctorat, institut national polytechnique de Grenoble, France.1997.

[19] M. Baatouche, F. Hachouf et F. Mezhoud, *segmentation d'images couleur par une approche neurofloue*, 18ème Colloque GRETSI, Toulouse, sept.2001.

[20] L. Baghli, *contribution à la commande de la machine asynchrone, utilisation de la logique floue, des réseaux de neurones et des algorithmes génétiques*. Thèse de doctorat, Université de Henri Poincaré, Nancy, France.1999.

[21] G. Bassili, *an adaptatif neuro-fuzzy method (AFNIS) for estimation songl–trial movement's related potential*,HdR université de Milano, Italia.1997.

[22] H. Bing et H. Dolf, *fuzzy neural networks model for building energy diagnosis*, Eighth International IBPSA Conference Eindhoven, Netherlands August 11-14, 2003.

[23] R. Bisdroff, *on natural fuzzification of Boolean logic*, Proceedings Information Processing and Management of Uncertainty, IPMU'96, Granada, 593-599.2004.

[24] S. Breton, *une approche neuronale utilisant la vision Binoculaire par reconstruction tridimensionnelle*, thèse de doctorat, Université de Ahaut-Alsase, France.1999.

[25] S. Brock, *Application of AFNIS controller for two-mass-system*,. ESIT, 14-15 September 2000, Aachen, Germany 2000.

[26] R. Comoum Rimi, *apprentissage par renforcement utilisant les réseaux de neurones, avec des applications au contrôle moteur*, thèse de doctorat, INPG France. 2002.

[27] G. Dreyfus, J. Martinez, M. Samuelides, M. Gordon, F. Badran, S. thiria et L. Hérault, *réseaux de neurones, méthodologies et application* ; Edition Eyrolles.2002.

[28] F. Santo Oseo, *INSS : un système hybride neuro-symbolique pour l'apprentissage automatique constructif*, thèse de doctorat, INPG France.1998.

[29] R. Fuller, *Neural fuzzy systems*, Université de Abo Akademi . ESF Séries A:443, 1995.

[30] Habbi H, Zelmat M, Ould Bouamama B; A *dynamic fuzzy model for a drum-boiler-turbine system*; Automatica 39, pp 1213-1219. 2003.

[31] K. Hadjar et M. Ben Ahmed, A*daptative multi agent neural approach for automated interpretation of the ECG*, High School of Management, Tunisia.

[32] S. Touaf, diagnostic *logique des systèmes complexes dynamiques dans un contexte multi agent,* Thèse de Doctorat de Université Joseph Fourier Grenoble, 2005.

[33] J.-S. R. Jang, *Neuro-Fuzzy Modeling: Architectures, Analyses, and Applications* Ph.D. Dissertation, EECS Department, Univ. of California at Berkeley, 1992.

[34] H. Kaboli et M. Savoji, *non linaire prediction of speech using AFNIS; comparison with neural networks*, Université de shahid bahcheti, Tahran, Iran.2001.

[35] K. Khedhri, *modélisation et simulation d'une application industrielle par le générateur de systèmes experts G2*, Thèse de Magister, Université de Batna Algerie.2002.

[36] E. Laurin, *Système intelligent d'assistance à la perception dans la conduite des véhicules*, Mémoire de Maîtrise en sciences appliques Université de Sherbrooke Canada.2000.

[37] H. Ming, *the improvement of a fuzzy neural network based back-propagation*, machine learning centre, Université de Hebi, China.1999.

[38] C. Mogara, *neuro fuzzy modelling with standard feed forward neural network*, ESIT 2000, 14-15 , Aachen, Germany. 2000.

[39] M. Mouchaouh, *conception d'un system de diagnostic adaptatif et prédictif basé sur la méthode fuzzy patern matching pour la surveillance en ligne des systèmes evolitifs*, thèse de doctorat de Reims chapagne-ardenne .2002.

[40] S. Ploix, *diagnostic des systèmes incertains : l'approche bornant*, thèse de doctorat INP de lorain, France.2004.

