

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abou-bekr Belkaid Tlemcen

Faculté des Sciences

Département Informatique



MEMOIRE

Pour l'obtention du diplôme de :

MAGISTER EN INFORMATIQUE

De l'Ecole Doctorale des systèmes et technologies
d'information et de communication STIC

Intitulé :

Amélioration des Performances des Classifieurs à base de Métaheuristiques

Présenté par :

Mlle. Fatima Bekaddour

Soutenu en Mai 2014 devant le Jury :

Président du jury.	Mr. Lahsaini Mohamed,	MCA, UABB Tlemcen - Algérie,
Examineur.	Mr. Benmaamer Badr,	MCA, UABB Tlemcen - Algérie,
	Mr. Bessaid Abdelhaffidh,	Pr, UABB Tlemcen - Algérie,
Invité.	Mr. Benazzouz Mourtada,	MCB, UABB Tlemcen - Algérie,
Directeur de mémoire.	Mr. Chikh Mohamed Amine,	Pr, UABB Tlemcen - Algérie,

Remerciements

Je remercie ALLAH le Tout-Puissant de m'avoir donné le courage, la volonté et la patience de mener à terme ce présent travail.

Mes remerciements vont particulièrement à Mon encadreur Mr Chikh M^{ed} Amine de l'université Abou-Bekr Belkaid Tlemcen pour m'avoir proposé ce sujet, pour son aide et le temps qu'il m'a consacré, tout en me faisant profiter de sa culture et de sa rigueur scientifique.

Je voudrais aussi remercier Mr Lahsaini Mohamed, maître de conférences à l'université Abou- Bekr Belkaid Tlemcen pour l'intérêt qu'il a porté à ce travail en acceptant d'être président du jury.

Mes remerciements s'adressent à Mr Benmaamer Badr, MCA à l'université de Tlemcen et Mr Benazzouz Mourtada maître de conférence classe B à l'université de Tlemcen pour l'intérêt qu'ils ont porté à ce travail en acceptant d'être examinateurs.

Mes derniers mots s'adressent tout particulièrement à ma famille : mon père, mes frères, et surtout ma mère dont les encouragements et le soutien ont été indispensables à l'aboutissement de mes études, et que sans elle je n'aurai jamais pu réussir.

Je dédie ce travail à la source de tendresse que sont

Mes très chers parents

Tout en étant convaincue que mon succès est une récompense

Pour tous leurs sacrifices, qu'ils trouvent ici l'expression de ma plus

Profonde gratitude.

À mes deux frères Akkacha et Hassen

À la mémoire de mon grand père Bekaddour Ahmed

Et mon professeur Mr Kara Chafik

Que leurs âmes reposent en paix

À toute la famille Bekaddour

À tous mes amis.

À tous ceux qui m'aiment

F. Bekaddour

Résumé

La performance d'une méthode de classification est d'un grand intérêt pour le choix, la comparaison et la validation des algorithmes de classification. La lisibilité des résultats et la réduction du cout d'échec total de la classification sont d'une importance cruciale pour l'amélioration de la performance des classifieurs. Dans ce mémoire de magister, nous proposons principalement deux approches de résolution à travers la description d'un modèle d'optimisation métaheuristique baptisé *ProSadm-HBA* (*ProSadm* : Programmation d'un Système d'Aide au Diagnostic Médical, en conjonction avec la métaheuristique *HBA* : Homogeneity Based-Algorithm) et *F-HBA* (Fuzzy Homogeneity Based-Algorithm). Nous avons validé nos résultats expérimentaux sur des bases de données médicales connues : *Pima* (Diabètes), *TH* (Troubles Hépatiques), *AP* (Appendicite). Les approches développées permettent de minimiser le nombre total d'échecs de la classification (*ProSadm-HBA*) tout en respectant la contrainte d'interprétabilité des classifieurs (*F-HBA*). Ces contributions peuvent être d'un grand intérêt pour les experts dans le domaine médical.

Mots clés: Performances, Sur-Apprentissage, Sur-Généralisation, Métaheuristique, ProSadm, ANFIS, HBA, Optimisation, Interprétabilité.

Abstract

Performance of a classification method is of great interest for choosing, comparing and validation of classification algorithms. The transparency of results and the reduction of the total misclassification cost are of vital importance as well as the improvement of the performance of classifiers. In this magister report, we propose two solving approaches through the description of a meta-heuristic optimization model denoted ProSadm-HBA (ProSadm in conjunction with HBA: Homogeneity Based-Algorithm) and F-HBA (Fuzzy Homogeneity Based-Algorithm) .we validated our experimental results based on three well known medical datasets: *PID* (Pima Indian Diabetes), *LD* (Liver Disorders) and *AP* (Appendicitis) . The developed approaches allow minimizing the total number of misclassification cases (ProSadm-HBA) while respecting the interpretability constraint of classifiers (F-HBA). These contributions can be of great interest for experts in the medical field.

Keywords: Performance, Overfitting, Overgeneralization, Metaheuristic, ProSadm, ANFIS, HBA, Optimization, Interpretability.

ملخص

تعد ماهية أداء طرق التصنيف ذات منفعة كبيرة لاختيار, مقارنة و تقييم خوارزميات التصنيف. تعتبر الشفافية و الحد من الأخطاء الناتجة عن التصنيف ذوا أهمية كبيرة لتحسين أداء طرق التصنيف . في رسالة الماجستير هذه, نقترح كحل, نموذجان أساسيان يعتمدان علي نظام تحسين من خلال استعمال *Meta-heuristic* و قد أسميناها *ProSadm - HBA* & (*F-HBA*). لقد قمنا بتقييم النتائج المتحصل عليها وذلك عبر استعمال قواعد بيانات طبية معروفة : السكري, اضطرابات الكبد و التهاب الزائدة الدودية . إن النماذج المطورة تسمح بتقليص أخطاء التصنيف (*ProSadm-HBA*) مع احترام شفافية أنظمة التصنيف (*F-HBA*) . يمكن لمثل هذه المساهمات أن تكون ذات فائدة كبيرة للخبراء في المجال الطبي.

الكلمات المفتاحية : الفعالية , ANFIS , ProSadm, Metaheuristic, Overgeneralization, Overfitting, تحسين , شفافية .

Table des matières

Remercîment	i
Dédicace	ii
Résumé	iii
Table des matières	iv
Liste des tableaux	viii
Liste des figures	xi
Liste des algorithmes	xiv
Liste des abréviations	xv
Introduction Générale	1
1- État de l'art	7

Chapitre I : Généralités sur les métaheuristiques

I.1 Introduction	9
I.2 Théorie de complexité.....	9
I.3 Optimisation.....	10
I.3.1 Principes d'optimum.....	10
I.3.2 Problèmes de décision et problèmes d'optimisation.....	11
I.3.3 Optimisation combinatoire.....	11
I.3.4 Les différents problèmes d'optimisation.....	12
I.3.5 Les différentes méthodes d'optimisation.....	13
I.3.5.1 Méthodes exactes.....	13
I.3.5.2 Méthodes approchées.....	14
I.4 Heuristique et métaheuristique	14
I.5 Intérêt de la métaheuristique.....	15
I.6 Principes du métaheuristique.....	15
I.7 Utilisation d'une métaheuristique.....	16
I.8 Propriétés des métaheuristiques.....	16
I.9 Principales métaheuristiques.....	17
I.9.1 Métaheuristiques de trajectoire.....	17
I.9.2 Métaheuristiques de population	20
I.9.2.1 Introduction.....	20
I.9.2.2 Algorithmes génétiques.....	20
I.9.2.3 Stratégies d'évolution.....	24
I.9.2.4 Programmation évolutionniste.....	25
I.10 Classification des métaheuristiques.....	25
I.11 Exploitation vs Exploitation	27
I.12 Frameworks et bibliothèques pour les métaheuristiques.....	28
I.13 Evaluation des métaheuristiques.....	29

Table des matières

I.14 Conclusion.....	30
----------------------	----

Chapitre II : Problème de Performance des Classifieurs

II.1 Introduction	32
II.2 Performance des classifieurs.....	33
II.3 Pré-Requis.....	33
II.3.1 Notations.....	33
II.3.2 Performance et efficacité.....	34
II.3.3 Mesure de la performance.....	35
II.4 Problème de la Performance des Classifieurs.....	37
II.4.1 Définition du PPC.....	37
II.4.1.1 Qualités des données.....	38
II.4.1.2 Choix des paramètres.....	39
II.4.1.3 Choix de modèle.....	40
II.4.1.4 Dimensionnalité des données.....	41
II.4.2 Issues de la performance.....	42
II.4.2.1 Surapprentissage & Surgénéralisation.....	42
II.4.2.2 Dilemme biais-variance.....	44
II.4.2.3 Surapprentissage/Surgénéralisation, Biais/Variance & Optima locaux..	44
II.4.3 Méthodes existantes pour la résolution du PPC.....	45
II.4.3.1 Arbre de décision (A.D).....	45
II.4.3.2 Approche K plus proches voisins (Kppv).....	47
II.4.3.3 Réseaux de neurones artificiels (R.N.A).....	48
II.5 Présentation des travaux selon les méthodes de classification.....	51
II.6 Bilan et synthèse.....	53

2- Approches proposées.....	54
------------------------------------	-----------

Chapitre III : Méthodes & Approches Adoptées

III.1 Introduction	56
III.2 Partie -I- Le système <i>ProSadm</i>	
III.2.1 Description du système	58
III.2.2 Aperçu du système	59
III.2.3 Méthodes utilisées.....	60
III.2.3.1 Apprentissage dans les réseaux de neurones.....	60
III.2.3.2 Quelques types des réseaux de neurones.....	61
III.2.3.3 Le perceptron multicouche.....	61
III.2.3.4 Quantification vectorielle Adaptative.....	64
III.3 Partie -II- La <i>logique Floue</i> & modèles <i>Neuro-Flous</i>	
III.3.1 Système d'inférence flou.....	67
III.3.2 Principe de fonctionnement du SIF.....	68

Table des matières

III.3.2.1 Fuzzification.....	68
III.3.2.2 Inférence floue.....	68
III.3.2.3 Défuzzification.....	68
III.3.3 Types des SIF.....	69
III.3.3.1 Système flou de Mandani	69
III.3.3.2 Système flou de Takagi-Sugeno.....	69
III.3.7 Vers les systèmes Neuro-Flous.....	70
III.3.8 Définitions des modèles Neuro-Flous.....	71
III.3.9 Types des systèmes neuro-flous.....	71
III.3.9.1 Système neuro-flou coopérative	71
III.3.9.2 Système neuro-flou concurrent.....	72
III.3.9.3 Système neuro-flou hybride.....	72
III.3.10 Architectures Neuro-Flous.....	72
III.3.11 ANFIS : un outil hybride pour le diagnostic.....	72
III.3.12 Intérêt de l'ANFIS.....	73
III.3.13 Architecture de l'ANFIS.....	73
III.3.14 Apprentissage de l'ANFIS	75
III.4 Partie -III- La métaheuristique <i>A.B.H</i>	
III.4.1 Introduction.....	78
III.4.2 Description de l'A.B.H.....	78
III.4.2.1 Quelques notions d'observations.....	78
III.4.2.2 Estimation non paramétrique.....	81
III.4.3 Processus de l'algorithme A.B.H.....	83
III.4.4 Résolution des sous-problèmes.....	85
III.4.4.1 Résoudre le sous-problème 2.....	85
III.4.4.2 Résoudre le sous-problème 3.....	87
III.4.4.3 Résoudre le sous-problème 5.....	88
III.4.4.4 Résoudre le sous-problème 6.....	91
III.5 Conclusion.....	93

Chapitre IV : Les modèles *ProSadm-HBA* & *F-HBA*

IV.1 Introduction.....	95
IV.2 Description du problème.....	95
IV.2.1 Définitions de base.....	95
IV.2.2 Définition du problème.....	96
IV.2.3 Objective.....	97
IV.3 Description des bases de données.....	97
IV.3.1 Base de données du diabète PID (Pima Indian Diabetes).....	98
IV.3.1.1 Contexte du diabète.....	98
IV.3.1.2 Description de la base de données de diabète.....	99
IV.3.1.3 Travaux menés sur la base de diabète.....	100

Table des matières

IV.3.2 Base de données du Troubles hépatiques (Liver disorders).....	100
IV.3.2.1 Contexte du Troubles Hépatites	100
IV.3.2.2 Description de la base du Troubles hépatiques	101
IV.3.2.3 Travaux menés sur la base du Troubles hépatiques.....	101
IV.3.3 Base de données d'Appendicite	102
IV.3.3.1 Contexte de l'Appendicite	102
IV.3.3.2 Description de la base de données d'Appendicite	102
IV.3.3.3 Travaux menés sur la base d'Appendicite.....	103
IV.4 Critères d'évaluation.....	103
IV.5 Le modèle ProSadm-HBA.....	104
IV.5.1 Architecture du modèle.....	104
IV.5.2 Implémentation & choix des paramètres.....	104
IV.5.3 Aperçue sur l'application.....	105
IV.6 Le modèle F-HBA.....	107
IV.6.1 Architecture du modèle.....	107
IV.6.2 Implémentation & choix des paramètres.....	107
IV.6.3 Aperçue sur l'application.....	109
IV.7 Conclusion.....	112

Chapitre V : Validation Expérimentale

V.1 Introduction.....	114
V.2 Méthodologie expérimentale.....	114
V.3 Résultats et analyse expérimentales.....	114
V.3.1 Considération 1.....	115
V.3.1.1 Analyse des performances de classification	115
V.3.1.2 Analyse des règles floues	116
V.3.2 Considération 2.....	128
V.3.2.1 Analyse des performances de classification	129
V.3.2.2 Analyse des règles floues	129
V.3.3 Considération 3.....	135
V.3.3.1 Analyse des performances de classification	136
V.3.3.2 Analyse des règles floues	137
V.4 Etude comparatif.....	141
V.4.1 Avec Métaheuristiques VS Sans Métaheuristiques.....	141
V.4.2 Sensibilité et Spécificité.....	141
V.4.3 Comparaison avec les résultats de la littérature.....	142
V.5 Conclusion.....	144
Conclusion Générale & perspectives.....	145
Annexes.....	149
Références Bibliographiques.....	160

Liste des Tableaux :

Chapitre II

Tab II.1 Matrice de confusion.....	35
Tab II.2 Principaux travaux pour la résolution du <i>PPC</i> dans certaines méthodes de classification.....	52

Chapitre III

Partie I :

Tab III.1 : Exemples de fonctions utilisées	59
--	----

Partie II :

Tab III.2 : La classification des paramètres du SIF.....	67
Tab III.3 : Comparaison entre la logique floue et les réseaux de neurones.....	70
Tab III.4 : Les paramètres à ajuster pour l'ANFIS.....	75

Chapitre IV

Tab IV.1 : Caractéristiques des bases de données utilisées.....	98
Tab IV.2 : Description des attributs de la base PID (Diabète).....	100
Tab IV.3 : Description des attributs de la base T.H (Troubles hépatiques).....	101
Tab IV.4 : Description des attributs de la base AP (Appendicite).....	102
Tab IV.5 : Matrice de confusion.....	103
Tab IV.6 : Configuration du modèle Neuro-Flou pour la reconnaissance du cancer de sein.....	107

Chapitre V

Tab V.1 : Résultats de minimisation pour $CT = \min (Taux_{FP} + Taux_{FN})$	115
Tab V.2 : Valeurs finales des paramètres (α^- , α^+) et (β^- , β^+) pour les bases de données médicales PID, T.H, A.P (Considération n°1).....	116
Tab V.3 : Degré de sollicitation pour les règles des cas FP	118
Tab V.4 : 1 ^{ère} Exemple des cas de FP	118
Tab V.5 : 2 ^{ème} Exemple des cas de FP	118
Tab V.6 : Degré de sollicitation pour les règles des cas Nc	118
Tab V.7 : 1 ^{ère} Exemple et 2 ^{ème} Exemple des cas Nc	119
Tab V.8 : Degré de sollicitation pour les règles des cas Vn	119
Tab V.9 : Exemple des cas de VN	120
Tab V.10 : Degré de sollicitation pour les règles des cas FN	120
Tab V.11 : Exemple des cas de FN	120

Tab V.12 : Degré de sollicitation pour les règles des cas Nc	121
Tab V.13 : 1 ^{ère} Exemple des cas de Nc (non classifié).....	121
Tab V.14 : 2 ^{ème} Exemple des cas de Nc (non classifié).....	121
Tab V.15 : Degré de sollicitation pour les règles des cas VP	122
Tab V.16 : 1 ^{ère} Exemple et 2 ^{ème} Exemple des cas de Vp	122
Tab V.17 : Degré de sollicitation pour les règles des cas FP	122
Tab V.18 : 1 ^{ère} Exemple des cas de FP	123
Tab V.19 : 2 ^{ème} Exemple des cas de FP	123
Tab V.20 : Degré de sollicitation pour les règles des cas Nc	123
Tab V.21 : 1 ^{ère} Exemple des cas Nc	124
Tab V.22 : 2 ^{ème} Exemple des cas Nc	124
Tab V.23 : Degré de sollicitation pour les règles des cas VP	124
Tab V.24 : Exemple des cas de VP	124
Tab V.25 : Degré de sollicitation pour les règles des cas VN	125
Tab V.26 : Exemple des cas de VN	125
Tab V.27 : Degré de sollicitation pour les règles des cas Nc (non classifiés).....	125
Tab V.28 : 1 ^{ère} Exemple des cas de Nc (non classifié).....	126
Tab V.29 : 2 ^{ème} Exemple des cas de Nc (non classifié).....	126
Tab V.30 : Degré de sollicitation pour les règles des cas FN	126
Tab V.31 : Exemple des cas de FN	126
Tab V.32 : Degré de sollicitation pour les règles des cas Nc	127
Tab V.33 : Exemple des cas de Nc	127
Tab V.34 : Degré de sollicitation pour les règles des cas Nc	127
Tab V.35 : Exemple des cas de Nc	128
Tab V.36 : Résultats de minimisation pour $CT = \min (Taux_{FP} + 20Taux_{FN} + 3 Taux_{NC})$	128
Tab V.37 : Valeurs finales des paramètres (α^-, α^+) et (β^-, β^+) pour les bases de données médicales PID, T.H, A.P (Considération n° 2).....	129
Tab V.38 : Exemple des cas de VP	130
Tab V.39 : Exemple des cas de VP	130
Tab V.40 : Degré de sollicitation pour les règles des cas VN	130
Tab V.41 : Exemple des cas de VN	131
Tab V.42 : Degré de sollicitation pour les règles des cas Fn	131
Tab V.43 : Exemple des cas de FN	131
Tab V.44 : Degré de sollicitation pour les règles des cas NC	132
Tab V.45 : 1 ^{ère} Exemple des cas de Nc (non classifié).....	132
Tab V.46 : 2 ^{ème} Exemple des cas de Nc (non classifié).....	132
Tab V.47 : Degré de sollicitation pour les règles des cas VP	133
Tab V.48 : Exemple des cas de VP	133
Tab V.49 : Degré de sollicitation pour les règles des cas VN	133
Tab V.50 : Exemple des cas de VN	134
Tab V.51 : Degré de sollicitation pour les règles des cas FN	134
Tab V.52 : Exemple des cas de FN	134
Tab V.53 : Degré de sollicitation pour les règles des cas Nc	135
Tab V.54 : Exemple des cas de Nc	135

Tab V.55 : Résultats de minimisation pour $CT = \min(\text{Taux}_{FP} + 100\text{Taux}_{FN} + 3\text{Taux}_{Nc})$.	136
Tab V.56 : Valeurs finales des paramètres (α^-, α^+) et (β^-, β^+) pour les bases de données médicales PID, T.H, A.P (Considération n° 3).....	137
Tab V.57 : Degré de sollicitation pour les règles des cas VP	137
Tab V.58 Exemple des cas de VP	137
Tab V.59 : Degré de sollicitation pour les règles des cas VN	138
Tab V.60 : Exemple des cas de VN	138
Tab V.61 : Degré de sollicitation pour les règles des cas FN	138
Tab V.62 : Exemple des cas de FN	139
Tab V.63 : Degré de sollicitation pour les règles des cas FP	139
Tab V.64 : Exemple des cas de FP	139
Tab V.65 : Degré de sollicitation pour les règles des cas Nc	140
Tab V.66 : Exemple des cas de Nc	140
Tab V.67 : Résultats expérimentaux des bases de données utilisées.....	141
Tab V.68 : Évaluation des approches proposées sur les bases des données du Diabète Troubles Hépatiques et Appendicite.....	142
Tab V.69 : Performance de classification pour la base de données de diabète PID.....	143
Tab V.70 : Performance de classification pour la base de données TH.....	143
Tab V.71 : Performance de classification pour la base de données d'appendicite.....	144

Liste des figures :

Chapitre I

Figure I.1 : Classification des complexités des problèmes.....	10
Figure 1.2 : Optimum local et global de f (deux-dimonsion).....	11
Figure 1.3 : Classification des problèmes d'optimisation	13
Figure 1.4 : Classification des méthodes d'optimisation	13
Figure 1.5 : Arbre de recherche Backtracking	14
Figure 1.6 : Evolution d'une solution dans la méthode de descente (Hill Climbing)..	18
Figure 1.7 : Mécanisme pour s'échapper d'optimums locaux	19
Figure 1.8 : Liste Taboue	19
Figure 1.9 : Les concepts principaux utilisés dans les algorithmes génétiques.....	21
Figure 1.10 : Opérateurs de croisement & de mutation en représentation binaire	21
Figure I.11 : Probabilités de sélection d'une solution.....	23
Figure I.12 : Une solution domine la population avec sa fonction Fitness.....	23
Figure 1.13 : Taxonomie des métaheuristiques à base des analogies de la nature	25
Figure I.14 : Taxonomie des métaheuristiques à base de la fonction Objectif.....	26
Figure I.15 : Taxonomie des métaheuristiques à base de la structure de voisinage.....	26
Figure I.16 : Taxonomie des métaheuristiques selon [Hertz 2003].....	26
Figure I.17 : Taxonomie des métaheuristiques selon le type de décision.....	27
Figure I.18 : Intensification vs Diversification	27

Chapitre II

Figure II.1 : Structure d'un réseau de neurone (feed-forward).....	34
Figure II.2 : Processus globale de la classification.....	37
Figure II.3 : Principales étapes dans le prétraitement des données.....	38
Figure II.4 : Illustration de la méthode de validation croisée (5-Fold).....	41
Figure II.5 : Sous-apprentissage /Fit et Sur-apprentissage.....	42
Figure II.6 : Phénomènes du sur-apprentissage et sur-généralisation.....	43
Figure II.7 : Représentation symbolique du dilemme biais-variance.....	44
Figure II.8 : Illustration portant sur le compromis biais / variance.....	45
Figure II.9 : Schéma d'une classification par l'approche Kppv.....	48
Figure II.10 : Courbes d'erreurs d'apprentissage et de validation.....	49
Figure II.11 : Courbe réelle d'erreur de validation.....	49
Figure II.12 : Évolution de la moyenne des carrés des poids en fonction du nombre d'itérations de l'algorithme d'apprentissage.....	50

Chapitre III

Partie I :

Figure IV.10 : Réseau Neuro-Flou développé pour la base PID (Diabète).....	108
Figure IV.11 : Réseau Neuro-Flou développé pour la base TH (Troubles Hépatiques).....	108
Figure IV.12 : Réseau Neuro-Flou développé pour la base AP (Appendicite).....	109
Figure IV.13 : Fenêtre Principale du modèle F-HBA	109
Figure IV.14 : La barre de menu.....	110
Figure IV.15 : La barre d'outils.....	110
Figure IV.16 : Fonctions d'appartenances Avant (a gauche) & Après (a droite) pour les bases de données PID ⁽¹⁾ , TH ⁽²⁾ , AP ⁽³⁾ respectivement.....	111

Chapitre V

Figure V.1 : Schéma représentatif des fonctions objectives utilisées.....	115
Figure V.2 : Meilleure Performances de classification des méthodes proposées.....	142

Annexes

Figure B.1 : Schéma d'un arbre de décision.....	152
Figure B.2 : (a) Arbre de décision originale,(b) A.D après post-élagage (REP, PEP) ..	153
Figure C.1 : Architecture de FALCON.....	155
Figure C.2 : Architecture de NEFCON.....	156
Figure C.3 : Exemple illustratif d'un chromosome composé de 4 gènes.....	159
Figure C.4 : Exemple illustratif de la fonction de croisement.....	159

Liste des algorithmes :

Chapitre I

Algorithme I.1 : Hill Climbing.....	18
Algorithme I.2 : Recherche Tabou.....	19
Algorithme I.3 : Modèle commun des algorithmes évolutionnistes.....	20
Algorithme I.4 : Modèle algorithmique abstraite pour les métaheuristiques.....	28

Chapitre II

Algorithme II.1 : Post-élagage indirecte.....	47
--	----

Chapitre III

Partie III

Algorithme III.1 : ABH (Algorithme a Base d'Homogénéité)	84
Algorithme III.2 : Résoudre le Sous-Problème 2	86
Algorithme III.3 : Résoudre le Sous-Problème 3	88

Annexes

Algorithme B.1 : K-plus proches voisins.....	153
Algorithme B.2 : K-plus proches voisins Flou.....	154
Algorithme C.1 : Expansion radial.....	158

Liste des abréviations :

ABH : Algorithme s Base d'Homogénéité
AD : Arbre de décision
AG : Algorithme Génétique
ANFIS : Adaptative Neuro-Fzzy Inference Système
AP : Appendicite(Appendicitis)
 C_{FN} : Cout de pénalité pour le Faux Négatif
 C_{FP} : Cout de pénalité pour le Faux Positif
 C_{NC} : Cout de pénalité pour les cas Non-Classifiés
 C_T : Cout total d'échec de classification
FALCON : Fuzzy Adaptative Learning Control Network
F-HBA : Fuzzy Homogeneity Based Algorithm
FN : Faux Négatif
FP : Faux Positif
FUN : Fuzzy Net
FUN : Fuzzy Net
GA : Genetic Algorithm
HBA : Homogeneity Based-Algorithm
I.A : Intelligence Artificielle
k-NN : k-Nearest Neighbor
Kppv : K- plus proches voisins
L.D : Liver disorders
L.F : Logique Floue
L.V.Q : Learning Vector Quantization
Nc : non-classifié
NEFCON: Neuro-Fuzzy Control
PID : Pima Indian Diabets
P.M.C: Perceptron Multi-Couches
PPC : Problème de la Performance des Classifieurs
PROSADM: Programmation d'un Système d'Aide au Diagnostic Médical
ProSadm-HBA : ProSadm en conjonction avec la métaheuristique HBA
QVA : Quantification Vectorielle Adaptative
R.N.A : Réseau de Neurone Artificiel
Se : Sensibilité
SIF : Système d'Inférence Flou
Sp : Spécificité
Tc : Taux de Classification
 T_{FP} : Taux du Faux Positif
 T_{FN} : Taux du Faux Négatif
TH : Troubles Hépatiques
 T_{NC} : Taux des cas Non-Classifiés
T.S.K : Takagi-Sogeno-Kang
UCI : University of California, Irvine

VN : Vrai Négatif

VP : Vrai Positif

VPN : Valeur Prédictive Négatif

VPP : Valeur Prédictive Positif

Introduction Générale

Beaucoup d'applications en Data-Mining , concernent l'analyse de données où nous connaissons leurs classes a priori .En général, les modèles inférés a partir des données d'apprentissage sont utilisés par la suite dans la reconnaissance de nouvelles données. Cette approche en Data-Mining est appelée *classification* ou *prédiction de la classe* .

Une tâche de *classification* (ou discrimination) consiste à classer un ensemble d'objets, ou les attribuer à chaque classe (ou catégorie) parmi plusieurs classes définies a l'avance. Un algorithme qui réalise automatiquement cette classification est appelé *classifieur*. [Cornuéjols 2003] .Le rôle d'un classifieur est de déterminer, parmi un ensemble fini de classes, à laquelle appartient un objet donné.

Beaucoup d'études de la classification se terminent souvent par un tableau récapitulatif qui présente les résultats de performance de l'application de diverses approches de Data-Mining sur un ensemble de données différente. Aucune méthode ne surpasse toutes les autres méthodes, tout le temps. En outre, la performance d'une méthode de classification en terme des taux de : faux-positifs (F_P), faux-négatifs (F_N) peuvent être totalement imprévisible. Si le modèle permet aux nouvelles données d'être considérés comme des cas non classifiés, alors il est possible pour les deux taux d'erreur précédents (F_P , F_N) d'être très faibles, mais ,en même temps , le taux d'avoir des nouveaux exemples non-classifiés peuvent être très élevé .

L'optimisation et l'amélioration de la performance des classifieurs occupent actuellement une place grandissante pour la résolution des problèmes complexes. Dans notre cadre, améliorer la performance revient à résoudre les problèmes liées à la classification que ce soit en qualité de données ou en paramètre algorithmiques / non-algorithmique.

L'optimisation couvre un large éventail de techniques certaines datant de plusieurs années, et est utilisée dans de nombreux domaines. Elle peut être définie comme la recherche d'une solution optimale, parmi un ensemble de solutions candidates, pour un problème donné. Les problèmes d'optimisation peuvent être combinatoires (discrets) ou à variables continues, avec un seul ou plusieurs objectifs (optimisation mono ou multi-objectif), avec ou sans contraintes. Cette liste n'est pas exhaustive i.e un problème peut être, par exemple, à la fois continu et mono-objectif.

Néanmoins, l'optimisation d'erreurs de l'apprentissage et de la généralisation des classifieurs et plus précisément le développement d'un modèle autonome qui réalise un compromis entre la performance d'apprentissage et la qualité de la généralisation apparaît comme un problème à part entière avec ses difficultés propres jouant un rôle prépondérant dans la minimisation d'erreur et l'amélioration des performances des classifieurs. Ce problème est connu dans la littérature par les phénomènes sur-apprentissage / sur-généralisation.

Une amélioration de la performance des classifieurs sera désignée dans la suite par l'acronyme *PPC* : " **P**roblème de **P**erformance des **C**lassifieurs ".

Il s'agit d'une balance séparable des modèles en apprentissage ou en généralisation qui peut conduire à une diminution de la performance des classifieurs, consistant à augmenter le nombre total des erreurs de la classification. L'objectif donc est de réduire ces erreurs.

Dans ce contexte du *PPC*, nous traitons aussi le problème de la perte de l'interprétabilité qui diminue la lisibilité et la compréhensibilité des systèmes, un problème à éviter dans le domaine médical.

En résumé, la problématique définie ci-dessus s'énonce par : Comment développer un système capable de modéliser, résoudre, améliorer et optimiser la performance d'un classifieur dans une classification supervisée?

Ainsi décrit, le problème de classification est considéré comme un problème NP. Une solution optimale du problème d'amélioration de la performance des classifieurs (*PPC*) est un plan d'actions de tenir compte de la qualité des données, c'est-à-dire constitué d'une série de prétraitement des données de la classification, et tenant compte aussi des paramètres de l'approche de classification que ce soit algorithmique ou non (structurelle). Dans le cadre de ce mémoire de magister, nous discutons le deuxième critère (*Paramètres algorithmiques/non-algorithmiques de l'approche de classification utilisée*) affectant la performance des classifieurs. D'après les travaux de recherche qui lui ont été consacrés, nous remarquons que l'origine du problème crucial citée au-dessus est de rechercher un compromis entre le sur-apprentissage et la sur-généralisation d'une approche de classification donnée. Le sur-apprentissage (Overfitting) signifie que le modèle extrait décrit bien le comportement des données connu mais présente une faible prédiction pour de nouvelles points de données. La sur-généralisation (Overgeneralization) se produit lorsque le système utilise les données disponibles puis tente d'analyser de grandes quantités de données qui n'ont pas servi à sa modélisation [**Pham 2007**]

Nous avons constaté que ces travaux utilisent pour la plupart des modèles ignorants les cas d'exemples non-classifiés ou associe des coûts de pénalités identiques pour les erreurs de la classification (F_P (faux-positif), F_N (faux-négatif), et N_C (des cas non-classifier)). Nous retrouvons aussi des méthodes qui s'intéressent à augmenter la performance de la classification par un control séparable entre le fitting et la généralisation. Ces approches s'avèrent limitées face à la complexité de ce problème.

Étant donné ces multiples aspects, il serait intéressant d'aborder en profondeur cette problématique dans ce mémoire de magister. Nous aurons recours à trois notions fondamentales : Le système ProSadm (*Programation d'un Système d'Aide au Diagnostic Médical*), le système flou (*Neuro-Flou*) et la méta-heuristique.

D'une part :

- Le système *ProSadm* connaît un certain succès et encouragement, basé sur des techniques dite intelligentes et constitue notre ancien contribution proposée pour le diagnostic médical (cancer du sein) .Ce système semble adaptée à de nombreux problèmes en raison de leur capacité à aborder les systèmes complexes .
- Le système Neuro-Flou (*ANFIS : Adaptive Neuro-Fuzzy Inference System*), constitue une hybridation de la logique floue et les réseaux de neurones pour exploiter la richesse des deux approches .Ce système ne nécessite pas d'expertise préalable, présente un grand choix des paramètres et un temps de convergence rapide [Patnaik 2012], [Hosseini 2011].

D'autre part :

- Les méta-heuristiques permettent de fournir des solutions réalisables de qualité acceptable souvent très proches de l'optimal et déterminées en un temps raisonnable (complexité polynomiale). Elles sont situées à mi-chemin entre les méthodes exactes qui se limitent à la difficulté de la résolution des problèmes réels tels que la classification ; et les heuristiques dont l'inconvénient relève de leur sensibilité aux résolutions des problèmes particuliers, vu qu'elles s'appuient sur les propres spécificités du problème étudié.

Objectifs

Notre travail consiste alors à associer les concepts de systèmes *ProSadm*, *Neuro-Flou* et de l'optimisation métaheuristique, et ce en proposant deux modèles d'optimisation métaheuristique nommés : *ProSadm-HBA* (le système *ProSadm* en conjonction avec l'algorithme *HBA (Homogeneity Based-Algorithm)* et *F-HBA : (Fuzzy Homogeneity Based Algorithm)*, qui allient d'une part, les caractéristiques d'apprentissage et d'interprétabilité des systèmes *ProSadm* et *ANFIS* respectivement et d'autre part, la mise en place d'une certaine "optimalité" supportée par les méta-heuristiques. Nous pensons que la combinaison des deux notions : *intelligence* et *métaheuristique* qui, séparément, fournissaient de bons résultats, représente une potentialité d'amélioration intéressante.

Dans un contexte bien déterminé, selon des motivations et des enjeux que nous allons définir, et considérant des hypothèses de travail que nous allons décrire, les modèles *ProSadm-HBA* et *F-HBA* doivent optimiser les performances de la classification des méthodes classiques (sans conjonction avec la métaheuristique *ABH : Algorithme a Base d'Homogénéité*) en minimisant le nombre total d'échec de classification et en tenant compte des cas non-classifiées et les coûts de penalties inégales associés aux erreurs de classification F_P (faux-positif), F_N (faux-négatif), et N_C (des cas non-classifier) ainsi que la lisibilité du classifieur.

Organisation du Mémoire

L'organisation de ce mémoire de magister suit une progression ordonnée. Ce document se décompose en deux parties correspondant au cheminement de notre démarche.

Partie 1 : État de l'art

La première partie expose notre contexte de travail dans deux chapitres. Le premier chapitre présente les méta-heuristiques, et définit les différentes terminologies et les concepts intrinsèques. Le deuxième chapitre survole, en premier lieu la littérature des problèmes de classification rencontrés et découlant l'importance et la logistique de la performance d'un classifieur.

En deuxième lieu, ce chapitre se concentre sur le problème étudié : le *PPC* (Problème de Performance des Classifieurs) en faisant un tour d'horizon des problèmes issus et les méthodes de résolution qui lui sont dédiées pour certaines approches de classification. Cela permet de souligner la complexité, l'importance et l'ouverture caractérisant le *PPC*.

Cette partie nous permet ainsi de nous positionner par rapport à notre problématique et de spécifier nos orientations et nos choix serviront de fondements à notre proposition.

Partie 2 : Approches proposées

La deuxième partie sera dédiée aux approches de résolution que nous proposons pour l'optimisation de la performance des classifieurs. Elle s'articule autour de trois chapitres : le premier chapitre présente d'abord les concepts clés de notre réflexion : le système *ProSadm* (**P**rogrammation d'un **S**ystème d'**A**ide au **D**iagnostic **M**édical), Le système d'inférence flou & L' *ANFIS* (**A**daptative **N**euro-**F**uzzy **I**nference **S**ystem) et la méthode méta-heuristique *ABH* (**A**lgorithme a **B**ase d'**H**omogénéité), afin de définir quelques notions que nous retrouvons au fil de ce mémoire de magister et de justifier des choix que nous adoptons dans notre proposition.

Le deuxième chapitre présente notre proposition constituée de deux contributions : d'abord, une première contribution à travers laquelle nous avons cherché à exploiter notre ancien système **ProSadm** et que nous l'avons nommé **ProSadm-HBA**. Ce travail a été étendu par l'enfoncement des performances du classifieur à travers l'introduction d'une approche d'optimisation.

Dans la première contribution, **ProSadm-HBA** a donné de bons résultats. Cependant ces résultats s'avère limité (boite noire) et restent toujours non interprétables, la chose qui n'est pas exigée dans le domaine médical. Pour pallier ce problème, nous présentons dans la deuxième contribution un modèle Flou (Neuro-Flou)

d'optimisation reposant sur la logique flou et la méthode d'optimisation méta-heuristique. Ce modèle est appelé **F-HBA** (*Fuzzy Homogeneity Based-Algorithm*). Ce deuxième chapitre analyse en détail l'architecture, l'implémentation des deux applications (**ProSadm-HBA** et **F-HBA**) et le jeu de données utilisées.

Le troisième chapitre valide nos approches : *ProSadm* et le modèle *Flou* (Neuro-Fou) sans et avec utilisation de méta-heuristique, en procédant dans un premier temps, à présenter la méthodologie expérimentale envisagée pour une étude détaillée des résultats obtenus. Ensuite, à exposer une analyse détaillée de nos expérimentations qui sont validées sur différents jeux de données. Enfin, à présenter une étude comparative de nos expérimentations.

Dans une conclusion générale, nous repositionnons l'ensemble de nos développements au regard de l'objectif initial de l'étude : " *Amélioration de la performance des classifieurs à base de métaheuristique* ". Nous résumons les principaux résultats de notre travail et nous abordons enfin une discussion sur les perspectives de travail qui découlent de cette mémoire de magister.

Première partie

État de l'art

Chapitre I :

Généralités sur les Métaheuristiques

Ce premier chapitre introduit les concepts de base liés à la complexité, l'optimisation et les métaheuristiques.

1.1 Introduction

Ce premier chapitre a pour but d'introduire les pré-requis nécessaires à la bonne compréhension du présent mémoire. Ainsi, dans un premier temps, les termes liés à l'*optimisation* et la *complexité* des problèmes sont définis. En outre, une étude d'état de l'art des problèmes et méthodes d'optimisation (*exactes*, *métaheuristiques*) pour résoudre le problème étudié doit être accomplie.

Dans un deuxième temps, nous donnons un aperçu sur les principales métaheuristiques en s'appuyant sur les caractéristiques et les classifications fondamentales des métaheuristiques.

Au moment de la conception des métaheuristiques, deux critères essentiels doivent être tenus en compte : l'*exploitation* et l'*exploration*. Par la suite, la relation entre ces deux concepts sera clarifiée. Nous présentons aussi quelques Frameworks qui ont été introduits à fin de faciliter l'implémentation des métaheuristiques. En fin, l'évaluation des performances des métaheuristiques reste la dernière étape dans le processus de résolution du problème d'optimisation.

