CHAPITRE 1

OPTIMISATION PAR ESSAIMS PARTICULAIRES AVEC FUZZY ARTMAP

Lors de la réalisation de ce mémoire, plusieurs méthodes ont été utilisées. Ce chapitre présente les divers aspects théoriques pour bien comprendre les expériences effectuées tout au long de ce mémoire.

Ce chapitre est divisé en trois sections. La première section présente la théorie du classificateur neuronique FAM. Il s'agit du classificateur à la base de ce mémoire et une bonne compréhension de son fonctionnement interne est nécessaire. La deuxième section présente les techniques d'apprentissage standard et l'effet de sur-apprentissage. Puis, la dernière section présente la stratégie d'apprentissage spécialisée utilisant l'optimisation par essaims particulaires (PSO) que nous avons développée. Cette stratégie utilise l'algorithme PSO afin d'éliminer la dégradation des performances présente dans ces réseaux.

1.1 Réseau de neurones fuzzy ARTMAP

Le fuzzy ARTMAP est un réseau de neurones artificiels appartenant à la famille ARTMAP qui permet l'apprentissage supervisé et non supervisé des observations dont les caractéristiques sont définies dans l'espace des nombres réels. Le principe de base des réseaux de neurones de type FAM a été introduit par Carpenter, Grossberg, Markuzon, Reynolds et Rosen en 1992 [2]. Ce type de réseau exploite le réseau fuzzy ART [6] qui utilise un algorithme non supervisé pour la génération des groupes (clustering). En fait, ce type de réseau est obtenu en remplaçant les modules ART1 [3,4] du système ARTMAP par un module fuzzy ART [5,6]. Le module fuzzy ART est obtenu en remplaçant l'opérateur d'intersection de type booléen du module ART1 par un opérateur ET de type flou. Le Tableau I présente les différences au niveau algorithmique

entre le module ART1 exploité par le modèle ARTMAP et le module fuzzy ART exploité par le modèle FAM.

Tableau I

Algorithme ART1 vs fuzzy ART [2]

ART1	fuzzy ART
FONCTION DE CHOIX	
$T_j = \frac{\lvert \mathbf{I} \cap \mathbf{w}_j \rvert}{\alpha + \lvert \mathbf{w}_j \rvert}$	$T_j = \frac{\lvert \mathbf{I} \wedge \mathbf{w}_j \rvert}{\alpha + \lvert \mathbf{w}_j \rvert}$
FONCTION DE VIGILANCE	
$\frac{\lvert \mathbf{I} \cap \mathbf{w} \rvert}{\lvert \mathbf{I} \rvert} \geq \rho$	$\frac{\lvert \mathbf{I} \wedge \mathbf{w} \rvert}{\lvert \mathbf{I} \rvert} \geq \rho$
FONCTION D'APPRENTISSAGE	
$\mathbf{w}_j^{(new)} = \mathbf{I} \cap \mathbf{w}_j^{(old)}$	$\mathbf{w}_j^{(new)} = \mathbf{I} \wedge \mathbf{w}_j^{(old)}$
$\cap$ = ET booléen (intersection)	$\wedge$ = ET flou (minimum)

Le FAM est simple à utiliser car un petit nombre de paramètres est nécessaire pour sa mise en oeuvre. En effet, son comportement est basé principalement sur le paramètre de choix (α), le paramètre de vigilance de base ($\bar{\rho}$), de MatchTracking (ε) et le paramètre d'apprentissage (β). Sa complexité algorithmique est faible et peut être mise en œuvre très efficacement en électronique numérique. Il existe d'autres algorithmes qui s'inspirent de la logique floue (eg, [7,8,]), mais le FAM se distingue de ceux-ci par le fait qu'il modifie ses poids synaptiques après chaque observation (apprentissage

séquentiel) au lieu d'effectuer un apprentissage "batch" après avoir inspecté l'ensemble des observations disponibles. Ceci est un avantage notable pour les applications d'apprentissage dites en-ligne. La figure 1 représente l'architecture du modèle FAM.