[41] A. Shigeo, *neural networks and fuzzy systems, theory and applications* .edition kluwer academic publishers London.1997.

[42] T. Coudert, *logique floue et systèmes multi agents pour un ordonnancement coopératif production/maintenance*, Ecole nationale d'Ingénieurs de Tabres.1997.

[43] R. Mellah et R. Toumi , *commande neuro-floue par mode glissant du robot PUMA 560 muni de vérins pneumatiques* , Journal Européen des systèmes automatisés, Vol 39; Numero 5/6, pages 739-765, 2005.

[44] C. Touzet, *contribution à l'étude et au développement de modèle connexionniste séquentiel d'apprentissage.* Thèse de doctorat, Université de Montpellier France.1990.

[45] A. Villmeur, *sûreté de fonctionnement des systèmes industriels fiabilité facteurs humains informatisation*, édition eyrolles.1988.

[46] P. Wira, *Réseaux neuromemetiques, modularité et statistique : estimation de mouvement pour l'asservissement visuel du robot*. Thèse de doctorat université de Haut-Alsas France.1992.

[47] M. Zemouri, *contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques : application à la maintenance*, Thèse de doctorat de l'université de Franche-Comté, France.2003.

[48] L. Zadeh, *Toward a Generalized Theory of Uncertainty (GTU) – An Outline*, Journal of Statistical Planning and Inference, Elsevier Science, Vol (105), pp. 233-264.2005.

[49] L Zadeh, *fuzzy sets; information and control*. Université de California, USA. pp. 8, 338-353,1965.

[50] L Zadeh, *Outline of a new approach to the analysis of complex systems and decision processes*, IEEE transactions on systems , Man, Cybernitics. VolSMC-3, PP. 28-44. 1973.

[51] G. Zweingelstein, *diagnostic des défaillances; théories et pratique pour les systèmes industriels*, col. traite des nouvelles technologies, séries diagnostic et maintenance, hermès .1995.

[52] A. Abraham, *cerebral quotion of neuro fuzzy tevhniques-hype or hallelujah*, school of computing and information technology, Victoria, Australia.2000.

[53] L. Wang et J. Mendel, *Generation fuzzy rules by learning from examples,* IEEE transactions on systems. 1414 -1427 . 1992.

[54] A. Abraham et B. *Nath, Hybrid intelligent systems design- a review of a decade of research*, school of computing and information technology, Université de Monach, Churchill, Australia.2001.

[55] D. Racoceanu, *contribution à la surveillance des systèmes de production en utilisant les techniques de l'intelligence artificielle*, Université de franche-Comté,

France.2006.

[56] M.D Mouss, *Diagnostic et conduite des systèmes de production par approche à base de connaissances*, Thèse de doctorat, Laboratoire d'Automatique et Productique, Université de Batna.Algerie.2005.

[57] S. Altug, M. Chou et H. Joel Trussell, *Heuristic constraints enforcement for training of and rule extraction from a fuzzy/neural architecture- Part II: implementation and application*, IEEE Transactions on Fuzzy Systems, pp 151 – 159. 1999.

[58] T. Ojala, *neuro-fuzzy systems in control*, department of electrical engineering. These de Master en science, Université de Tempre, Finland.1994.

[59] L. Khodja, *contribution à la classification floue non supervisée*, thèse de doctorat, Université de savoie. novembre 1997.

[60] L. Rutkowski, *Flexible neuro-fuzzy systems ,Structures, Learning and Performance Evaluation.* Kluwer Academic Publishers New York, 2004

[61] C. Lakhmi. Jain; N.M. Martin, *Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications.* CRC Press, ISBN: 0849398045.1998.

[62] **R.Mahdaoui, L.H. MOUSS**, *Diagnostic industriel par approche Neuro-floue, application à un système de production,* CIP Sétif, pp 77. 3-4 novembre 2007.