I.2 Théorie de complexité

La théorie de la complexité [Arora 09] [Papadimitriou 03] s'intéresse à l'étude formelle de la difficulté des problèmes en informatique. En fait, plusieurs algorithmes peuvent être utilisés pour résoudre un seul problème donné. La question qui se pose est de trouver la méthode la plus performante. Une des approches utilisées pour les comparer, calcule le temps mis par un algorithme pour s'exécuter sur une machine. Cette approche dépend principalement des caractéristiques du matériel utilisé. Par conséquent, nous devons considérer une approche indépendante de toute machine ou tout langage de programmation. Pour cela une notation mathématique pratique a été introduite : la notation *grand-O*. Nous traitons dans cette section uniquement la complexité des problèmes décidables. La notion d'algorithme a été formalisée par l'introduction des *machines de Turing*. Nous présentons ci-dessous quelques définitions concernant la notion de complexité :

Un problème est dit *décidable* s'il existe un algorithme, une procédure qui termine en un nombre fini d'étapes, qui le décide (qui réponde par oui ou par non à la question posée par le problème). S'il n'existe pas de tels algorithmes, le problème est dit *indécidable*. [Talbi 09]

- *Complexité des algorithmes* :

Un algorithme a besoin de deux ressources fondamentales à fin de résoudre un problème posé : *Temps* et *Espace*. La notation *grand-O* (*big O*) sert à exprimer généralement la complexité d'un algorithme. Ceci est mesurée en termes d'analyse de pire des cas. [Talbi 09]

Nous savons que la complexité *temporelle* d'un algorithme est le nombre d'opérations élémentaires nécessaires pour résoudre un problème de taille n de données. Ainsi, la complexité *spatiale* d'un algorithme mesure l'espace mémoire nécessaire pour

exécuter un algorithme. Un algorithme est dit *polynomiale* si sa complexité est en $O(p(n))$. $P(n)$ définit comme suit : [Talbi 09]

$$P(n) = a_k \cdot n^k + \dots + a_j \cdot n^j + \dots + a_1 \cdot n + a_0. \quad [\text{Avec } (a_k > 0) \ \& \ (a_j \geq 0, \forall 1 \leq j \leq k-1)]$$

Un algorithme est dit *exponentielle* si sa complexité est en $O(c^n)$ [c est un nombre réel strictement supérieur à 1]. [Talbi 09]

- Complexité des problèmes :

Il existe principalement deux classes de problèmes les plus représentatives P (problème polynomial) et NP (Problème Non-déterministe Polynomial) [Sipser 92] [Fortnow 09]. (Voir figure I.1)

Un problème est dit *polynomial* s'il peut être résolu par une machine déterministe en un temps polynomial (voir Annexe A). Exemples des problèmes polynomiaux : problème de plus court, tri de tableaux... Un problème appartenant à la classe P est considéré comme un problème *facile* (easy) . [Talbi 09]

La classe NP regroupe l'ensemble des problèmes qui peuvent être résolu par une machine non déterministe en un temps polynomiale. (Voir Annexe A). [Talbi 09]

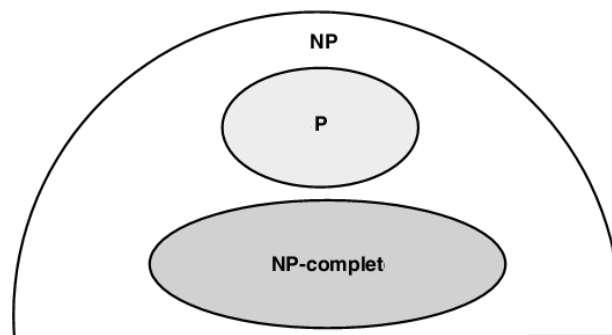


Figure I.1 Classification des complexités des problèmes [Talbi 09]

Une troisième classe de problèmes existe, et est particulièrement importante, il s'agit des problèmes *NP-complets* [Karp 72] [Johnson 85]. Un problème $D \in NP$ est dit *NP-complet* (sous la réduction de Karp) si tout problème de NP se réduit à D (Voir Annexe A). Exemple des problèmes NP-Complet : problème SAT : satisfaisabilité (satisfiability problem).

I.3 Optimisation

I.3.1 Principes d'Optimum :

D'après [Andreasson 2005] , le terme « optimisation » ,trouve son origine dans les mots latin « optimum » qui signifie « l'idéal ultime » et « optimus » qui veut dire «le meilleur».

On considère indifféremment des problèmes d'optimisation de minimisation et de maximisation. La **figure I.2** illustre une distinction entre un optimum local et global. Un optimum global est un optimum de tout le domaine X , tandis qu'un optimum local est un sous-ensemble de X . [Weise 2009]

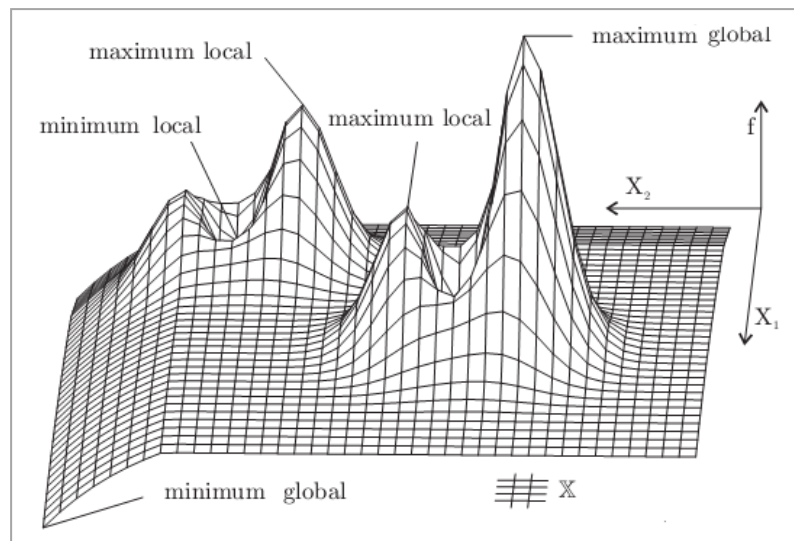


Figure I.2 : Optimum local et global de f (deux-dimension) .[Weise 2009]

I.3.2 Problèmes de décision et problèmes d'optimisation :

La théorie de complexité cherche à connaître la difficulté de répondre à un problème algorithmique. On distingue deux types de problèmes :

-Un problème de décision: C'est un problème dont la solution est formulé en termes *oui/non*.

-Un problème d'optimisation. Le but est de trouver une solution maximisant (resp. minimisant) une fonction objective donnée. A chaque problème d'optimisation on peut associer un problème de décision, dont le but est de déterminer s'il existe une solution pour laquelle la fonction objectif soit supérieure (resp. inférieure) ou égale à une valeur donnée.

Exemple: Etant donnée un graphe $G=(X,E)$.

- *Problème de décision:* existe-t-il un chemin de longueur $\leq L$?
- *Problème d'optimisation :* Trouver un chemin de longueur minimale.

I.3.3 Optimisation combinatoire :

L'optimisation combinatoire est une branche relativement récente, liée en générale à la théorie des graphes, l'informatique théorique et la recherche opérationnelle [Korte 2002]. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation [PAPADIMITRIOU 82], et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire [Hao 1999]. Un problème d'optimisation combinatoire, peut être formulé comme suit :

Un *problème d'optimisation combinatoire*, inspiré de [Sakarovitch,1984]. Soit S l'ensemble des solutions possibles d'un problème. Soit $X \subseteq S$ l'ensemble des solutions admissibles vérifiant un ensemble de contraintes C . Soit $f: X \rightarrow \mathbb{R}$, une fonction que l'on nomme fonction objectif. Un problème d'optimisation se définit par :

$$\begin{cases} \min f(x) \\ x \in X \end{cases}$$

Lorsque l'ensemble S des solutions est discret, on parle de problème d'optimisation combinatoire.

I.3.4 Les différents problèmes d'optimisation

Il existe plusieurs classes d'optimisation, dont nous citons les plus connues (**figure I.3**) :

- **Optimisation Monovariabe et Multivariable :**

Les problèmes d'optimisation mono-variable désignent habituellement tous les problèmes dont la fonction objectif ne dépend que d'une seule variable. Les problèmes d'optimisation multivariable rassemblent les problèmes dont la fonction objectif est définie par plus d'une variable.

- **Optimisation Uni-modale et Multimodale :**

Un problème dont f ne compte qu'un seul optimum est appelé problème *uni-modale*. Dans ce cas, un optimum local est aussi un optimum global. A l'inverse, Lorsqu'une fonction admet plusieurs optima *locaux*, elle est dite *multimodale*

- **Optimisation Mono Objectif et Multi Objectif :**

Les problèmes d'optimisation mono-objectif regroupe les problèmes n'ayant qu'un objectif unique à optimiser. Les problèmes d'optimisation multi-objectif possèdent plus d'un objectif, a optimiser.

- **Optimisation continue et discrète:**

A la différence de l'optimisation *continue* qui recherche une solution dans un ensemble infini d'objets (ex : l'ensemble des nombres réels) , l'optimisation *discrète* regroupe les problèmes dont la solution recherchée est une combinaison d'éléments parmi un ensemble fini d'objets.

- **Optimisation avec et sans contraintes :**

Les problèmes d'optimisation sans contraintes sont des problèmes pour lesquels toute solution s appartenant à S est considérée comme une solution faisable et acceptable. Les problèmes d'optimisation avec contraintes sont des problèmes dont une solution s appartenant à S ne peut être considérée comme solution faisable que si elle satisfait l'ensemble des contraintes prédéfinie.

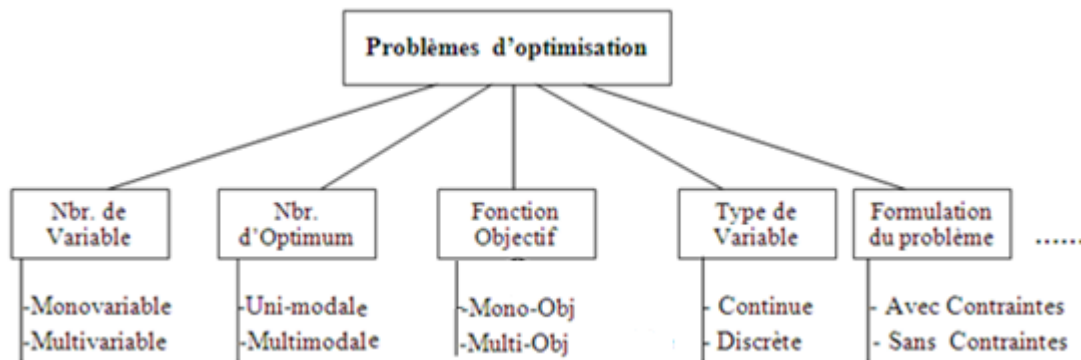


Figure I.3 Classification des problèmes d'optimisation

I.3.5 Les différentes méthodes d'optimisation

Deux types de méthodes sont souvent mis en opposition, dans la résolution des problèmes d'optimisation (**Figure 1.4**) : les méthodes exactes et les méthodes approchées. Les méthodes exactes ont l'avantage d'obtenir des solutions dont l'optimalité est garantie. Par contre, pour les problèmes NP-complets, ces algorithmes sont très coûteuses en temps d'exécution (algorithmes à temps non polynomial). Les méthodes approchées génèrent des solutions de haute qualité en un temps de calcul raisonnable, mais il n'existe aucune garantie de trouver la solution optimale [Talbi 2009].

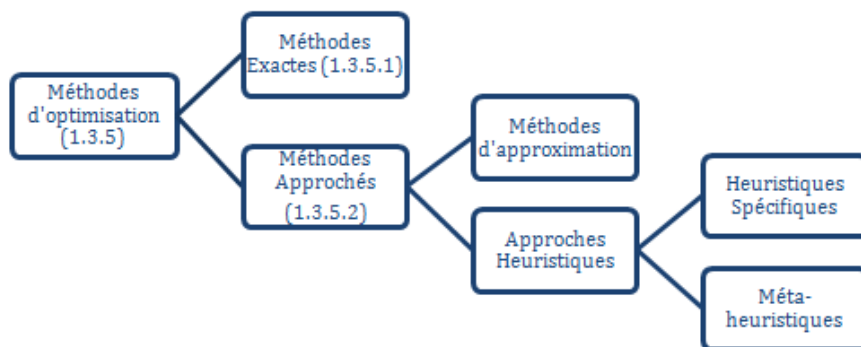


Figure I.4 : Classification des méthodes d'optimisation [Talbi 09]

I.3.5.1 Méthodes exactes :

Le terme de méthodes exactes [Dudzinski 87] regroupe l'ensemble des méthodes permettant d'obtenir la solution optimale d'un problème. Elles s'opposent aux heuristiques, qui permettent d'obtenir une solution approchée. A nos jours, il existe des outils logiciels (ex: **CPLEX**¹, **Lindo**², **XPRESS**³), permettant de résoudre l'ensemble des problèmes d'optimisation. Parmi les méthodes exactes les plus connues, nous citons par exemple la méthode de *backtracking*, (aussi appelées méthodes de *retour sur trace*).

a)- Le *backtracking* (BT) est un algorithme de recherche systématique. Le principe de cette méthode consiste à instancier à chaque pas, une nouvelle variable x_i à une valeur de son domaine. Seules les contraintes portant uniquement sur des variables déjà instanciées sont testées. Si la nouvelle instanciation partielle ainsi obtenue n'est pas une

¹ IBM ILOG CPLEX Optimization Studio home page ; [13]

² www.lindo.com ;

³ <http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx>

solution partielle, on effectue un backtrack (retour en arrière) sur x_i . Autrement dit, x_i est instanciée à une autre valeur de son domaine, au cas où il en existe au moins une. Dans le cas contraire, si S (solution(s)) est vide le problème est insatisfiable, sinon on effectue un backtrack sur la dernière variable instanciée de S . Le processus se poursuit jusqu'à ce qu'une solution complète soit construite.

La **figure 1.5** représente l'arbre de recherche du backtracking appliquée au problème des 4 reines.

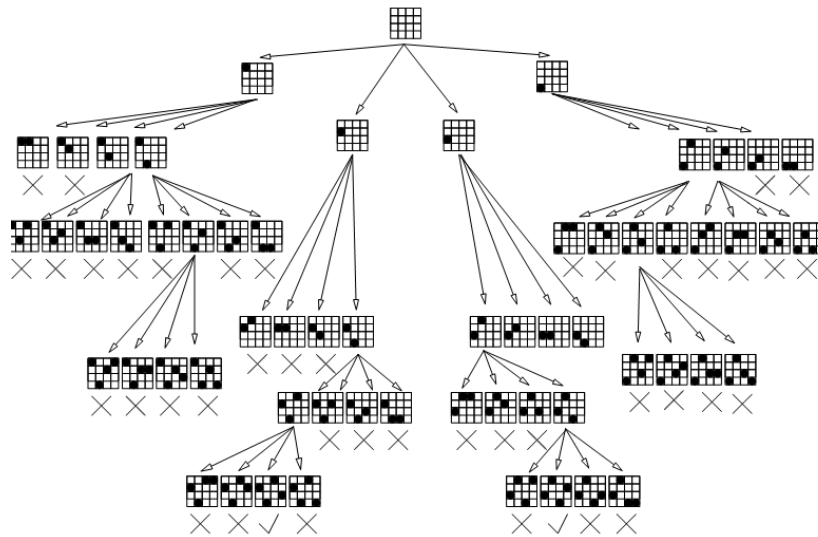


Figure 1.5 Arbre de recherche Backtracking

Le problème des n -reines consiste à placer n reines sur un échiquier $(n \times n)$ de sorte qu'aucune reine ne soit en prise avec une autre. Ceci revient à placer au plus une reine par ligne, par colonne et par diagonale.

I.3.5.2 Méthodes approchées :

Pour ces méthodes, on abandonne la contrainte de trouver une solution optimale et on cherche un algorithme polynomial qui trouve une solution, quelque fois optimale souvent acceptable. Ces méthodes approchées permettent néanmoins de trouver une bonne solution en un temps de calcul raisonnable comparées aux méthodes exactes.

Ces méthodes approchées sont divisées en deux sous-catégories : les algorithmes d'approximation et les heuristiques. Contrairement aux heuristiques, les algorithmes d'approximation garantissent la limite (bound) de la solution trouvée par rapport à l'optimal. Enfin, il existe encore deux sous-classes de méthodes heuristiques : les heuristiques spécifiques à un problème donné, et les méta-heuristiques. La section suivante présente la différence entre ces deux sous-classes.

I.4 Heuristique & Métaheuristique

Avant l'adoption du mot métaheuristique, on parlait plutôt d'heuristique (du verbe grec heuriskein, qui signifie « trouver »). Une heuristique d'optimisation peut être définie comme une méthode approchée rapide et adaptée à un problème donné.

Comme l'heuristique, et à la différence des algorithmes d'optimisation exactes, une métaheuristique n'offre pas de garantie d'optimalité des solutions obtenues. En plus, elle ne précise pas si ses solutions obtenues sont proche de l'optimum ou pas [Talbi 09]. La différence réside dans le fait qu'une heuristique vise à résoudre un problème particulier, en s'appuyant sur ses propres spécificités.

1.5 Intérêt de la Métaheuristique

En général, les approches exactes fournissent une solution optimale au problème traité, mais elles restent généralement très coûteuses en temps d'exécution pour des problèmes de grande taille. A l'inverse, les méthodes approchées ne donnent aucune garantie de trouver la solution optimale, mais elles visent à générer des solutions de haute qualité en un temps de calcul raisonnable.

En particulier, lors de l'apparition des premières métaheuristiques, beaucoup de chercheurs se sont lancés dans l'utilisation et le développement de ces méthodes, ce qui a conduit à une avancée importante pour la résolution pratique des problèmes complexes. De nos jours, les métaheuristiques sont devenues de plus en plus populaire dans les différents domaines de la recherche. L'un des indicateurs de cette situation est l'augmentation importante du nombre des conférences portant sur la conception et l'application des métaheuristiques [Talbi 09]

[Hao 1999] résume les principales avantages des métaheuristiques :

- Généralité et application possible à une large classe de problèmes,
- Simplicité et rapidité.
- Efficacité pour de nombreux problèmes,
- Possibilité de compromis entre qualité des solutions et temps de calcul.

1.6 Principes du Métaheuristique

Les premières métaheuristiques datent des années 80. Au cours des dernières années, les métaheuristiques ont reçu de plus en plus de popularité. A l'inverse des méthodes exactes, les métaheuristiques permettent de résoudre des problèmes complexes de grande taille en délivrant des solutions satisfaisantes dans un temps de calcul raisonnable [Talbi 09].

Le mot Métaheuristique, introduit pour la première fois par *Glover* dans son article [Glover 1986]. Il est dérivé de la composition de deux mots grecs:

- 'heuristique' qui vient du verbe *heuriskein* ($\epsilon\upsilon\pi\iota\sigma\kappa\epsilon\iota\nu$) et qui signifie 'trouver'.
- Le terme *meta* qui signifie "supérieure, de haut niveau" .

D'après Blum et Roli [Blum 03] il n'existe actuellement aucune définition communément acceptée pour les métaheuristiques. Dans cette section, nous présentons quelques définitions des métaheuristiques.

En 1996, I.H. Osman et G. Laporte [Osman 96], définissaient la métaheuristique comme un *processus itératif qui guide une heuristique subordonnée en combinant*

intelligemment différents concepts d'exploration et d'exploitation de l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales .

En 1999, S. Voß et Roucairol [Voß 99] ont défini les métaheuristiques comme *un processus itératif maître qui guide et modifie des heuristiques subordonnées dans le but d'efficacement produire des solutions de haute qualité. Une métaheuristique peut manipuler une ou plusieurs solutions complètes(ou incomplètes) à chaque itération. Les heuristiques subordonnées peuvent être des procédures de haut(ou bas) niveau, ou de simple recherche locale ou juste des méthodes de construction.*

En 2013, Le réseau Metaheuristics (metaheuristics.org) définit les métaheuristiques comme *un ensemble de concepts qui peuvent être utilisés pour définir des méthodes heuristiques appliquées à un ensemble de problèmes de nature différente. En d'autres termes une métaheuristique peut être vue comme un modèle algorithmique général pouvant être appliqué à différents problèmes d'optimisation moyennant des modifications relativement faibles.*

1.7 Utilisation d'une métaheuristique

La complexité d'un problème d'optimisation donne une indication sur la difficulté de ce problème .Aussi le temps de recherche pour la résolution des problèmes d'optimisation est un issue important pour la sélection d'un algorithme d'optimisation. D'après [Talbi 09] , l'utilisation des métaheuristiques est justifiée par les principales propriétés suivantes caractérisant les problèmes d'optimisations :

- Un problème facile (classe **P**) avec un nombre d'instances élevées.
- Un problème facile (classe **P**) avec des contraintes temps-réel dures.
- Un problème difficile (classe **NP-dure**) avec des instances de taille modérée et/ou structure complexe.
- Problèmes d'optimisation où l'évaluation des contraintes et/ou de la fonction(s) objective sont coûteux en temps de calcul.

1.8 Propriétés des métaheuristiques

Dans [Blum 03], les auteurs résumant les propriétés fondamentales attachées à la notion de métaheuristique :

- Les métaheuristiques sont des stratégies qui guident le processus de recherche.
- L'objectif des métaheuristiques est d'explorer efficacement l'espace de recherche afin de trouver des solutions (proche de) optimale.
- Les techniques mises en œuvre dans les métaheuristiques vont de la simple recherche locale à des procédures complexes d'apprentissage.
- Les métaheuristiques sont des algorithmes approximatifs et généralement de nature non-déterministe.
- Elles peuvent intégrer des mécanismes évitant d'être piégé dans une zone de l'espace de recherche.

- Les métaheuristiques sont décrites suivant un niveau d'abstraction indépendant du problème spécifique à traiter.
- Les métaheuristiques peuvent utiliser des connaissances spécifiques au problème sous la forme des heuristiques contrôlées à un niveau supérieur.
- Les métaheuristiques les plus modernes introduisent des mécanismes pour adapter et guider la recherche.

Même si Les métaheuristiques n'ont pas de définition bien précise, il est possible de les regrouper en classes selon plusieurs critères .

I.9 Principales métaheuristiques

Dans cette section, nous donnons un aperçu des principales méta-heuristiques . Il existe principalement deux types de méta-heuristiques : Métaheuristiques de Trajectoire et métaheuristiques à base de population.

I.9.1 Métaheuristiques de Trajectoire

La première classe de métaheuristiques regroupe les « Métaheuristiques de Trajectoire ». Le principe de ces méthodes consiste à faire évoluer une solution initiale en s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif en tentant de se diriger vers des solutions optimales (ex : Hill climbing, recherche Tabou).

- Hill Climbing (HC):

La méthode Hill Climbing (aussi appelé Par escalade, La recherche par descente), est un algorithme d'optimisation et de recherche assez ancienne. [Weise 2009]. C'est une métaheuristique à base de solution unique très utilisé, due a sa simplicité et sa facilité de mis en place [Luke 2013]. Le but est d'améliorer la qualité d'une solution par exploration de son voisinage. Cette technique s'applique à un seul point (s) a la fois (point courant) de l'espace de recherche. Durant chaque itération, un nouveau point s' est sélectionné parmi les voisins du point courant. - Si $f(s')$ est meilleur que $f(s)$, alors le nouveau point (s') devient le nouveau point courant . - Sinon : un autre voisin est sélectionnée et testée par rapport au point courant. La méthode termine si aucune amélioration n'est possible, ou atteint un temps bien définie [Michalewicz 2004] La *figure I.6* schématise ce processus qui est aussi détaillé par l'**algorithme 1.1**.

Algorithme 1.1 l'Algorithme Hill Climbing

```

- // Initialisation
- Générer une solution initial  $s_0$ .
-  $s \leftarrow s_0$ .
- Tant que (critère d'arrêt n'est pas satisfait) faire
  Générer  $V(s)$ .
  Evaluer  $V(s)$ .
  Si (pas de meilleur voisin trouvé) alors
  | Stop.
  // Choisir le meilleur voisin  $s' \in V(s)$ .
   $s \leftarrow s'$ 
Retourner solution obtenue.

```

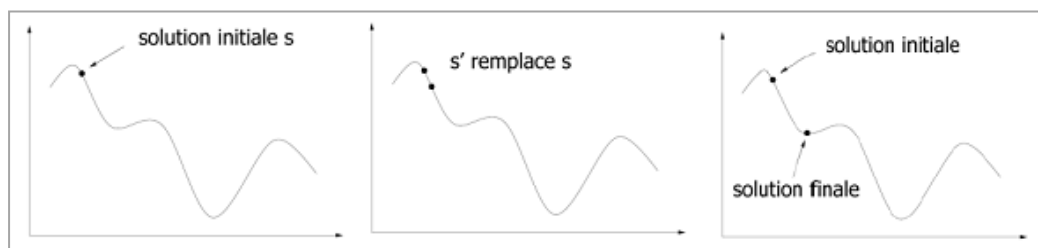


Figure 1.6 Evolution d'une solution dans la méthode de descente (Hill Climbing)
[Braune 10]

- Il existe plusieurs stratégies de sélection du voisin : [Talbi 2009]
 - Choix du premier voisin qui améliore la solution courante (*first improvement*).
 - Choix du meilleur voisin qui améliore la solution courante (*best improvement*).
 - Sélection aléatoire appliquée aux voisins améliorant la solution courante (*Random selection*).

Les inconvénients majeurs de cette méthode sont: [Michalewicz 2004]

- L'algorithme termine généralement avec un optimum local.
- Manque d'information qui ne permet pas de savoir si un optimum local trouvé est proche d'un optimum global.
- L'optimum obtenu dépend de la configuration initiale.

Une des versions qui améliore l'algorithme HC, consiste à faire des redémarrages lorsqu'un optimum local est trouvé, en repartant d'une nouvelle solution initiale générée aléatoirement. (Random-Restart Hill Climbing).

- Recherche Tabou : R.T (Tabu Search) :

La recherche Tabou a été introduite par Fred Glover en 1986 [Glover 1986], [Glover 89i], [Glover 89ii] pour permettre à la méthode de descente de s'échapper des optima locaux. L'origine du nom "Recherche tabou (Tabu search)" provient de la structure de mémoire qui déclare à chaque itération un ensemble de solutions dite tabou à fin d'éviter un comportement cyclique de la méthode. Le principe de cette méthode consiste (comme la méthode par descente) à examiner les solutions voisines de la

solution courante et choisit le meilleur (non tabou) de celle-ci. Cependant, lorsqu'il n'y a plus de meilleures solutions dans le voisinage, le meilleur voisin remplace la solution courante même si celui-ci est moins bon puis la recherche recommence.

La recherche Tabou repose sur l'utilisation de la mémoire (pour s'échapper d'un optimum local) sous la forme d'une liste appelée liste tabou (de longueur limitée l) où les solutions les plus récemment visités sont mémorisés. La liste est mise à jour chaque itération et la solution visitée durant l'itération actuelle est introduite selon la politique *FIFO* (first in first out). (Voir **figure I.8**). [Braune 10]. La méthode s'arrête soit après un nombre fixé d'itérations, soit après un nombre fixé d'étapes n'ayant pas amélioré la solution.

L'**algorithme 1.2** montre une structure algorithmique générale de la recherche Tabou. Les symboles s et s' représente la solution courante et prochaine respectivement. T dénote la liste Tabou. N est l'ensemble des voisins de la solution courante s . [Braune 10]

Algorithme 1.2 Recherche Tabou

- Créer une solution initiale s ;
 - Initialiser la liste Tabou T ;
 - **Tant que** (critère d'arrêt n'est pas satisfait) **faire**
 - Déterminer l'ensemble des voisins N de la solution courante s ;
 - Sélectionner la meilleure solution non-tabou s' de N .
 - Basculer vers la solution s' ($s \leftarrow s'$).
 - Mettre à jour la liste T .
 - Mettre à jour la meilleure solution trouvée (si nécessaire);
 - Fin**
 - **Retourner** la meilleure solution trouvée.
-

La **figure I.7** illustre le comportement de l'espace des solutions d'un problème de minimisation. L'algorithme intensifie la recherche à partir d'une solution initiale (s_1), jusqu'à atteindre le premier optimum local (s_5). Pour s'échapper de cet optimum local, l'algorithme poursuit la recherche sur la nouvelle s_6 (s_4 est une solution taboue). (De même pour s_7). Dans ce cas, le processus de recherche entre dans une phase de diversification qui se poursuit jusqu'à solution s_9 , suivit par une phase d'intensification. [Braune 10]

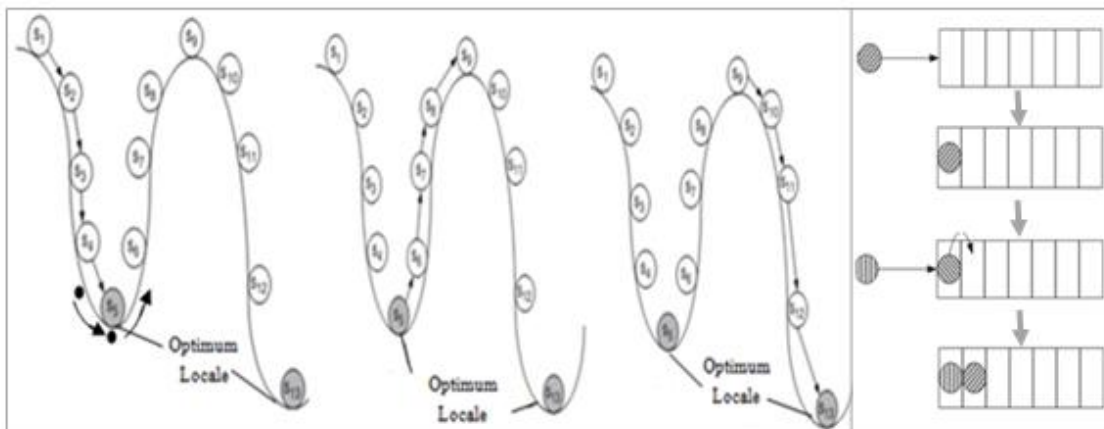


Figure 1.7 Mécanisme pour s'échapper d'optimums locaux [Braune 10] [19]

Figure 1.8 liste Taboue durant la recherche [Braune 10]

I.9.2 Métaheuristiques de Population

I.9.2.1 Introduction

Contrairement aux méthodes précédentes, les approches à base de population consistent à travailler avec un ensemble de solutions simultanément, que l'on fait évoluer graduellement. Ceci permet d'améliorer l'exploration de l'espace de recherche. Dans cette seconde catégorie, on recense : Les algorithmes évolutionnistes (Evolutionary Algorithms) qui forment une classe importante des métaheuristiques à base de population.

En 1859, Darwin a présenté la théorie d'évolution dans son livre : *On the Origin of Species*. En 1980, ces théories de la création de nouvelles espèces et leur évolution ont inspiré les informaticiens à concevoir des algorithmes évolutionnistes. Plusieurs algorithmes évolutionnistes ont évolué de façon indépendante au cours des 40 dernières années : *Algorithmes génétiques* (AG : développé par Holland 1970, USA), *Stratégies d'évolution* (Rechenberg 1964 : Germany), *Programmation évolutionniste* (Fogel, 1966, USA). Chacune de ces algorithmes constituent des approches différentes ; cependant, ils sont tous inspiré des mêmes principes d'évolution naturelle. Les approches évolutionnaires s'appuient sur un modèle commun présenté par l'**algorithme 1.3**. [Talbi 09].

Algorithme 1.3 Modèle commun des algorithmes évolutionnistes

```

- Générer ( $P(0)$ ) // Population initiale
-  $t = 0$ ;
- Tant que (critère d'arrêt n'est pas satisfait) faire
    Evaluer ( $P(t)$ ).
     $P'(t)$  = Sélection ( $P(t)$ );
     $P''(t)$  = Reproduction ( $P'(t)$ );
    Evaluer ( $P''(t)$ );
     $P'(t+1)$  = Remplacement ( $P(t), P''(t)$ );
     $t = t + 1$ ;
Retourner Meilleur individu ou meilleur population trouvée.

```

I.9.2.2 Algorithmes génétiques :

Les algorithmes génétiques sont des méthodes de recherche stochastiques et des techniques d'optimisation inspirés par l'évolution naturelle (mutation, croisement, ...) pour générer de meilleures solutions à travers des générations. Ils sont initialement conçus pour résoudre des problèmes d'optimisation à variables discrètes. Ensuite, ils ont été appliqués à l'apprentissage automatique dans les années 1980. [Talbi 09].

Dans un premier temps, une *population* de points de recherche initiales appelé «individus» est générée. Chaque individu est un codage ou une *représentation* (binaire, entiers, ...) pour une *solution* candidate d'un problème donné (de même qu'un chromosome est formé d'une chaîne de gènes) (voir **figure I.9.a**). Dans chaque *chromosome*, les gènes sont les variables du problème et leurs valeurs possibles sont appelées *allèles* (alleles). (Voir **figure I.9.b**). La fonction de codage associée à chaque

phénotype (la solution du problème réel) une représentation de génotype. Le *génotype* est utilisé au cours de l'étape de *reproduction* de l'algorithme alors que le phénotype est nécessaire pour l'évaluation du coût d'un individu. (Voir **figure I.9.c**). [Talbi 09]

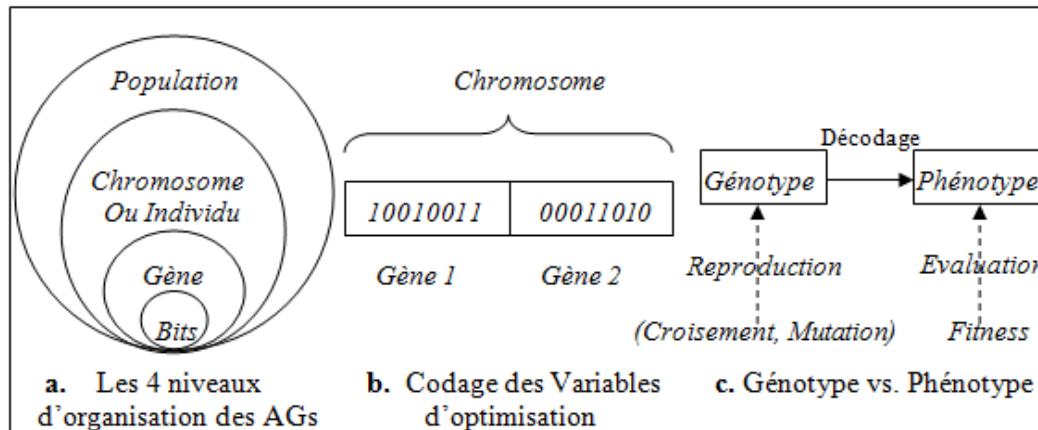


Figure I.9: Les concepts principaux utilisés dans les algorithmes génétiques [Talbi 09]

Une fonction d'évaluation est utilisée pour associer une mesure de «fitness» pour chaque solution indiquant sa qualité. Au cours de l'évolution, selon le principe darwinienne de la survie du plus fort, les meilleurs individus (fittest individuals) sont sélectionnés pour la reproduction selon différentes stratégies de sélection (par exemple élitiste, sélection par roulette.....).

Ensuite, les opérateurs de reproduction (ou transformation) : *croisement* et *mutation* (voir **Figure I.10**) sont appliqués sur chaque couple des individus sélectionnés pour générer de nouveaux individus (*enfants (offspring)*). Les stratégies de remplacement sont utilisées pour produire la prochaine génération des meilleurs individus (fitter individuals). Ce processus est répété Jusqu'à atteindre un critère d'arrêt (ex: après *m* générations ou un délai donné).

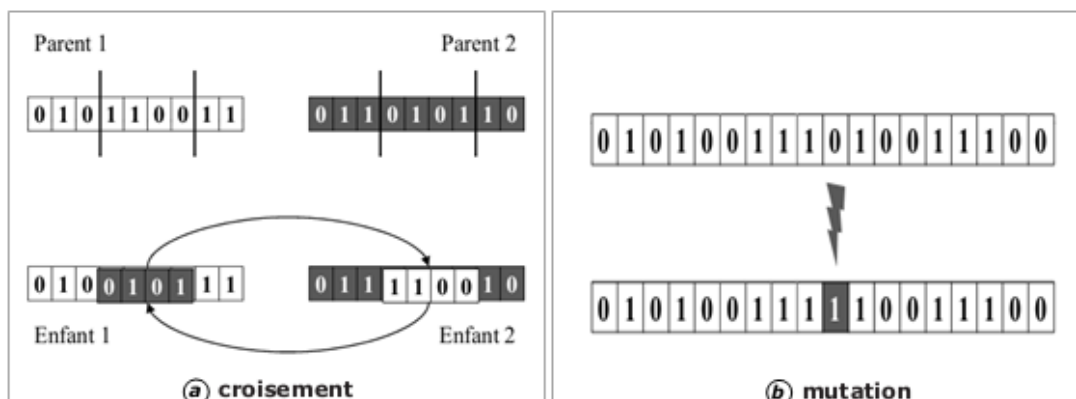


Figure I.10: Opérateurs de croisement & de mutation en représentation binaire [Talbi 09]

Initialisation de la Population :

Deux questions découlent lors d'initialisation de la population : d'abord, quelle est la taille optimale de la population et comment initialiser (qualité de la population) les

individus de la première génération $P(0)$? Aucune de ces questions n'a de réponse théorique bien établie.

La taille de la population, généralement fixe au cours du déroulement de l'algorithme et notée N , est fréquemment de l'ordre de plusieurs centaines d'individus dans les algorithmes génétiques. En effet, les justifications théoriques de ces algorithmes indiquent qu'il faut des populations très grandes pour que l'algorithme fonctionne optimalement. Cela est aussi souhaitable pour conserver une bonne diversité à la population, permettant ainsi d'échapper aux optima locaux. Inversement, les considérations pratiques de coût computationnel militent pour l'emploi de populations de taille limitée, sous peine de ne pas pouvoir répéter suffisamment les étapes nécessaires à l'évolution de la population. [Cornuéjols 2009]

Par ailleurs, la qualité de la population initiale peut contribuer à l'obtention rapide des meilleurs individus (fittest individuals). Les heuristiques peuvent être utilisées pour avoir des solutions initiales optimales. Cependant, pour ces stratégies d'initialisation, un autre mécanisme additionnel est utilisé pour empêcher la population d'être dans la même région de l'espace de recherche. Par exemple, une part de la population pourrait être générée aléatoirement afin d'introduire une diversification de l'espace de recherche alors que le reste de la population est optimisé [Braune 2010].

Stratégies de sélection :

Le mécanisme de sélection dans les AGs permet de déterminer quels individus de la population sont utilisés pour produire la prochaine génération. Ils existent plusieurs méthodes de sélection dans la littérature. Une méthode intuitive sélectionne les meilleurs parents (the fittest parents), car ils sont plus probable de générer des descendants ayant de bonne qualité. Dans ce qui suit, nous citons deux autres méthodes de sélection.

a. Sélection par roulette : (roulette-wheel-selection)

L'exemple suivant illustre le principe de la *sélection par roulette* [Braune 2010]. Etant donné les solutions suivantes et leurs valeurs de fitness (problème de maximisation).

$$f(s_1) = 10, f(s_2) = 20, f(s_3) = 30, f(s_4) = 40, f(s_5) = 50$$

La probabilité de sélection de s_1 est plus petite que la probabilité de sélection de s_5 . Avec, la probabilité de sélection d'une solution est égale à :

$$p(s_i) = \frac{f(s_i)}{\sum_{k=1}^n f(s_k)} \quad (1.1)$$

Ainsi, les probabilités de sélection (voir **figure I.11**) des solutions citées au-dessus sont:

$$p(s_1) = \frac{10}{10+20+30+40+50} = \frac{10}{150} = \frac{1}{15}, p(s_2) = \frac{2}{15}, p(s_3) = \frac{3}{15}, p(s_4) = \frac{4}{15}, p(s_5) = \frac{5}{15}$$

Considérons l'exemple suivant (voir **figure I.12**) composé de cinq solutions et leurs fitness correspondantes :

$$f(s_1) = 100, f(s_2) = 5, f(s_3) = 20, f(s_4) = 15, f(s_5) = 10$$

La **figure I.12** illustre un problème majeur lors de l'utilisation de la sélection par roulette. Comme la recherche progresse, des solutions avec des valeurs de fitness supérieur peuvent survenir. Dans ce petit exemple, une seule solution avec sa fitness très élevée existe. Ces solutions dominent alors la population, car leurs probabilités de sélection sont très élevées, et la chance de sélection autres individus est minimale. Ceci amène a une convergence prématurée.