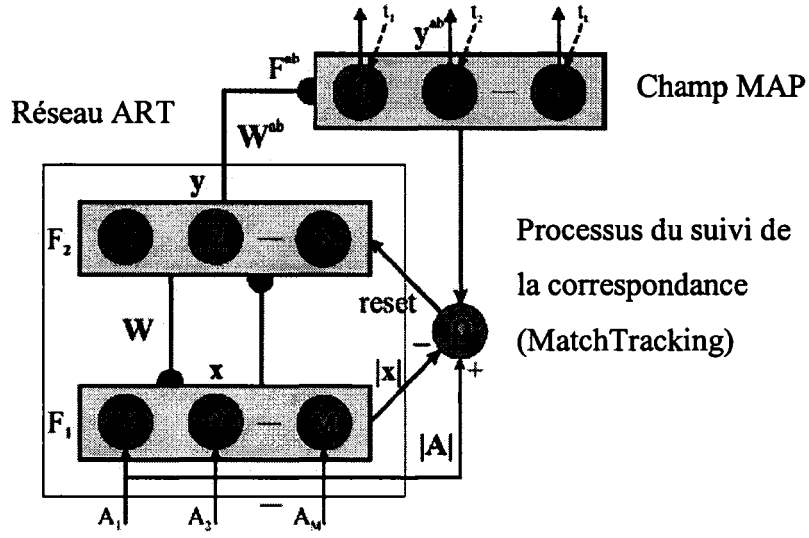

Figure 1 Architecture du Fuzzy ARTMAP [32]

Le réseau fuzzy ARTMAP contient trois couches distinctes, soit la couche F_1, F_2 et F^{ab} (le champ MAP). La valeur des caractéristiques du patron **A** est propagée à la couche F_1. La couche F_1 produit le vecteur d'activation $\mathbf{x} = \mathbf{A} \wedge \mathbf{W}$ et est reliée à la couche F_2 par les poids synaptiques **W**. La couche F_2 représente les catégories créées dans le réseau par lequel le patron est classifié. Elle est reliée au champ MAP par les poids $\mathbf{W}^{ab}$ et produit le vecteur d'activation **y**. Le champ MAP produit le vecteur d'activation $\mathbf{y}^{ab}$. En fait, il active une classe *K* selon la catégorie de la couche F_2 sélectionnée. Lors de la phase d'apprentissage, le champ MAP active, si besoin, le processus du suivi de la correspondance (MatchTracking), qui utilise les valeurs du patron **A** et de la couche F_1 (**x**) afin de modifier la recherche de la catégorie appropriée pour le patron présenté.

A - Phase d'apprentissage :

1 - Initialisation :

Lors de l'initialisation du réseau, aucun nœud de la couche F_2 n'est assigné. Les entrées (**a**, **t**) sont présentées au réseau, où **a** représente le vecteur de données des observations et **t** la classe qui lui est associée. Le réseau FAM exige que toutes les caractéristiques a_i des observations présentées au système soient codées en complément et qu'elles soient toutes comprises entre 0 et 1 inclusivement. Le complément d'un patron de m dimensions est composé de $M = 2 \cdot m$ dimensions et est défini par :

$$\mathbf{A} = (a; a^c) = \left(a_1, a_2, ..., a_m; a_1^c, a_2^c, ..., a_m^c\right) \tag{1.1}$$

$$a_i^c = (1 - a_i) \tag{1.2}$$

$$a_i \in [0,1] \tag{1.3}$$

2 - Activation des catégories :

L'observation **A** active la couche F_1 et elle est propagée vers F_2 à travers les connexions synaptiques de poids **W**. L'activation des nœuds j de la couche F_2 est déterminée selon la loi de Weber :

$$T_j(\mathbf{A}) = \frac{\left|\mathbf{A} \wedge \mathbf{w}_j\right|}{\alpha + \left|\mathbf{w}_j\right|}, \quad j = 1,2,..N \tag{1.4}$$

$$\left|\mathbf{w}_j\right| = \sum_{i=1}^{M} w_i \tag{1.5}$$

Le paramètre de choix α possède une valeur supérieure ou égale à 0. Ce paramètre influence le nombre d'itérations de recherche des catégories avant d'assigner une nouvelle catégorie déterminant ainsi la profondeur de recherche de la couche F_2. Ainsi, une grande valeur de ce paramètre favorise la création de nouvelles catégories alors qu'une petite valeur favorise la réutilisation des catégories existantes. La couche F_2 sélectionne un seul gagnant J = argmax $\{T_j ; j=1,2,..,N\}$. Ainsi, seulement le nœud J, ayant la plus grande valeur d'activation, restera actif. Si plus d'un nœud a la même valeur d'activation et qu'elle est maximale, le nœud avec le plus petit indice sera

sélectionné. Le test de vigilance s'exécute alors avec le nœud *J*. Il compare le degré de similitude entre $\mathbf{w}_J$ et **A** avec le paramètre de vigilance ρ :