Annexes

Dans cette annexe nous présentons quatre algorithmes d'apprentissages en utilisant les notions suivantes:

- Γ : un ensemble de données d'apprentissage avec $|\Gamma|=s$,qui représente un problème de classification de défaillances,tel que la forme $p \in \Re^n$ qui doivent être classées dans m mode de défaillances,tel que $C_i \subseteq \Re^n$.
- $(p,t) \in \Gamma$: une forme d'apprentissage consiste d'un vecteur d'entrée $p \in \Re^n$ et un vecteur de sortie (désiré) $t \in [0,1]^m$. L'index de la classe de **P** est calculé par le grand valeur de **t** : classe (p) =argmax$_j\{t_j\}$.
- $R = (A, C)$: une règle floue de classification avec un antécédent $ant(R) = A$ et conséquent $con(R) = C$, avec $A = (\mu_{j1}^{(1)},.....,\mu_{jn}^{(n)})$ et C est un classe (mode de défaillance).
  Le degré d'accomplissement d'une règle R pour une forme p est
  $R(p) = A(p) = \min\{\mu_{j1}^{(1)}(p_1),.....\mu_{jn}^{(n)}(p_n)\}$.
- $\mu_j^{(i)}$ est la j^{eme} ensemble floue de partition floue initial du variable d'entée x_i.

Il existe q_i ensemble flou pour le variable x_i.

- $\mathbf{c}_A$: un vecteur de m entrées pour présenter les degrés d'appartenance cumulées pour chaque classe de toutes les formes avec A(p) >0 ; $\mathbf{c}_A[j]$ est le j^{eme} entrée de $\mathbf{c}_A$.
- $P_R \in [-1,1]$ une valeur qui représente la performance de règle R :

$$P_R = \frac{1}{s}\sum_{(p,t)}(-1)^c R(p), \quad avec\ c = \begin{cases} 0 & si\ calsse(p) = con(R), \\ 1 & si \quad autrement \end{cases}$$

- $v_r^{(j)}$: l'ensemble floue du variable de sortie y_j ($j \in \{1,.....,m\}$ qui est incluse dans le conséquent de règle floue R_r.
- C_j *: le* vecteur de sortie qui enregistre le degré d'appartenance de forme d'entrée pour la classe *j*.

Algorithme 1. *Algorithme d'apprentissage de règles par NEFDIAG.*

Début algorithme

Pour toutes forme (p,t) **faire**

Pour toutes entrée x_i **faire**

$$\mu_{ji}^{(i)} = \underset{\mu_j^{(i)}, j\in\{1,...q_i\}}{\arg\max} \left\{\mu_j^{(i)}(p_i)\right\}$$

Fin pour

Créer l'antécédent $A = (\mu_{j1}^{(1)}, ..., \mu_{jn}^{(n)}$;

Si ($A \notin$ liste des antecedents) **alors**

Ajouter A da, s la liste des antecedents.

Finsi

Finpour

Pour toutes forme (p,t) **faire**

Pour toutes $A \in$ liste d'antécédents **faire**

$\mathbf{c}_A$[classe(p)]= $\mathbf{c}_A$[classe(p)]+A(p)

f**inpour**

finpour

---

pour toutes $A \in$ liste d'antécédents **faire**

j = $\underset{i\in\{1...m\}}{\arg\max} \{c_A[i]\}$;

Créer une règle R avec l'antécédent A et conséquent C_j ;

Ajouter R dans la base de règle candidates ;

$$P_R = \frac{1}{s} c_A[j] - \sum_{i\in\{1,...m\}, i\neq j} c_A[i]$$

Finpour

Raffiner la base de règles candidates (sélectionner les meilleures règles) « voir algorithme 2 »

Fin algorithme

Algorithme 2. *Sélectionner les meilleures règles pour la base de règles*
Début algorithme

K = 0 ; stop = false ;
R**épéter**
R′ = argmax{P_R} ;
Si taille base de règles fixées a**lors**
Si k< k_{max} **alors**
Ajouter R′ dans la base de règle ;
Effacer R′ de liste des règles candidates ;
K :=k+1 ;
Sinon
Stop =true;
Finsi
Fin algorithme