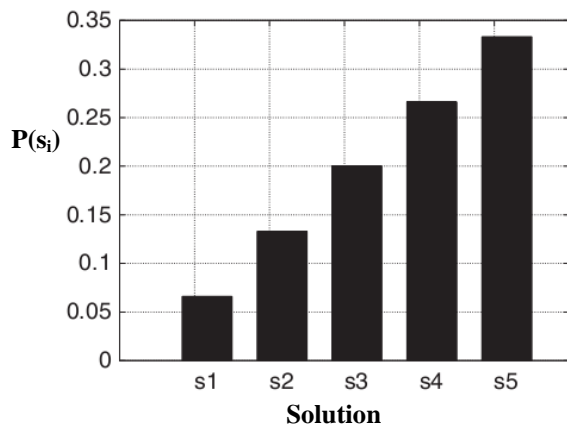


Figure I.11 : Probabilités de sélection d'une solution [Braune 10]

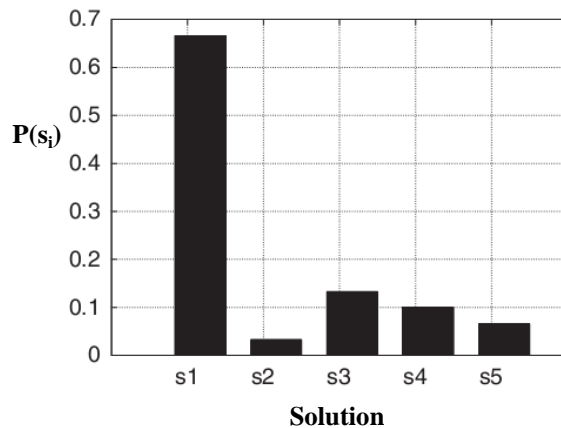


Figure I.12 : Une solution domine la population avec sa fonction Fitness [Braune 10]

b. Sélection proportionnelle au rang :

Pour éviter le problème de la sélection par roulette cité au-dessus, la *sélection proportionnelle au rang* peut être utilisée. Ce type de sélection utilise le rang d'une solution comparé à d'autres solutions plutôt d'utiliser directement les valeurs de fitness. [Braune 2010].

Etant donné l'exemple cité dans la sélection par roulette. La première étape consiste à trier en ordre croissant les valeurs de fitness (pour un problème de maximisation). On obtient : $(s_2, s_5, s_4, s_3, s_1)$. Aussi, au lieu d'utiliser les valeurs de la fonction fitness, la sélection proportionnelle au rang utilise le rang des solutions. La solution s_2 a le rang 1 ..., et la solution s_1 a le rang 5. Puis, on calcule la probabilité de sélection de façon similaire que la sélection par roulette :

$$s_i = \frac{r(s_i)}{\sum_{k=1}^n r(s_k)} \quad (1.2)$$

Avec r : le rang d'une solution s_k . On obtient alors les probabilités de sélections suivantes :

$$s_2 = \frac{1}{15}, s_5 = \frac{2}{15}, s_4 = \frac{3}{15}, s_3 = \frac{4}{15}, s_1 = \frac{5}{15}.$$

Ainsi, en utilisant la sélection proportionnelle par rang, les solutions dominants ont moins d'influence. [Braune 2010].

c. Sélection par tournoi :

Dans ce type de sélection, un sous-ensemble des solutions de la population sont choisis aléatoirement. Ensuite, la meilleure solution (dans ce sous-ensemble) est

sélectionnée. Etant données les solutions suivantes : $\{s_1, s_2, s_3, s_4, s_5\}$. Jusqu'à présent, on ne tient pas compte des valeurs de fitness correspondantes à ces solutions. Dans un premier temps, la taille de tournoi (tournament size) doit être choisie. Ainsi, deux individus sont choisis aléatoirement, par exemple : s_4, s_3 , et leurs valeurs de fitness sont déterminées. Ces valeurs déterminent uniquement quelle solution (parmi les deux solutions s_4, s_3) est meilleure que l'autre.

Considérons que : $f(s_3) = 20$; $f(s_4) = 15$. Ainsi, s_3 est choisie pour la recombinaison. D'autres parents nécessaires pour la recombinaison seront choisis de la même manière dans des tournois indépendants. [Braune 2010].

Opérateurs de croisement :

L'opérateur de croisement est un opérateur qui combine deux parents sélectionnés dans l'espoir de produire de mieux descendants. Un exemple de l'opérateur de croisement est : PMX (two-points partially mapped crossover) qui consiste à sélectionner des sous-chaines à l'aide de deux points choisie aléatoirement qui sont utilisés comme points de coupe. Puis en prenant les deux parents, on échange les sous-chaines correspondantes. (Voir **figure I.10**).

Opérateurs de mutation :

L'opérateur de *mutation* est un opérateur unaire qui est utilisé pour introduire une petite perturbation sur un individu donné. La mutation est utilisée comme un mécanisme de diversification pour les AGs (Algorithmes Génétiques), afin d'introduire un nouvel individu à partir d'un seul individu, en modifiant les bits à une (bit-flip) ou plusieurs positions choisies au hasard dans les chromosomes. (voir **figure I.10**). [Konar 2000]. Afin d'éviter la recherche purement aléatoire, cette opérateur doit être utilisé avec une petite probabilité p_m entre [0.001,0.01]

Stratégies de remplacement :

L'opération de remplacement consiste à décider quels individus seront admis dans la prochaine génération entre les deux parents de l'ancienne génération et les enfants obtenue à partir de l'étape de reproduction. Ils existent deux stratégies principales de remplacement :

- Remplacement de la population (*Generational replacement*): Cette stratégie consiste à remplacer tous les parents par leur descendants pour former la prochaine génération. L' AG utilisant cette méthode de remplacement sont appelés : *Generational GAs*.
- Stratégie des états stationnaires (*Steady-State replacement*): Un seule enfant est généré pour chaque couple de parents et il remplace le pire parent dans la prochaine génération, ce qui fait l'état d'équilibre. Chaque génération n'implique donc que le remplacement de certains individus et non le remplacement de toute la population.

I.9.2.3 Stratégies d'évolution

Forment une autre sous-classe des Algorithmes évolutionnistes, originalement développé par Rechenbeg et Schewefel en 1964 .Elles ont été initialement conçues pour résoudre des problèmes à variables continues. Elles utilisent généralement la méthode

de remplacement d'élite (consiste à conserver le meilleur individu de la population actuelle dans la population de génération suivante). L'opérateur de croisement est rarement utilisé. L'opérateur de sélection est déterministe et basé sur la sélection proportionnelle au rang. Leur principale avantage est leur efficacité temporelle. [Talbi 09].

I.9.2.4 Programmation évolutionniste :

La programmation évolutionniste, développée par Fogel à partir de 1966. Elle est fondée essentiellement sur l'opérateur de mutation et n'utilise pas d'opérateur de croisement. Le mécanisme de sélection est déterministe. Le processus de remplacement est probabiliste et basé sur la sélection par tournoi.

I.10 Classification des métaheuristiques

Les méthodes de recherche de type métaheuristiques peut être définies comme des méthodes " génériques " utilisées pour la recherche d'une solution approchée aux problèmes d'optimisation. Dans la section précédente, nous avons présenté un aperçu des principales métaheuristiques. Cette section synthétise les caractéristiques de ces métaheuristiques et les classe à partir de cinq critères fondamentaux.

- Naturel vs non naturel:

D'après Blum et Roli [Blum 03], un des critères intuitifs de la classification des métaheuristiques est leur origine d'inspiration. D'un côté, des algorithmes inspirés de la nature telle que les algorithmes génétiques. D'un autre côté, des métaheuristiques non inspirées de la nature comme la recherche tabou. D'après [Braune 10], cette définition possède une signification bien limitée. En effet, il n'est toujours pas possible d'attribuer définitivement une métaheuristique à l'une des deux classes.

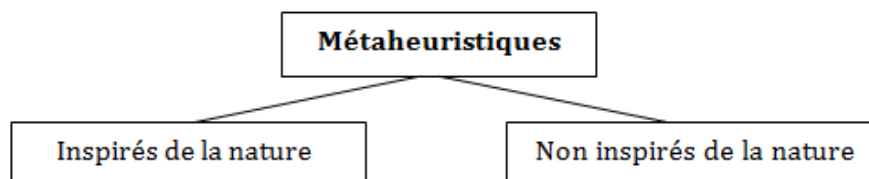


Figure I.13 Taxonomie des métaheuristiques à base des analogies de la nature

- Dynamique vs statique :

Dans [Blum 03], Les métaheuristiques peuvent être aussi classifiées selon l'usage de la fonction objectif. Certains algorithmes conservent cette fonction telle quelle est au bout du processus de recherche. Tandis que d'autres, la modifient [Braune 10].

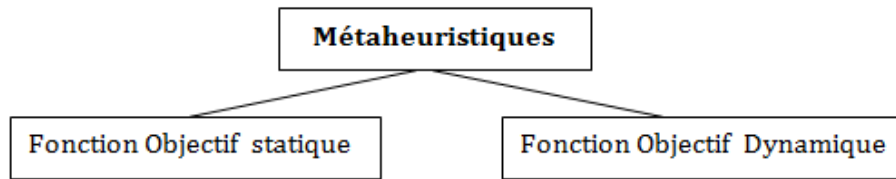


Figure I.14 Taxonomie des métaheuristiques à base de la fonction Objectif

- **Structure de voisinage simple vs variable :**

Dans cette classification, nous tenons compte du nombre de structures de voisinages comme l'indique la **figure I.15**. Les métaheuristiques à base de recherche « Standard », utilise un seul schéma de voisinage (tel que : la recherche Tabou) [Braune 10]. Les approches de voisinage multiples permettent d'utiliser un ensemble de structures de voisinages (diversification d'espace de recherche). [Blum 03].

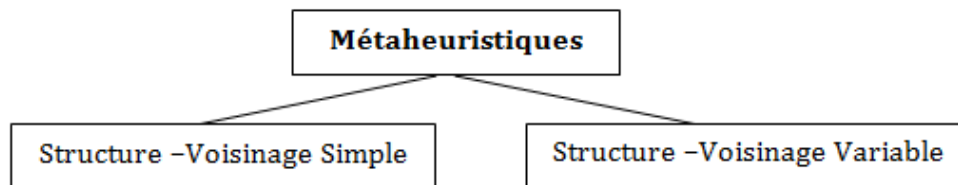


Figure I.15 Taxonomie des métaheuristiques à base de la structure de voisinage

- **Recherche à base de population vs unique :**

Un autre façon de classer les métaheuristiques est de distinguer celles qui utilisent une population de solutions (ex : les algorithmes génétiques) de celles qui ne manipulent qu'une seule solution à la fois. Les méthodes qui tentent itérativement d'améliorer une solution sont appelées aussi méthodes de *recherche locale* ou méthodes de *trajectoire*.

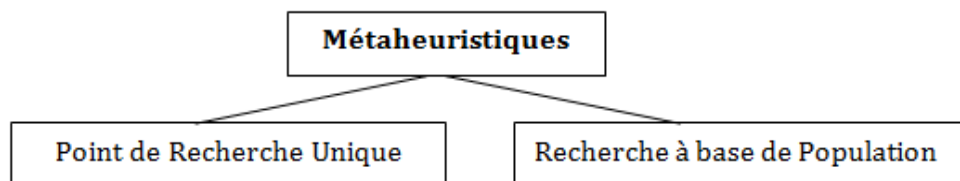


Figure I.16 Taxonomie des métaheuristiques selon [Hertz 2003]

- **Déterministe vs Aléatoire**

Une métaheuristique de type déterministe permet la résolution d'un problème d'optimisation en prenant des décisions déterministes (ex : recherche Tabou). Contrairement aux métaheuristiques stochastiques, les métaheuristiques déterministes permettent d'avoir la même solution finale en partant de la même configuration initiale.

Pour les métaheuristiques stochastiques, la même configuration initiale amènera à des différentes solutions finales (ex : A.G). [Talbi 09]

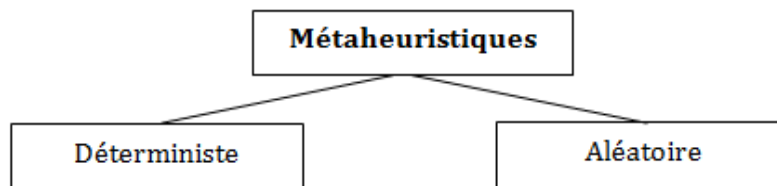


Figure I.17 Taxonomie des métaheuristiques selon la nature de décision

1.11 Exploitation vs Exploration

L'exploration (aussi appelé *Diversification*) permet de chercher de nouvelles solutions dans l'espace de recherche. Cette phase permet de soulever le problème d'optimum local. L'exploitation (aussi appelé *Intensification*) permet de concentrer la recherche sur les zones prometteuses et d'essayer d'obtenir les meilleures solutions possibles.

La conception d'une métaheuristique doit tenir compte des deux critères contradictoires (voir **figure I.18**) : *l'exploration* de l'espace de recherche et *l'exploitation* de meilleures solutions trouvées [Talbi 09]. Ces deux concepts ont été initialement introduit dans le contexte de la recherche Tabou [Glover 89i], [Glover 89ii]. Plus tard, ils ont été généralisés et considérés comme une propriété essentielle pour les métaheuristiques [Braune 10].

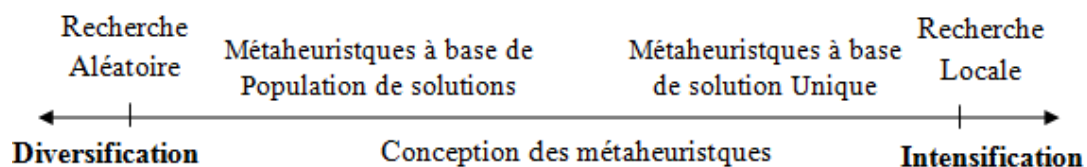


Figure I.18 Intensification vs Diversification : les deux critères contradictoires dans la conception des métaheuristiques [Talbi 09]

Généralement, les métaheuristiques à base de recherche unique sont beaucoup plus orienté exploitation tandis que les métaheuristiques a base de population sont orienté beaucoup plus exploration [Talbi 09]. Les algorithmes de recherche en termes d'exploration (resp. exploitation) sont de nature aléatoire (resp. recherche locale a base d'amélioration itérative) [Talbi 09].

Durant la recherche, les algorithmes aléatoires génèrent des solutions aléatoires. Alors que les algorithmes de recherche locale sélectionnent la meilleure solution du voisinage qui permet d'améliorer la solution courant [Talbi 09].

L'**algorithme 1.4** montre une description abstraite du fonctionnement générale d'une métaheuristique. La boucle principale présente un branchement a l'une des deux alternatives : phase d'intensification ou bien de diversification. Pour ces deux étapes,

l'algorithme créera une nouvelle solution tant que le critère d'arrêt n'est pas satisfait. En conséquence, l'algorithme retourne la meilleure solution trouvée.

Algorithme 1.4 modèle algorithmique abstraite pour les métaheuristiques

- Créer une (des) solution(s) initiale(s) ex : usage d'heuristique spécifique pour un problème donné

- **Tant que** (critère d'arrêt n'est pas satisfait) **faire**

Si (intensifié) **Alors**
Créer une nouvelle solution par étape d'intensification.

Sinon
Créer une nouvelle solution par étape de diversification.

Fin
Mettre à jour la meilleure solution trouvée (si nécessaire);

Fin

Retourner la meilleure solution.

Le succès d'une métaheuristique dépend principalement d'un compromis entre la diversification et l'intensification. Ne pas préserver cet équilibre, conduit à une bonne exploitation (convergence rapide vers des optimum locaux) ou une bonne exploration (mauvaise exploitation).

I.12 Frameworks & bibliothèques pour les Métaheuristiques

Face à la variété et la complexité des problèmes d'optimisations, il semble nécessaire d'établir un outil permettant d'analyser et/ou concevoir les méthodes métaheuristiques. À ce stade, plusieurs Framework permettant la modélisation et l'implémentation des métaheuristiques ont été proposées.

Il existe trois approches fondamentales utilisées pour le développement des métaheuristiques [Talbi 09] :

- **Débuter à partir de rien** (From scratch or no reuse) : Les programmeurs sont tentés de développer eux-mêmes leurs codes, ce qui augmente la difficulté de cette approche (besoin de temps et d'énergie, difficile à maintenir et faire évoluer ...).
- **Réutilisation du code** (Only code reuse) : permet la réutilisabilité du code disponible sur Web sous forme de programme (free individual programs) ou des bibliothèques (libraries).
- **Réutilisation du code & de conception** (Both design and code reuse) : L'objectif de la réutilisabilité du code et de la conception est de refaire le moins possible du code chaque fois qu'un nouveau problème d'optimisation doit être traité.

Les principales différences entre les Framework s'appuyant sur les points suivants :

- Type de la *fonction objective* (Mono- objectif et/ou Multi- objectif).
- Type du *problème d'optimisation* (Combinatoire et/ou Continue).

- Type des *métaheuristiques* utilisées (Métaheuristiques a base de population et/ ou a base de solution unique).
-

- Principaux Framework et Bibliothèques pour les métaheuristiques : [Fink 2003]

- **EasyLocal++**⁴ [Gaspero 2002] : (Gaspero and Schaerf, 2002) est une bibliothèque de classes en C++ qui supporte les métaheuristiques a base de *solution unique* (ex : recherche Tabou) pour la résolution des problèmes *mono-objectif*.

- **iOp** [Voudouris 2002]

Est une bibliothèque de classes en JAVA développé au sein du centre de recherche British Telecom. Elle support les métaheuristiques de *recherche locale* et les métaheuristiques *a base de population*. C'est une bibliothèque qui facilite l'hybridation des algorithmes. Elle est utilisé pour la résolution des problèmes d'optimisation *combinatoire mono-objectif*.

- **ParadisEO**⁵ :

Est une Framework open source (C++). Elle set développé par l'INRIA de Lille .C'est une extension de Framework EO initialement conçu pour l'implantation d'algorithmes évolutionnistes [Cahon 2004]. L'architecture du Paradis EO permettent la réutilisabilité du code afin d'accélérer le développement des métaheuristiques. Aussi l'architecture permette le développement de nouvelles métaheuristiques ou de métaheuristiques hybrides. Cette architecture est composée de trois couche : *Helpers* fournit les composants de base des métaheuristiques ex : opérateur de mutation), *Runners* (correspond aux différentes métaheuristiques) , *Solvers* (chargé d'exécuté (de coordonner) la(les) métaheuristiques). *ParadisEO* fournit des modules qui traite des métaheuristiques à base de population, l'optimisation multi-objectifs et les métaheuristiques à base de solution unique.

I.13 Evaluation des métaheuristiques

Afin d'évaluer les performances des méthodes d'optimisation exactes, l'efficacité est un indicateur principale pour ces algorithmes puisqu'elles garanties l'obtention d'un optimum global. D'autres critères de mesure de qualité des solutions doivent être pris en considération durant l'évaluation de l'effectivité des métaheuristiques. Ces critères de performances peuvent être groupés en trois classes [Talbi 09] :

- **Qualité de solution** : les indicateurs de performances sont généralement basés sur la distance entre la solution obtenue et la solution optimale globale ou la meilleure solution connue.
- **Effort de calcul** (Computational effort) : l'efficacité d'un algorithme peut être démontrée par une analyse théorique ou empirique .

⁴ <http://tabu.diegm.uniud.it/EasyLocal++/>

[29]

⁵ <http://paradisEO.gforge.inria.fr/>

- *Analyse théorique* : ça concerne le calcul de la complexité de l'algorithme dans le pire cas. (**Voir partie 1**)

- *Analyse empirique* : ça concerne le temps de calcul nécessaire pour une métaheuristique de résoudre les instances d'un problème donné.

• **Robustesse** : insensibilité contre des petites déviations des données d'entrées ou des paramètres des métaheuristiques. La diminution de la variabilité des solutions implique une augmentation de la robustesse de l'algorithme.

D'autres critères d'évaluations qualitatives peuvent être utilisés pour évaluer la performance des métaheuristiques telle que : la facilité d'utilisation, la simplicité, flexibilité,[Talbi 09]

I.14 Conclusion

Nous avons décrit dans ce chapitre le cadre général des métaheuristiques et de connaître au mieux les concepts liés à l'optimisation et ce dans le but de contourner la problématique étudiée.

Nous constatons que la philosophie et le mode de recherche des métaheuristiques mettent en point un bon outil qui peut être appliquées pour la résolution des problèmes d'optimisation des classifieurs.

Dans le chapitre suivant, nous présentons un état de l'art exhaustif de ces problèmes. Nous nous penchons sur le Problème de Performance des Classifieurs (**PPC**) auquel nous nous intéressons dans ce mémoire de magister.

Chapitre II :

Problème de Performance des Classifieurs (*PPC*)

Ce chapitre survole, en premier lieu la littérature des problèmes de classification rencontrés et découlent l'importance et la logistique de la performance d'un classifieur. En deuxième lieu, ce chapitre se concentre sur le problème étudié : *PPC* en faisant un tour d'horizon des problèmes issus et des méthodes de résolution qui lui sont dédiées.

II.1 Introduction

Depuis quelques années, de nombreuses équipes de recherche tentent de développer des systèmes d'aide à la décision pour des applications complexes du monde réelle. A titre d'exemple, dans le champ médical, plusieurs voies sont explorées telles que la classification automatique des anomalies afin d'aider les médecins à la détection précoce des maladies.

Une tâche de *classification* (ou discrimination) consiste à classer un ensemble d'objets, ou attribuer à chacun une classe (ou catégorie) parmi plusieurs classes définie à l'avance. Un algorithme qui réalise automatiquement cette classification est appelé *classifieur*. [Cornuéjols 2003]

Pour de nombreux domaines d'application tels que le traitement de signal et la reconnaissance des formes, les approches de data-mining sont nécessaires pour développer des modèles approximatifs *performants*. Entre temps, plusieurs applications (comme le diagnostic médical) impose l'étape d'apprentissage qui est désormais effectuer en temps réel. D'autres applications, telle que le web-mining qui doit traiter un ensemble de données très volumineux. [Sheng 1999]

Par conséquent, un des problèmes les plus compliqués est de développer un classifieur avec une structure compact qui peut achever une bonne performance dans un temps de calcul raisonnable. Ils y a deux aspects lorsque ces deux issues sont investigués : l'*architecture* du classifieur et le *scénario d'apprentissage*. Nous considérons dans le cadre de ce mémoire de magister le cas d'apprentissage supervisé. Nous discutons plusieurs approches, telle que : *l'arbre de décision*, *l'algorithme du cas plus proche voisin* et *les réseaux de neurones*. En apprentissage supervisé, la *performance* s'adresse au problème de développement d'un classifieur qui peut achever une performance optimale pour un ensemble de données qui n'est pas inclut dans l'échantillon d'apprentissage. Les méthodes d'apprentissage supervisé souffrent d'un problème intrinsèque qui est le dilemme *Surapprentissage / surgénéralisation*.

Dans tout les cas, l'*évaluation* de l'apprentissage devrai être conduite avec soin. En général, on mesure la performance après que l'apprentissage a eu lieu sur un certain nombre de données que l'on appelle échantillon de test. Il faut s'assurer que la mesure de la performance s'effectue sur un échantillon de test différent de l'échantillon d'apprentissage. Autrement, ce ne sera pas la capacité de *généralisation* qui sera testée, mais une capacité à l'apprentissage par cœur. [Cornuéjols 2003]

On peut donc en tirer deux enseignements [Dreyfus 2008] :

- D'une part, il ne faut généralement pas estimer la performance d'un modèle à partir des résultats de l'apprentissage ;
- D'autre part, il faut estimer le mieux possible l'erreur de prédiction.

L'objectif de ce chapitre est de présenter les principaux problèmes affectant la performance des classifieurs. Nous présentons une revue de littérature des travaux qui décrivent différents approches de classification (AD, Kppv, RN).

II.2 Performance des Classifieurs

Avant de définir le contexte de la performance, une question importante est : *pourquoi on a besoin de la performance ?*. La performance d'un algorithme d'apprentissage est un point crucial. En effet, c'est à partir de l'évaluation de cette performance que l'on peut comparer, choisir ou valider les différentes méthodes.

On cherche à évaluer, en particulier, l'erreur de généralisation des modèles prédictifs que nous avons construit, c'est à dire le taux d'erreur de prédiction de la sortie d'un ensemble d'exemples indépendants de ceux utilisées pendant l'apprentissage. Pour cela, on définit tout d'abord une fonction qui mesure l'erreur entre la sortie du modèle et la sortie réelle.

Les performances d'un classifieur, quelconque peuvent être représentées par sa matrice de confusion, pour la quelle, on ne s'intéresse pas seulement au nombre de prédictions correctes, mais on veut également savoir si une prédiction a été faite positive alors que l'exemple montrait réellement un résultat négatif (*faux positif*) ou l'inverse (*faux négatif*) (Voir section II.3.3).

En apprentissage supervisé, deux issues importantes et étroitement liés doivent être clarifié : la *performance* et l'*efficacité*. La performance s'adresse à la capacité de généralisation d'un classifieur. L'efficacité traite la complexité temporelle et spatiale du classifieur. (Voir section suivante). En pratique, le nombre des données disponibles peut être rétreints. Plusieurs méthodes d'estimation de l'erreur de classification existent, et visent à optimiser l'utilisation de ces données ou comme solution pour certain problème liée à la classification. (Voir les sections II.4.1 et II.4.3).

Les performances d'un classifieur dépendent fortement de :

- *La qualité des données utilisées.*
- *Paramètres algorithmiques/non-algorithmiques de l'approche utilisée.*

Dans le cadre de ce mémoire de magister, nous discutant le deuxième critère (*Paramètres algorithmiques/non-algorithmiques de l'approche de classification utilisée*) affectant la performance des classifieurs.

II.3 Pré-requis

II.3.1 Notations

Dans un environnement d'apprentissage supervisé, étant donnée $D = \{X_n, T_n\}_{n=1}^N$, avec N : nombre des paires d'apprentissage $X_n \in \mathbf{R}^d$ (d : vecteur d-dimension d'attributs de n exemples et T_n : leurs classe correspondantes). Pour faciliter la compréhension des notions de la performance, supposons que T_n est un scalaire et qu'il existe une fonction $f(.)$ telle que : $T_n = f(X_n)$. T_n : un réel indiquant la classe des n -exemples. [Sheng 1999]

Etant donnée un réseau de neurones (voir figure II.1), composé d'une couche d'entrée de taille (X_d), une couche cachée (de taille L , fonction d'activation sigmoïde). Soit $W^{(1)}$: la matrice des poids de la 1^{ère} couche et sa $J^{ème}$ colonne $W_j^{(1)}$ ($1 \leq J \leq L$) : le vecteur poids connectant le $J^{ème}$ neurone caché aux entrées. $W^{(2)}$ est un vecteur de poids de la 2^{ème} couche et ses J éléments $W_j^{(2)}$ qui note les poids connectant le $J^{ème}$

neurone caché a la sortie. \mathbf{h}_n : est un vecteur qui représente les sorties des neurones cachés correspondant a l'entrée \mathbf{X}_n . Le J^{eme} élément ($h_{j,n}$ avec $1 \leq J \leq L$) de \mathbf{h}_n est égale a : [Sheng 1999]

$$h_{j,n} = g(\mathbf{w}_j^{(1)T} \mathbf{x}_n) \quad (2.1)$$

Avec $g(\cdot)$ est la fonction de transfert sigmoïde, la sortie du réseau \mathbf{Y}_n correspondant a l'entrée \mathbf{X}_n est écrit comme suite : [Sheng 1999]

$$y_n = \mathbf{w}^{(2)T} \mathbf{h}_n \quad (2.2)$$

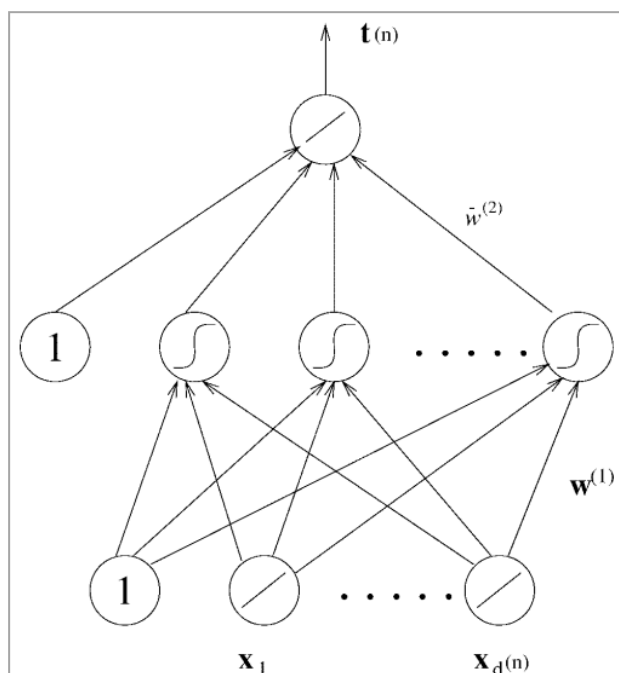


Figure II.1 : Structure d'un réseau de neurone (feed-forward) [Sheng 1999]

II.3.2 Performance et efficacité

a) - Performance :

Etant donnée $f_D(\mathbf{X}, \mathbf{W})$ un modèle avec (\mathbf{W} : ensemble des paramètres entraîné sur une base d'apprentissage). La performance de $f_D(\mathbf{X}, \mathbf{W})$ peut être mesuré en terme de la différence entre de la fonction $f(\mathbf{X})$ a être approché et son approximation $f_D(\mathbf{X}, \mathbf{W})$ à travers la norme carré (squared norm). [Sheng 1999]

$$\int |f(\mathbf{x}) - f_D(\mathbf{x}; \mathbf{w})|^2 p(\mathbf{x}) d\mathbf{x} \quad (2.3)$$

Avec $p(\mathbf{x})$: fonction de densité probabilité (probability density function) de \mathbf{x} . Pour un modèle de classification, l'erreur de généralisation est défini comme étant la probabilité de la classification incorrecte. [Sheng 1999]

b) - Efficacité :

L'espace de complexité S de $fD(X, W)$ est le nombre des paramètres libres L de $fD(X, W)$ lorsque l'erreur de généralisation n'est pas supérieur à $\epsilon_g > 0$. Par exemple, pour un réseau de neurones (feed-forward) de deux couches, L peut être le nombre des neurones cachés ou le nombre total des poids. La complexité temporelle d'un système d'apprentissage adaptative est défini comme le temps d'apprentissage attendu nécessaire pour obtenir un système d'apprentissage lorsque l'erreur de généralisation est délimité par ϵ_g . La propriété de la complexité spatiale et temporelle définit l'efficacité d'un algorithme d'apprentissage duquel l'erreur de généralisation n'est pas supérieur à la quantité $\epsilon_g > 0$. $fD(X, W)$ est dite *efficace en espace* si S est polynomiale en terme de dimension d des vecteurs d'attribues, ϵ_g et d'autres paramètres liés. Si la complexité en temps est polynomiale en termes de ces quantités, alors la méthode d'apprentissage est dite *efficace en temps*. Si l'efficacité peut être achevé à la fois en espace et en temps, alors l'algorithme est dite *efficace*. [Sheng 1999]

c) - Relation entre performance et efficacité :

La performance et l'efficacité sont étroitement liées. Pour faciliter la compréhension de cette relation, supposons qu'un algorithme d'apprentissage obtient les poids de chaque neurone caché en un temps polynomiale. Cette algorithme n'est pas efficace si le nombre des neurones cachées nécessaire est exponentielle en terme de la dimension de vecteur d'attribue. Ceci est due à cause du temps d'apprentissage totale qui est estimé comme le temps nécessaire pour apprendre un neurone caché multiplié par le nombre des neurones dans un réseau. En d'autre terme, une complexité spatiale polynomiale est nécessaire pour achever une complexité de temps polynomiale pour apprendre le réseau. Ensemble, l'efficacité spéciale et temporelle peut définir l'efficacité de l'algorithme d'apprentissage d'un réseau de neurones. [Sheng 1999]

L'efficacité doit être défini avec la performance i.e. : l'erreur de généralisation d'un réseau résultant de 2 couches doit être plus petits que la quantité désiré lors de l'examinassions de la complexité spatiale et temporelle. Ceci est dû au concept d'efficacité qui n'aurait pas de sens sans l'exigence de la performance de généralisation. [Sheng 1999]

II.3.3 Mesure de la performance

L'évaluation de la performance d'un classifieur se fait par le taux de bonnes classifications (Taux *classification*). Ainsi, par exemple, si pour 100 exemples de tests, 91 ont été prédits correctement par le modèle de prédiction, on pourra dire que ce modèle a un taux de classification de 91% (souvent écrit 0.91). Mais ce critère d'évaluation n'est pas le seul à prendre en compte.

De nouveaux critères peuvent entrer en jeu. En effet, On ne s'intéresse pas seulement au nombre de prédiction correctes, mais on veut également savoir si une

Sortie Réelle	Sortie Désirée	
	Correcte	Incorrecte
Correcte	vrais positifs	Faux positifs
Incorrecte	Faux négatifs	vrais négatifs

Tab II.1 Matrice de confusion

prédiction a été faite positive alors que l'exemple montrait réellement un résultat négatif (*faux positif*) ou l'inverse (*faux négatif*) (voir **tableau II.1**). On peut alors définir les termes suivants :

- ✓ Taux de faux positif (**FP**) : représente le nombre des patients non malades prédit comme malade.

$$T_{FP} = \frac{\text{Faux positifs}}{\text{Vrais négatifs} + \text{Faux positifs}} \quad (2.4)$$

- ✓ Taux de faux négatif (**FN**) : représente le nombre des patients malades prédit comme non malade.

$$T_{FN} = \frac{\text{Faux négatifs}}{\text{Vrais positifs} + \text{Faux négatifs}} \quad (2.5)$$

- ✓ Sensibilité (**Se**) : mesure le taux de vrai positif (**VP**). Il est calculé par la formule suivante :

$$Se = \frac{\text{Vais positifs}}{\text{Vrais positifs} + \text{Faux négatifs}} \quad (2.6)$$

- ✓ Spécificité (**Sp**) : mesure le taux de vrai négatif (**VN**). Elle est donnée par la formule suivant :

$$Sp = \frac{\text{Vais négatifs}}{\text{Vrais négatifs} + \text{Faux positifs}} \quad (2.7)$$

- ✓ Taux de Classification (Accuracy) : c'est la reconnaissance correcte rapportée au total.

$$T_{Classification} = \frac{\text{Vrais positifs} + \text{Vais négatifs}}{\text{Vrais positifs} + \text{Vrais négatifs} + \text{Faux positifs} + \text{Faux négatifs}} \quad (2.8)$$

- ✓ Valeur Prédictive Positive (**Vpp**) : représente le taux de classification des cas malades correctement reconnus par le classifieur par rapport au nombre total des cas malades détectés par le classifieur .On la calcule par l'équation :

$$Vpp = \frac{\text{Vais positifs}}{\text{Vrais positifs} + \text{Faux positifs}} \quad (2.9)$$

- ✓ Valeur Prédictive Négative (**Vpn**) : représente le taux de classification des cas non malades correctement reconnus par le classifieur par rapport au nombre total des cas non malades détectés par le classifieur. On la calcule par l'équation :

$$Vpn = \frac{\text{Vais négatifs}}{\text{Vrais négatifs} + \text{Faux négatifs}} \quad (2.10)$$

En plus des critères numériques, il existe un certain nombre de qualités qui permettent de distinguer un modèle parmi d'autres. [Cornuéjols 2003]

- ✓ L'intelligibilité (la compréhensibilité).
- ✓ La simplicité des résultats de l'apprentissage.
- ✓

II.4 Problème de la Performance des Classifieurs

II.4.1 Définition du PPC

L'objectif de la classification est de reconnaître des entités en catégories à partir d'observations effectuées sur celles-ci. Ce processus se décompose en 3 étapes (voir figure II.2) :

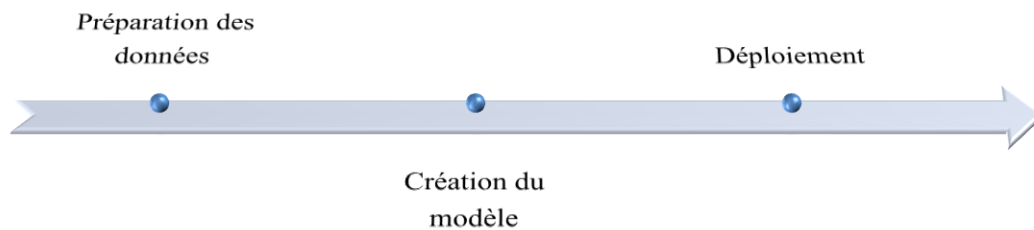


Figure II.2 : Processus globale de la classification

1. **Préparation des données** : la phase de préparation des données est particulièrement importante dans le processus de classification. Pour être exploitées par des méthodes de classification, les données doivent subir à une série de prétraitements en vue d'obtenir des données écartées de problèmes. Les principales étapes du prétraitement seront les suivantes :

- *Nettoyage* : constitue le traitement du bruit, des valeurs manquantes ou aberrantes
- *Transformation* : exemple normalisation des données.
- *Sélection d'attributs* (Feature Selection en anglais) : permet de réduire le nombre de variables à celles qui sont les plus pertinentes.

2. **Création du modèle** : Le but de cette phase étant de trouver le modèle le plus adapté au problème. De ce fait, plusieurs modèles seront testés sur les données afin de comparer leurs performances.

3. **Déploiement** : Pour obtenir les meilleurs résultats possibles l'étape de déploiement applique le meilleur modèle sélectionné à l'étape précédente à de nouvelles données.

D'après la **figure II.2** et comme indiqué au-dessus, la performance d'un classifieur dépend fortement de la qualité des données qu'on lui fournit et l'approche de classification utilisée (paramètres : algorithmique / non-algorithmique).

Dans les deux parties suivantes (sections **II.5**, **II.6**), certaines méthodes de classification ont été présentées en mettant de côté quelques difficultés potentielles. Cette partie va être l'occasion de les aborder.

II.4.1.1 Qualité des données

Un prétraitement des données est nécessaire avant de les utiliser dans un algorithme d'apprentissage. Les différents problèmes à considérer sont les suivants (voir **figure II.3**) :

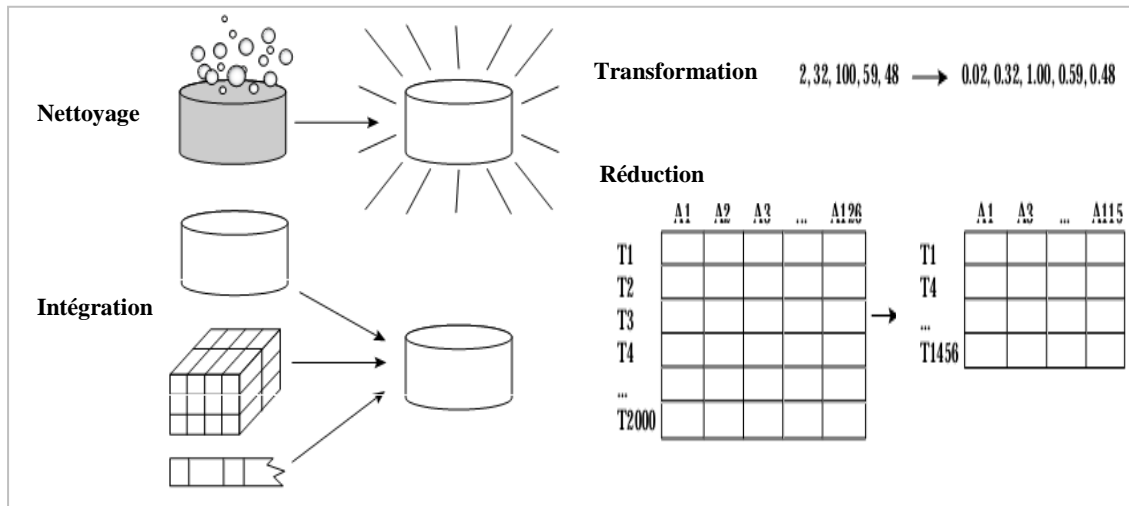


Figure II.3 Principales étapes dans le prétraitement des données

- **Nettoyage des données :**

Les données disponibles sont rarement décrites parfaitement. Souvent les défauts des instruments de mesure artificiels ou humains provoquent des erreurs. [Cornuéjols 2003]. Les données peuvent comporter des valeurs *manquantes* (incomplètes), être *imprécises*, et parfois elles sont *inhomogènes* (résultant de plusieurs sources). Ceci est a cause de :

- Instrument de mesure défectueux.
- Problème de saisie.
- Problème de transmission.
- Limitation technologique.
- Incohérence dans les conventions de nommage.