$$\frac{|\mathbf{A} \wedge \mathbf{w}_j|}{M} \geq \rho \tag{1.6}$$

Si le test est concluant, l'activation des classes survient, sinon, le nœud *J* est désactivé et le système cherche un autre nœud respectant le test de vigilance. Si aucun nœud ne satisfait l'équation (1.6), un nœud de la couche F_2 non assigné est attribué à cette entrée. Le paramètre de vigilance détermine la taille maximale des catégories, soit $|w_j| \leq 2 \cdot (1-\rho)$ et sa valeur est comprise entre 0 et 1. De plus, ce paramètre influence le nombre de catégories créées. Une grande valeur de ce paramètre favorise la création de nouvelles catégories. À l'opposé, une petite valeur de ce paramètre tend à réutiliser le plus grand nombre de catégories possibles et ainsi diminue le nombre de catégories créées.

3 - Activation des classes :

Lorsque la couche F_2 sélectionne le nœud *J* gagnant, elle active le champ MAP via les poids synaptiques $\mathbf{W}^{ab}$ et assigne une classe à l'observation présentée. Si la réponse de ce nœud n'est pas la même que celle associée au patron d'entrée (**t**) , le processus du suivi de la correspondance (MatchTracking) s'enclenche. Ce processus augmente la valeur du paramètre de vigilance ρ (équation (1.7)) dans le but d'effectuer une nouvelle recherche à travers la couche F_2. Celle-ci continuera jusqu'à ce qu'un nœud classifie correctement l'observation présentée, ou jusqu'à qu'un nœud non assigné soit associé à ce patron.

$$\rho = \frac{|\mathbf{A} \wedge \mathbf{w}_J|}{|\mathbf{A}|} + \varepsilon \tag{1.7}$$

La valeur du paramètre de MatchTracking ε peut être négative ou positive. Une grande valeur positive favorise la création de nouvelles catégories lorsque la première tentative de classification d'un patron échoue. Une grande valeur négative facilite la réutilisation des catégories déjà existantes lorsque la classification a initialement échoué et ainsi tend à minimiser le nombre de catégories créées lors de l'apprentissage.

4 - Apprentissage :

L'apprentissage d'une observation **a** met à jour les poids synaptiques $\mathbf{w}_J$, et crée un nouveau lien associatif vers F^{ab} si J correspond à un nœud de la couche F_2 nouvellement associé. La mise à jour des poids synaptiques de la couche F_2 est effectuée selon l'équation :

$$w_J^{'} = \beta(A \wedge w_J) + (1-\beta) \cdot w_J \tag{1.8}$$

où : β est la vitesse d'apprentissage

Cet algorithme peut être ajusté pour un apprentissage lent, $0 < \beta << 1$, ou pour un apprentissage rapide, $\beta = 1$. Lors de l'utilisation du mode d'apprentissage rapide, le réseau FAM formera des hyper-rectangles de largeur minimale pour entourer l'observation **a**. Une petite valeur du paramètre β diminue la vitesse de changement de la taille des catégories alors qu'une grande valeur permet un changement de taille de plus en plus rapide.

B - Phase de test :

Lors de la phase de test, l'observation présentée au réseau est associée à un nœud de la couche F_2 par la fonction de choix. La classe du champ d'association du nœud gagnant est associée avec l'observation testée, classant ainsi cette observation. Aucun test de vigilance ou de MatchTracking n'est effectué.

1.2 Stratégies d'apprentissage standard et impact du sur-apprentissage

Un réseau de neurones nécessite une phase d'apprentissage par laquelle le réseau construit les liens lui permettant d'effectuer la classification du type de données présentées. Lors de cette phase, le réseau doit posséder un critère d'arrêt d'apprentissage. Si le réseau apprend trop longtemps, un phénomène de sur-apprentissage risque de survenir, entraînant une dégradation des performances en généralisation. Le sur-apprentissage est un phénomène provenant de la sur-spécialisation d'un système pour un problème donné. Ainsi, dans un réseau de neurones, à partir d'un certain nombre d'époques[1] d'entraînement, le système se sur-spécialise par rapport à la base de données d'apprentissage. Ce faisant, il perd sa capacité de généralisation par rapport aux données de test provenant de la même source mais qui n'ont pas encore été traitées par le système. La figure 2 représente ce phénomène.