Algorithme 3. *Apprentissage des ensembles floues dans NEFDIAG.*

Début d'algorithme

1- *sélectionner la forme suivante (p,t) de* Γ,propager la,par NEFDIAG et *détermine le vecteur de sortie C.*

2- *pour chaque unité de sortie C_i ; déterminer la valeur de delta*

$$\delta C_i = ti - Oc_i$$

3- *pour chaque unité de règle R avec $O_R > 0$*

a- *déterminer la valeur de δ*

$$\delta_R = O_R(1 - O_R) \sum_{C \in U_3} W(R,C)\delta_C .$$

b- *trouver x' tel que :* $w(x', R)(o_{x'}) = \min_{x \in U_1} \{w(x, R)(\delta_C)\}$

c -*pour l'ensemble flou w(x',R) déterminer les valeurs de δ pour les paramètres a,b,c utilisant le pas d'apprentissage σ>0.*

$$\delta_b = \sigma . \delta_R .(c-a) sgn(O_{x'} - b),$$

$$\delta a = -\sigma \delta_R ..(c-a) + \delta_b,$$

$$\delta c = \sigma \delta_R ..(c-a) + \delta_b.$$

Appliqué le changement aux w(x',R).

4 –si *l'itération est terminé, ou les critères de fin sont validé ;* ***alors*** *stop*

Sinon *aller a* ***(1).***

Fin d'algorithme

Algorithme 4. *Mises à jour des ensembles floue de conséquents*

{ Les paramètres d'entrées suivantes de cet algorithme sont

ν : L'ensemble floue qui doit calculer ses paramètres.

e : Valeur d'erreur.

τ : Degré d'accomplissement pour la règle qui utilise ν dans ça conséquent.

t : Valeur de vecteur de sortie désiré pour le domaine de ν

a, b, c, d Sont les paramètres de ν }

Début algorithme

Si ν est triangulaire **alors**

Shift = $\sigma \cdot \tau \cdot (c-a) \cdot \tau \cdot (1-\nu(t))$;

$\Delta b = \Delta b \ shift$;

Si $\nu(t) > 0$ a**lors**

$\Delta a = \Delta a + \sigma \cdot \tau \cdot (b-a) + shift$

$\Delta c = \Delta c - \sigma \cdot \tau \cdot (b-a) + shift$

Sinon

$\Delta a = \Delta a + \text{sgn}(t-b)\sigma \cdot \tau \cdot (b-a) + shift$

$\Delta c = \Delta c + \text{sgn}(\sigma \cdot \tau \cdot (b-a)) + shift$

Finsi

Fin algorithme.

Grammaire pour la spécification en mode texte

Nous pouons utiliser un fichier pour définir les paramètres de notre système; ensuite NEFDIAG fait une petite compilation pour générer un système neuro-flou utilisant la grammaire de type 0 suivante:

RNF :

[<Commentaire>] <Paramètres> [[<commentaire>]<données>] fin

Commentaire : ligne commence par #

Paramètres : PARAMETRES <nouvelle ligne>
<L1><L2><L3><max_regles><antecbase.nombre tableau>
<rang de var entée : tableau de <min max> pairs><nouvelle ligne>
[UTILISE SOMME<nouvelle ligne>/ UTILISE MAX
< nouvelle ligne>]
[INDIVIDUEL <nouvelle ligne>/ MEME < nouvelle ligne>]
[NOMSVAR<nouvelle ligne><nom_var>]
<nom_var> :<<chaîne> nouvelle ligne ><chaîne>……>

<Données> : <Ensemble floue>/ <connections>/ <poids>
<Ensemble floue> : [NOMS <nouvelle ligne>] FLOUE
<Nouvelle ligne>
<< a b c ls rs [n]><a b c ls rs [n]> ……><nouvelle ligne >
<Connexion> : MATRICE <nouvelle ligne>
<Tableau antécédents> <nouvelle ligne>

.

<a b c ls rs[n]> : a= point droite, b :centre, c :point gauche,
Ls=1 si ensemble de début, rs = 1 si ensemble de la fin