Le traitement du bruit dans les données n'est pas un problème facile à résoudre. Ainsi, différentes techniques sont possibles pour y remédier :

- Compléter manuellement les données.
- "Listwise data deletion" : suppression de l'entité x_i ayant au moins une valeur manquante.
- Imputation moyenne : remplacement de la valeur manquante par la moyenne de l'attribut correspondant.
- Imputation "hot-deck" : remplacement de la valeur manquante par celle de l'exemple le plus proche.
- L'une des méthodes les plus utilisées pour décrire des données imprécises est la logique floue.

- **Transformation :**

Certains algorithmes d'apprentissage sont incapables de traiter directement des attributs à valeur continue. Il est nécessaire de les transformer en attributs à valeurs discrète. Une autre raison pour discrétiser un attribut à valeur continue est de réduire le volume des données. [Cornuéjols 2003]

D'autres formes de la transformation des données sont la réduction de dimension (Voir section II.4.1.4) et la normalisation des données. La normalisation consiste à la mise à l'échelle des valeurs des attributs. Les méthodes de discrétisation sont nombreuses par exemple :

- La méthode *min-max* : [Kamber 2006]

Etant donné un attribut A et n observations : v_1, v_2, \dots, v_n .

La normalisation *min-max* effectue une transformation pour un ensemble de données d'origine. Etant donné : \min_A et \max_A , le minimum et le maximum d'un attribut A respectivement. La normalisation *min-max* fait la correspondance d'une valeur v de A en une nouvelle valeur v' en utilisant la formule suivante :

$$v' = \frac{v - \min_A}{\max_A - \min_A} \quad (2.11)$$

- La méthode *z-score* : [Kamber 2006]

Dans la normalisation *z-score*, les valeurs d'un attribut A sont normalisées en utilisant la moyenne (mean) et la *déviatoin standard* de A . Une valeur v de A est normalisé en v' en utilisant l'équation suivante :

$$v' = \frac{v - \mu}{\sigma} \quad (2.12)$$

Avec : $\mu = \frac{1}{n}(v_1 + v_2 + \dots + v_n)$; $\sigma = \frac{1}{n} \sum_{i=1}^n (v_i - \mu)^2$.

II.4.1.2 Choix des paramètres

Il existe plusieurs paramètres qui doivent être ajustés pour chaque méthode de classification. L'ajustement des paramètres doit être fait avec soins, car ils ont une grande influence sur l'efficacité et l'effectivité de l'approche de classification. Un ensemble de valeurs optimal et universelle pour les différentes valeurs des paramètres d'une méthode de classification donné est inexistante.

Il existe principalement deux stratégies d'ajustement des paramètres :

- Initialisation hors ligne (off-line) :

Dans cette stratégie d'initialisation, les valeurs des différents paramètres sont fixées avant l'exécution de l'approche de la classification. Généralement, le développeur de l'approche ajuste un paramètre à la fois, et ces valeurs optimales sont déterminées empiriquement.

- Initialisation en ligne (online) :

Dans cette stratégie d'initialisation, les paramètres sont contrôlés et mis à jour durant l'exécution de l'approche de classification. Par exemple, une modification d'une valeur d'un paramètre est accomplie par utilisation de métaheuristique.

II.4.1.3 Choix de modèle

La méthode de minimisation du risque conduit à concevoir des modèles de complexités différentes et à choisir celui qui est susceptible d'avoir les meilleures propriétés de généralisation [Dreyfus 2008]. Comme indiqué plus haut, la capacité de généralisation d'un modèle ne peut être estimée à partir des résultats d'apprentissage. Une telle procédure conduirait à sélectionner un modèle de biais faible et de variance élevée (cas de sur-ajustement) (voir section II.4.2)

Pour sélectionner le meilleur modèle parmi des modèles de complexités différentes, il convient de les comparer sur la base de prédictions qu'ils effectuent sur des données qui n'ont pas servi à l'apprentissage [Dreyfus 2008]. Nous décrivons ci-dessous, les méthodes de validation les plus couramment utilisées.

- Méthodes de validation :

À partir d'un échantillon de taille n , il existe de nombreuses méthodes pour estimer la qualité de l'apprentissage. Pour avoir une bonne estimation, il faut recourir à un ensemble d'exemples qui n'ont pas servi pour l'apprentissage : il s'agit de la base de test. Cette base de données est obtenue en réservant une partie des exemples initiaux et en ne les utilisant pas pour la phase d'apprentissage.

La base de test doit également contenir des exemples étiquetés (dont les classes sont connues), afin de comparer les prédictions faites par le modèle à la valeur réelle de la classe. Cependant pour des bases de données de petites tailles, il est pénalisant d'écarté une partie des exemples pendant la phase d'apprentissage.

Les techniques de validation croisée (cross validation), permettent d'obtenir une estimation des performances en exploitant la totalité du jeu de données. Ceci est obtenu en faisant plusieurs tests sur différents ensembles d'apprentissage et de validation, et en calculons la moyenne des résultats.

La resubstitution :

Le jeu de données pour l'apprentissage et le test sont identiques et correspondent à l'ensemble de l'échantillon. Si l'on note n_e , le nombre d'erreurs commises, on aura :

$$Erreur = \frac{n_e}{n} \quad (2.13)$$

Validation simple (hold-out)

Cette approche est utilisée lorsque l'on dispose d'un grand nombre de données. [Dreyfus 2008]. L'idée de cette approche est de séparer une base de données initiales en deux : base d'apprentissage et base de test. Par exemple une moitié des données est utilisée pour l'apprentissage, l'autre moitié pour le test.

$$Erreur = \frac{n_e}{n/2} \quad (2.14)$$

Validation croisée : *K-Fold*

L'algorithme de validation croisée *k-Fold* consiste à segmenter aléatoirement une base de données en k (folds) sous-ensembles (de même taille) disjoints numérotés de 1 à k . Ensuite, l'algorithme écarte le 1^{er} bloc (servira pour le test), et utilise les $k-1$ autres blocs afin de constituer l'ensemble d'apprentissage. Puis, il suffit de suivre l'approche classique de construction du modèle de prédiction sur l'ensemble d'apprentissage, en validant ses performances sur l'ensemble de test (voir **figure II.4**). On recommence cette opération en réservant chaque bloc successivement comme bloc de test. La valeur populaire pour k est: 10.

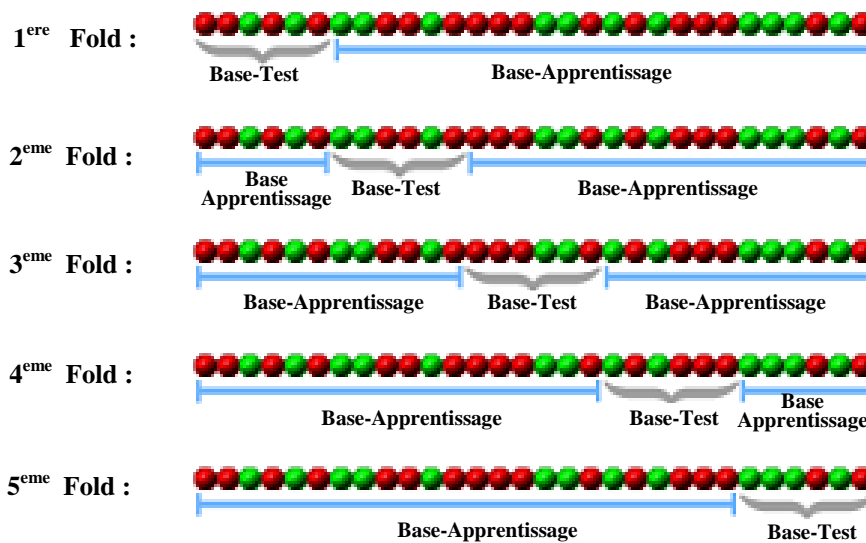


Figure II.4 : Illustration de la méthode de validation croisée (5-Fold)

Si l'on note $n_e(m)$ le nombre d'erreurs de classement commises sur le m -ième sous-ensemble, l'erreur est la moyenne des $n_e(m)$. (Avec $K = P$).

$$Erreur = \frac{\sum_{m=1}^p n_e(m)}{n} \quad (2.15)$$

Validation croisée : *Leave One Out*

Cette méthode est une forme particulière de la méthode de validation *k-Fold*, en prenant $k = n$ (avec n : étant le nombre d'exemples). A chaque itération, l'apprentissage est faite sur tous les exemples moins un, et tester sur un seul exemple, afin de vérifier s'il est prédit correctement. Cette méthode est plus coûteuse, en terme de temps de calcul, que la validation croisée *k-fold*, car elle nécessite beaucoup plus de tests.

II.4.1.4 Dimensionnalité des données

La sélection de variables constitue un élément important dans une procédure de conception d'un modèle par apprentissage ; elle contribue à la diminution de la complexité d'un modèle. Comme souligner plus haut, la sélection d'attributs consiste à éliminer les attributs les moins pertinents pour l'apprentissage. Le but est de diminuer la dimensionnalité du problème. Si on possède une description des données par un ensemble de D attributs, le problème est de chercher un sous-ensemble de d attributs qui préserve au mieux les informations nécessaires à l'algorithme d'apprentissage.

[Dreyfus 2008]. L'augmentation du nombre des caractéristiques n'améliore pas systématiquement la qualité de l'apprentissage. En effet, on est confronté au phénomène appelé « the curse of dimensionality ». Aussi, la complexité des algorithmes augmente lorsque la dimension D croît et certains attributs sont séparément pertinents, mais le gain est faible lorsqu'ils sont combinés.

II.4.2 Issues de la performance

Il ya trois facteurs importants qui affectent la performance des classifieurs : les phénomènes *sur-apprentissage / sur-généralisation* ; le dilemme *biais / variance* et la possibilité d'avoir une fonction objectif avec de nombreux minima *locaux*. Dans ce qui suit, nous discutons le premier issue et sa relation avec les deux autres issues de la performance.

II.4.2.1 Surapprentissage & Surgénéralisation

Le phénomène du sur-apprentissage (en anglais : overfitting) est l'un des problèmes les plus importants lorsque l'on travaille dans le domaine de l'apprentissage. En effet, il faut que le prédicteur apprenne suffisamment des données d'apprentissage pour pouvoir prédire de nouveaux exemples.

Cependant, il arrive souvent que les exemples de la base d'apprentissage comportent des valeurs bruitées. Ainsi, il ne faut pas qu'il "apprenne trop", sinon il colle parfaitement aux données d'apprentissage et perd de sa généralisation lorsqu'on lui présente de nouvelles données. (Voir figure II.5)

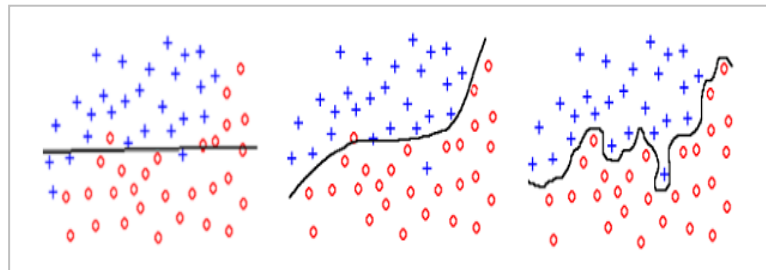


Figure II.5 Sous-apprentissage Fit Sur-apprentissage

Une cause très fréquente du surajustement est la présence du bruit dans les données d'apprentissage. En effet, il n'existe pas de dispositif de mesure qui délivre des résultats parfaite sans erreur. Aussi, depuis longtemps les praticiens savaient en effet qu'il leur fallait contrôler la complexité de leur modèle d'apprentissage pour ne pas tomber victime de surapprentissage. [Cornuéjols 2003]

Plus le nombre d'exemples utilisés pour construire le modèle augmente, plus l'erreur d'apprentissage diminue et tend vers 0. Cependant, ce qui compte vraiment, c'est l'erreur en généralisation. Pour sa part, quand le nombre d'exemples utilisées augmente, l'erreur en généralisation commence par diminuer puis elle augmente : c'est précisément la ou elle est minimale que l'on a construit le meilleur modèle, celui qui fait une erreur minimale. C'est ce modèle qui est capable de produire la meilleure généralisation de l'apprentissage, c'est-à-dire, qui a la plus petite probabilité de mal classer une donnée quelconque. Au-delà de ce modèle optimal, quand l'apprentissage se

poursuit, le modèle se complique, la probabilité d'erreur augmente, et le modèle produit du sur-apprentissage. (Voir figure II.6) [PREUX 2011]

Un bon moyen pour éviter ce phénomène est de partager la base d'apprentissage en deux sous-ensembles (Voir section II.4.1.3). Le premier sert à l'apprentissage et le deuxième sert à l'évaluation de l'apprentissage.

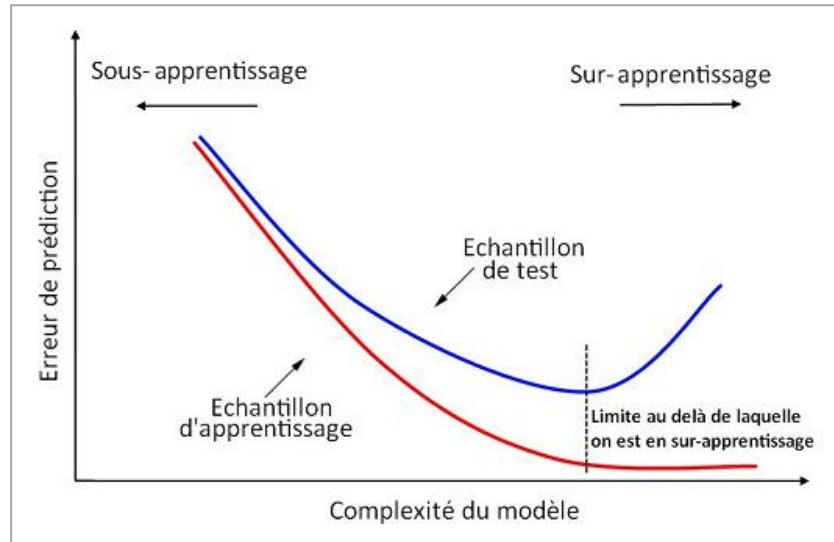


Figure II.6 : Phénomènes du sur-apprentissage et sur-généralisation [Preux 2011]

La théorie de la *régularisation statistique* introduite par Vladimir Vapnik permet de réguler les phénomènes liés au sur-apprentissage. (Voir section II.4.3.3).

Aussi, en classification, la méthode la plus simple pour éviter le sur-apprentissage est de réduire le nombre d'attributs. En effet, plus le modèle sera créé à partir d'un nombre limité de variables et plus il sera général. [Cornuéjols 2003]

La figure II.5 montre que le problème majeur résultant le sur-apprentissage (overfitting) est la perte de généralité. Une solution d'un processus d'optimisation est général si elle est non seulement valide pour l'ensemble d'entrées : a_1, \dots, a_n qui ont été utilisés pour l'apprentissage au cours du processus d'optimisation, mais aussi pour des autres entrées différentes $a \neq a_i \forall i: 0 < i \leq n$ si ces entrées (a) existe. [Weise 2009]

La sur-généralisation (en anglais: overgeneralization ou oversimplification) est l'opposé du surapprentissage. . Considérant que, le surapprentissage désigne l'émergence de candidats de solutions trop complexes, les solutions issues de la surgénéralisation ne sont pas assez compliquées (Voir figure II.5). Bien qu'elles semblent bien adaptées et représentent les échantillons d'apprentissage utilisés au cours du processus d'optimisation, elles ne parviennent pas à donner de bons résultats pour les cas ne font pas partie des exemples d'apprentissage. [Weise 2009]

Une cause commune de la surgénéralisation est que l'ensemble d'apprentissage ne représente qu'une fraction de l'ensemble d'entrées possibles. De telle couverture incomplète peut échouer à représenter quelques caractéristiques des données qui peut ensuite mener à des solutions simplifiées. Une autre raison possible pour la sur-

simplification est que la structure de la de fitness peut conduire à la convergence prématurée et empêcher l'optimiseur au surpassant d'un certain niveau de qualité des candidats de la solution. [Weise 2009]

II.4.2.2 Dilemme biais-variance

Le meilleur modèle réalise un compromis entre la performance de l'apprentissage et la qualité de la généralisation. Si le modèle postulé est trop peu complexe, l'apprentissage et la généralisation sont peu performants ; si le modèle est trop complexe, l'apprentissage est satisfaisant, mais la généralisation ne l'est pas. Ce compromis entre la qualité de l'apprentissage et celle de la généralisation, gouverné par la complexité du modèle est connu sous le terme de *dilemme biais-variance*. Le dilemme *biais-variance* constitue un problème central pour la pratique de l'apprentissage [Dreyfus 2008]. Un modèle qui a un *biais* faible apprend très bien les points de l'apprentissage, mais il peut avoir une *variance* élevée car il peut être fortement tributaire de détails de l'ensemble d'apprentissage (modèle sur-ajusté). En revanche, un modèle peut avoir un *biais* élevé (il n'apprend pas parfaitement les éléments de l'apprentissage) mais une variance faible (il ne dépend pas des détails de l'ensemble d'apprentissage). [Dreyfus 2008].

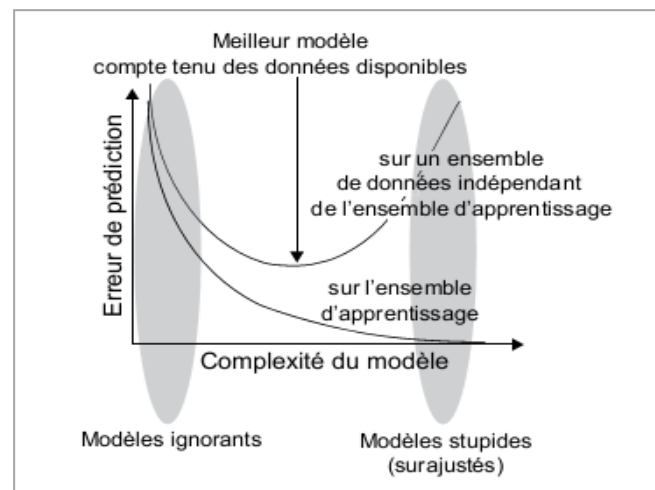


Figure II.7 Représentation symbolique du dilemme biais-variance [Dreyfus 2008]

La **figure II.7**, résume la problématique du dilemme biais-variance. Le meilleur modèle constitue un compromis entre l'ignorance (modèles incapables d'apprendre) et la stupidité (modèles surajustés qui apprennent très bien et sont capables de généraliser).

II.4.2.3 Sur-apprentissage / Sur-généralisation, Biais / Variance et Optima locaux:

Nous illustrons ces issues à l'aide de la **figure II.8**. Dans celle-ci, la fonction g , qui représente le modèle idéal, est supposée inconnue ; la fonction f est une approximation de g ; les croix représentent les éléments de l'ensemble d'apprentissage.

Un modèle ayant un *biais trop faible* génère une *variance trop forte*. Dans l'exemple de la **figure II.8** (partie **a.**), la fonction f s'ajuste de manière presque parfaite aux données d'apprentissage : son biais est quasiment nul. Toutefois sa forme varie

beaucoup trop, car elle est totalement en fonction des données. Ce risque représente le phénomène de *surapprentissage* (surajustement).

Un modèle ayant un *biais trop fort* possède une *faible variance*. Dans l'exemple de la figure II.8 (partie b.), la fonction affine f s'écarte énormément du modèle idéal, son biais est très important. Toutefois cet écart ne dépend que très peu de l'ensemble d'apprentissage, sa variance est donc faible. Ce risque s'appelle la *surgénéralisation*.

Un bon modèle génère une fonction ayant un bon équilibre entre ces paramètres. La fonction f de la figure II.8 (partie c.), a un comportement proche de l'idéal. Son biais et sa variance sont faibles, le modèle donnera une réponse moyenne satisfaisante (bonne capacité de généralisation) tout en dépendant le moins possible de l'échantillon d'apprentissage (bonne capacité de généralisation).

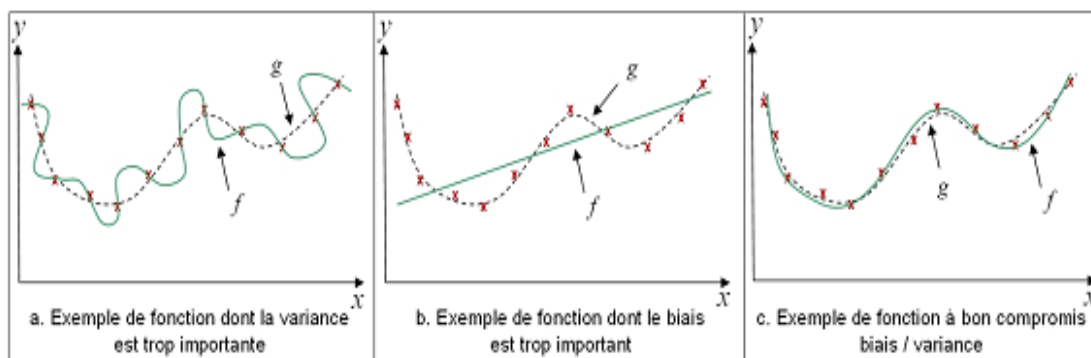


Figure II.8 : Illustration portant sur le compromis biais / variance

Une question qui doit être aussi clarifiée est de savoir si le *biais* et la *variance* (*surapprentissage/surgénéralisation*) sont liés aux *optima locaux*, ou l'effectivité d'un algorithme à trouver une bonne solution. Selon [Sheng 1999], le biais, ne dépend que de la structure d'un modèle de classification (Par exemple : dans le cas des réseaux de neurones, le biais ne dépend pas du choix des valeurs de poids). Ainsi, le biais est indépendant de tout algorithme d'apprentissage.

Etant données un ensemble de données d'apprentissage D de taille N , on peut trouver toujours un modèle correspondant à $f_D(x, w)$. [Sheng 1999]. Il est supposé que $f_D(x, w)$ correspond à un minimum global de la fonction d'erreur i.e. : qu'il est le meilleur modèle qui peut être obtenue à partir d'un ensemble de données d'apprentissage. Pour définir la variance pour une application et un algorithme donné, le résultat théorique peut perdre sa généralité. [Sheng 1999].

En pratique, pour un ensemble d'apprentissage donné et une fonction d'erreur choisie, différents algorithmes résulte de différents modèles, correspondant à différents $f_D(x, w)$. Si un algorithme (par exemple : AG) est toujours mieux de trouver une solution globalement optimale, et un autre algorithme (par exemple : la descente de gradient) qui est le plus souvent bloqué à un minimum local, alors, le premier aura une variance plus petite que celui du dernier cité. [Sheng 1999].

II.4.3 Méthodes utilisées pour la résolution du PPC

II.4.3.1 Arbre de décision (A.D) :

Etant donné un ensemble X de N exemples notés x_i . Chaque exemple x est étiqueté, c'est-à-dire qu'il lui est associée une classe que l'on note $y \in Y$. A partir de ces exemples, on construit un arbre dite de décision tel que [PREUX 2011]:

- Chaque nœud correspond à un test sur la valeur d'un ou plusieurs attributs.
- Chaque branche partant d'un nœud correspond à une ou plusieurs valeurs de ce test.
- A chaque feuille est associée une valeur de l'attribut cible.

Un arbre de décision peut être exploité de différentes manières [PREUX 2011]:

1. En y classant de nouvelles données ;
2. En en extrayant un jeu de règles de classification concernant l'attribut cible.
3. En interprétant la pertinence des attributs.

Voir *Annexe B* pour plus de détails sur cette technique de classification. Il existe principalement deux méthodes de contrôle du problème de surapprentissage pour la méthode d'arbre de décision.

a)- Approche pré-élagage (pre-pruning):

Le principe de cette première stratégie consiste à fixer quelques règles d'arrêt qui permet de stopper la construction de l'arbre lors de la phase d'élaboration. Il y avait récemment beaucoup d'effort qui a mis l'accent sur l'amélioration des méthodes de *pré-élagage*:

- [Kohavi , 1996] a proposé le classifieur NBTree (a hybrid of decision-tree and naive-classifiers) . Le système NBTree fournit quelques règles d'arrêt prématuré en comparant deux alternatives : - le partitionnement de l'espace des instances VS l'arrêt de la partition et produire un seul classifieur bayésien.
- [Zhou 2002] a suggéré une approche d'arbre de décision hybride pour la croissance d'un arbre de décision binaire. Le réseau de neurones artificiels est utilisé pour déterminer par la suite certaines règles d'arrêt prématuré.
- [Cohen 2007] a proposé une approche pour la construction d'un AD en utilisant un critère d'homogénéité pour le partitionnement de l'espace.

Cependant, les approches ci-dessus ont une difficulté dans le choix de la valeur seuil pour un arrêt prématuré. Une valeur qui est trop élevée peut entraîner un problème de sous-apprentissage, alors qu'une valeur trop faible de seuil peut ne pas être suffisante pour surmonter le problème de surapprentissage des modèles.

b)- Approche de post-élagage (post-pruning):

Cette approche consiste à construire le modèle (l'arbre de décision) complet, puis élaguer quelques partitions de l'arbre. Selon les approches de post-élagage (*REP*, *PEP*)

décrites dans [Quinlan 1987], le processus d'élagage élimine certaines partitions de l'arbre. (Pour plus de détaille voir *Annexe B*).

- **REP (Reduced Error Pruning)** : le principe de cette technique d'élagage se fait comme suit : Tant qu'il existe un arbre que l'on peut remplacer par une feuille sans faire croître l'estimation de l'erreur réelle alors élaguer cet arbre.
- **PEP (Pessimistic Error Pruning)** : L'arbre est élagué en examinant le taux d'erreur à chaque nœud et assumant que le vrai taux d'erreur est considérablement pire. La préoccupation centrale de cet algorithme, est de minimiser cette estimation, en considérant ce taux d'erreur comme une version très optimiste du taux d'erreur réel.

Dans [Mededjel 2007], les auteurs proposent une autre approche de post-élagage indirecte qui porte sur les règles générées à partir d'un arbre de décision. Leurs approche utilise des critères de validation inspirés de la technique de découverte des règles d'association telle que : le support '*S*', la confiance '*C*', l'intérêt '*I*' et la fiabilité '*F*'. cette approche est illustré dans l'**Algorithme II.1** suivant.

Etant donnée la règle : $A \Rightarrow B$. Avec : **A** : condition, **B** : résultat. Alors :

$$S = \text{fréquence}(\text{condition} \wedge \text{résultat}) \quad ; \quad C = \frac{\text{fréquence}(\text{condition} \wedge \text{résultat})}{\text{fréquence}(\text{condition})}$$

$$I = \frac{\text{fréquence}(\text{condition} \wedge \text{résultat})}{\text{fréquence}(\text{condition}) \times \text{fréquence}(\text{résultat})} \quad ; \quad F = |\text{Confiance} - \text{fréquence}(\text{résultat})|$$

Algorithme II.1 Post-élagage indirecte

1. Générer l'ensemble des règles à partir de l'A.D.
 2. Pour chaque règle, on applique les mesures d'association (*S*, *C*, *I*, *F*), et suivant les résultats obtenus :
 - Une règle est élaguée (supprimée), si $I < 1$.
 - Sinon, pour chaque règle, appliquer ces même mesure sur chaque item de la condition, un item est élaguée s'il n'améliore pas l'*I* de sa règle.
 3. Recalculer les mêmes mesures pour les nouvelles règles puis les trier selon les valeurs descendantes en considérons les métriques (*I*, *F*, *C*, *S*) respectivement. Les nouvelles règles seront appliquées par priorité.
-

L'augmentation de la généralisation n'est pas toujours nécessaire et elle n'est pas toujours bénéfique. Un arrangement plus complexe de partitions a été prouvé d'augmenter la complexité des arbres de décision. En outre, le traitement de la généralisation d'un A.D peut conduire à un problème de surgénéralisation car les conditions d'élagage sont basées sur des informations localisées [Pham 2007].

II.4.3.2 Approche **K** plus proches voisins (**Kppv**) :

La méthode *k* plus proches voisins (*k-NN*: *k*-Nearest Neighbor), est l'un des méthodes de classification les plus simples. Un exemple est prédit de classe *C* si la

classe la plus représentée parmi ses k voisins est la classe C . Pour plus de détails sur cette méthode de classification, voir **Annexe B**.

Le choix de la valeur du paramètre K , affecte de façon directe la performance de cette approche. Une mauvaise valeur (du paramètre K) peut conduire à des problèmes de sur-apprentissage et sur-généralisation [Tan 2005]. Par exemple, sur la **figure II.9**, si $k = 1;2;3$, l'exemple à classifier (noté "?") serait prédit comme étant de la classe "1" (notée par 'X'), mais si $k= 5$, il serait prédit comme étant de la classe "2" (notée par des 'O').

L'un des moyens de réduire l'impact du paramètre K , consiste à pondérer l'influence des voisins les plus proches en fonction de leur distance par rapport au point de test (exemple: *DWV*: Distance Weighted Voting [Dudani 1976] ; *F-KNN* : Fuzzy k-Nearest Neighbor [Keller 1985] et [Qu 2005] .

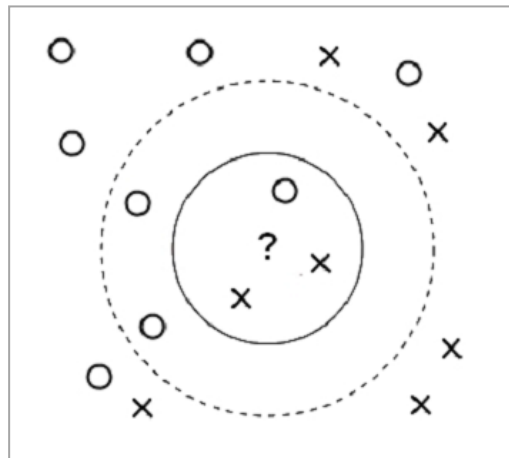


Figure II.9 : Schéma d'une classification par l'approche Kppv

Cependant, l'utilisation du classifieur du *K-plus proche voisin* a ses inconvénients. Le classement d'un exemple de test peut être très coûteuse due aux calculs du degré de similarité entre le point de teste et chaque point d'apprentissage. Ceci est à cause de ces points qui sont basés sur des informations localisées uniquement. Enfin, il est difficile de trouver une valeur appropriée pour le paramètre K afin d'éviter les problèmes du surapprentissage et surgénéralisation.

II.4.3.3 Réseaux de neurones artificiels (R.N.A)

Un réseau de neurone artificiel est un modèle composé d'éléments simples (neurones) inspirés par le système nerveux biologique. Le fonctionnement du réseau (de neurone) est fortement influencé par la connections des éléments entre eux qui peut être entraîné en ajustant les valeurs des connections (ou poids) entre les éléments (neurone).

En général, un *RNA* dispose d'un ensemble de nœuds d'entrée $[X_1, X_2, \dots, X_m]$ et un nœud de sortie Y . Etant donnée n valeurs pour m -tuple $[X_1, X_2, \dots, X_m]$. Aussi, soit les sorties réelles $[Y'_1, \dots, Y'_n]$ et $[Y_1, Y_2, \dots, Y_n]$ sont les sorties désirées. Soit $E = \sum_{i=1}^n |Y_i - Y'_i|^2$ désigner la somme totale des différences carrées entre les sorties réelles et désirées. Le but de *RNA* est de déterminer un ensemble des poids, afin de minimiser la valeur de E . Pendant la phase d'apprentissage d'un *RNA*, les paramètres

des poids sont ajustées jusqu'à ce que les sorties du perceptron deviennent compatibles avec les sorties des points d'apprentissage. Dans le processus de mise à jour de poids, ces poids ne doit pas être changé aussi considérablement parce que E est calculée uniquement pour le point de l'apprentissage actuelle.

Afin d'éviter les problèmes de surapprentissage et surgénéralisation, la conception d'un *RNA* doivent être pris en considération. Un réseau qui n'est pas suffisamment complexe peut échouer à détecter entièrement une entrée dans une base de données complexe, conduisant à une surgénéralisaion. D'autre part, un réseau qui est trop complexe peut non seulement s'adapter (fit) a l'entrée, mais aussi les points bruyants, conduisant ainsi à un problème de surapprentissage.

Selon [Moody 1992]. Plus le nombre de poids augmente, relative au nombre des d'exemples d'apprentissage, plus le surapprentissage amplifie le bruit dans les systèmes de classification. La réduction de la taille des poids [Pham 2007] peut réduire le nombre effectif des poids menant à la régularisation (weight decay) [Moody 1992] et l'arrêt prématuré (early stopping) [Weigend 1993]. Dans ce qui suit, nous présentons le principe de ces deux techniques.

- Arrêt prématuré (Early Stopping) :

Dans la plupart des documents qui introduit l'apprentissage supervisé des réseaux neurones, on peut trouver un schéma similaire à celui de la **figure II.10**. Cette figure montre l'évolution dans le temps des erreurs calculer sur un ensemble d'apprentissage et un ensemble de validation (la courbe d'erreur d'apprentissage et la courbe d'erreur de validation). Cependant, l'erreur commise sur un ensemble de validation pour des applications réelles, n'évolue pas aussi bien comme le montre la **figure II.10**, mais ressemble plus à la **figure II.11**.

Un arrêt prématuré consiste à utiliser la procédure suivante : [Prechelt 1998]

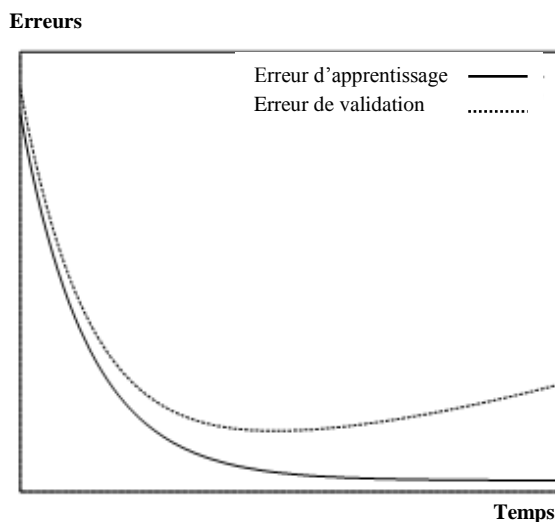


Figure II.10 courbes d'erreurs d'apprentissage et de validation [Prechelt 98]

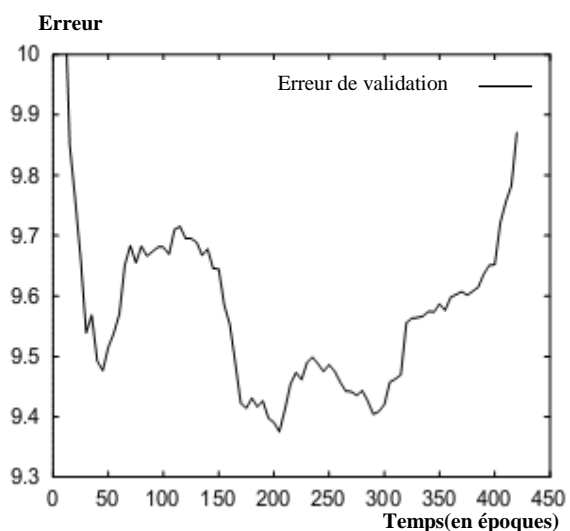


Figure II.11 courbe réelle d'erreur de validation [Prechelt 98]

- Subdiviser la base d'apprentissage en deux : ensemble d'apprentissage et ensemble de validation.

- Faire l'apprentissage sur l'ensemble d'apprentissage et évaluer de temps en temps l'erreur sur l'ensemble de validation (ex : après chaque 5 époques).
- Arrêter l'apprentissage dès que l'erreur sur l'ensemble de validation est plus élevé qu'il l'était la dernière fois vérifié.
- Utiliser les paramètres issus de l'étape précédente comme résultats de l'exécution l'apprentissage.

L'arrêt prématuré (Early stopping) est largement utilisé parce qu'il est simple à comprendre et à implémenté .Cependant, ce qu'il nous faut est de trouver un prédicat qui nous dit quand arrêter l'apprentissage. Nous appelons ce prédicat un critère d'arrêt (Stopping Criteria) . [Prechelt 1998]

- Modération des poids (weight decay):

La méthode de modération des poids, consiste à ajouter un terme à la fonction de coût usuelle. La **figure II.12** montre, pour différentes architectures, l'évolution de la somme des carrés des poids pendant l'apprentissage. Pendant l'apprentissage, certains paramètres des réseaux à 5 et 10 neurones cachés augmentent et finissent par atteindre des valeurs très élevées. Ce n'est pas le cas pour le réseau à deux neurones cachés. La méthode de régularisation par modération des poids a précisément pour objectif d'empêcher les paramètres de prendre des valeurs exagérées, en minimisant, pendant l'apprentissage, une fonction de coût J' qui rassemble entre la fonction de coût et d'un terme de régularisation proportionnel à la somme des carrés des paramètres : [Dreyfus 2008]

$$J' = J + \frac{\alpha}{2} \sum_{i=1}^p w_i^2 \quad (2.16)$$

Où p est le nombre de paramètres du réseau, et α est un paramètre dont la valeur détermine l'importance relative à la pénalité. Cette méthode présente l'avantage d'être simple à mettre en œuvre, puisque le gradient de J' se calcule très simplement à partir du gradient de J et du vecteur des poids du réseau w : [Dreyfus 2008]

$$\nabla J' = \nabla J + \alpha w \quad (2.17)$$

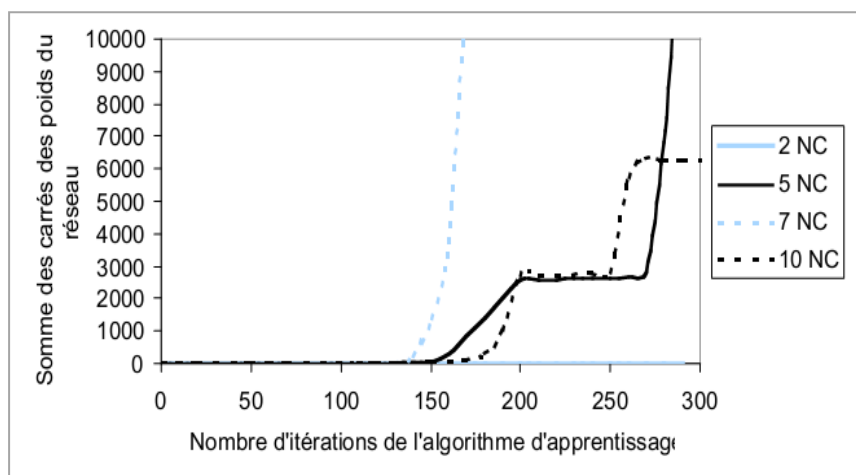


Figure II.12 : Évolution de la moyenne des carrés des poids en fonction du nombre d'itérations de l'algorithme d'apprentissage. [Dreyfus 2008]

Il suffit d'ajouter la quantité αw au vecteur ∇J calculé par l'algorithme de rétro-propagation. [Dreyfus 2008]

En résumé, les *RNAs* présentent les inconvénients suivants :

- Il est difficile de trouver une topologie de réseau approprié à un problème donné afin d'éviter les problèmes de surapprentissage et surgénéralisation.
- Besoin de temps pour faire apprendre un *RNA* lorsque le nombre de nœuds cachés augmente.