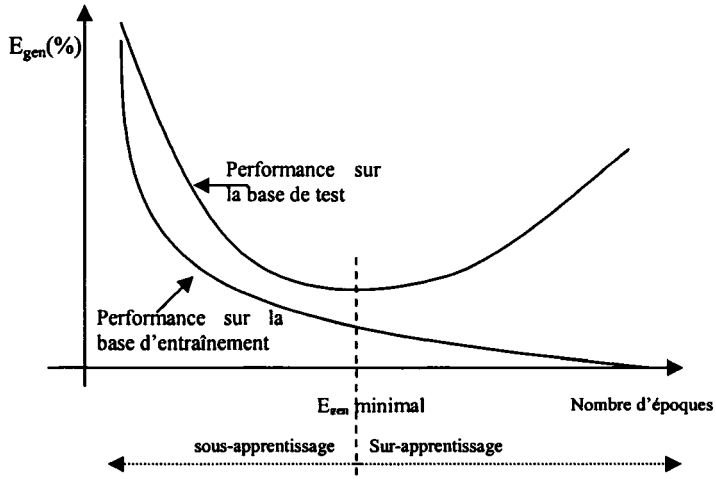

Figure 2 Schématisation de l'erreur en généralisation en fonction du nombre d'époques lors de la phase d'apprentissage d'une classification

[1] Une époque d'entraînement est atteinte lorsque tous les patrons contenus dans la base d'apprentissage ont été présentés au réseau.

Cette figure présente l'erreur en généralisation en fonction du nombre d'époques d'apprentissage. On constate que l'erreur en généralisation sur la base d'apprentissage ne fait que décroître. Ainsi, le réseau de neurones devient meilleur d'époque en époque pour classifier la base d'entraînement. Par contre, à partir d'un certain nombre d'époques, on remarque que l'erreur en généralisation sur la base de test augmente. Cet effet est dû au fait que le réseau s'est sur-spécialisé pour la base d'entraînement et qu'il perd sa capacité de généralisation. Ainsi, il faut arrêter l'entraînement du système à l'époque la plus proche possible de la frontière entre le sous-apprentissage et le sur-apprentissage. Avec les réseaux FAM, le sur-apprentissage se manifeste par une dégradation des performances en généralisation et en ressources. La stratégie d'apprentissage par validation hold-out permet de réduire les effets engendrés par le sur-apprentissage.

Cependant, le phénomène de sur-apprentissage est légèrement controversé dans la communauté du FAM. Boaz Lerner et Boaz Vigdor [18] obtiennent de meilleures performances lors de l'apprentissage par convergence des patrons d'entraînement comparativement à l'utilisation d'une méthode de validation appliquée sur le nombre d'époques d'apprentissage. En d'autres mots, il ressort de leur étude que le nombre d'époques d'entraînement ne crée pas de sur-apprentissage pour le réseau FAM. À l'opposé, deux équipes indépendantes ont démontré que le FAM subissait bel et bien une erreur de sur-apprentissage due au nombre d'époques d'entraînement [22,23,26].

Dans certaines situations, le FAM a tendance à créer plus de catégories que nécessaire [19]. Bref, il perd sa capacité de généralisation à cause de la création d'un trop grand nombre de poids synaptiques à l'intérieur du réseau. Par contre, ce phénomène survient même lors de l'application de la stratégie d'apprentissage validation hold-out, laquelle est immunisée contre la dégradation des performances due au nombre d'époques d'entraînement. Quelques solutions pour régler ce problème ont été proposées dans la communauté scientifique [19,20,21].

Il existe plusieurs stratégies pour indiquer à un réseau l'arrêt de l'apprentissage. Les quatre stratégies les plus courantes sont :

Une époque *(1EP)* : Entraînement du réseau sur une seule époque. Le seul critère d'arrêt de cette méthode est la durée de l'apprentissage, soit une époque. Avant l'apprentissage, l'ordre de présentation des données est généré aléatoirement.

Convergence des poids synaptiques *($CONV_W$)* : Entraînement du réseau avec comme condition d'arrêt la convergence des poids synaptiques. L'ordre de présentation des observations est réparti au hasard entre chaque époque. La convergence des poids synaptiques implique que la somme des différences au carré entre les poids obtenus par deux époques successives soit inférieure à 0,01. Ceci indique que les valeurs des poids obtenues entre deux époques n'ont pratiquement pas changé.

Convergence des patrons d'apprentissage *($CONV_P$)* : Entraînement du réseau avec comme condition d'arrêt la convergence des patrons d'apprentissage. L'ordre de présentation des observations est réparti au hasard entre chaque époque d'entraînement. La convergence des patrons d'apprentissage implique que la phase d'entraînement se termine lorsque toutes les observations de la base d'apprentissage sont parfaitement classées.

Validation hold-out *(HV)* : Pour résoudre le problème de sur-apprentissage, des méthodes de validation croisée ont été proposées [9,10,17]. La technique la plus courante est la méthode de validation hold-out. Cette méthode a été développée à partir des techniques de validation croisée, mais elle n'utilise qu'un seul ensemble de données pour la validation et n'utilise pas de « croisement » lors de la validation. La différence entre la validation croisée et la validation de type hold-out est importante car la validation croisée est supérieure pour des petits ensembles de données [30]. La validation hold-out implique la séparation de la base de données d'apprentissage en deux

parties distinctes et fixes : une base d'apprentissage et une base de validation. La base de validation servira de test après chaque époque d'apprentissage et permettra de sceller l'apprentissage du réseau lors de l'obtention du minimum d'erreur en généralisation (E_{gen} minimal), voir figure 2. Une fois l'apprentissage terminé (convergence des patrons de la base d'apprentissage) le réseau créé à l'époque ayant obtenu le meilleur taux de reconnaissance sur la base de validation est conservé. Sa performance est, par la suite, évaluée sur la base de test. La base de validation permet donc de faire un choix sur le nombre d'époques d'apprentissage et ainsi garantir que la base de test n'est traitée qu'une seule fois par le réseau, soit lors du test final. L'ordre de présentation des observations est réparti au hasard entre chaque époque.

1.3 Stratégie d'apprentissage avec optimisation par essaims particulaires

Cette section présente la stratégie d'apprentissage spécialisée pour les réseaux FAM que nous avons développée. Cette stratégie utilise PSO pour sélectionner la valeur des quatre paramètres internes FAM, soit le paramètre de choix α, de vigilance de base $\bar{\rho}$, de MatchTracking ε et de vitesse d'apprentissage β.

D'autres travaux ont été effectués pour étudier l'impact de certains paramètres. Le paramètre de vigilance a été testé avec diverses valeurs [18] ainsi qu'avec une valeur variable pendant l'entraînement [39,40]. Bien que ces auteurs ont démontré un avantage à utiliser une valeur de vigilance variable, l'effet secondaire de cette approche est une prolifération des catégories diminuant ainsi le taux de compression. Dubarwski [41,42] a utilisé une approche stochastique pour orienter l'apprentissage des réseaux neuronaux. Ses travaux portent sur les paramètres α, β et ρ des réseaux fuzzy ARTMAP. Les résultats obtenus sont raisonnablement comparables à ceux obtenus par un humain sur un problème à deux dimensions.

1.3.1 Optimisation par essaims particulaires

Plusieurs scientifiques ont créé des modèles interprétant le mouvement des vols d'oiseaux et des bancs de poissons. Plus particulièrement, Reynolds [11] et Heppner et al. [12] ont présenté des simulations sur un vol d'oiseaux. Reynolds était intrigué par l'aspect esthétique du déplacement des oiseaux en groupe et Hepnner, un zoologue, était intéressé à comprendre les règles permettant à un grand nombre d'oiseaux de voler en groupe : soit de voler sans se heurter, de changer soudainement de direction, de s'écarter et de se rapprocher de nouveau. Cette étude a grandement inspiré le développement de l'algorithme PSO.

En effet, lors de simulations des modèles mathématiques décrivant les vols d'oiseaux, Wilson [13] a suggéré que ces types de modèles pourraient très bien s'appliquer à la recherche de points caractéristiques dans un espace de recherche. Sa réflexion se base sur le fait que, lors de l'installation d'une mangeoire à oiseaux dans une cour, même si aucun oiseau ne l'a jamais visitée, après quelques heures de patience un grand nombre d'oiseaux viendront y manger. Lors des simulations de Wilson, la volée d'oiseaux cherchait une mangeoire dans un espace donné et finissait par découvrir son emplacement. En utilisant les algorithmes de modélisation de Heppner [12] et de Reynolds [11], et en modifiant le modèle mathématique de Wilson [13], Kennedy et Eberhart [14] ont transformé le tout en un vol d'oiseaux cherchant la « mangeoire » la plus grosse dans un lot de mangeoires contenues dans une région prédéterminée. L'algorithme d'optimisation PSO a ainsi vu le jour.

1.3.1.1 Description algorithmique

L'algorithme PSO est généralement introduit en relatant son développement conceptuel. Tel que mentionné, cet algorithme a vu le jour sous la forme d'une simulation simplifiée d'un milieu social, tel que le déplacement des oiseaux à l'intérieur d'une volée. Pour cet

algorithme une redéfinition des termes est nécessaire; une *population* représente le vol d'oiseaux et un *agent ou particule* représente chaque oiseau de la volée.