II.5 Présentation des travaux selon les méthodes de classification

Dans cette section, nous présentons un **tableau II.2** qui résume les recherches consacrées à quelques problèmes liés à la performance des classifieurs pour certaines méthodes de classification supervisée. La première et Quatrième colonnes de ce tableau montrent le nom de la méthode de classification et les références des travaux dédiés qui sont extraits de la littérature. La deuxième et la troisième colonne présente les problèmes discutés pour chaque méthode de classification ainsi que les méthodes de résolution adoptées. Nous constatons que les facteurs affectant plus la performance des classifieurs est la recherche d'un bon compromis entre les phénomènes du sur-apprentissage et sous-apprentissage.

Méthode de classification	Problème étudié	Méthode de résolution	Références
Arbre de décision (A.D)	Taille de la l'arbre	<p>1- <i>Pré-élagage</i> :</p> <ul style="list-style-type: none"> *NB-Tree (Naïve Bayes and Decision Tree) *HDT (Hybrid Decision Tree). * DFID (Decision-tree Framework for Instance-space Decomposition) <p>2- <i>Post-élagage</i> :</p> <ul style="list-style-type: none"> *REP (Reduced Error Pruning) *PEP: (Pessimistic Error Pruning. *Mesure d'association (Post-élagage indirect) 	<p>[Kohavi 1996]</p> <p>[Zhou 2002]</p> <p>[Cohen 2007]</p> <p>[Quinlan 1987]</p> <p>[Quinlan 1987]</p> <p>[Mededjel 2007]</p>
K-plus proche voisin (Kppv)	Choix du paramètre K	<ul style="list-style-type: none"> * The Distance-Weighted k-Nearest-Neighbor Rule. * A Fuzzy K-Nearest Neighbor Algorithm * An improved KNN Algorithm – Fuzzy KNN. 	<p>[Dudani 1976]</p> <p>[Keller 1985]</p> <p>[Qu 2005]</p>
Réseaux de neurones (R.N)	Choix d'une topologie approprié pour un problème donné.	<ul style="list-style-type: none"> * Weight Decay. * Early Stopping 	<p>[Moody 1992]</p> <p>[Weigend 1993]</p>

Tab II.2 Principaux travaux pour la résolution du PPC dans certaines méthodes de classification

II.6 Bilan et synthèse

Dans ce chapitre, nous avons décrit successivement les pré-requis nécessaires à la bonne compréhension de la notion de performance des classifieurs. Des problèmes liés à la performance sont aussi discutés, une revue de littérature est présentée proposant quelques techniques pour la résolution du problème de la performance pour certaines techniques de classification.

Ce chapitre nous a permis essentiellement d'avoir une vue générale sur les méthodes de résolution dédiées à l'amélioration des performances des classifieurs. Nous avons décelé les limites suivantes par rapport aux objectifs que nous nous sommes fixés :

- Des modèles telles que les réseaux de neurones *RNAs* souffrent de problème d'interprétabilité, malgré leur capacité d'apprentissage et de généralisation.
- Difficulté de développement d'un modèle approprié pour un problème donné : Un modèle qui n'est pas suffisamment complexe conduit à une surgénéralisation. A l'inverse, un modèle trop complexe peut non seulement s'adapter (fit) à l'entrée, mais aussi aux points bruyants, conduisant à un problème de surapprentissage.
- Minimisation de cout totale d'échecs de classification et augmentation du taux de reconnaissance.
- Recherche d'un bon moyen pour une balance simultanée entre le « fitting » et la « généralisation » au lieu d'un control séparé.
- La prise en considération des cas non-classifiés. Dans le champ médical, un cas non-classifié est bien requis. Par exemple, un patient peut être atteint/ou non d'un type de cancer, ou il s'agit d'un cas non-classifié (i.e. : le médecin ne peut pas décider si un patient a un cancer ou pas).Malheureusement, les méthodes de classification courant ignore les couts de pénalités pour les cas non-classifiés (Unclassifiable type).
- Les approches de data-mining (médicale) courent, souvent assignent des couts de pénalités identiques pour les cas de faux positifs (*FP*) et négatifs (*FN*),

Au vue des ces limites, il nous semble que certaines voies de recherche telles que les métaheuristiques et la théorie de Zadah peuvent se révéler fructueuses, particulièrement, en ce qui concerne l'interprétabilité et la résolution de problèmes d'optimisation des classifieurs. C'est pourquoi nous avons décidé de développer deux modèles : un premier modèle appelé *ProSadm-HBA* (la métaheuristique *HBA*: *Homogeneity Based Algorithm* est utilisée en combinaison avec notre ancien système *ProSadm* : *Programmation d'un Système d'Aide au Diagnostic Médical* [Bekaddour 2012.a]) ,et un deuxième modèle nommé *F-HBA* (*Fuzzy Homogeneity Based Algorithm*) ,pour la contribution à la résolution des problèmes des performances des classifieurs. Les spécificités de ces deux contributions seront exhaustivement détaillées dans le chapitre 4. Mais avant cela, dans le chapitre 3, nous présentons les méthodes et les approches adoptés dans ces deux modèles. Nous donnons également, de manière plus synthétique nos arguments d'utiliser chaque méthode.

Deuxième partie

Approches proposées

Chapitre III :

Méthodes & Approches Adoptées

Ce chapitre comporte trois parties. La première présente le système *ProSadm* (*Programmation d'un Système d'Aide au Diagnostic Médical*). La deuxième partie introduit des notions de bases sur la logique *Floue* et les modèles *Neuro-Flous*. La troisième présente la métaheuristique *A.B.H* (*Algorithme a Base d'Homogénéité*).

III.1 Introduction

Dans les chapitres précédents, nous avons dressé une étude bibliographique exhaustive concernant les problèmes des performances des classifieurs, et quelques méthodes de résolution qui leur ont été dédiées. Nous avons conclu par un travail de synthèse et critique de cet état de l'art qui nous a permis de situer notre travail de magister au sein des travaux existants, de montrer certaines limites et insuffisances quant aux approches de résolution du problème étudié, de mettre en avant son analogie avec les problèmes connus de : surapprentissage , surgénéralisation et d'interprétabilité des modèles de classification , et en conséquence de justifier nos choix et propositions quant à la modélisation et la résolution de ces problèmes. Dans le reste de cet mémoire nous allons alors nous intéresser aux différentes méthodes et concepts adoptés dans nos modèles proposés : *ProSadm-HBA* et *F-HBA* (*Fuzzy Homogeneity Based Algorithm*) , à savoir :

- L'application *ProSadm* (*Programmation d'un Système d'Aide au Diagnostic Médical*) : étant notre travail d'ingénieur dédié à la réalisation d'un outil d'aide au diagnostic médical ([*Bekaddour 2012.a*]).

- Le système d'inférence flou & L' *ANFIS* (*Adaptative Neuro-Fuzzy Inference System*) : étant la théorie de la logique Floue et le modèle Neuro-Flou .

- La métaheuristique *A.B.H* (*Algorithme a Base d' Homogénéité*) [ou *H.B.A* (*Homogeneity Based Algorithm*)] : étant une méthode d'optimisation approchée. Nous expliquons également la nature et l'utilité de leur alliance par rapport à notre modèle proposé.

III.2 . Partie I : Le système ProSadm

(Programmation d'un Système d'Aide au Diagnostic Médical) :

Artificial intelligence is the science of making machines do things that would require intelligence if done by men.

– Marvin Minsky –

III.2.1 Description du système

Les techniques de l'intelligence artificielle sont de plus en plus employées dans le développement des systèmes dit intelligents. Ces techniques sont utilisées pour élaborer des applications dans de nombreux domaines. En particulier, l'introduction de l'IA dans le domaine médical a permis, non seulement de faciliter la tâche aux experts humains, mais également l'automatisation de la prise de décision. Dans [Bekaddour 2012.a], nous avons développé un système nommé *ProSadm* qui constitue un premier pas vers un système d'aide au diagnostic médical basé sur des approches neuronales (*P.M.C (Perceptron Multi-Couches)*, *Q.V.A(Quantification Vectorielle Adaptative)*)).

L'application *ProSadm (Programmation d'un Système d'Aide au Diagnostic Médical)* est composée de trois niveaux essentiels :

- ✓ *Niveau noyau* : cette couche possède une instanciation des objets à utiliser et de structures de données avec leurs initialisations d'une part, et d'implémentation permettant l'interaction entre les différents objets.

- ✓ *Niveau fichier* : ceci concerne la création, l'ouverture, l'enregistrement des réseaux neuronaux, les résultats du test et de la réduction de la base qui sont sous forme de fichiers d'extension (*.mat), ainsi que le chargement des bases pour l'apprentissage ou le test et qui peuvent être d'extension (*.mat' ou *.m').

- ✓ *Niveau interface* : il ne s'agit pas uniquement d'un intermédiaire entre l'utilisateur et notre application mais aussi d'une interaction avec les deux autres couches situées aux dessus. Notre application comporte deux interfaces principales correspondant aux deux méthodes de classification d'un coté ; et de trois interfaces commun reflétant le chargement du résultat de test, la réduction de la base et l'interprétation des performances et des résultats finaux d'un autre coté.

Le tableau III.1 suivant présente quelques fonctions utilisées durant le codage de notre application *ProSadm* :

Nom_fonction:	Description_fonction :
Créer_Callback	Création du réseau (LVQ ou PMC)
Apprentissage_Callback	Apprentissage du réseau
Teste_Callback	Teste du réseau
Afficher_Callback	Affichage des résultats de teste
Enregistrer_Callback	Le sauvgarde des paramètres du réseau
Visualiser_Callback	L'interprétation des résultats de teste
Quitter_Callback	Libération de la mémoire des variables globales
Imprimer_Callback	Impression des résultats
Charger_Callback	Chargement de la base pour le teste et l'apprentissage
Acp_Callback	Application de la méthode d'A.C.P pour la réduction de la taille de base
Ouvrir_Callback	Le chargement du réseau et du base pour le teste

Tab III.1: Exemples de fonctions utilisées

III.2.2 Aperçu du système

La **figure III.1** suivante présente quelques interfaces de l'application *ProSadm* :



Fig III.1 : Interfaces Principales du *ProSadm*

Dans ce qui suit, nous présentons quelques notions de base sur les réseaux de neurones en particulier les modèles (*P.M.C (Perceptron Multi-Couches)*, et *Q.V.A (Quantification Vectorielle Adaptative)*) adoptés par le système *ProSadm*.

III.2.3 Méthodes utilisées

III.2.3.1 Apprentissage dans les réseaux de neurones

L'apprentissage est la caractéristique principale des réseaux de neurones. Il peut être considéré comme un problème de mise à jour des poids de connexions au sein du réseau. Il existe essentiellement trois types d'apprentissage dans les réseaux de neurones [Fife 1996] :

a- Apprentissage supervisé :

Ce type d'apprentissage est caractérisé par la présence d'un «professeur» qui possède une connaissance approfondie de l'environnement. En pratique, les connaissances de ce

professeur prennent la forme d'un ensemble de Q couples de vecteurs d'entrée et de sortie que nous noterons $\{(p_1, d_1), (p_2, d_2), \dots, (p_Q, d_Q)\}$, où p_i désigne un stimulus (entrée) et d_i la cible pour ce stimulus, c'est-à-dire les sorties désirées du réseau. Chaque couple (p_i, d_i) correspond donc à un cas de ce que le réseau devrait produire (la cible) pour un stimulus donné. Pour cette raison, l'apprentissage supervisé est aussi qualifié d'apprentissage par des exemples.

b- Apprentissage non supervisé :

Ce type d'apprentissage est caractérisé par l'absence complète du professeur. Nous ne disposons donc que d'un environnement qui fournit des stimuli, et d'un réseau qui doit apprendre sans intervention externe. L'apprentissage non-supervisé s'appuie généralement sur un processus compétitif.

c- Apprentissage par renforcement :

Il existe également certains problèmes intermédiaires, où seules quelques données sont étiquetées. Dans l'apprentissage semi-supervisé, les données d'apprentissage sont constituées de deux types d'instances : « l » étiquetées $\{(x_i, l_i)\}_{i=1}^l$ et « u » non-étiquetées $\{(x_j)\}_{j=l+1}^{l+u}$. L'objectif consiste à entraîner le classifieur à partir des données étiquetées et non étiquetées, de telle sorte qu'il est mieux entraîné le classifieur supervisé à base des données étiquetées uniquement [Zhu 2009].

III.2.3.2 Quelques types des réseaux de neurones

Le schéma III.2 suivante montre quelques types des réseaux de neurones :

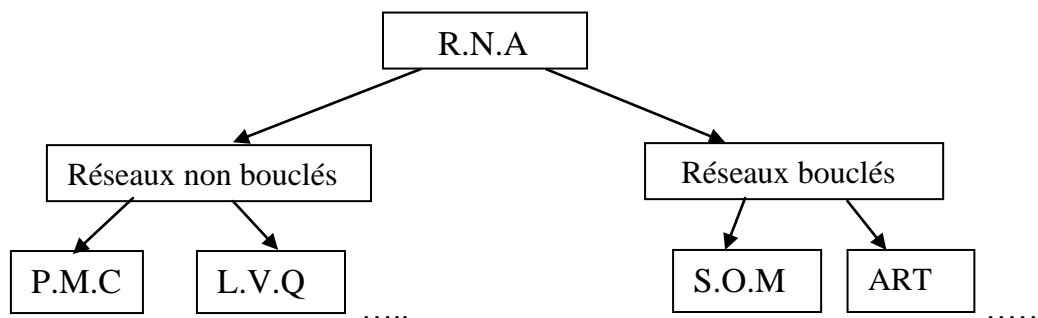


Figure III.2 : Principales types de réseaux de neurones

Dans ce qui suit, nous détaillons deux types de ces réseaux (PMC, QVA) adoptées par notre ancien système *ProSadm* (*Programmation d'un Système d'Aide au Diagnostic Médical*).

III.2.3.3 Le perceptron multicouche :

Le *PMC* avec rétro-propagation est l'un des modèles des réseaux de neurones les plus populaires [Hideyuki 1997], [Konar 99]. Un exemple du perceptron multicouche est illustré dans la figure III.3 au-dessous. La couche la plus à gauche représente la couche d'entrée, elle n'effectue aucun calcul, contrairement aux éléments des autres couches. La couche centrale est la couche cachée (il peut y avoir plus d'une couche cachée dans des réseaux complexes). La couche la plus à droite de neurones est la

couche de sortie, qui produit des résultats. Il n'y a pas d'interconnexions entre les neurones dans la même couche, mais tous les neurones dans une couche donnée sont entièrement connectés aux neurones dans les couches adjacentes. Ces interconnexions sont associées à des valeurs numériques W_{JM} qui sont ajustées au cours de la phase d'apprentissage. [Tso 2009]

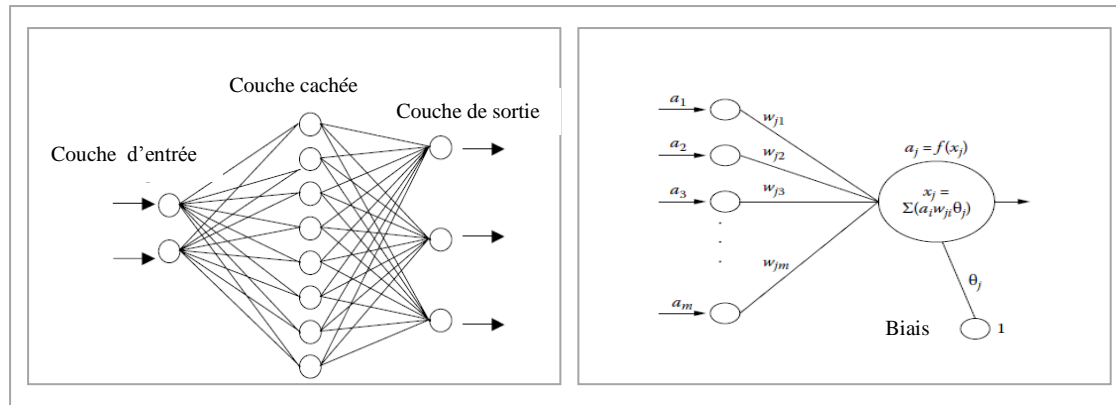


Figure III.3 (a) Exemple de PMC à trois couches. [Tso 2009] (b) Exemple du procédé de propagation en avant du neurone j [Tso 2009]

L'apprentissage du P.M.C comporte deux passes définies dans la **figure III.4** suivante :

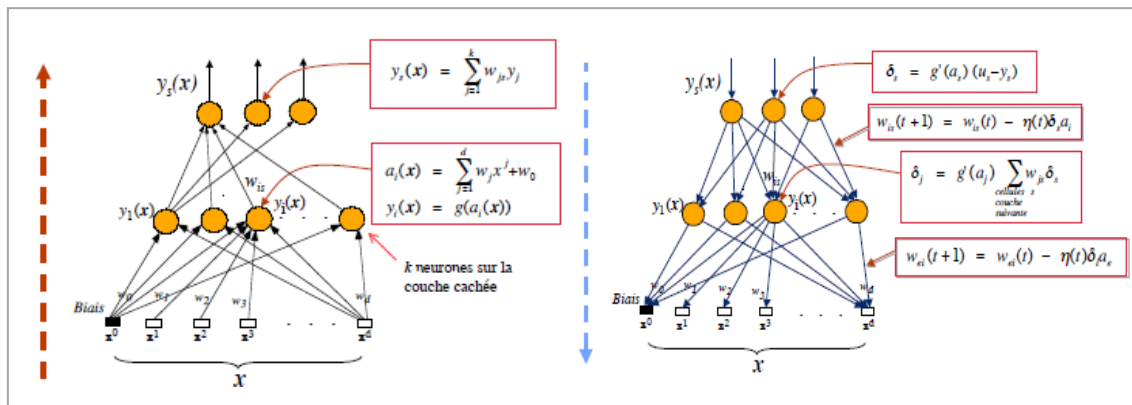


Figure III.4 Passes Avant & Arrière du P.M.C

Les étapes d'apprentissage du perceptron multicouche se déroulent comme suite :

1. Présentation d'un exemple parmi l'ensemble d'apprentissage.
2. Calcul de l'état du réseau.
3. Calcul de l'erreur = fct (sortie réelle - sortie désirée) (e.g. = $(y^l - u^l)^2$).
4. Calcul des gradients par l'algorithme de rétro-propagation de gradient
5. Modification des poids synaptiques
6. Critère d'arrêt : Seuil d'erreur. Nombre de présentation d'exemples,
7. Retour en 1

➤ *Apprentissage par Rétropropagation :*

Une des principales capacités d'un réseau de neurones est d'apprendre, des modèles de lois d'apprentissage ont été réalisés dont la rétro-propagation est la plus connue. Cette technique est basée sur la propagation de l'erreur.⁶

⁶ http://82.245.102.21/html/reseau_de_neurones [62]

La puissance de cet algorithme vient de sa capacité d'ajuster les poids de connexion pour effectuer une tâche particulière. Pour ce faire, cet algorithme utilise une série d'exemples pour l'apprentissage ou chaque vecteur d'entrée est associé à une réponse désirée ou un vecteur de sortie. Ces exemples sont traités un par un par le réseau et les poids sont ajustés en conséquence.

Les étapes de cette algorithme sont décrit au-dessous dans la **figure III.5**, Ils sont présentés dans l'ordre dont ils seraient utilisés pendant l'apprentissage : [freeman 1991]

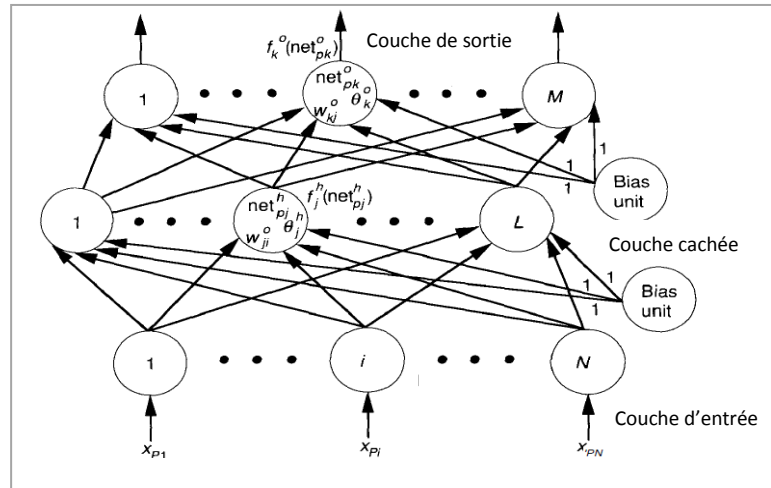


Figure III.5 : Architecture à trois couches du réseau de rétro-propagation [freeman 91]

- Appliquer le vecteur d'entrée, $x_p = (x_{p1}, x_{p2}, \dots, x_{pN})^T$ à la couche d'entrée.
- Calculer les valeurs reliant l'entrée du réseau aux unités couche cachée:

$$\text{net}_{pj}^h = \sum_{i=1}^N w_{ji}^h x_{pi} + \theta_j^h \quad \text{Avec } \theta : \text{biais de poids} \quad (3.1)$$

- Calculer les sorties de la couche cachée: $i_{pj} = f_j^h(\text{net}_{pj}^h)$ (3.2)
- Déplacer vers la couche de sortie. Calculer la valeur d'entrée à chaque unité :

$$\text{net}_{pk}^o = \sum_{j=1}^L w_{kj}^o i_{pj} + \theta_k^o \quad (3.3)$$

- Calculer les sorties : $o_{pk} = f_k^o(\text{net}_{pk}^o)$ (3.4)
- Calculer l'erreur pour les unités de sortie : $\delta_{pk}^o = (y_{pk} - o_{pk}) f_k^{o' }(\text{net}_{pk}^o)$ (3.5)
- Calculer l'erreur pour les unités caches $\delta_{pj}^h = f_j^{h' }(\text{net}_{pj}^h) \sum_k \delta_{pk}^o w_{kj}^o$ (3.6)

- Ajuster les poids de la couche de sortie : $w_{kj}^o(t+1) = w_{kj}^o(t) + \eta \delta_{pk}^o i_{pj}$ (3.7)
- Ajuster les poids de la couche cachée : $w_{ji}^h(t+1) = w_{ji}^h(t) + \eta \delta_{pj}^h x_i$ (3.8)

Ensuit : calculer l'erreur :
$$E_P = \frac{1}{2} \sum_{k=1}^M \delta_{pk}^2 \quad (3.9)$$

Cette quantité est la mesure de la façon dont le réseau apprend. Lorsque l'erreur pour chaque paire des vecteurs d'apprentissage est petite, l'apprentissage peut être interrompu.

III.2.3.4 Quantification vectorielle Adaptative (Q.V.A):

C'est un algorithme d'apprentissage supervisé inventé par Kohonen en 1988 [Hassoun 96]. Il existe plusieurs versions de cette algorithme telle que : *Q.V.A1*, *Q.V.A2*....

Le vecteur de poids pour une unité de sortie est souvent appelé vecteur « codebook » de la classe qui représente l'unité.

Soit $\{W_j\}$ dénote l'ensemble des vecteurs Voronoi et $\{X_i\}$ un vecteur d'entrée. L'exemple de la **figure III.6** illustre ce principe [Haykin 99].

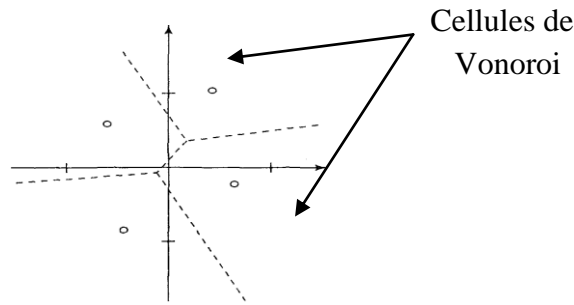


Figure III.6 : Exemple de diagramme de Voronoi [Haykin 99].

Si on suppose que C_{wi} représente la classe du vecteur de Voronoi et C_{xi} dénote la classe du vecteur d'entrée X_i alors le poids de Voronoi W_c est ajusté comme suite :

- Si $C_{wi} = C_{xi}$ alors : $w_c(n + 1) = w_c(n) + \alpha_n [x_i - w_c(n)] \quad 0 < \alpha_n < 1 \quad (3.10)$

- Sinon $w_c(n + 1) = w_c(n) - \alpha_n [x_i - w_c(n)] \quad (3.11)$

Les autres vecteurs Voronoi sont inchangables. La **figure III.7** montre ce changement de poids.



Figure III.7 : Avant l'apprentissage Après l'apprentissage

Le réseau *Q.V.A* est un réseau hybride composé d'une couche compétitive et autre linéaire.[Demuth 98]. L'architecture d'un réseau de neurones *Q.V.A* est présentée dans la **figure III.8** suivante :

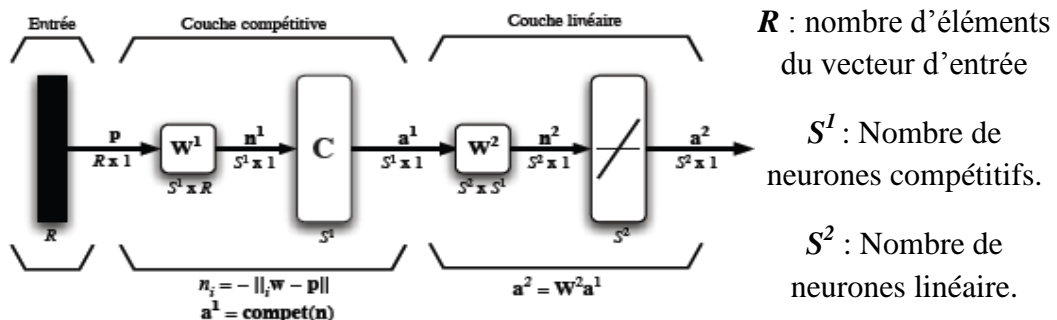


Figure III.8: Architecture du réseau Q.V.A (L.V.Q)

La couche linéaire transforme les classes du couche compétitive en catégorie cible définie par l'utilisateur. Les classes appris par la couche compétitive sont référées sous « subclasses » et les classes de la couche linéaire sont des classes cibles.

La motivation de l'algorithme d'apprentissage pour le réseau *Q.V.A* est de trouver l'unité de sortie la plus proche du vecteur d'entrée. Si x et w , appartiennent à la même classe, le poids doit être déplacé vers le nouveau vecteur d'entrée, par contre s'ils appartiennent à des classes différentes, alors déplacer le poids loin de ce vecteur d'entrée [Tso 09]

L'algorithme du réseau *Q.V.A* peut être résumé dans les étapes suivantes :

Etape 0 : Initialisation des poids W_i de chaque neurone d'entrée.

Etape 1 : Calcule de la distance euclidienne entre le vecteur classifié et le poids W_i :

$$\|X - W_i\|,$$

Etape 3 : Sélection du neurone i ayant la plus faible distance.

Etape 4 : Modification du poids W_i selon la formule suivante :

- Si $T = C_i$ alors W_i (nouveau) = W_i (ancien) + $\alpha [X - W_i$ (ancien)]
- Si $T \neq C_i$ alors W_i (nouveau) = W_i (ancien) - $\alpha [X - W_i$ (ancien)]

Pour les autres neurones, W_i est inchangeable : $W_{i+1} = W_i$

Les étapes 1-4 seront répétées jusqu'à atteindre un critère d'arrêt (nombre époques, ...)

La méthode *Q.V.A* (*Quantification Vectorielle Adaptative*) permet d'entraîner les réseaux de neurones compétitifs de manière supervisée. Comme la classification se fait par le calcul de distance entre les poids des vecteurs d'entrée, il n'y a pas de mécanisme strict qui permettent de définir si les vecteurs d'entrés sont dans la mêmes classe ou non. Celles-ci peuvent être imposées par l'utilisateur.

Malgré que les réseaux de neurones (PMC, QVA) sont connues par leurs capacités d'apprentissage, ce dernier souffre d'un problème majeur réside dans la non interprétabilité des résultats. La logique floue [Zadeh 65] permet de pallier ce problème (boîtier noir). L'hybridation du réseau de neurone avec la logique floue permettre de bien augmenté la performance des classifieurs (performance et lisibilité des résultats).

III.3. Partie II: la logique Floue & modèles Neuro-Flous
(ANFIS- un outil hybride pour le diagnostic)

*I know it was going to be important. That much I knew .But I
never dreamed it would become a worldwide phenomenon.*

Lotfi Zadah

III.3.1 Système D'inférence Floue

Un classifieur est considéré comme « opaque » lorsque sa modélisation n'est pas interprétable et que la façon dont il est parvenu à ce résultat ne peut être expliquée a posteriori. [Chikh 2005]

On parle aussi de boîte noire, ce qui est le cas des réseaux de neurones. Les modèles flous, sont très souvent utilisés en intelligence artificielle notamment pour la reconnaissance ou la classification. D'où l'intérêt de leurs applications dans le domaine médicale, notamment lorsque le mécanisme de décision doit être interprétable pour l'utilisateur.

Un système d'inférence floue peut être classifié selon quatre catégories présentées dans le **tableau III.2** suivant : [Sharma 2011]

<i>Classe :</i>	<i>Paramètre :</i>
Logique :	Le mécanisme de raisonnement Les opérateurs flous Les types des fonctions d'appartenance La méthode de déffuzzification
Structurelle :	Le nombre des fonctions d'appartenance, des règles, des variables linguistiques
Connective :	Antécédents et conséquences des règles, ..
Opérationnel :	Les valeurs des fonctions d'appartenances.

Tab III.2 : La classification des paramètres du SIF [Sharma 2011]

- 1)- *Les paramètres logiques* : incluent les opérateurs de la logique floue appliquée pour les méthodes d'implication, d'agrégation et de defuzzification ainsi que le type des opérateurs adaptés pour And, Or...
- 2)- *Les paramètres structurels* : incluent le nombre des variables participant à l'inférence, le nombre des fonctions d'appartenance qui définit chaque variable linguistique ainsi que le nombre des règles utilisées.
- 3)- *Les paramètres connectifs* : définissent la topologie du système. Ceci incluent la connexion entre les différentes instances linguistiques (les antécédents, conclusions, et les poids des règles).
- 4)- *Les paramètres opérationnels*: Ces paramètres définissent la correspondance entre la représentation linguistique et numérique des variables.

Un système d'inférence flou, aussi appelé : le modèle flou, le système à base de règles floues, ou: le système flou est construit à partir de trois composants conceptuels :

- ✓ *Base de règles* : Une base des règles floues est composée de règles qui sont généralement déclenchées en parallèle.
- ✓ *Base de données* : définit les fonctions d'appartenance utilisées pour les règles floues.

- ✓ *Le mécanisme de raisonnement* : représente la procédure d'inférence appelée aussi la généralisation du *modus-ponens* ou raisonnement approximative.

III.3.2 Principe de fonctionnement du SIF

Généralement, un système d'inférence flou comprend trois étapes [Tso 2009], décrit dans la **figure III.9** suivante :

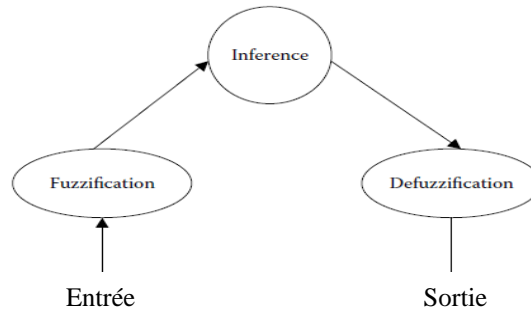


Figure III.9 : Système d'inférence flou [Tso 2009]

III.3.2.1 Fuzzification : Cette première étape détermine le degré d'appartenance de chaque variable d'entrée. Celui-ci est déterminé à l'aide des fonctions d'appartenance définies dans le système, tel est l'exemple de la **figure III.10**.

Exemple :

(NG :Négative Grand ;NM :Négative Moyen)

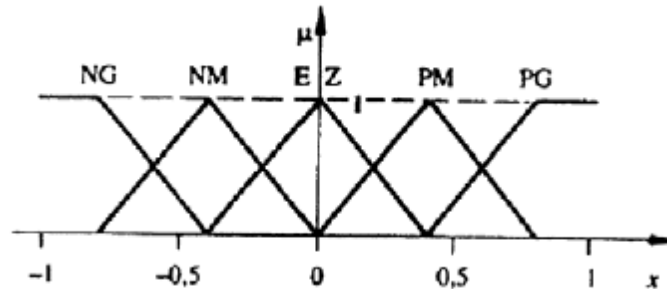


Figure III.10: Exemple de la Fuzzification

Pour $x=0.5 \rightarrow$ on associe $\mu_{PM}(0.5) = 0.75$ et $\mu_{PG}(0.5) = 0.25$

Pour $x=0.1 \rightarrow$ on associe $\mu_{EZ}(0.1) = 0.9$ et $\mu_{PM}(0.1) = 0.1$

D'où chaque variable linguistique d'entrée x fait correspondre à une valeur linguistique (NG, NM, EZ, PM, PG) avec un degré d'appartenance.

III.3.2.2 Inférence floue : donne la relation qui existe entre les variables d'entrées et les variables de sorties.

III.3.2.3 Défuzzification : Les méthodes d'inférence fournissent une fonction d'appartenance résultante pour la variable de sortie. Il s'agit donc d'une information floue qu'il faut transformer en valeur numérique.

La *fuzzification* consiste à transformer les entrées numériques en parties floues. Ceci alimente alors le mécanisme d'*inférence* qui, à partir des valeurs d'entrées et selon la *basse de connaissance*, détermine la valeur correspondante de la sortie. Enfin, la *défuzzification* joue le rôle inverse de la fuzzification, en convertissant les parties floues relatives aux sorties du mécanisme d'inférence en sorties numériques.

III.3.3 Type des SIF

Il existe principalement deux types de systèmes flous :

III.3.3.1 Système flou de Mandani :

Dans ce type de systèmes flous, la prémisse et la conclusion sont floues. Après la réalisation de l'inférence floue, une étape du « défuzzification » est obligatoire pour le passage du symbolique au numérique. Un exemple de ce SIF est illustré dans la **figure III.11** [Wu 2011].

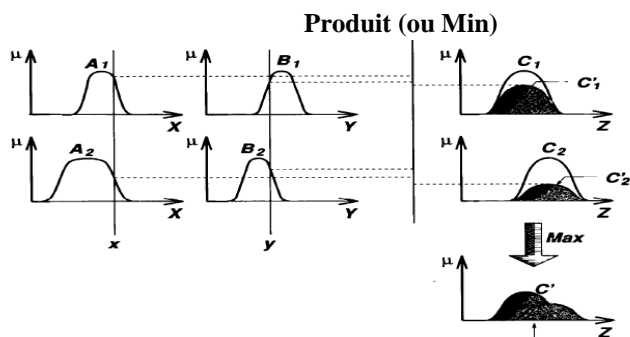


Figure III.11 : Exemple du SIF de Mandani [Wu 2011]

III.3.3.2 Système flou de Takagi-Sugeno :

Ce modèle est proposé par Takagi, Sugeno et Kang [Jang 1997]. Il est composé d'une base de règles floues de la forme :

Si x est Ai et y est Bi Alors z = f(x, y). avec f est une fonction nette.

Dans ce type de SIF, la conclusion correspond a une constante ou une expression polynomiale .Il permet d'obtenir directement la sortie défuzzifiée à partir des règles linguistiques. La **figure III.12** montre un exemple de SIF de Sugeno.

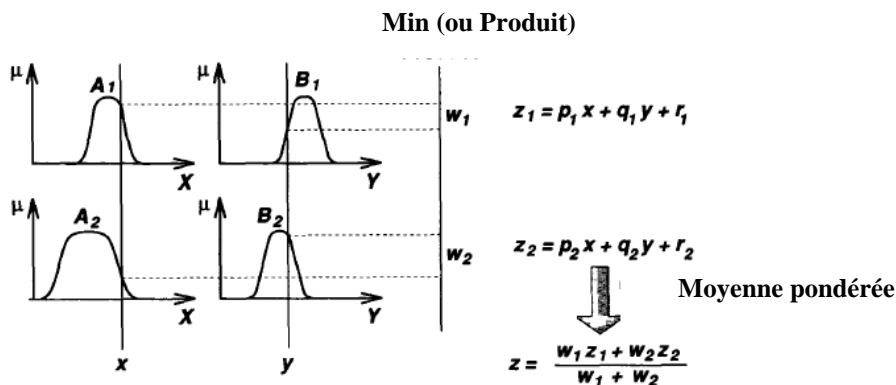


Figure III.12 : Exemple du SIF de Sugeno [Wu 2011]

Les systèmes flous sont connus par leurs capacités de traitement qui se fait au niveau symbolique ainsi que leurs représentation de l'imprécision et l'incertitude d'un expert humain [Zadeh 1996]. Ceci a augmenté le nombre des domaines d'application qui utilisent la logique floue parmi les quelles, on trouve la médecine.

La logique floue et les réseaux de neurones forment aujourd'hui la base de la majorité des systèmes intelligents d'aide au diagnostic et à la décision. Il serait intéressant de fusionner les deux pour exploiter la richesse des deux approches.

III.3.4 Vers les systèmes Neuro-Flous

La logique floue et les réseaux de neurones artificiels sont des technologies complémentaires. Ces techniques ont chacune leurs forces et leurs faiblesses. La combinaison de ces deux approches dans un système intégré semble plus prometteuse pour le développement des systèmes intelligents. [Magaly 2001].

Afin de résumer l'apport des approches neuro-floues, le tableau III.3 regroupe les avantages et les inconvénients des réseaux de neurones et de la logique floue [VIEIRA 2004].

Réseaux de neurones	Logique Floue
Avantages	
<ul style="list-style-type: none"> • Capacité d'apprentissage. • Capacité de généralisation. • Robustesse. 	<ul style="list-style-type: none"> • Représentation des connaissances incertaines • Facilité d'interaction. • Facilité d'interprétation des résultats. • Facilité d'extension de la base de connaissances.
Inconvénients	
<ul style="list-style-type: none"> • Boîte noire (manque d'interprétabilité). • Difficulté de déterminer le nombre de couches /neurones. 	<ul style="list-style-type: none"> • Incapacité de généralisation. • Dépend de l'exigence d'un expert pour déterminer les règles d'inférence. • Non robuste, face à des modifications topologique

Tab III.3 Comparaison entre la logique floue et les réseaux de neurones. [VIEIRA 2004]

Ce tableau montre clairement que la combinaison de ces deux techniques permet de tirer profits des avantages de chaque une des deux approches. D'un côté, les réseaux de neurones peut améliorer leur transparence, ce qui les rendre plus proche des systèmes flous. D'un autre côté, les systèmes flous s'auto-adapte (réglage automatique des paramètres), ce qui les rendre plus proche des réseaux connexionnistes. [Lin 96]

Les systèmes neuro-flous ont suscité l'intérêt croissant des chercheurs dans les domaines scientifiques et d'ingénierie [Lin 96], [Jang 97]. En particulier, les systèmes neuro-flous hybride semblent avoir un intérêt croissant dans le domaine de reconnaissance des formes [Magaly 2001], [Baraldi 98].

III.3.5 Définition des modèles Neuro-Flous

Une définition des systèmes neuro-flous est donnée dans [Nauck 1997]:

“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 locale et produit uniquement des changements locaux dans le système flou d'origine.”

On peut aussi noter que les systèmes neuro-flous (ANFIS) peuvent être utilisés comme des approximateurs universels [Jang 97]

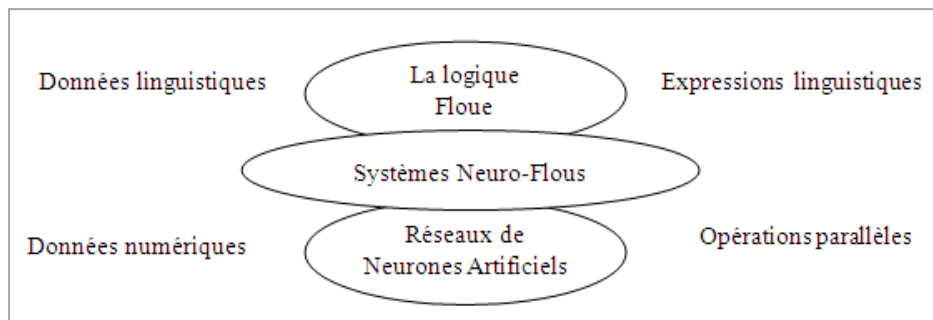


Figure III.13 Le système neuro-flou

L'objectif des systèmes neuro-flous présenté dans la **figure III.13**, montre que l'utilisation conjointe des réseaux de neurones et de la logique floue, permet de tirer les avantages des deux méthodes : les capacités d'apprentissage de la première et l'interprétabilité et la lisibilité de la seconde.