Lors de l'initialisation de PSO, chaque particule est positionnée aléatoirement dans l'espace de recherche. Pour définir ce positionnement il faut connaître la plage des valeurs de chaque dimension (paramètres) à optimiser. Puis, la vitesse initiale de chaque particule est aléatoirement choisie. Pour choisir cette vitesse, l'algorithme requiert une vitesse maximale absolue pour chaque dimension. La vitesse peut être négative mais le signe n'indique que le sens dans lequel la particule se déplace.

Il est fortement conseillé d'initialiser la première particule à l'aide d'une valeur connue [15]. Lors de l'optimisation des paramètres dans l'espace de recherche, ceux-ci possèdent généralement des valeurs par défaut. Ces valeurs deviennent le point de départ de la recherche. L'initialisation de la première particule avec de telles valeurs permet d'obtenir, dès la première itération, une valeur de pertinence *(qualité)* de base. Cette valeur peut servir de référence pour fins de comparaison et d'estimation de la performance en optimisation obtenue à l'aide de PSO.

Chaque particule connaît son emplacement (x_i), sa vitesse de déplacement (v_i), la position où elle a obtenu sa meilleure qualité (P_i) ainsi que la position de la meilleure qualité obtenue par l'ensemble des particules (P_g) et ce, à chaque itération (t) de l'algorithme PSO. En tenant compte de ces valeurs, la position et la vitesse de déplacement de chaque particule est mise à jour. La figure 3 présente la mise d'une particule avec l'algorithme PSO.

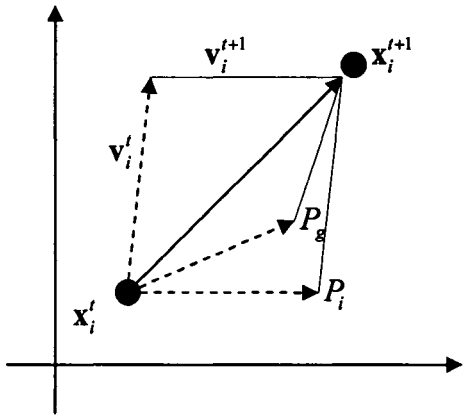

Figure 3 Mise à jour d'une particule avec l'algorithme PSO

Le but de l'algorithme PSO est d'optimiser une fonction continue dans un espace donné. Dans la majorité des cas, l'algorithme d'optimisation recherche le maximum ou le minimum global de l'espace de recherche. Voici la description des étapes de l'algorithme PSO :

Étape 0 :

Si le critère d'arrêt est vérifié, alors l'algorithme se termine. S'il ne l'est pas, une nouvelle itération commence en retournant à l'Étape 1 avec la première particule $(i=1)$.

Le critère d'arrêt correspond généralement à un nombre d'itérations prédéfinies, mais on peut également spécifier un critère d'arrêt en fonction de la meilleure valeur de qualité $G(\overline{p}_g)$ obtenue pour l'ensemble des particules.

Pour toutes les particules de la population, exécuter les Étapes 1 à 5.

Étape 1 :

Calcul de la qualité $G(\overline{\mathbf{x}}_i)$ de la particule i en fonction de son vecteur de position $(\overline{\mathbf{x}}_i)$.

Étape 2 :

Établir si la qualité $G(\overline{\mathbf{x}}_i)$ obtenue par la particule *i* est supérieure à la meilleure qualité que cette particule a obtenue antérieurement. Si $G(\overline{\mathbf{x}}_i) > G(\overline{p}_i)$, la présente position de la particule $\overline{\mathbf{x}}_i$ est sauvegardée comme étant la meilleure position $\overline{p}_i$ obtenue à ce jour pour la particule *i*.

Étape 3 :

Établir si la qualité $G(\overline{p}_i)$ obtenue par la particule *i* est plus grande que la meilleure qualité $G(\overline{p}_g)$ obtenue pour l'ensemble de la population. Si tel est le cas, l'indice de la particule ayant obtenu la meilleure qualité *g* prend la valeur *i*.

Étape 4 :

Mettre à jour la vitesse de déplacement $\overline{\mathbf{v}}_i(t)$ de la particule *i*. Cette mise à jour tient compte de la vitesse précédente de la particule $\overline{\mathbf{v}}_i(t-1)$, de sa position présente $(\overline{\mathbf{x}}_i)$, de la position de la meilleure qualité $\overline{p}_i$ obtenue par cette particule ainsi que de la position de la meilleure qualité globale $\overline{p}_g$ obtenue par la population. De plus, deux paramètres, φ_1 et φ_2, sont utilisés pour ajuster l'importance des termes $(p_{id} - x_{id}(t-1))$ et $(p_{gd} - x_{id}(t-1))$ de l'équation de mise à jour de la vitesse (1.11). Une fois cette vitesse mise à jour, il faut vérifier si la nouvelle vitesse $\overline{\mathbf{v}}_i(t)$ de la particule *i* est contenue dans les limites autorisées $\overline{\mathbf{v}}_i \in (-\mathbf{V}_{max}, +\mathbf{V}_{max})$. Si tel n'est pas le cas, la nouvelle vitesse est réduite à la borne la plus proche.