III.3.6 Types des systèmes Neuro-Flous

En général, toutes les combinaisons de techniques basées sur les réseaux de neurones artificielles et la logique floue sont appelés *Systèmes Neuro-Flous*. Différents travaux se sont intéressés à l'intégration des RNA et les SIF [Ming 2002] ,[Tano 1996] ,[Jang 93],[Kwan 1994] ,[Bing 2003], [Fullér 1995]

Il existe trois grandes catégories de combinaisons des réseaux de neurones avec la logique floue [Nauck 1997], [VIEIRA 2004] .

III.3.6.1 Système Neuro-Flou cooperative (Cooperative Neuro-Fuzzy System):

Pour un système coopérative, les réseaux de neurone sont utilisées uniquement dans une phase initiale. Dans ce cas, le réseau de neurone est employé pour déterminer les (sub-blocks) sous-blocs (les règles et les ensembles flous) d'un système flou. Après cette phase, le système flou fonctionne sans usage du réseau de neurone (voir **figure III.14**). L'inconvénient majeur des systèmes neuro-flous cooperative reside dans leur structure qui n'est pas totalement interprétable. [VIEIRA 2004].

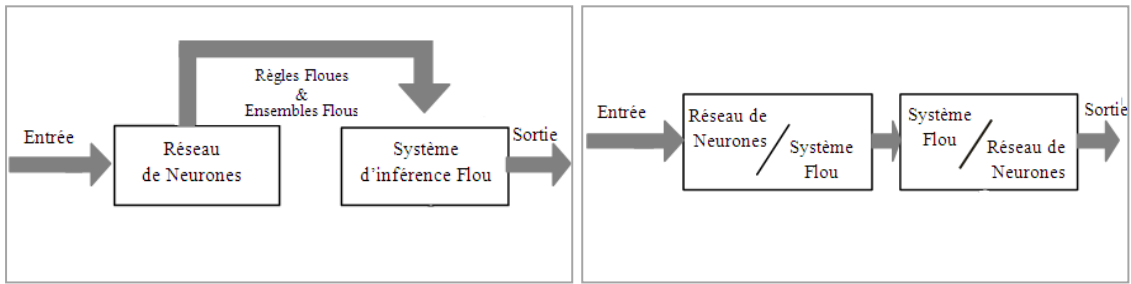


Figure III.14 Modèle Neuro-flou coopérative [VIEIRA 2004]

Figure III.15 Modèle Neuro-flou concurrent [VIEIRA 2004]

III.3.6.2 Système Neuro-Flou concurrent (Concurrent Neuro-Fuzzy System):

Dans le sens strict du terme, un système concurrent n'est pas un système neuro-flou. Dans un système neuro-flou concurrent, le système flou et le modèle connexionniste sont appliquées (voir **figure III.15**) l'une après l'autre comme des techniques de pré(positif)-traitement. [VIEIRA 2004].

III.3.6.3 Système Neuro-Flou Hybride (Hybrid (Integrated) Neuro-Fuzzy System):

Les approches neuro-floues modernes sont de cette forme. Dans cette catégorie, les réseaux de neurones sont utilisés pour faire apprendre les paramètres des systèmes flous. Les chercheurs utilisent le terme neuro-flou aux systèmes neuro-flous hybride [VIEIRA 2004].

III.3.7 Architectures Neuro-Floues

Diverses combinaisons des méthodes et architectures neuro-floues ont été développées depuis 1988. [Abraham 2001] a recensé plusieurs architectures de systèmes neuro-flous (voir *Annexe C*).

Notre étude s'intéressera particulièrement au modèle *ANFIS* (*Adaptive Neuro Fuzzy Inference System*). La suite est dédiée à la présentation de cette architecture neuro-floue. Une explication par l'exemple du fonctionnement de cette architecture sera faite afin de mieux comprendre les mécanismes évoqués précédemment.

III.3.8 ANFIS - un outil hybride pour le diagnostic

Le modèle *ANFIS* (*Adaptive Neuro-Fuzzy Inference System*) est un réseau de neurone flou proposé en 1993 par Jang [Jang 93]. Le réseau comporte cinq couches représenté dans la **figure III.16** suivante :

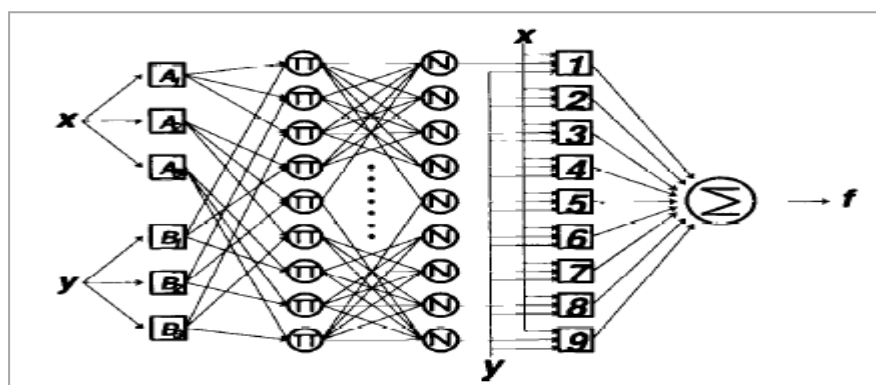


Figure III.16 : Réseau ANFIS [Jang 97]

Les nœuds sont de deux types différents selon leur fonctionnalité: des nœuds adaptatifs (carrés) et des nœuds fixes (circulaires) [Jang 92].

La conception du modèle *ANFIS* peut être subdivisée en 3 stages le développement du modèle, apprentissage du réseau et validation et test du système [Opeyemi 2012]

III.3.9 Intérêt de l'ANFIS

Dans le cadre de ce mémoire de magister, nous nous intéressons aux troisième type des systèmes d'inférence Neuro-Flous: *ANFIS* (Adaptative Neuro-Fuzzy Inference System) . *ANFIS* est un système attrayant en vue de sa structure et son algorithme d'apprentissage standard [Du 2006] .

D'après [Patnaik 2012], [Hosseini 2011], *ANFIS* présente les avantages suivants :

- Raffine les règles floues *Si-Alors* pour décrire le comportement d'un système complexe.
- Présente un grand choix d'usage des fonctions d'appartenance.
- Temps de convergence rapide.
- Expertise préalable non requise.

Cependant, le modèle *ANFIS* est couteux en calcul (computationally expensive) due au problème de *malédiction de la dimension* (*Curse of dimensionality*). [Du 2006] .

III.3.10 Architecture de l'ANFIS

Pour simplifier la compréhension, nous considérons un modèle T.S.K, composé de deux entrées (x et y), une seule sortie globale (f) et de deux règles. Tel qu'il est montré dans la **figure III.17**

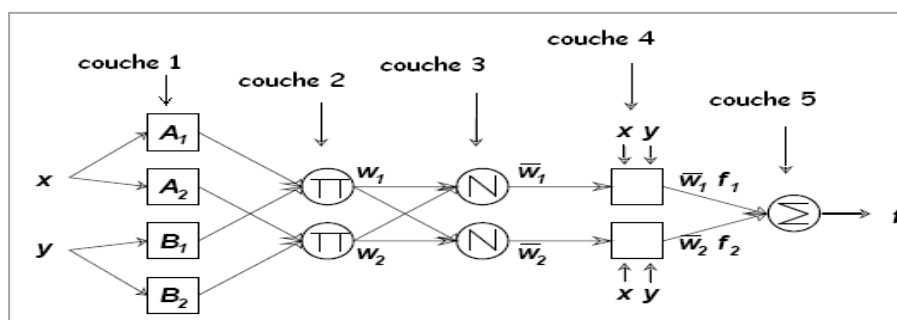


Figure III.17 Architecture de l'ANFIS [Jang 97]

Règle 1 : Si x est A_1 et y est B_1 alors $f_1 = p_1 x + q_1 y + r_1$.

Règle 2 : Si x est A_2 et y est B_2 alors $f_2 = p_2 x + q_2 y + r_2$.

La sortie O_i^k d'un nœud i de la couche k (appelée *noeud* (i,k)) dépend des signaux provenant de la couche $k-1$ et des paramètres du nœud (i,k) [Jang 97] .

✓ *Couche 1* : cette couche comporte autant de neurones qu'il y'a de sous ensembles flous dans le système d'inférence flou. Les nœuds de cette couche sont tous de types adaptatifs. Cette couche réalise la fuzzification des entrées c'est à dire qu'elle détermine les degrés d'appartenance de chaque entrée (O_i^1):

$$O_{1,i} = \mu_{A_i}(x) \quad \text{Pour } i=1,2 \quad (3.12)$$

$$O_{1,i} = \mu_{B_{i-2}}(y) \quad \text{Pour } i=3,4 \quad (3.13)$$

Ces paramètres sont appelés *les paramètres prémisses*. Les sous ensemble flous choisies peut être de forme : Gaussien, triangle, trapézoïdal...

Les fonctions généralisées correspondant sont:

$$\text{- Triangle :} \quad \mu(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \quad (3.14)$$

$$\text{- Trapézoïdale :} \quad \mu(x) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \quad (3.15)$$

$$\text{- Gaussienne} \quad \mu(x) = \exp\left(-\frac{(x-c)^2}{\sigma^2}\right) \quad (3.16)$$

Avec $\{a, b, c, d, \sigma\}$ sont l'ensemble des paramètres.

✓ *Couche 2* : Les nœuds de cette couche sont des nœuds fixes. Ils reçoivent les signaux de sortie de la couche précédente et envoient leur produit en sortie .cette couche engendre le degré d'activation d'une règle. Ceci dépend des opérateurs présents dans les règles (ET ou OU).

$$O_{2,i} = W_i = \mu_{A_i}(x) \mu_{B_i}(y) \quad i=1,2 \quad (3.17)$$

✓ *Couche 3* : Chaque neurone dans cette couche calcule le degré de vérité normalisé d'une règle floue donnée. Le résultat à la sortie de chaque nœud représente la contribution de cette règle au résultat final.

$$O_{3,i} = \overline{W}_i = \frac{W_i}{W_1 + W_2} \quad i=1,2 \quad (3.18)$$

✓ *Couche 4* : Les noeuds dans cette couche sont des noeuds adaptatifs. Chacun de ces noeuds est relié à un neurone de normalisation correspondant et aux entrées initiales du réseau. La sortie d'un noeud i est donnée par :

$$O_{4,i} = \overline{W}_i f_i = \overline{W}_i (p_i x + q_i y + r_i) \quad (3.19)$$

Où \bar{w}_i est la sortie de la troisième couche, et $\{p_i, q_i, r_i\}$ est l'ensemble des paramètres. Ces paramètres sont appelés *les paramètres conséquents*.

✓ *Couche 5* : Cette couche comprend un seul neurone circulaire qui effectue la somme des signaux provenant de la couche précédente pour donner la sortie finale du réseau :

Une autre :
$$O_{5,1} = \bar{w}_i = \frac{\sum_i W_i f_i}{\sum_i W_i} \tag{3.20}$$
 est montrée dans la suivante.

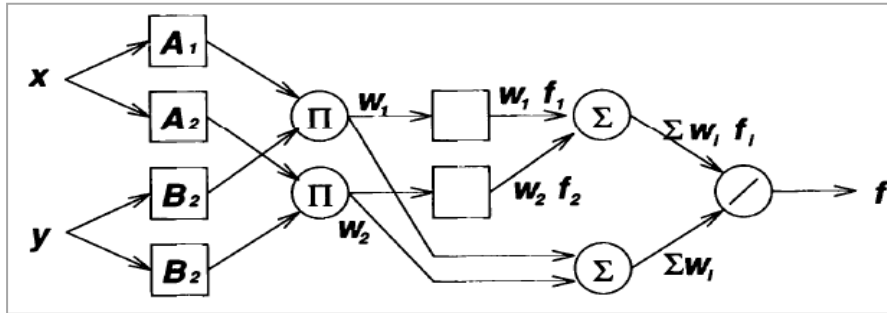


Figure III.18 : Autre architecture pour le modèle ANFIS [Jang 97]

III.3.11 Apprentissage de l'ANFIS

L'ajustement des paramètres de l'Anfis est réalisé lors de la phase d'apprentissage. Cette étape commence par la construction d'un réseau initiale, ensuite applique une méthode d'apprentissage par rétro-propagation [Jang 97]. Jang a proposé d'appliquer une méthode hybride.

L'algorithme d'apprentissage hybride est une association de la méthode de descente de gradient et de la méthode d'estimation des moindres carrés. La méthode de descente de gradient permet d'ajuster les prémisses en fixant les paramètres conséquents alors que la méthode LSM (Least square Method) ajuste les paramètres conséquents en fixant les prémisses.[Jang 97] . Ceci est présenté dans le **tableau III.4** suivant :

	Passage vers l'avant	Passage en arrière
Paramètres des fonctions d'appartenances (a _i , b _i , c _i ...)	Fixe	Rétropropagation
Paramètre de coefficient (p, q, r) (conclusion)	Moindres Carrés	Fixe

Tab III.4 : Les paramètres à ajuster pour l'ANFIS [Jang 97]

Dans l'architecture *ANFIS* déjà proposée, la sortie globale peut être exprimé en tant que des combinaisons linéaires des paramètres conséquents. La conclusion (la sortie) peut être réécrite comme suite:

$$\begin{aligned}
 f &= \frac{w1}{w1 + w2} f1 + \frac{w2}{w1 + w2} f2 & (3.21) \\
 &= (\overline{w_1 x}) p_1 + (\overline{w_1 y}) q_1 + (\overline{w_1}) r_1 + (\overline{w_2 x}) p_2 + (\overline{w_2 y}) q_2 + (\overline{w_2}) r_2
 \end{aligned}$$

Dans la deuxième partie ,nous avons détaillé le modèle *ANFIS* (**A**daptative **N**euro-**F**uzzy **I**nferece **S**ystem) que nous adoptons pour notre deuxième contribution (*F-HBA*) en expliquant son architecture et son principe de fonctionnement .

Nous définirons dans la section suivante et dernière de ce chapitre le troisième concept clé de notre démarche de résolution : l'optimisation métaheuristique que nous adoptons pour optimiser le problème sous-jacent.

III.4. Partie III: La métaheuristique A.B.H
(Algorithme a Base d'Homogénéité)

The root to the low accuracies is the overfitting and overgeneralization behaviors of a given classification algorithm when it is processing a dataset.

– Huy Nguyen Anh Pham –

III.4.1 Introduction

Les applications du monde réel (ex : le diagnostic médical) présentent trois types différents de couts de pénalité [Pham 2008] :

- 1^{er} cout : faux positif (point positif classifié comme négatif).
- 2^{eme} cout : faux négatif (point négatif classifié comme positif).
- 3^{eme} cout : point qui ne peut être classifié par un des modèles de classification.

D'après les considérations susmentionnés, les approches de fouille de données (data-mining) courant, souvent assignent des couts de pénalités identiques pour les cas de faux positifs (*FP*) et négatifs (*FN*), ou ignore complètement le cout de pénalité des cas non-classifiés .Aussi, ces approches abouties a des résultats inacceptabl ou inattendus .[Pham 2009]

Les coûts du *FP* et *FN* peuvent être dramatiquement différents dans les applications médicales. [Pham 2009]. Pour les conditions de vie menaçant ou le temps est essentiel, si un cas est reconnu comme *FN*, alors son (ses) conditions médical n'est pas (ou males) traités. Ainsi, le temps a de valeur et la situation peut devenir fatal pour un patient. D'autre part, un diagnostic de faux positif (*FP*) peut entrainer d'autres cout financière et d'inquiétude pour un patient.

Dans le champ médical, un cas non-classifié est bien requis .Le diagnostic d'un patient comme un cas non-classifié peut imposée d'autres tests médicaux additionnels. Cependant, ce cas particulier ne peut nécessairement résulter à un faux diagnostic.

Pour ces raisons cités au-dessus, nous appliquons une nouvelle métaheuristique nommée *A.B.H* (Algorithme a *Base* d'*Homogénéité*) développé par Pham et Triantaphyllou [Pham 2007],[Pham 2008],[Pham 2009],[Pham 2011].

L'objectif de cette métaheuristique (*A.B.H* (Algorithme a *Base* d'*Homogénéité*)) est d'achevé une balance entre le fitting et la généralisation qui minimise le cout totale d'échec de classification noté C_T (voir équation 3.23).

$$C_T = \min (C_{FP} \times \text{Taux}_{FP} + C_{FN} \times \text{Taux}_{FN} + C_{NC} \times \text{Taux}_{NC}) \quad (3.22)$$

En particulier, (C_{FP} , C_{FN} , C_{NC}) présentent les couts de pénalité pour les cas du faux-positif, faux-négatif et non classifiés respectivement. (Taux_{FP} , Taux_{FN} , Taux_{NC}) sont les taux du faux-positif, faux-négatif et non classifiés respectivement. [Pham 2008].

Dans les sections suivantes, nous définissons les principes de base de cette métaheuristique qui sont présentés dans les sections III.4. [2-3-4]. Ces sections montrent qu'une balance entre le fitting et la généralisation peut améliorer la performance des méthodes de classification existantes.

III.4.2 description de l'A.B.H

III.4.2.1 Quelques notions d'observations

La figure III.19 (a) suivante présente deux modèles inférés (les cercles qui entour les données d'apprentissage).Les petits cercles représentent les données positives, les points (Q,P) (petits triangles) sont des données additionnelles (supposant que la classe

de ces deux points est inconnue). Le but est d'utiliser les données d'apprentissage disponibles pour inférer des modèles afin de classer ces deux points. [Pham 2007]

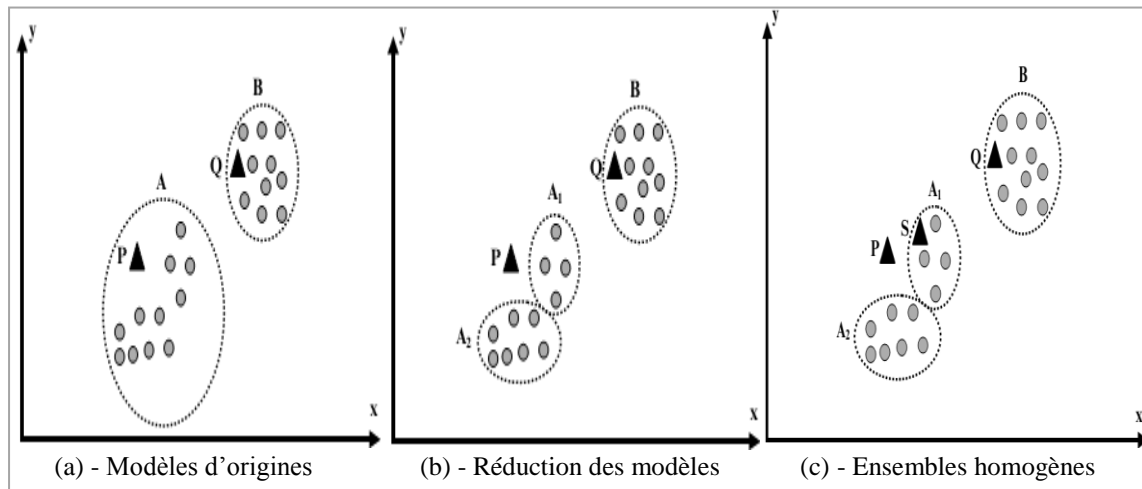


Figure III.19 : Le modèle A n'est pas un ensemble homogène. Les modèles B , A_1 , A_2 sont des ensembles homogènes. [Pham 2007]

Cette figure montre que les points (P et Q) sont couverts par les modèles A et B respectivement. Ils sont supposés d'être des exemples positifs. Le modèle A décrit dans la **figure III.19(a)** montre que ce modèle est couvert par des régions (de l'espace de recherche) qui ne sont pas adéquatement remplies par les points d'apprentissage positifs. Exemple: le coin supérieur gauche et la partie inférieure du modèle A . Il est possible que les points non-classifiés qui appartiennent à ces régions sont supposés de façon erronée de la classe des points positifs couverts par A . [Pham 2007]

Ainsi, la supposition que le point P est un point positif n'est cependant pas très précise. D'autres parts, le modèle B ne dispose pas de telle région. Ainsi, les points non-classifiés couverts par le modèle B sont plus précis, supposés d'être de la même classe que les points positifs couverts par le même modèle. Ainsi, la supposition que le point Q est positif est plus précise. [Pham 2007]

Ces observations amènent à la première supposition [Pham 2007, 2008, 2009] :

Supposition 1: La performance des systèmes de classification inférés peut augmenter si les modèles dérivés sont plus *compact* et *homogène* (more compact and homogenous). Un modèle C de taille n_c est homogène, si ce modèle peut être divisé en petits bins de même taille h et que la densité de ces bins sont presque équivalents les uns et les autres.

Un modèle non homogène (A), peut être remplacé par d'autres ensembles homogènes notés A_1 , A_2 (voir **figure III.19(b)**). À ce stade, les régions couvertes par les deux nouveaux modèles A_1 , A_2 sont de plus en plus homogènes que le modèle d'origine A . Étant donné ces conditions, le point P est supposé d'être non-classifié et le point Q est un point positif. [Pham 2007]

Comme a été mentionné dans les paragraphes précédentes, la propriété d'homogénéité des modèles peut influencer sur le nombre des cas non-classifiés des systèmes de classification inférés. En plus, si un modèle est un ensemble homogène alors, le nombre des points peut affecter la performance des systèmes inférés. [Pham 2008]

Par exemple, la **figure III.19(c)** suppose que les modèles $A1$, $A2$, B sont des ensembles homogènes et que le nouveau point S est couvert par le modèle $A1$. Cette figure montre que le nombre des points dans B est plus grand que ceux dans le modèle $A1$. [Pham 2008]

Malgré que les deux points (Q et S) sont couverts par des ensembles homogènes, la supposition que Q est un point positif est plus précise que la supposition que S est un point positif. Les observations citées ci-dessous amènent à la deuxième supposition suivante: [Pham 2007,2008, 2009]

Supposition 2: La performance des systèmes de classification inférés peut être aussi affectée par la mesure de la densité (density measure). Cette mesure est définie par le nombre des points de données incluse dans chaque modèle inféré par unité de surface ou de volume. Par conséquent cette densité est appelée: degré d'homogénéité (homogeneity degree). Si C est homogène, alors $DH(C)$ note son degré d'homogénéité.

- **Axiome 1:** Etant donné un modèle C (Pattern C with size one). Alors C est un ensemble *homogène*. [Pham 2007]

- **Théorème 1:** Etant donné C , un ensemble *homogène*. Si C est divisé en deux parties (C_1 et C_2), alors ces deux parties sont aussi des ensembles *homogènes*. [Pham 2007]

[Pham 2007] utilise la contradiction pour prouver **Théorème 1**. Soit C un ensemble homogène. Ceci implique qu'il existe une variable aléatoire uniforme Z qui représente la distribution des points dans C . Aussi, $Z1$ et $Z2$ sont deux variables aléatoires qui représentent la distribution des points dans $C1$ et $C2$. Z étant la somme de $Z1$ et $Z2$. Supposons que $Z1$ et $Z2$ sont des ensembles non homogènes. Ainsi, $Z1 + Z2$ n'est pas une variable aléatoire uniforme. Ceci est une contradiction du fait que Z est une variable aléatoire uniforme.

- **Heuristique 1:** Si h est égal à la valeur minimale dans S et que cette valeur est utilisée pour calculer la $d(x)$ en utilisant l'équation (3.29), alors $d(x)$ s'approche de la densité réelle (true density). Par exemple: [Pham 2007]

- Etant donné C un modèle de taille 5 (i.e: il existe cinq points dans C).

- Supposons que les distances(10) entre tous les paires de points possibles sont calculées.

- Supposons que ces distances sont: (6,1,2,2,1,5,2,3,5,5).

- S : un ensemble des distances les plus fréquentes = {2,5} (occurrence=3).

Heuristique 1 propose que la valeur approprié de $h = 2$. Cette heuristique semble d'être raisonnable. En pratique : C étant le modèle comportant un nombre fini de points. La valeur h ne peut être arbitrairement petite i.e. $h \in]\min, \max[$ des distances calculer pour toutes les paires de points dans C . [Pham 2007]

Si $h \rightarrow 0$, alors $d(x)$ s'approche de la densité réelle. Ainsi, la valeur h décrit dans l'heuristique 1 est une sélection raisonnable car elle s'approche de la distance minimale. [Pham 2007]

III.4.2.2 Estimation non paramétrique:

Rappelons qu'un modèle C de taille n_c est un ensemble homogène si ce modèle peut être subdivisé en petits bins de même taille h et que la densité de ces bins sont presque égaux les une des autres (ou la déviation standard est suffisamment petite). En d'autres termes, si C est superposé dans un hypergrid de taille h et la densité de ces bins dans C sont presque égaux les une des autres, alors C est un ensemble homogène. [Pham 2007,2008, 2009]

Par conséquent, l'estimation de la densité joue un rôle important pour la détermination si un ensemble est homogène ou pas. Il existe principalement deux approches de base de l'estimation de la densité : [Pham 2007,2008, 2009]

- **Estimation paramétrique** : pour la quelle, la forme de la fonction de densité est supposé connue (ex : gaussienne,..).
- **Estimation non paramétrique** : pour la quelle, la forme de la fonction de densité n'est pas supposés connue et que la densité est estimés à partir des données d'apprentissage disponibles.

Les sections suivantes utilisent l'estimation de la densité non paramétrique. Cette approche divise le modèle C en petits bins de taille h . La densité au centre de x de chaque bin peut être approximé par la fraction du point C qui appartient au bin correspondant et du volume de ce bin. Par exemple : un bin en $3-D$ est un cube de taille h (voir figure III.20). Soit n le nombre des points dans C et $d(x)$ la densité de x alors : [Pham 2007,2008, 2009]

$$d(x) = \frac{1}{n} \left[\frac{\text{nbr d'exemples regroupé dans un bin de centre } x}{\text{Volume d'un bin}} \right] \quad (3.23)$$

En utilisant cette idée :

$$d(x) \approx \frac{k}{n \times V} \quad (3.24)$$

Avec $x \in R$, k : nombre de point qui appartient a R . V : volume de R .

L'approche : Fenêtre de Parzen (Parzen Window) [Duda 1973] a été introduite comme étant l'approche la plus approprié pour l'estimation de la densité. Etant donnée une région R (R est un hypercube de taille h en D -dimension). Pour trouver le nombre

des points $\in \mathbf{R}$, l'approche **Parzen Window** définit une fonction noyau (Kernel Function) $\Phi(\mathbf{u})$ comme suit :

$$\varphi(u) = \begin{cases} 1, & |u| \leq 1/2, \\ 0, & \text{Sinon} \end{cases} \quad (3.25)$$

La quantité $\Phi(\mathbf{u})=1$ si x_i est à l'intérieur de l'hypercube de taille h centré en x . Sinon = 0.

Ainsi k : est le nombre des points dans l'hypercube. Avec :

$$k = \sum_{i=1}^n \varphi\left(\frac{x - x_i}{h}\right) \quad (3.26)$$

En **D-dimension**, la fonction noyau est donnée par ;

$$\varphi\left(\frac{x - x_i}{h}\right) = \prod_{m=1}^D \varphi\left(\frac{x^m - x_i^m}{h}\right) \quad (3.27)$$

En utilisant l'équation (3.28) et (3.25), On obtient :

$$d(x) \approx \frac{1}{n_C \times h^D} \sum_{i=1}^{n_C} \prod_{m=1}^D \varphi\left(\frac{x^m - x_i^m}{h}\right) \quad (3.28)$$

$$\varphi(u) = \frac{1}{(2 \times \pi)^{\frac{D}{2}}} \exp\left(-\frac{1}{2}u^2\right) \quad (3.29)$$

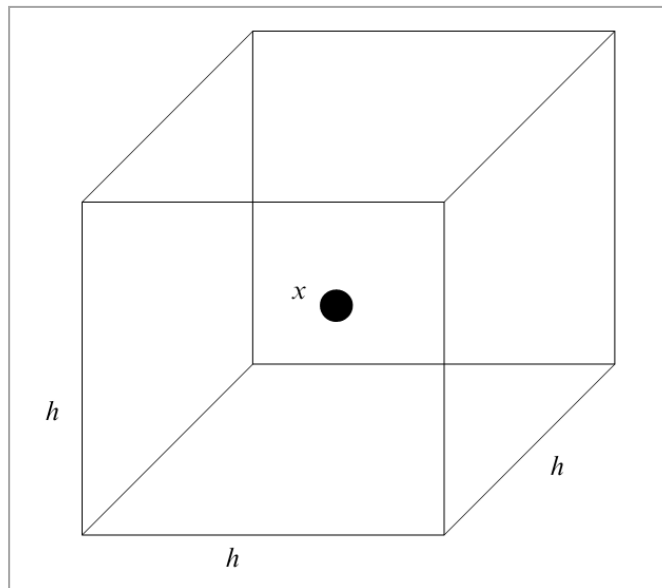


Figure III.20 un Bin de taille h centré en x (3-D) [Pham 2007]

Le choix de la valeur du paramètre de lissage h (smoothing parameter) joue un rôle important pour l'estimation de la densité. Si $h \rightarrow \infty$, alors la densité du point \mathbf{x} dans \mathbf{C} , $d(\mathbf{x})$, s'approche du faux densité (false density). et Si $h \rightarrow 0$, alors la fonction noyau (kernel function) s'approche de la fonction de Dirac (dirac delta function) et $d(\mathbf{x})$ s'approche de la densité réelle (true density). [Pham 2007,2008]

III.4.3 Processus de l'algorithme A.B.H

Pour une méthode de classification et durant l'optimisation du cout d'échec total de classification définie par l'équation(3.22), la méthode ne peut faire une séparation de control de généralisation et le "fitting" indépendamment.[Pham 2007,2008,2009,2011]

L'idée principale du A.B.H (ou H.B.A Homogeneity Based-Algorithm) est de faire une balance simultanée entre le fitting et la généralisation en ajustant les systèmes inférés à travers l'usage des concepts d'ensembles homogènes et du degré d'homogénéité. :[Pham 2007,2008]

Phase I : Appliquer une approche de classification sur la base d'apprentissage T_1 pour inférer les systèmes de classification (positive et négative).Supposons que chaque système de classification est composé d'un ensemble de modèles (patterns). Engendré Les modèles inférés par des hypersphères.

Phase II : Déterminer si les hypersphères dérivés de la première phase sont homogènes ou pas. Si Oui : aller a la **phase III**. Sinon ; diviser les ensembles non homogènes a des petites hypersphères. Répéter cette phase jusqu'à ce que tous les hypersphères soit homogènes.

Phase III : pour chaque ensemble homogène, si son degrés d'homogénéité $DH \geq \beta^+$ (pour un ensemble positive, $DH \geq \beta^-$ pour un ensemble négative), alors étendre ce modèle. Sinon, subdivisé cet ensemble a des sous ensembles homogènes. L'approche s'arrête si tous les ensembles homogènes sont traités.

Algorithme III.1 A.B.H (Algorithme a Base d' Homogénéité)

Entrée : Une base de données T .

Une méthode de classification.

Un seuil de densité γ .

Sortie : Un nouveau système de classification S .

- 1- Diviser T en T_1 et T_2 . {Phase I}
- 2- Initialiser aléatoirement les valeurs des paramètres de contrôle : α^+ , α^- , β^+ et β^- .
- 3- Appeler le **sous-problème 1** & T_1 pour inférer les deux modèles de classification.
- 4- Appeler le **sous-problème 2** pour former des hypersphères aux modèles inférés.
- 5- **Pour** chaque hypersphère C faire : {Phase II}
 - Appeler le **sous-problème 3** avec les entrées C et γ pour déterminer si C est homogène ou pas.
 - Si (C n'est pas homogène) alors appeler le **sous-problème 4** (aller vers étape (5))
- 6- Trier les degrés d'homogénéités (dans un ordre décroissantes).
- 7- **Pour** chaque ensemble homogène C faire : {Phase III}
 - Si ($DH(C) \geq \beta^+$) (pour un ensemble positif) ou ($DH(C) \geq \beta^-$) (pour un ensemble négatif) alors :
 - Appeler **sous-problème 5** avec les entrées $DH(C)$ et (α^+ ou α^-) pour étendre C
 - Sinon :
 - Appeler le **sous-problème 6** pour réduire C .

Notes :

- Appliquer l'approche A.G (algorithme génétique) pour les étapes de 5 à 7 en utilisant l'équation III.22 comme une fonction fitness et T_2 comme une base de données de calibration pour trouver le modèle de classification S_1 et les valeurs des seuils optimaux (α^+ , α^- , β^+ et β^-).
- Pour les points non classifiés de S_1 sur T_2 , le **A.B.H** utilise les étapes de 3 à 7 avec les valeurs des seuils optimaux (α^+ , α^- , β^+ et β^-) pour inférer les modèles de classification additionnelles S_2 .

10- $S = S_1 \cup S_2$.

Etant donné un ensemble *homogène* C et son degrés d'homogénéité noté $HD(C)$.la métaheuristique **A.B.H** utilisent les cinq paramètres suivants (qui sont calculés par l'**AG** (Algorithme Génétique)) : **[Pham 2007,2008]**

- α^+ , α^- : deux seuils utilisés pour étendre les ensembles homogènes positif et négatif respectivement.
- β^+ , β^- : deux seuils utilisés pour réduire les ensembles homogènes positif et négatif respectivement.
- γ : seuil de densité utilisé pour tester si une hypersphère positive ou négative est approximativement un ensemble homogène ou pas.

III.4.4 résolution des sous-problèmes

Les trois phases décrites au-dessus (voir *Algorithme III.1*) amènent à la formulation des six sous-problèmes suivants:[Pham 2007,2008] :

- *Sous-problème 1* : Appliquer une approche de Data-Mining (ex :R.N.A, A.D,...) pour inférer les systèmes de classification.
- *Sous-problème 2* : Engendrer les modèles inférées par des hypersphères.
- *Sous-problème 3* : Déterminer si un ensemble est homogène ou pas. Si oui, calculer son degré d'homogénéité.
- *Sous-problème 4* : Si une hypersphère n'est pas homogène, alors le subdiviser a des petites hypersphères.
- *Sous-problème 5* : Etendre un ensemble homogène C en utilisant la notion du degré d'homogénéité $HD(C)$ et la valeur du seuil d'expansion plus quelques conditions d'arrêts.
- *Sous-problème 6* : Subdiviser l'ensemble homogène C a des petits ensembles homogènes.

Pour résoudre le *Sous-problème 1*, une méthode de classification est appliquée pour dériver les modèles de classification. Aussi, une solution du *Sous-problème 2* est similaire à une solution du *Sous-problème 4*. Ainsi, nous présentons dans les sections suivantes quelques algorithmes pour la résolution des *Sous-problèmes : 2, 3,5*. [Pham 2007,2008].

III.4.4.1 Résoudre le Sous-Problème 2 :

L'objectif du *sous-problème 2* est de trouver un nombre minimal des hypersphères qui couvrent tous les points d'un modèle originale. Dans le cadre de l'algorithme ABH, l'heuristique III.2 décrite dans *Algorithme III.2* propose la résolution du *sous-problème 2*. [Pham 2008]

Pour simplifier la compréhension, nous considérons l'ensemble des points d'apprentissage (en 2D) positifs et négatifs **figure III.21(a)**. Supposons que le *sous-problème 1* a appliqué un algorithme de classification (ex : arbre de décision) pour inférer l'arbre de décision présenté dans la **figure III.21(b)**. Cet arbre de décision sépare les données d'apprentissage en 4 groupes (voir **figure III.21(c)**). La **figure III.21(d)** présente un nombre minimal d'hypersphères couvrant l'ensemble des modèles positifs (B, D) et négatifs (A, C, E). [Pham 2007]

L'algorithme débute par l'estimation des densités des n_c points en utilisant l'équation (3.28). Supposons que la valeur de $K : 1 \rightarrow n_c$. Dans un premier temps, l'algorithme choisie K points dans C ayant les densités les plus élevées. Dans un second temps, elle utilise ces K points come des centroides pour l'approche **K-moyenne** (voir *Annexe C*). [Pham 2008]

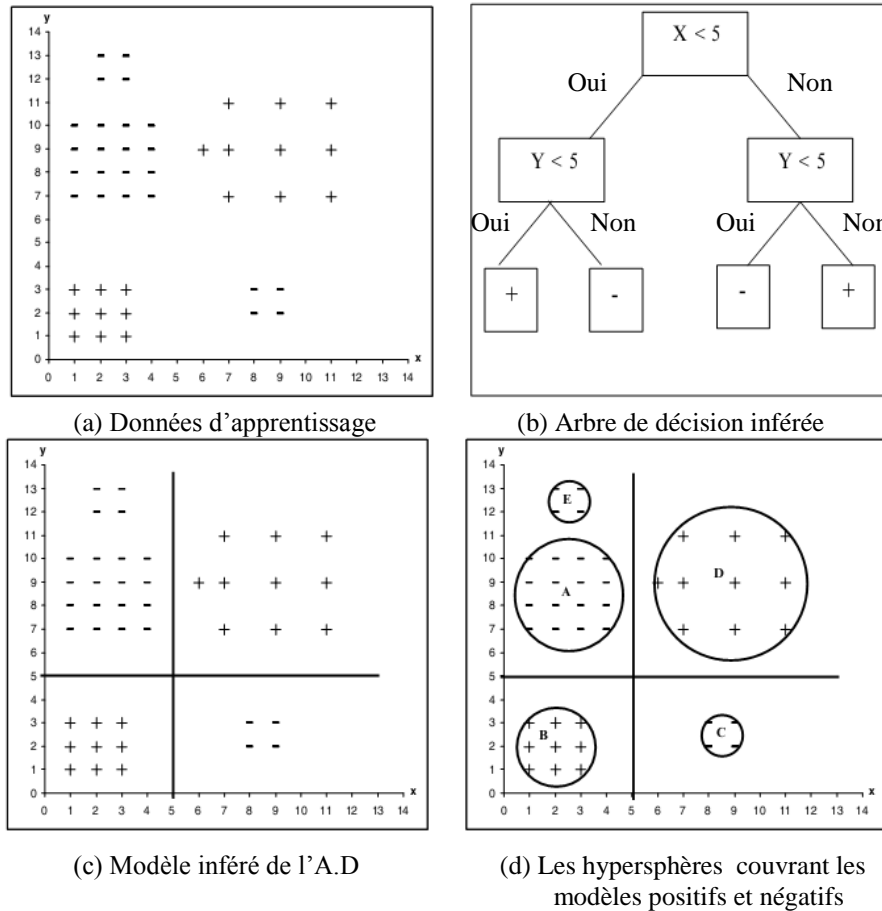


Figure III.21 : Exemple illustratif de la **phase I** [Pham 2008]

Si les **K** hypersphères obtenues de l'approche de clustering couvèrent **C**, alors l'algorithme s'arrête. Sinon, l'algorithme se répète avec une augmentation de un du paramètre **K**. Clairement, l'algorithme s'arrête après quelques itérations à cause de l'**axiome 1**. Rappelent que le *sous-problème 4* décompose un ensemble non homogène **C** à des petites hypersphères afin de minimiser les nombre des hypersphères qui couvrent **C**. ceci peut être résolue par un algorithme similaire a l'algorithme **III.2** [Pham 2008]

Algorithme III.2 résoudre le sous-problème 2

Entrée : Modèle **C** de taille n_c .

Sortie : **K** hyper-sphère.

- Estimer la densité des points n_c en utilisant l'équation (III.29).

- Pour **K=1** à n_c faire

Sélectionner **K** points dans **C** ayant les densités les plus élevées.

Utiliser l'approche **K-means** pour trouver les **K** hypersphères.

Si (**K** hypersphères couvrent **C**) **Alors**

Stop.

Sinon

K=K+1.