Étape 5 :

Mettre à jour la position $\overline{\mathbf{x}}_i(t)$ de la particule *i*. Cette mise à jour tient compte de la position précédente de la particule $\overline{\mathbf{x}}_i(t-1)$ ainsi que de la nouvelle vitesse $\overline{\mathbf{v}}_i(t)$ calculée à l'étape 4. Une fois la position de la particule *i* mise à jour, il faut vérifier si la

nouvelle position $\overline{\mathbf{x}}_i(t)$ est contenue dans l'espace de recherche spécifié par $\overline{\mathbf{x}}_i \in (\mathbf{X}_{\min}, \mathbf{X}_{\max})$. Si tel n'est pas le cas, la nouvelle position est ramenée à la borne la plus proche.

L'algorithme 1 présente un sommaire de l'algorithme PSO tel que présenté par J. Kennedy et R. Ebernhart [14,15].

Algorithme 1 : Optimisation par essaims particulaires [14,15]

Initialisation des paramètres $\overline{\mathbf{x}}_i, \overline{\mathbf{v}}_i, \mathbf{V}_{\max}, \mathbf{X}_{\min}, \mathbf{X}_{\max}$

for t = 1 au temps maximum

  for i = 1 au nombre de particules

    if $G(\overline{x}_i) > G(\overline{p}_i)$ //G() évaluer la qualité

      $\overline{\mathbf{p}}_i = \overline{\mathbf{x}}_i$ // p_{id} meilleure position

    EndIf

    g = i // arbitraire

    for j = index des voisins de la particule i

      If $G(\overline{\mathbf{p}}_j) > G(\overline{\mathbf{p}}_g)$ then g = j //index de la meilleure particule globale

    next j

    $\overline{\mathbf{v}}_i(t) = \overline{\mathbf{v}}_i(t-1) + \varphi_1(\overline{\mathbf{p}}_i - \overline{\mathbf{x}}_i(t-1)) + \varphi_2(\overline{\mathbf{p}}_g - \overline{\mathbf{x}}_i(t-1)), \quad \overline{\mathbf{v}}_i \in (-\mathbf{V}_{\max}, +\mathbf{V}_{\max})$

    $\overline{\mathbf{x}}_i(t) = \overline{\mathbf{x}}_i(t-1) + \overline{\mathbf{v}}_i(t), \quad x_i \in (\mathbf{X}_{\min}, \mathbf{X}_{\max})$

  next i

next t, jusqu'à obtention du critère d'arrêt

1.3.1.2 Valeur des paramètres φ_1 et φ_2

L'équation la plus importante dans l'algorithme PSO est la mise à jour de la vitesse de déplacement des particules $\overline{\mathbf{v}}_i(t)$:

$$\overline{\mathbf{v}}_i(t) = \overline{\mathbf{v}}_i(t-1) + \varphi_1(\overline{\mathbf{p}}_i - \overline{\mathbf{x}}_i(t-1)) + \varphi_2(\overline{\mathbf{p}}_g - \overline{\mathbf{x}}_i(t-1)) \quad (1.9)$$

Les valeurs attribuées à φ_1 et φ_2 ont fait l'objet d'une recherche. Après un très grand nombre de simulations et bien des tentatives, Kennedy et Eberhart [14] ont proposé un modèle simple pour ces deux valeurs. Ces deux variables seront d'une valeur aléatoire comprise entre 0 et 2 pour chaque mise à jour de chaque vitesse, et cela pour chaque particule (agent). Bien que plusieurs autres méthodes d'attribution des paramètres φ_1 et φ_2 ont été proposées, ce modèle reste le plus performant [14]. Ainsi, après modifications, la nouvelle formule de mise à jour de la vitesse des particules est :

$$\overline{\mathbf{v}}_i(t) = \overline{\mathbf{v}}_i(t-1) + \varphi_1(\overline{\mathbf{p}}_i - \overline{\mathbf{x}}_i(t-1)) + \varphi_2(\overline{\mathbf{p}}_g - \overline{\mathbf{x}}_i(t-1)) \quad (1.10)$$