Fin

Fin

III.4.4.2 Résoudre le Sous-Problème 3 :

Etant donnée une hypersphère C , le *sous-problème 3* décide si C est homogène ou pas. En utilisant l'idée d'estimation non paramétrique décrite au-dessus, C est divisé en petits bins de taille h et approximative la densité du centre x de chaque bin. Si les densités aux centres sont approximativement équivalentes les unes a des autres, alors C est homogène. **[Pham 2007,2008 ;2009]**

Pour simplifier la compréhension de l'**algorithme III.3**, la **figure III.22** présente deux cercles positifs (A et B) en $2D$, superposés dans un hypergrid de taille $h=1$. En utilisant l'**équation 3.28**, la **figure III.22** montre que l'ensemble des bins dans le cercle A ont la même densité $= \frac{1}{16*1} = 0.0625$. Ainsi A est *homogène*. A l'inverse, la densité de quelques bins dans $B = 0$. Ainsi B n'est *pas homogène*. **[Pham 2007]**

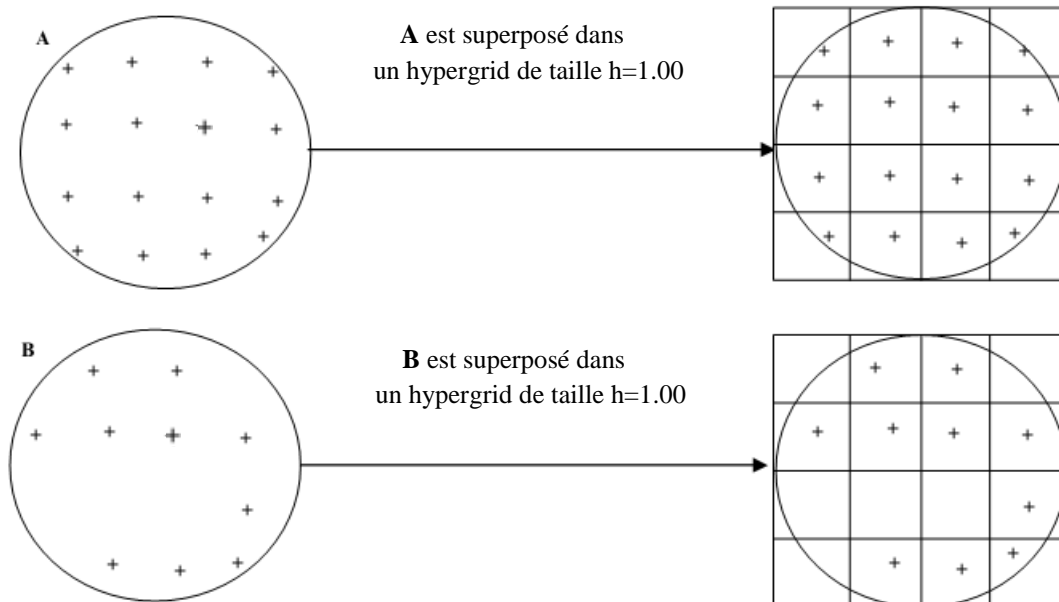


Figure III.22 Exemples illustratifs des ensembles homogènes **[Pham 2008]**

En plus, une autre condition est appliquée au lieu de cette condition stricte (besoin de la même densité aux centres des bins). Si la déviation standard des densités aux centre de chaque bin est approximativement $\leq \gamma$ ($\gamma = 0.01$) alors C est homogène. (Voir **Algorithme III.3**). **[Pham 2007, 2008, 2009]**

Algorithme III.3 résoudre le sous-problème 3

Entrée : hyper sphère C et seuil γ .
 Sortie : décidé si C est homogène ou pas.

- Calculer les distances entre toutes les paires de points dans C .
- Soit h la distance mentionné par l' **Heuristique 1**.
- Superposer C dans un hypergrid de taille h .
- Approximation de la densité au centre x de chaque bin.
- Calculer la déviation standard des densités aux centres des bins.

Si (déviation standard $\leq \gamma$) **Alors**
 C est *homogène*.
 Calculer $DH(C)$ en utilisant l'équation III.31

Sinon
 C n'est pas homogène.

Fin

Fin

Comme mentionné dans la section III.4.2, le degré d'homogénéité est facteur important qui peut affecter le cout total d'échec de classification des systèmes de classification inférés. Si un point non classifié est couvert par un ensemble homogène C ayant un $DH(C)$ le plus élevé, alors ce point est supposé d'être de la même classe que les points couvèrent par l'ensemble homogène C . Ainsi, la définition $DH(C)$ est une étape importante pour l'amélioration de la performance des systèmes de classification. [Pham 2008].

Intuitivement, $DH(C)$ dépend de la valeur h , définie par *Heuristique 1* et le nombre des points dans C (n_c). Ceci est donné par l'équation 3.30. Notons que, $DH(C)$ est inversement proportionnelle au paramètre h et directement proportionnelle à n_c . [Pham 2008].

$$DH(C) = \frac{\ln(nc)}{h} \quad (3.30)$$

Considérons maintenant la figure III.21, en utilisant eq 3.30, nous obtenons:

$$DH(A) = \frac{\ln(16)}{1} \approx 2.77, \quad DH(B) = \frac{\ln(9)}{1} \approx 2.19, \quad DH(C) = DH(E) = \frac{\ln(4)}{1} \approx 1.38,$$

$$DH(D) = \frac{\ln(10)}{1} \approx 1.151.$$

III.4.4.3 Résoudre le Sous-Problème 5 :

Rappelons que le control du fitting et la généralisation pour les systèmes de classification est achevé par expansion ou réduction des ensembles homogènes inférés en utilisant leurs degrés d'homogénéité. [Pham 2007, 2008]

Etant donnée un ensemble homogène positif F et son degré d'homogénéité $DH(F)$, les valeurs des seuils β^+ , α^+ . (Une définition similaire existe pour un ensemble homogène). D'après **Algorithme III.1**, si $DH(F) \geq \beta^+$ alors l'ensemble homogène F est

étendue en utilisant la valeur du paramètre α^+ . Sinon F est subdivisé en d'autres petites hypersphères. [Pham 2007]

Considérons maintenant la *figure III.21*, (A, B, C, D, E) sont des cercles avec ($DH(A) \approx 2.77$, $DH(B) \approx 2.19$, $DH(C)=DH(E) \approx 1.38$, $DH(D) \approx 1.151$) respectivement. Supposons que ($\beta^+ = 1$, $\beta^- = 1.5$) et ($\alpha^+ = \alpha^- = 2$). Comme le montre la *figure III.23*, les ensembles homogènes (A, B, D) sont étendus (indiqué par des cercles à traits pleines). Et les ensembles (C, E) sont subdivisés en 4 petits cercles (indiqué par les petits cercles à traits pleines). [Pham 2007]

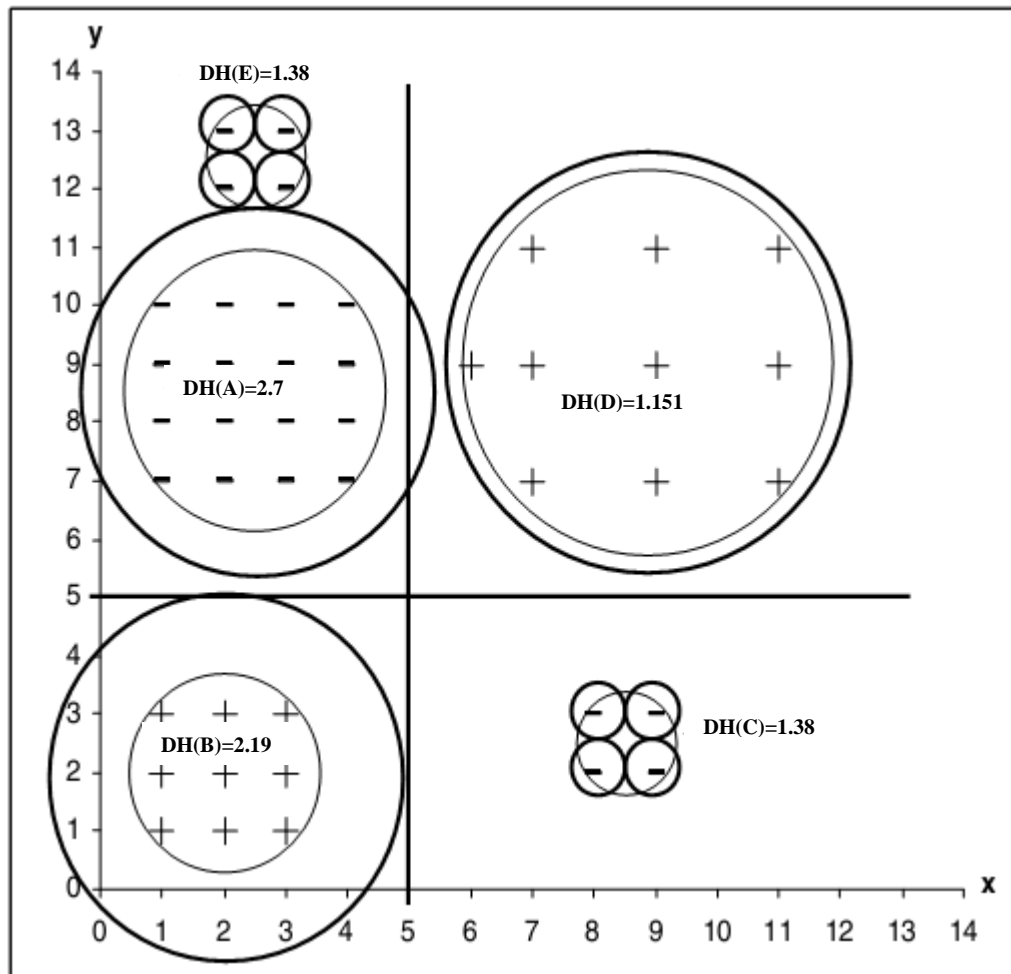


Figure III.23 : Exemple illustrative du sous-problème 5 [Pham 2007]

Remarque :

L'application de subdivision des modèles est décrite dans la section suivante **III.4.4.4** (le *sous-problème 6*). Il existe principalement deux types d'extension (pour plus de détails voir *Annexe C*):

- **Extension radial** : étend le modèle homogène F (dans toutes les directions) au modèle M en utilisant l'équation suivante : [Pham 2007, 2008, 2009]

$$R_M = R_F + \frac{R_G - R_F}{2} \times \frac{1}{L \times DH(F)} \quad (3.31)$$

- **Extension linéaire** : étend un modèle homogène F dans une direction donnée.

Pour faciliter la compréhension des approches d'expansion radiale et linéaire, nous considérons les deux figures suivantes :

1- La **figure III.24** présente l'exemple qui utilise les mêmes modèles inférés et présentés dans la **figure III.21 (d)**. Supposons que $L=1$. Cette figure montre que l'ensemble A ($R_A=2.121$) est couvert par trois cercles.

- G (hypersphère a traits double) avec $R_G = 2 * R_A = 4.242$.
- Hypersphère a traits pleines qui montre la région finale étendue.
- Hypersphère M et son radius calculé comme suit :

$$R_M = R_F + \frac{R_G - R_M}{2} \times \frac{1}{L \times w(A)} = 2.121 + \frac{4.242 - 2.121}{2} \times \frac{1}{1 \times 2.77} \approx 2.5.$$

L'équation (3.31) calcule les valeurs de R_M en quatre itérations 2.8 / 3.06 / 3.23 / 3.25, jusqu'à ce que R_M satisfait les conditions d'arrêtes mentionné dans la section précédente. [Pham 2007]

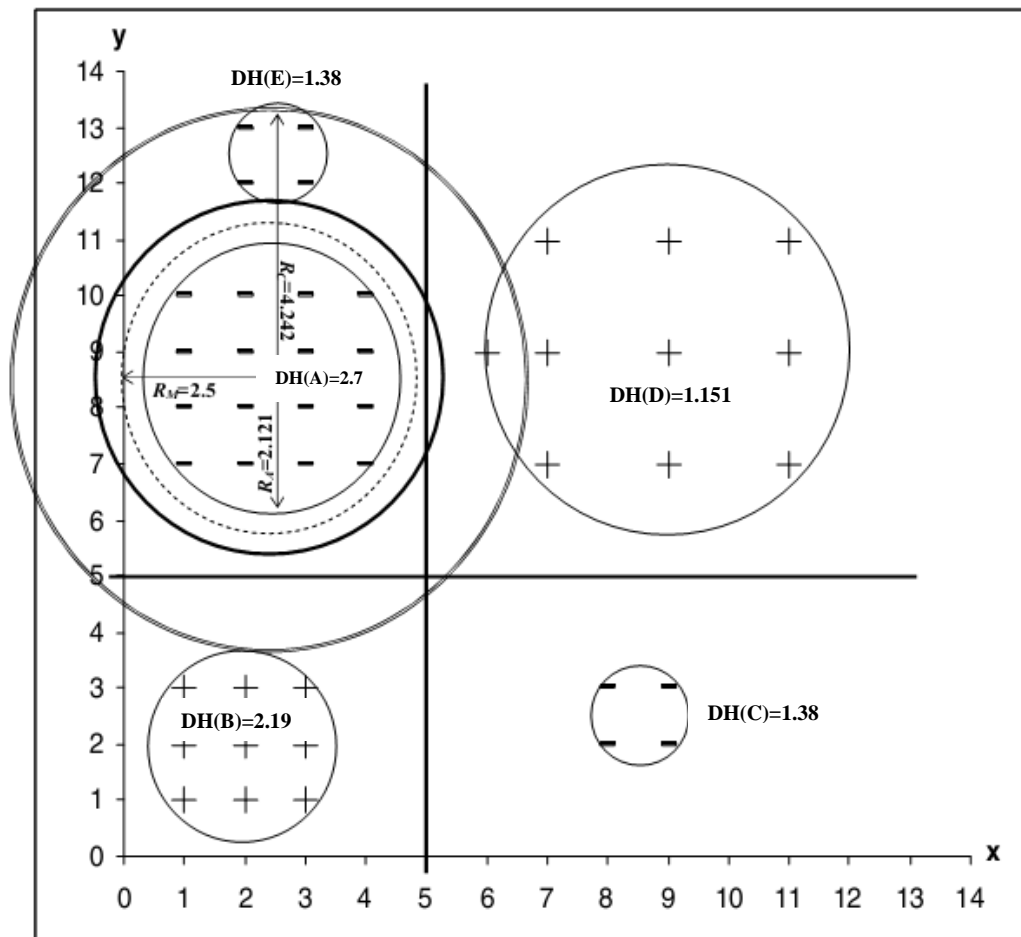


Figure III.24 : Exemple illustratif de l'expansion radiale [Pham 2007]

La **figure III.24** montre aussi qu'il existe une partie de l'espace de recherche qui est inférée comme région positif (par la méthode de classification **A.D**). Cependant, elle est dérivée comme région négatif après utilisation de l'**A.B.H**. Ceci indique que l'**A.B.H** peut dériver de mieux systèmes de classification. [Pham 2007]

2- la **figure III.25** qui présente un ensemble homogène **A** étendue (en utilisant l'expansion radiale $\alpha = 2$) en une hypersphère **U** **figure III.25(a)** (cercle à traits double).

Ensuite, l'hypersphère **U** est divisé en deux parties selon l'axe **X**. l'expansion radiale étend chaque partie (traits pleines) (**figure III.25.b**). Une approche similaire existe pour l'axe **Y** décrit et définit dans la **figure III.25(c)**. La figure finale étendue est définit par l'union des traits pleines présentées dans la **figure III.25.(d)**.

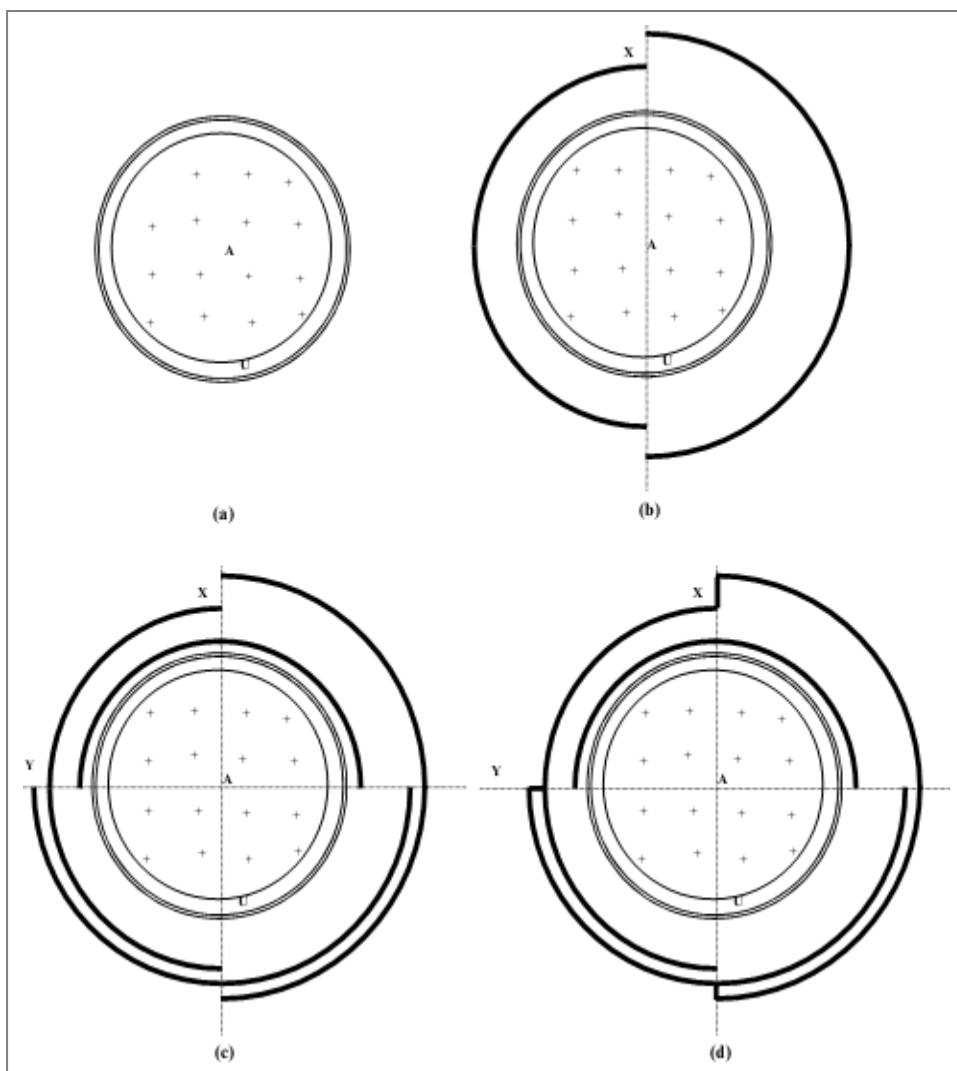


Figure III.25 : Exemple illustratif de l'expansion linéaire [Pham 2007]

III.4.4.4 Résoudre le Sous-Problème 6 :

Etant donnée un ensemble homogène **F** positif. Rappelons que si $DH(F) \leq \beta^+$, alors l'ensemble **F** est subdivisé en sous-modèles. D'après Théoreme1, ces sous-

modèles sont aussi homogènes. Considérons maintenant la figure **III. 23** ($\beta^+ = 1$, $\beta^- = 1.5$). Ainsi, l'ensemble homogène C est subdivisé en quatre petits cercles (une définition est similaire pour l'ensemble E). Ces petits cercles sont considérés comme des ensembles homogènes et leurs degrés d'homogénéité $DH=0$. Ainsi, cet ensemble n'est plus étendu. [Pham 2007]

Dans cette dernière partie du chapitre **III**, nous avons présenté la méthode **A.B.H** (Algorithme à Base d'Homogénéité) qui définit en premier temps une fonction objectif principale comme un problème d'optimisation en terme des taux du faux positif (T_{FP}), faux négatif (T_{FN}) et des cas non-classifiés (T_{NC}).

Ensuite, la méthode **A.B.H** organise les modèles inférés en régions représentés par des ensembles homogènes. Ces régions sont raffinées à base de leur densités en utilisant la métaheuristique **A.G** (Algorithme Génétique) pour minimiser le coût total d'échec de classification.

L'intérêt de la métaheuristique **A.B.H** est de trouver un contrôle simultané des deux systèmes (positif, négatif) afin de réduire significativement les trois coûts de pénalités (C_{FP} , C_{FN} , C_{NC}), lors de la combinaison de la métaheuristique **A.B.H** avec d'autres algorithmes de classification.

III.5 Conclusion

Ce troisième chapitre a été l'occasion de justifier et présenter les méthodes et les approches dont il sera fait usage dans la suite de ce mémoire de magistère. Au fur et à mesure de cette présentation, nous avons précisé à quels aspects nous souhaitons nous attacher plus particulièrement et à quels concepts nous faisons appel dans les modèles que nous proposons à savoir *ProSadm-HBA* et *F-HBA*.

D'abord le modèle *ProSadm* (*Programmation d'un Système d'Aide au Diagnostic Médical*), notre ancien système basé sur des techniques intelligentes qui a été développé dans le cadre du diagnostic du cancer du sein [Bekaddour 2012.a]. Nous avons ainsi défini ce que nous entendons par *ProSadm*, son principe, les méthodes adoptées (*LVQ*, *PMC*) par ce système, ses interfaces principales.

Ensuite, nous avons présenté la théorie de logique floue et les modèles Neuro-Flous afin d'améliorer l'interopérabilité du système. Nous avons ainsi cherché à définir ce que c'est un système d'inférence floue et les modèles hybrides neuro-flous en particulier le modèle *ANFIS* (*Adaptative Neuro-Fuzzy Inference System*). Ce modèle que nous l'avons déjà utilisé [Bekaddour 2012 .a] , a montré sa simplicité d'interaction ,flexibilité et sa capacité d'interprétation des résultats de classification .Ce qui nous a poussé à réutiliser à nouveau ce modèle et essayer d'enrichir par ajout de la méthode d'optimisation.

Et enfin, l'optimisation métaheuristique *H.B.A* (*Homogeneity Based-Algorithm*) pour améliorer la qualité des résultats obtenus. Nous avons ainsi cherché à définir la métaheuristique, le principe de la méthode, sa complexité, les heuristiques qu'elles adoptent...

Une fois argumentés et définis, nous partons de ces propos pour passer au chapitre suivant où nous allons décrire de manière exhaustive nos contributions : les modèles *ProSadm-HBA* et *F-HBA*. Nous détaillons leur architecture, le choix de leurs paramètres et nous donnons un aperçu global de chacun des approches proposées.

Chapitre IV:

Les modèles

ProSadm-HBA & F-HBA

Ce chapitre présente notre proposition constituée de deux contributions : *ProSadm-HBA* (un modèle reposant sur notre ancien système ProSadm et la métaheuristique **HBA**) & *F-HBA* (**F**uzzy **H**omogeneity **B**ased Algorithm).

IV.1 Introduction

L'objectif que nous nous étions fixé et annoncé au début de ce mémoire de magistère est l'amélioration des performances des classifieurs des données médicales à base des métaheuristiques en particulier l'algorithme **HBA (Homogeneity Based-Algorithm)**.

Dans ce chapitre, nous présentons d'abord notre première contribution à travers laquelle nous avons cherché à exploiter notre ancien système **ProSadm** (voir [Bekaddour 2012.a]) et que nous l'avons nommé **ProSadm-HBA**. Ce travail a été étendu par l'enfoncement des performances du classifieur à travers l'introduction d'une approche d'optimisation. Bien que les métaheuristiques ont été proposées depuis longtemps, elles ont reçu un grand intérêt et ont montré leur efficacité et à nos jours, elles restent applicables pour la résolution de nombreux problèmes d'optimisation.

Dans la première contribution, **ProSadm-HBA** a donné de bons résultats. Cependant ces résultats s'avèrent limités (boîte noire) et restent toujours non interprétables, la chose qui n'est pas exigée dans le domaine médical. Pour pallier ce problème, nous présentons dans la deuxième contribution : un modèle flou d'optimisation reposant sur la logique floue et la méthode d'optimisation métaheuristique **HBA [Pham 2011]**. Ce modèle est appelé **F-HBA** pour *Fuzzy Homogeneity Based Algorithm*, il sert les deux concepts : la logique floue et la métaheuristique. La logique floue comme méthode d'extraction des connaissances, et la métaheuristique H.B.A pour l'optimisation.

Ces deux contributions ainsi que le problème auxquelles elles sont dédiées, sont représentées par les schémas conceptuels des **figures IV.2 & IV.3 & IV.9**.

IV.2 description du problème

IV.2.1 Définitions de base

Pour faciliter la compréhension du problème, considérons l'ensemble des données d'apprentissage schématisé dans la figure **IV.1**. Les cercles et les carrés dans cette figure correspondent à un échantillon d'observations définissant deux classes en 2 dimension. Généralement, un *point de donnée* est un vecteur défini sur n variables avec ses valeurs attribuées. Dans la **figure IV.1**, n est égal à 2 et ces deux variables sont indiquées par les axes X et Y . Ces points de données décrivent le comportement du système analysé. Supposons que nous avons uniquement deux classes, alors appelons arbitrairement une des classes la classe *positive* et l'autre la classe *negative*. Un *point de donnée positive* est un point de donnée qui a été évalué d'appartenir à la classe positive. Une similaire définition pour le *point de donnée negative*. Étant donné un ensemble d'exemples positifs et négatifs, schématisés dans la figure **IV.1** qui présentent des données d'apprentissage (exemples de classification). [Pham 2007]

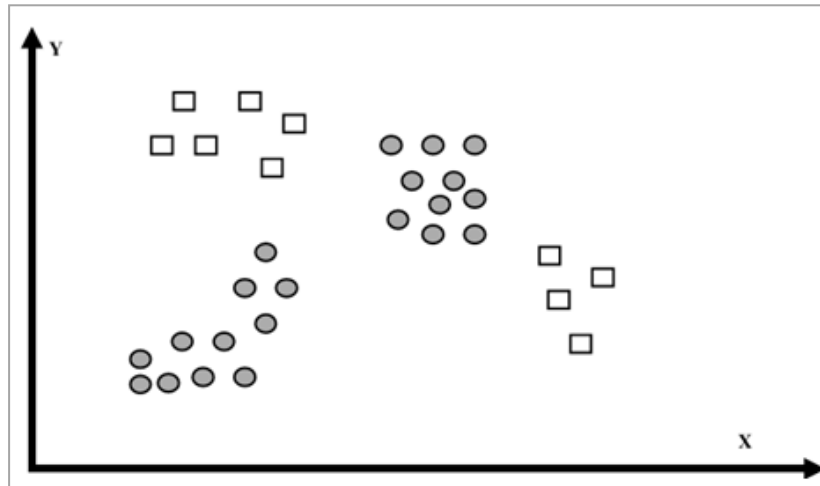


Figure IV.1 : Problème de bi-classification en 2-D [Pham 2007]

IV.2.2 Définition du problème

Supposons qu'une des méthodes de classification est appliquée sur l'ensemble des données présentées dans la **figure IV.1** (dans notre cas nous utilisons le modèle *ProSadm, Flou*). Supposons par la suite qu'il y a deux systèmes de classification qui ont été inférés à partir de ces données d'apprentissage. Un système décrit les données positives (Système positive) et l'autre décrit les données négatives (Système négative). Dans la **figure IV.2.a** les ensembles **A** et **B** définies le modèle positive & **C** et **D** correspond au modèle négative. Il existe deux type d'erreurs : si un des systèmes classifie un point de donnée positive comme négative ou un point de donnée négative comme positive. Le premier cas s'appelle Faux négative et le deuxième s'appelle Faux négative. La **figure IV.2** indique qu'il existe des points non classifiés (des objets qui appartiennent aux 2 classes simultanément ou à aucune des classes). [Pham 2007]

-Pour n : point qui n'est pas couvert par un des modèles .Le système le déclare comme un point non classifié. [Pham 2007]

-Pour m : point qui est couvert par deux modèle simultanément. Le système le déclare comme un point non classifié. [Pham 2007]

Considérons maintenant la **figure IV.2.b** qui réduit significativement les modèles **A, B, C & D** mais couvert toujours l'ensemble originale des données d'apprentissage .Cette figure montre que m et n ne sont plus couvert par un des modèles inférés. m et n sont des points non classifiés. En addition, des points qui ont été couvert simultanément par les deux modèles positive et négative sont maintenant couvert par un des modèles inférés ou d'aucun modèle .Cette **figure IV.2.b** a un cout de pénalité total élevé par rapport a la première considération (**figure IV.2.a**). Si l'on prend cette idée de réduire les ensembles couvrant autant que possible, alors on aura un cercle sur chaque point de donnée d'apprentissage .Ainsi, le cout de pénalité totale sera maximum et le système sera capable de classifiée uniquement les données d'apprentissage. Ce scénario est appelé *surapprentissage*. [Pham 2007]

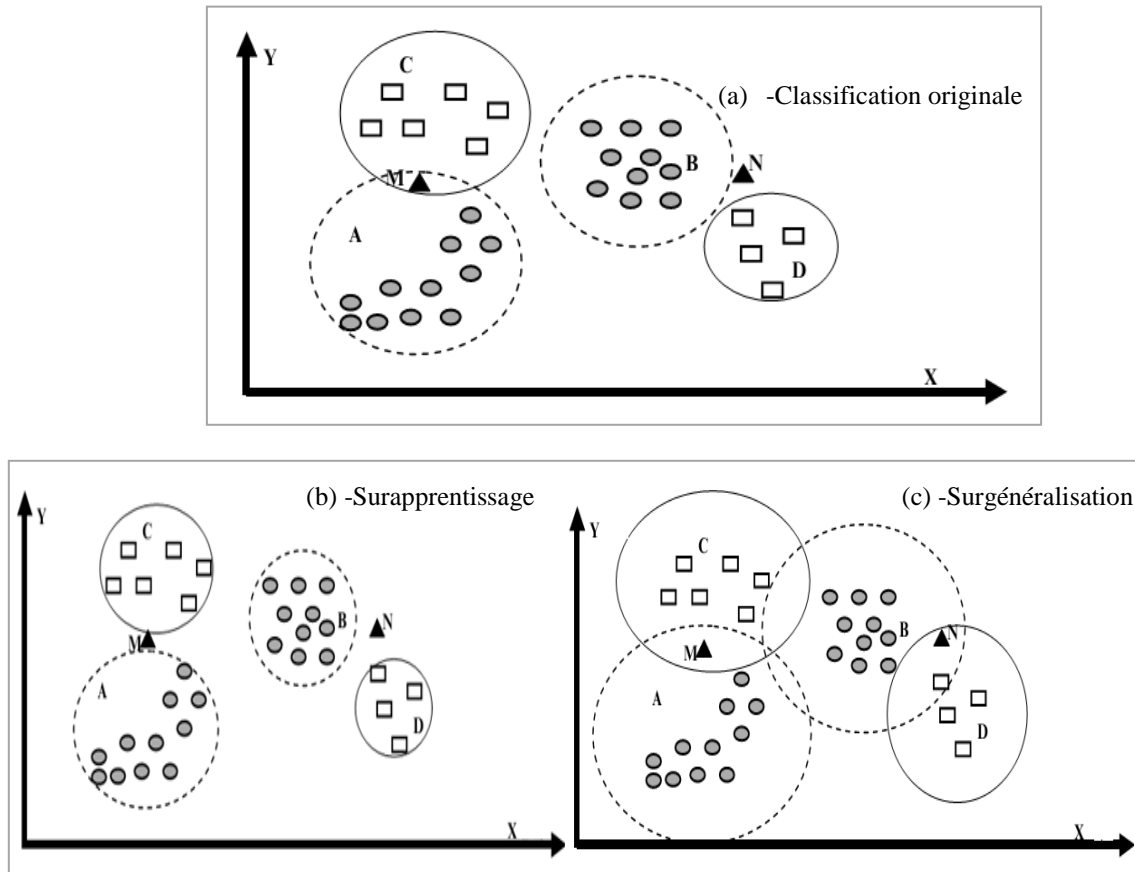


Figure IV.2 : exemple illustratif du surapprentissage et surgénéralisation [Pham 2007]

En revanche, supposons que les modèles **A**, **B**, **C** et **D** originaux présentés par la **figure IV.2.a** sont maintenant significativement étendus (voire **figure IV.2.c**). Cette figure montre que les points m et n et d'autres points sont maintenant couverts par les deux modèles (positive et négative) à la fois. Ce scénario présente un nombre élevé des points non classifiés à cause du désaccord entre les deux systèmes de classification (positive et négative). Ce scénario est appelé *surgénéralisation*. [Pham 2007]

La séparation indépendante entre le « fitting » et la « généralisation » est déconseillé. Ainsi, le besoin d'une approche qui fait une balance simultanée entre le fitting et la généralisation est obligatoire pour l'ajustement des systèmes inférés (positive et négative) obtenus par une méthode de classification. [Pham 2007]

IV.2.3 Objective

Notons que C_{FP} , C_{FN} et C_{NC} , trois coûts de pénalités pour le faux positif, le faux négatif et les objets non classifiés respectivement [Pham 2011]. Le problème étudié dans ce mémoire de magister est la modification d'un modèle de classification existant telle que la formule de C_T (i.e. le coût totale d'échec de classification) soit minimisé ou significativement réduit.

$$C_T = \min (C_{FP} \times \text{Taux}_{FP} + C_{FN} \times \text{Taux}_{FN} + C_{NC} \times \text{Taux}_{NC}) \quad (4.1)$$

IV.3 Description des bases de données

Dans ce mémoire de magistère nous utilisons trois bases de données médicales. Ces bases de données sont souvent utilisées par les méthodes de "Data Mining" pour analyser et valider ces approches. Les principales caractéristiques de ces ensembles de bases sont représentés dans le tableau suivant :

Bases de Données	Nbr d'attributs	Nbr de cas	cas positifs	cas négatifs	Taille-Base d'apprentissage	Taille-Base de Test
PID (diabète)	8	768	268	500	576	192
Troubles hépatiques	6	345	145	200	276	69
Appendicite	7	106	85	21	85	21

Tab IV.1 : Caractéristiques des bases de données utilisées

Ces bases de données utilisées ont été largement employées dans le domaine de classification. Ci-dessous, nous donnons des exemples de travaux issus de ces bases de données.

IV.3.1 Base de données du diabète PID (Pima Indian Diabetes)

IV.3.1.1 Contexte du diabète

Le diabète est l'une des maladies les plus dangereuses, connu aussi avec le nom de "tueur silencieux". Son incidence est en augmentation. L'augmentation du nombre de diabétique est tellement rapide que l'organisation mondiale de la santé (OMS) l'a identifié comme étant une épidémie. [Saidi 2012]

Le diabète est une condition chronique caractérisée par une insuffisance absolue ou relative de la sécrétion de l'insuline par le pancréas. L'insuline est une hormone qui régule la concentration de sucre dans le sang. [Settouti 2011]. Les tests suivants sont utilisés pour le diagnostic (présence du diabète):

- Glycémie plasmatique occasionnelle : c'est le niveau de glucose faite à n'importe quel moment de la journée, sans tenir compte des repas. Elle doit être ≥ 200 mg/dl et accompagner de symptômes classiques du diabète.

- Glycémie plasmatique à jeun : mesure la glycémie chez une personne qui n'a rien mangé pendant au moins 8 heures. Ce test est utilisé pour détecter le diabète. Elle doit être ≥ 126 mg/dl.

- Hyperglycémie provoquée par voie orale : la glycémie est mesurée après le jeûne d'au moins 8 heures et ensuite 2 heures après consommation d'une boisson contenant du glucose. Elle doit être ≥ 200 mg/dl.

On distingue 2 principaux types de diabète : le diabète de type 1 et le diabète de type 2.

- **Le diabète de type 1:** survient quand le pancréas ne peut pas fabriquer de l'insuline. Toutes les personnes atteintes de diabète de type 1 doivent recevoir des injections régulières d'insuline. Les chercheurs pensent que l'affection s'acquiert quand un virus endommage le pancréas ou incite le système immunitaire à attaquer les cellules bêta du

pancréas (*réaction auto-immune*). Les symptômes sont les suivants : excrétion excessive d'urine (polyurie), sensation de soif (polydipsie), perte de poids, altération de la vision et fatigue. Ces symptômes peuvent apparaître brutalement.⁷

- **Le diabète de type 2** apparaît quand le pancréas ne sécrète pas suffisamment d'insuline, ou quand le corps n'utilise pas l'insuline convenablement. Il se révèle généralement à l'âge adulte bien qu'il puisse parfois toucher les enfants. Le diabète de type 2 est principalement causé par une résistance à l'action de l'insuline. Cela signifie que l'organisme ne peut pas utiliser l'insuline aussi bien qu'il le devrait, quelle que soit la quantité produite. Par conséquent, le glucose qui est dans le sang ne peut pas être transporté à l'intérieur des cellules. Au fil du temps, l'excès de glucose dans le sang mène à une détérioration graduelle du pancréas dont la production d'insuline diminue et complique encore plus la prise en charge du taux de glucose sanguin. Les symptômes du diabète de type 2 apparaissent habituellement de façon progressive. Les personnes atteintes d'un diabète de type 2 et dont la glycémie n'est pas maîtrisée ont souvent une soif légère, mais persistante. Elles urinent fréquemment, ressentent souvent une légère fatigue et se plaignent d'une vision trouble.⁷

Le diabète joue un rôle d'une importance majeure dans la survenue d'une maladie cardiaque, Les personnes atteintes de diabète sont également beaucoup plus susceptibles de subir des amputations du pied ou d'autres parties de la jambe à cause de problèmes circulatoires⁷. Pour parvenir à lutter efficacement contre le diabète, il est important de sensibiliser le public et les milieux professionnels aux facteurs de risque et aux symptômes. La prévention est assurée par des bilans régulière : [**Ammar 2008**]

- De l'état nutritionnel du malade.
- De la glycémie.
- Du retentissement sur l'œil.
- Des retentissements cardio-vasculaires.
- Surveillance rénale.

- Tout diabétique doit être pris en charge par une diététicienne qui établit un régime adapté à sa pathologie.

IV.3.1.2 Description de la base de données de diabète

Comme décrit dans [**Settouti 2011**] & [**Saidi 2012**], l'ensemble de données a été choisi du dépôt d'UCI (University of California, Irvine) qui réalise une étude sur 768 femmes Indiennes Pima dont 268 sont des cas diabétiques et 500 non diabétiques. Le tableau suivant présente une description de ces attributs :

Attributs	Description
Npreg	Nombre de grossesses
Glu	Concentration du glucose plasmatique
BP	Tension artérielle diastolique
SKIN	Epaisseur de pli de peau du triceps
Insuline	Dose d'insuline
BMI	Index de masse corporelle
PED	Fonction de pedigree de diabète (l'hérédité)
Age	Age

Tab IV.2 : Description des attributs de la base **PID** (Diabète)

La variable de classe indique si une personne est atteinte de diabète ou non.

IV.3.1.3 Travaux menés sur la base de diabète

Au & Chan [Au 2001] ont amélioré le taux de classification en utilisant une approche floue. Leur approche a achevé une performance de 77.6%. Blachnik & Duch [Blachnik 2006] ont utilisé plusieurs méthodes dont C4.5 a donné une meilleure performance 74.48%. Pham et Triantaphyllou [Pham 2009] ont appliqué la métaheuristique HBA (Homogeneity-based algorithm) en conjonction avec la machine à vecteur de support, les réseaux de neurones et l'arbre de décision. Leur meilleur performance a atteint $\approx 94.8\%$. Settouti & Chikh [Settouti 2012] ont obtenue un taux de classification de 83.85% en hybridant l'algorithme c-moyens flous et le modèle ANFIS.

IV.3.2 Base de données du Troubles hépatiques (Liver Disorders).

IV.3.2.1 Contexte du Troubles Hépatites

Le foie est le plus gros organe interne du corps. Ses fonctions principales consistent à:

- Stocker des nutriments;
- Fabriquer des protéines;
- Nettoyer le sang en le débarrassant des médicaments, de l'alcool et des substances chimiques potentiellement nocives qu'il contient, et en traitant ces substances chimiquement pour qu'elles puissent être excrétées par le système digestif ou l'appareil urinaire⁷.