Où : $\varphi_1 = 2 \cdot rand()$ et $\varphi_2 = 2 \cdot rand()$

1.3.1.3 Poids inertiel

Shi et Eberhart [27] ont modifié la formule de mise à jour de la vitesse en introduisant la notion de poids inertiel. Le poids inertiel (*w(t)*) est un paramètre utilisé pour équilibrer la force de la vitesse précédente de la particule. La formule de mise à jour de la vitesse d'une particule est maintenant :

$$\overline{\mathbf{v}}_i(t) = \overline{\mathbf{v}}_i(t-1) \cdot w(t) + 2 \cdot rand() \cdot (\overline{\mathbf{p}}_i - \overline{\mathbf{x}}_i(t-1)) + 2 \cdot rand() \cdot (\overline{\mathbf{p}}_g - \overline{\mathbf{x}}_i(t-1)) \quad (1.11)$$

De nombreux tests ont également été effectués pour trouver la valeur optimale de *w(t)* [16]. L'une des meilleures techniques trouvée consiste en une fonction linéaire diminuant la valeur de *w(t)* de 0.9 à 0.4 sur le nombre maximal d'itérations à effectuer. Ainsi, la valeur de *w(t)* diminue légèrement après chaque itération PSO. Au début d'une optimisation, les particules feront de grands déplacements. Ceci permettra d'explorer une grande partie de l'espace. Puis, à mesure que le nombre d'itérations augmente, la

grandeur des déplacements des particules diminuera, permettant ainsi de raffiner la recherche.

1.3.1.4 Nombre de particules

Le choix du nombre de particules utilisées lors de l'optimisation est un autre sujet traité par Kennedy et Eberhart [14]. Bien que le nombre de particules optimum soit variable selon les types de problèmes à optimiser, il est généralement conseillé d'utiliser de 10 à 30 particules lors de l'emploi de l'algorithme PSO. Pour améliorer le résultat de l'optimisation, il est également recommandé de faire quelques cycles PSO lors de l'utilisation d'un petit nombre de particules, car cet algorithme est sensible aux valeurs initiales des particules. On peut diminuer cette sensibilité en utilisant un grand nombre de particules (50 et plus) et ainsi n'exécuter qu'un seul cycle d'optimisation PSO.

1.3.2 Stratégie d'apprentissage spécialisée pour le fuzzy ARTMAP

En utilisant PSO nous avons développé une stratégie d'apprentissage spécialisée pour les réseaux FAM afin d'éliminer la dégradation des performances présente dans ces réseaux.

Nous savons que les quatre paramètres internes des réseaux FAM peuvent influencer les performances de ces réseaux et que la valeur par défaut de ces paramètres est presque toujours utilisée. En employant PSO, nous allons chercher de meilleures valeurs pour les quatre paramètres FAM dans le but d'obtenir de meilleures performances en généralisation. Ainsi, pour chaque problème de classification testé, notre stratégie d'apprentissage spécialisée, utilisant l'algorithme PSO, optimisera les performances en généralisation de FAM à l'aide d'une base de validation dans un espace de recherche de quatre dimensions (paramètre de choix α, de vigilance de base $\bar{\rho}$, de MatchTracking ε et de vitesse d'apprentissage β).

Les quatre stratégies d'apprentissage standard sont implémentées lors du calcul de la qualité des particules. Lors de l'utilisation de la stratégie d'apprentissage standard 1EP pour le calcul de la qualité, on dénotera cette stratégie d'apprentissage spécialisée : PSO(1EP). Les stratégies spécialisées utilisant les autres stratégies d'apprentissage standard sont dénotées: PSO($CONV_P$), PSO($CONV_W$) et PSO(HV). Il existe donc deux critères d'arrêt dans chaque stratégie d'apprentissage spécialisée. L'une des deux conditions d'arrêt provient de la stratégie d'apprentissage standard lors du calcul de la qualité de chaque particule. La seconde condition d'arrêt est propre à l'algorithme d'optimisation PSO et est basée sur le nombre d'itérations de recherche. Si après 10 itérations PSO aucune particule n'a obtenu une meilleure performance en généralisation (calcul de la qualité) que la meilleure obtenue globalement, la recherche est arrêtée. De plus, l'optimisation PSO est limitée à 100 itérations. Une fois la centième itération complétée, l'optimisation PSO s'arrêtera.

Lors de l'entraînement, les valeurs des quatre paramètres internes FAM seront limitées dans un espace de recherche. La méthodologie utilisée pour les stratégies spécialisées pour les réseaux FAM est décrite en détail à la section 2.2.1.