Il existe une large gamme de troubles du foie, entre autres les hépatites, les tumeurs. L'hépatite est une inflammation du foie qui peut être provoquée par un virus, et parfois par certains médicaments ou toxines tels que l'alcool et les drogues. Ces maladies du foie peuvent se manifester par différents symptômes tels que la fatigabilité ; la jaunisse (la coloration jaune de la peau et du blanc des yeux), des nausées, une urine foncée (de la couleur du thé)... Les médecins dépistent les hépatites à l'aide d'analyses sanguines et d'un dossier médical.^{7, 8}

Les scientifiques ont identifié 4 types principaux d'hépatite virale : l'hépatite **A**, l'hépatite **B**, l'hépatite **C** et l'hépatite **D**. L'hépatite **A** est véhiculée par l'eau et se transmet principalement par l'intermédiaire des égouts, de l'eau et des aliments

⁷ <http://sante.canoe.ca>

⁸ <http://www.pipelette.com/sante/les-solutions-contre-les-troubles-hepatiques.html>

contaminés. L'hépatite **B** est transmise par le sang et les liquides corporels .L'hépatite **C** se propage par contact direct sanguin. L'hépatite **D** se transmet par l'intermédiaire d'aiguilles et de transfusions sanguines contaminées.⁷

Le meilleur moyen de prévenir les troubles hépatiques est d'adopter une alimentation saine et équilibrée. De ce fait, on devra éviter les encas et les goûters pris à des heures irrégulières. Par contre, on doit consommer des fruits et des légumes au cours des trois repas quotidiens⁸

IV.3.2.2 Description de la base de données du Troubles hépatiques

Cette base provient du centre de recherche médical BUPA : British United Provident Association (Richard S. Forsyth, Mapperley Park, Nottingham, UK). Elle est composée de 345 patients dont 145 cas présentent un diagnostic positif et 200 cas présentent un diagnostic négatif. Un diagnostic indique si une personne est atteinte d'une hépatite ou pas .Les principales caractéristiques de cette base de données sont présentées dans le tableau IV.3 suivant :

Attributs	Description
VGM	Volume globulaire moyen
Alkphos	Alkaline phosphatase
SGPT	Sérum glutamo pyruvique transaminase
SGOT	Sérum glutamo oxalacétique transaminase
Gammagt	Gamma glutamyl transpeptidases
Drinks	consommation alcoolique (number of half-pint equivalents of alcoholic beverages drunk per day)

Tab IV.3 : Description des attributs de la base **T.H** (Troubles hépatiques)

Les cinq premiers descripteurs correspondent à des tests sanguins donnant la sensibilité des troubles du foie.⁹

IV.3.2.3 Travaux menés sur la base du Troubles hépatiques

La deuxième base de données concerne les troubles hépatiques comme décrit dans [Asuncion 2007] . [Pham 2009] ont utilisé la métaheuristique HBA (**H**omogeneity-**B**ased **A**lgorithm) en conjonction avec machine a vecteur de support SVM, les réseaux de neurones RN et l'arbre de décision AD. Les taux de classification obtenus sont 100%, 98.6%, 95.7 % respectivement. P.Rajeswari & G. Sophia Reena [Rajeswari 2010] ont obtenu le meilleur taux de performance de 96.52% en utilisant un classifieur bayésien naïf. En 2011, B.Ramana ,M.Babu & N. B. Venkateswarlu [Ramana 2011] ont utilisé plusieurs algorithmes de classification et leurs étude a montrée que le classifieur bayésien naïf , C 4.5 , l'algorithme de rétro-propagation ,la méthode K-plus proche voisin , SVM ont obtenus des performances de classification de 56.52% , 68.69 % , 71.59% ,62.89 % et 58.26%, respectivement.

⁷ <http://sante.canoe.ca>

[101]

⁸ <http://www.pipelette.com/sante/les-solutions-contre-les-troubles-hepatiques.html>

⁹ Asuncion, A., Newman, D. J. (2007). UCI-Machine Learning Repository. School of Information and Computer Sciences, University of California, Irvine, CA, USA.

IV.3.3 Base de données d'Appendicite

IV.3.3.1 Contexte de l'Appendicite

On appelle appendicite une inflammation soudaine ou un gonflement de l'appendice. L'appendice est une structure en forme de tube qui part de l'endroit où le gros intestin (le *colon*) commence. Il est de la taille d'un crayon et est généralement long d'environ 10 cm (4 pouces).⁷

L'inflammation de l'appendice est généralement le résultat d'une obstruction. L'une des extrémités de l'appendice est ouverte à l'endroit où elle rejoint le gros intestin. L'appendicite peut survenir lorsque l'appendice est obstrué par une masse dure de fèces ou par un corps étranger se trouvant dans l'intestin. Cette obstruction peut directement provoquer l'inflammation de l'appendice et peut favoriser une infection bactérienne.⁷

Les premiers symptômes de douleur se manifestent généralement près du nombril et évoluent graduellement vers la partie inférieure droite de l'abdomen. La douleur est souvent accompagnée des symptômes suivants : - Nausées ou vomissements - Perte d'appétit - Faible fièvre. Diarrhea¹⁰

L'appendicite est généralement diagnostiquée lors d'un examen clinique, par un médecin. Le médecin effectuera un nombre de tests qui pourront également donner des informations sur l'étendue et la localisation de l'inflammation. Après un examen physique, une analyse de sang peut être effectuée pour rechercher une infection. Parfois, on a recours à une échographie pour aider à déterminer le diagnostic. L'intervention chirurgicale est l'unique traitement des appendicites aiguës. On enlève souvent l'appendice dans les heures qui suivent le diagnostic.⁷

IV.3.3.2 Description de la base de données d'Appendicite

Appendicite (Appendicitis) . Cette base de données créée par **Shalom Weiss** de l'université **Rutgers** est composée de 106 exemples avec sept descripteurs numériques décrit dans le tableau **IV.4.** qui représente des tests de laboratoires. Ces exemples sont répartie en deux classe : 80.2% des données appartient a la classe '1' (diagnostic positive) et 19.8 % appartient a la classe '2' (diagnostic négative).

Attributs	Description
1	WBC1
2	MNEP
3	MNEA
4	MBAP
5	MBAA
6	HNEP
7	HNEA

Tab IV.4 : Description des attributs de la base **AP** (Appendicite) [Weiss 89]

⁷ <http://sante.canoe.ca>

¹⁰ <http://www.dknews-dz.com/cms/sciences-medecine-technologie/sante/3995-appendicite-comment-la-diagnostiquer-rapidement>

IV.3.3.3 Travaux menés sur la base d'Appendicite

Citons en d'abord les travaux de Weiss et Kapouleas [Weiss 1989] qui ont utilisé différentes approches de classification (Cart , Réseau de neurone , k-plus proches voisins ...) .Ils ont atteint une meilleur performance de 89.6%. En 2003 Nakashima, Nakai, & Ishibuchi (2003) [Nakashima 2003] ont proposé l'usage d'un système de classification flou pour un mappage d'un espace d'entrées à un système de classification à base de règles floues. Leur méthode a donnée un taux de classification de 84%. Blachnik & Duch [Blachnik 2006] ont utilisé plusieurs méthodes dont Nefclass (Neuro-Fuzzy Classification) a donné une meilleure performance 87.7%. [Pham 2009] ont utilisé la métaheuristique HBA (Homogeneity-Based Algorithm) en conjonction avec machine a vecteur de support, les réseaux de neurones et l'arbre de décision. Les meilleurs taux de classification obtenus sont 100%, 96.4%, 89.3% pour SVM-HBA, ANN-HBA, DT-HBA respectivement.

IV.4 Critères d'évaluation

Afin d'évaluer nos approches et précisément la métaheuristique **HBA** (*Homogeneity Based-Algorithm*) et les modèles **ProSadm** (*Programmation d'un Système d'Aide au Diagnostic Médicale*), *Flou*(*Neuro-Flou*) adoptés, nous proposons dans le chapitre suivant une étude comparative selon les paramètres présentés au-dessous (pour plus de détaille voir section II.2):

- ✓ Taux de faux positif (**FP**).
- ✓ Taux de faux négatif (**FN**).
- ✓ Rappel (Sensibilité) (**Rp**).
- ✓ Spécificité (**Sp**)
- ✓ Taux de Classification

Ces paramètres sont présentés dans la matrice de confusion (appelé aussi tableau de contingence) donnée par le tableau IV.5. C'est un tableau qui sert à mesurer les performances des résultats d'une classification.

		Sortie Désirée	
		Correcte	Incorrecte
Sortie Réelle	Correcte	vrais positifs	Faux positifs
	Incorrecte	Faux négatifs	vrais négatifs

Tab IV.5 Matrice de confusion

Dans ce travail, nous utilisons d'autres critères d'évaluations cités au-dessous :

- ✓ Taux des cas non-classifiés (**N_C**).
- ✓ **Cout Total** des objets non classifiable (Total mis classification). Dans le cadre ce mémoire de magister, les valeurs de **C_T** sont calculées comme suite :

$$C_T = \min (\text{TauxFP} + \text{TauxFN})$$

$$C_T = \min (1*\text{TauxFP} + 20*\text{TauxFN} + 3 \text{Taux}_{N_C})$$

$$C_T = \min (1*\text{TauxFP} + 100*\text{TauxFN} + 3\text{Taux}_{N_C})$$

- ✓ **Taux d'amélioration** : de x par rapport y est donné par la formule suivante :

$$\text{Taux d'amélioration (x,y)} = [(y-x) / y] * 100$$

IV.5 Le modèle ProSadm-HBA

IV.5.1 Architecture du modèle

Comme a été mentionné, la métaheuristique **HBA** (**H**omogeneity-**B**ased **A**lgorithm) est utilisée en combinaison avec notre ancien système **ProSadm**. Le but est d'améliorer la performance de la classification. La figure suivante montre un schéma général de la démarche proposée (**ProSadm-HBA**).

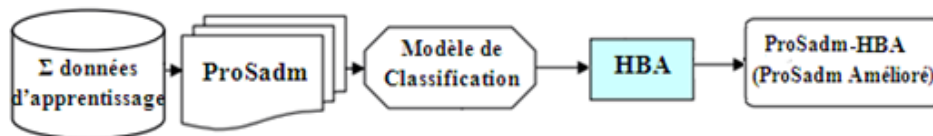


Figure IV.3 : Schéma général de **ProSadm-HBA** (1^{ère} Contribution)

Etant donné une base de données médicale, nous appliquons dans un premier temps une approche de classification traditionnelle (notre modèle **ProSadm**) pour inférer des paires de modèles classifications. A savoir, le modèle *positif* pour le quel une zone de décision est définie sur des points positifs de données d'apprentissage, et le modèle *négatif* pour le quel une zone de décision est définie sur des points négatifs de données d'apprentissage [Pham 2009]. Ensuite la métaheuristique **HBA** ajuste les modèles inférés en utilisant les étapes présentées dans le chapitre précédent. A cette manière, il est a espéré que la performance de la méthode de classification traditionnelle sera améliorée [Pham 2009].

IV.5.2 Implémentation et choix des paramètres:

Les figures **IV.4, 5** illustrent des exemples des réseaux implémentés pour le nouveau système **ProSadm-H.B.A** (**P.M.C** le **P**erceptron **M**ulti**C**ouches et le réseau **L.V.Q** (**L**earning **V**ector **Q**uantization)).

- Pour le réseau **P.M.C** , nous utilisons un réseau de trois couches (nous fixons le nombre de couches cachées a **1** à fin de minimiser la complexité du réseau) : une couche d'entrée qui reçoit les différents objets accompagnés de leurs caractéristiques ; une couche cachée qui fait l'apprentissage en utilisant la méthode de retro-propagation et une couche de sortie comportant les différentes classes cibles des objets introduits. Les paramètres telle que (Le nombre d'itérations, les neurones dans la couche cachée, ..) seront initialisés expérimentalement.

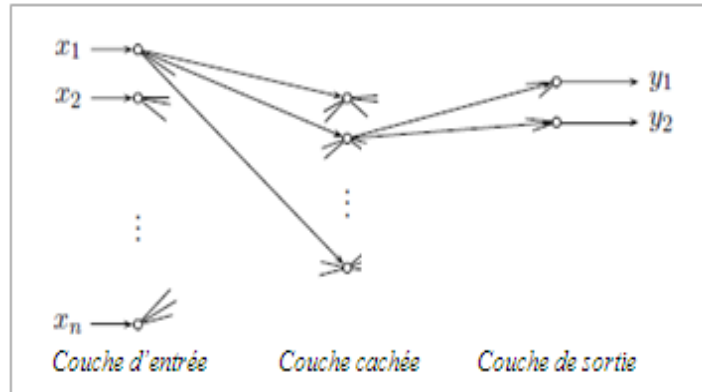


Figure IV.4 : Réseau **P.M.C** implémenté pour le système **ProSadm-HBA**

- Pour le réseau de **L.V.Q** : il comporte trois couches comme le montre la **figure IV.5** une couche d'entrée, une couche de compétition et une couche de sortie. Pour la couche de compétition et durant l'apprentissage : nous calculons les distances entre les poids et les différents vecteurs d'entrées, le neurone aura sélectionné comme gagnant s'il possède la distance la plus faible. Son poids sera ajusté en l'éloignant ou en le rapprochant du vecteur d'entrée suivant la sortie du réseau. Le nombre de neurones cachés, le pourcentage de distribution des neurones cachés, le nombre d'époques et autres paramètres sont importants pour la réalisation de ce réseau et seront choisis expérimentalement.

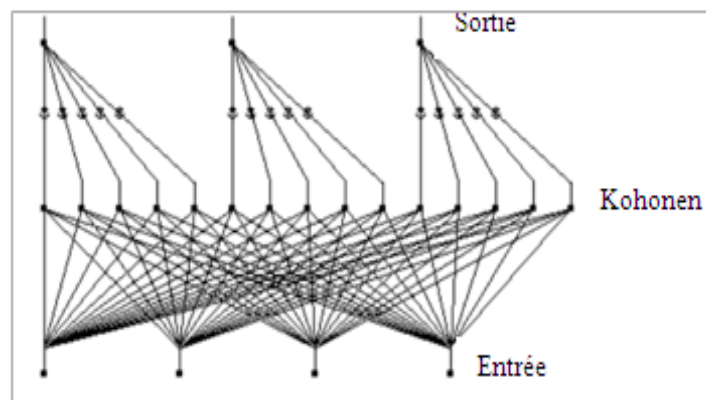


Figure IV.5 : Réseau **QVA** implémenté pour le système **ProSadm-HBA**

IV.5.3 Aperçue sur l'application :

Comme a été mentionné, le **HBA (Homogeneity-Based Algorithm)** est utilisé en combinaison avec notre ancien model **ProSadm**. Le but est d'améliorer la performance de classification. La figure suivante montre un schéma général de l'application proposée

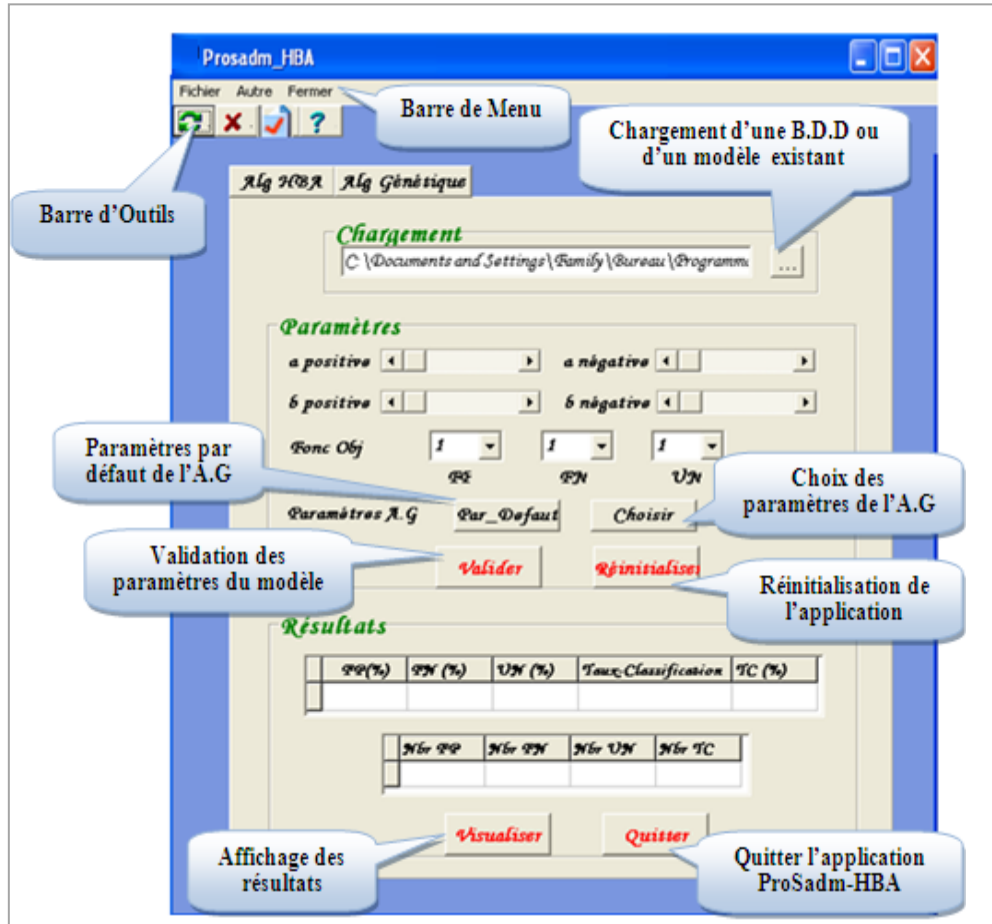


Figure IV.6 : Fenêtre Principale de l'application ProSadm-HBA

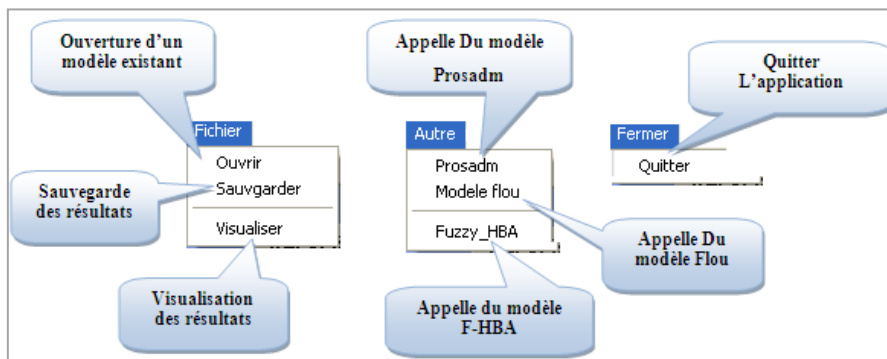


Figure IV.7 : La barre de menu

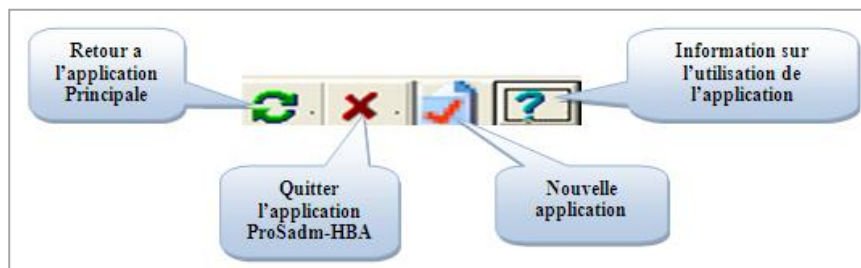


Figure IV.8 : La barre d'outils

Cette application et l'application suivante comportent trois niveaux :

- *Niveau noyau* : regroupe l'implémentation des algorithmes de classification.
- *Niveau fichier* : concerne la création, l'ouverture, l'enregistrement des réseaux implémentés et des résultats.
- *Niveau interface* : Il ne s'agit pas uniquement d'un intermédiaire entre l'utilisateur et notre application mais aussi d'une interaction avec les deux autres niveaux cités au dessus.

IV.6 Le modèle F-HBA

IV.6.1 Architecture du modèle

Le modèle **ProSadm-HBA** a donné de bons résultats comme le montre le chapitre V suivant. Cependant ces résultats restent toujours non interprétables ce qui n'est pas exigé dans le domaine médical. Résoudre ce problème au moyen de l'application **F-HBA** (**F**uzzy **H**omogeneity **B**ased-**A**lgorithm) que nous présentons permet d'augmenter la transparence du classifieur. La figure IV.9 présente un schéma général de l'approche **F-HBA**.

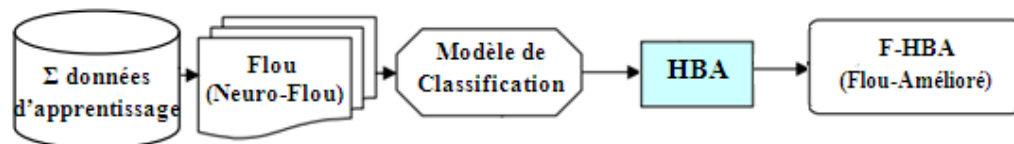


Figure IV.9 : Schéma général de F-HBA (2^{ème} Contribution)

IV.6.2 Implémentation & choix de paramètres

Notant que le nombre des fonctions d'appartenance est un point important pour l'implémentation des systèmes neuro-flous. [Bekaddour 2012.a]. Cela affecte le nombre des règles générés. Les contraintes qu'on rencontre généralement lors de développement de ces modèles sont les suivants [Ammar 2008] :

- Le nombre des règles floues générées
- Les règles générées doivent avoir le même langage utilisé par l'expert.
- Trouver un modèle qui répond au compromis suivant :
 - Un taux de classification acceptable, une erreur quadratique moyenne petite et un nombre de règles raisonnable.

L'exemple suivant cité dans [Bekaddour 2012.b] illustre mieux ces contraintes.

Configuration :	Nombre_regles :	Erreur-apprentissage :	Taux_Classif %
2*2*2*2*2	64	0.29843	0.9825
2*3*3*3*2*2	216	0.16176	0.8904
2*2*2*3*2*3	144	0.16982	0.9386

Tab IV.6 : Configuration du modèle Neuro-Flou pour la reconnaissance du cancer de sein [Bekaddour 2012.b]

Ce tableau montre clairement que le nombre de règles dépend du choix de la configuration (i.e. nombres de descripteurs & nombre des fonctions d'appartenances pour chaque descripteur) du modèle utilisé.

Avant l'apprentissage, nous fixons la configuration suivante pour le modèle flou (neuro-flou) :

- Le **SIF** (System d'Inférence Flou) utilisé dans ce mémoire de magister est de type Takagi-Sugeno.
- Le nombre de fonctions d'appartenances est fixé à 2 à fin de réduire la taille des règles générées.
- Les fonctions d'apparences choisies sont de type fonctions de bell généralisées (generalized bell functions) pour garder la lisibilité des résultats

Les modèles Neuro-Flous générés ainsi leurs réseaux équivalents sont présentés par les figures **IV.10**, **IV.11**, **IV.12** suivantes pour les bases de données **PID**, **T.H** et **A.P** citées au dessus.

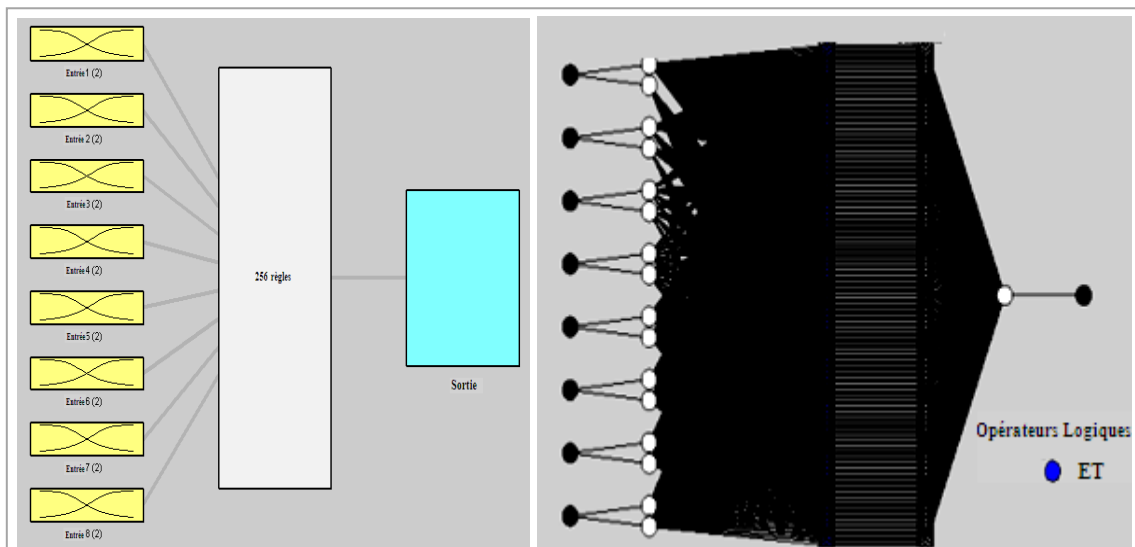


Figure. IV.10 : Réseau Neuro-Flou développé pour la base **PID** (Diabète)

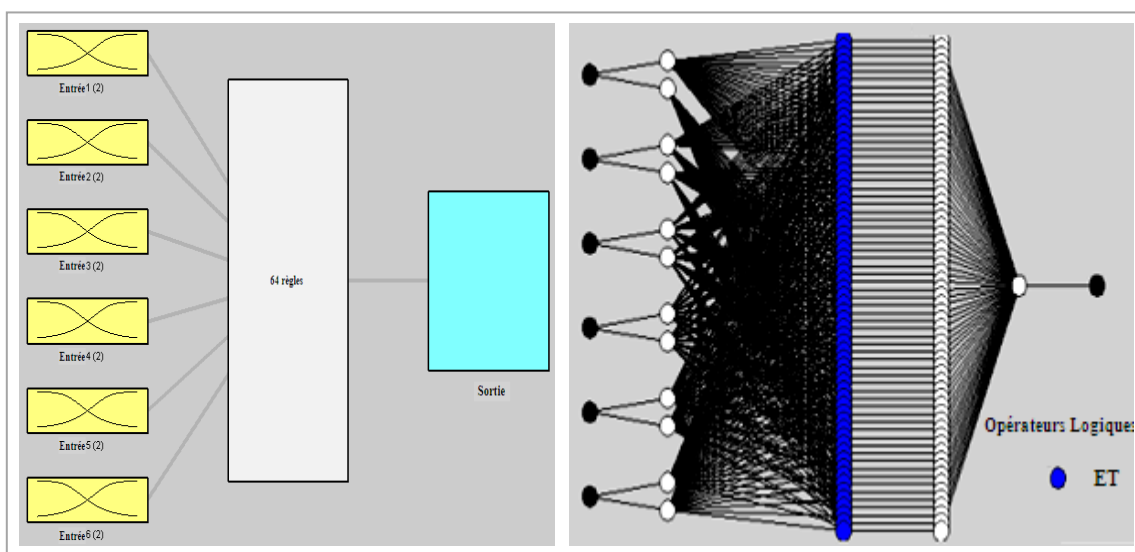


Figure IV.11 : Réseau Neuro-Flou développé pour la base **TH** (Troubles Hépatiques)

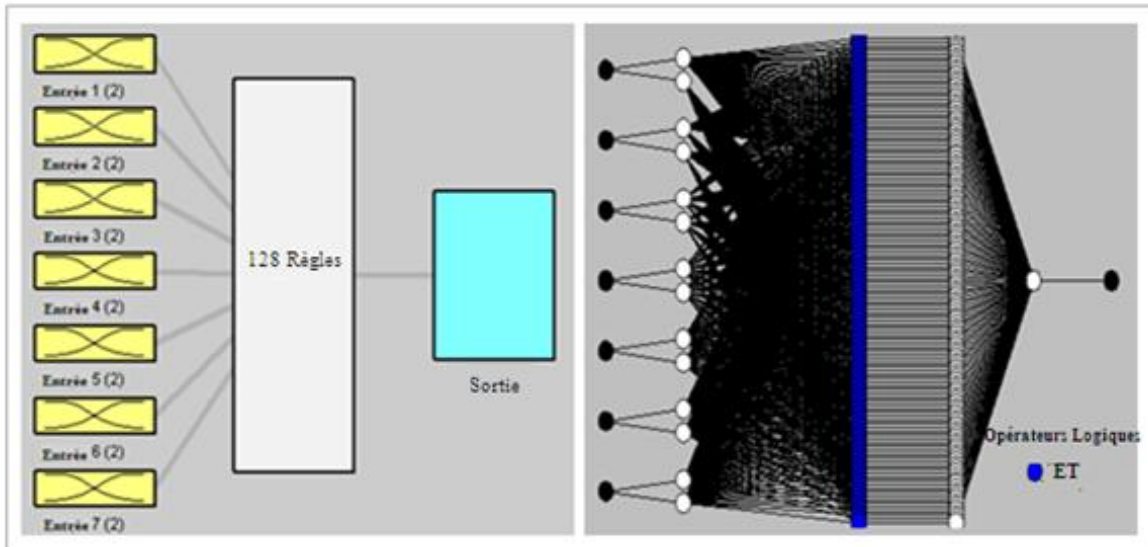


Figure IV.12 : Réseau Neuro-Flou développé pour la base AP (Appendicite)

IV.6.3 Aperçu sur l'application

La figure IV.13 schématise l'interface principale de l'application F-HBA proposée.

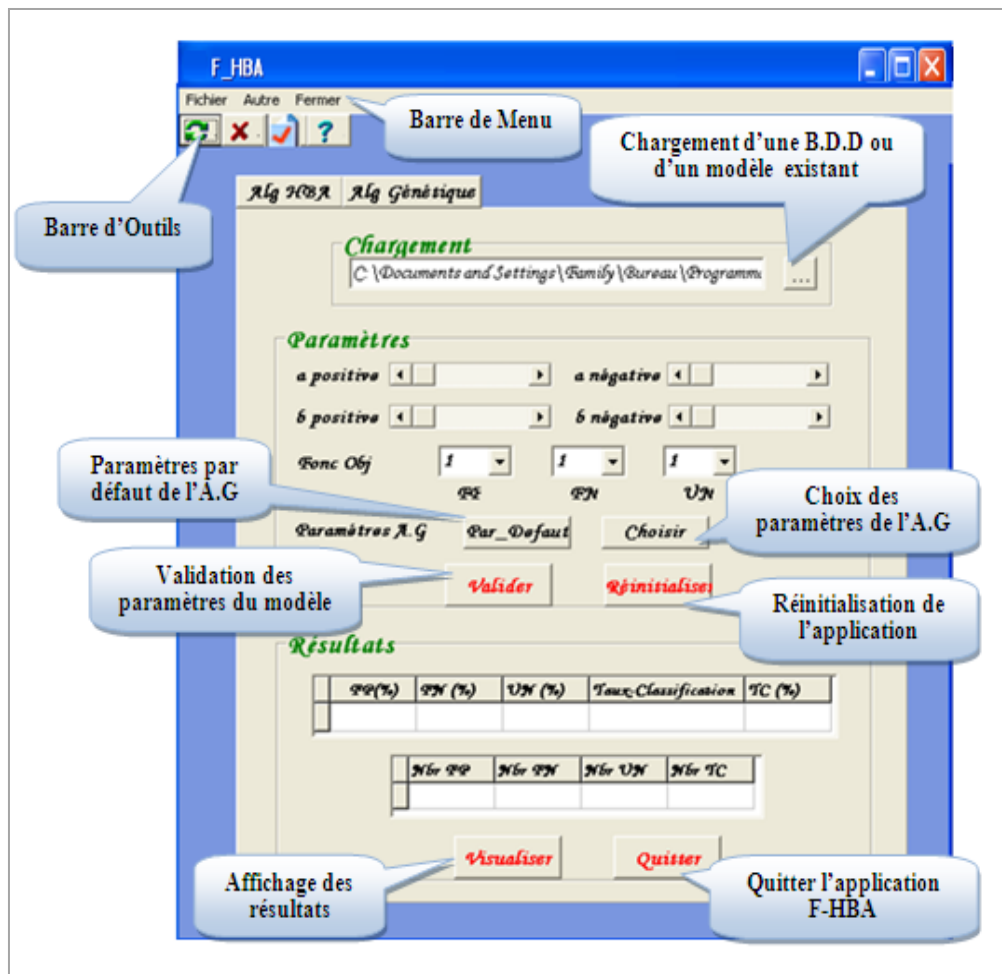


Figure IV.13 : Fenêtre Principale du modèle F-HBA

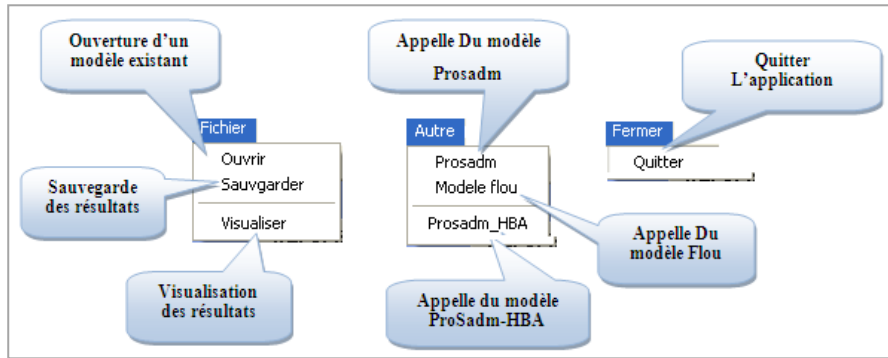


Figure IV.14 : La barre de menu

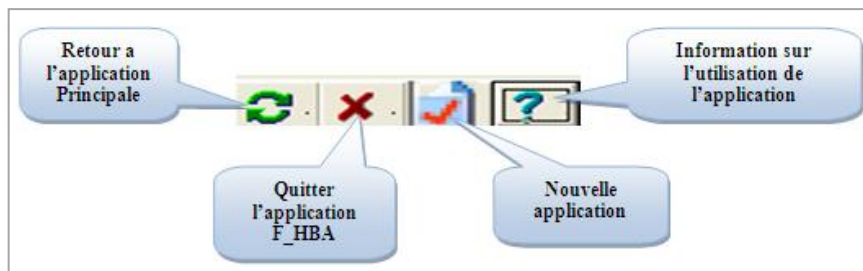
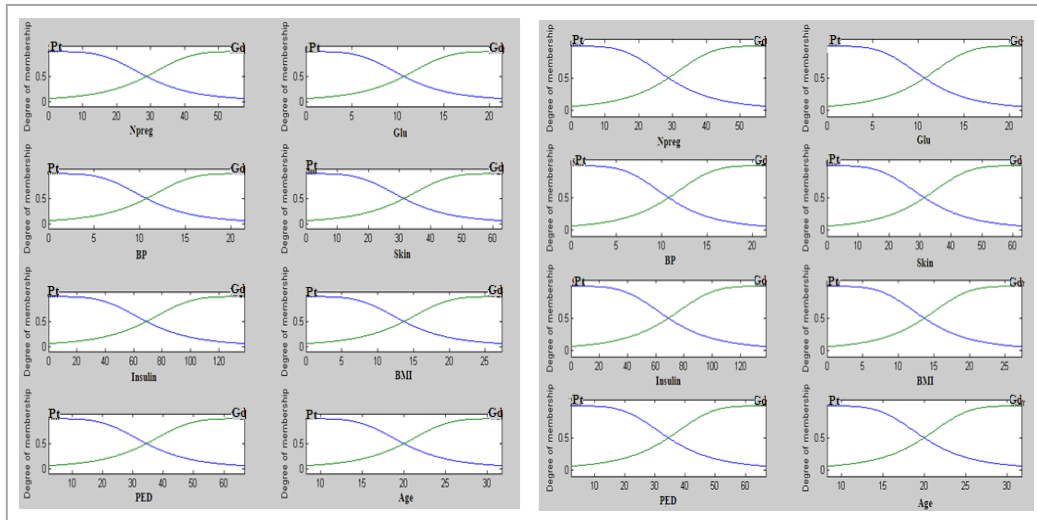


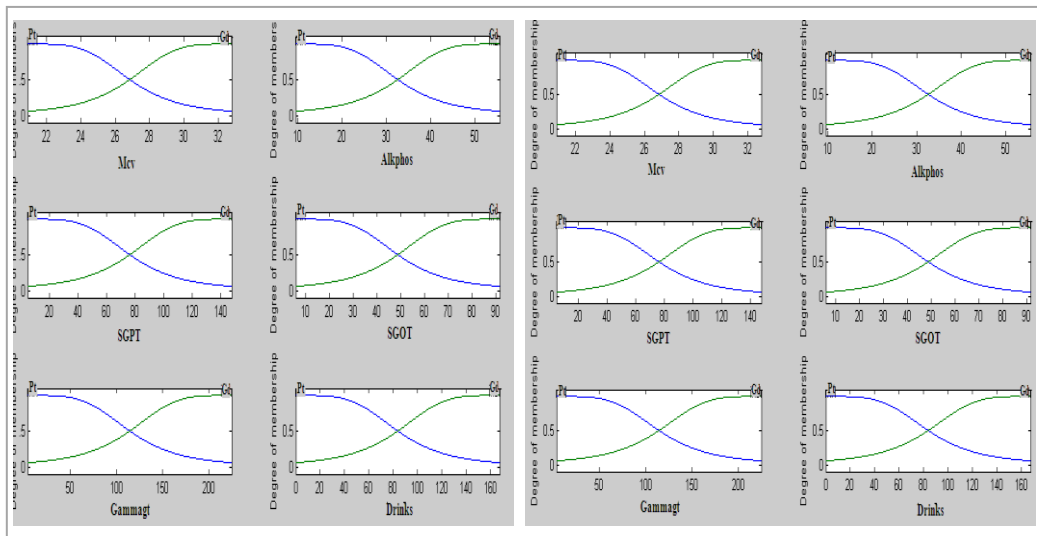
Figure IV.15: La barre d'outils

Les fonctions d'appartenances du modèle **Flou (F-HBA)** avant et après l'apprentissage pour les trois bases de données utilisées sont présentées dans la **figure IV.16** suivante :

(1)



(2)



(3)

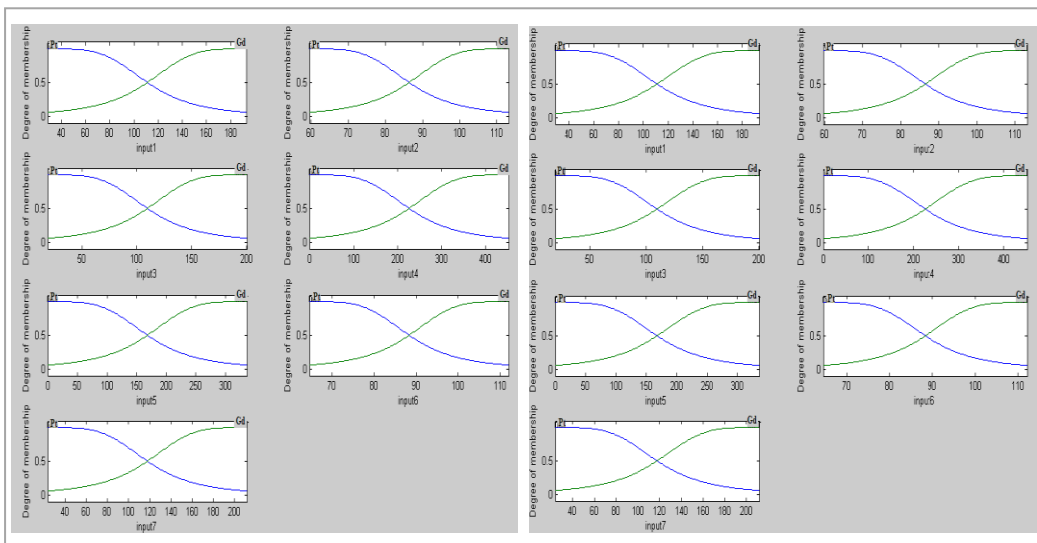


Figure IV.16 : Fonctions d'appartenance Avant (à gauche) & Après (à droite) pour les bases de données **PID⁽¹⁾**, **TH⁽²⁾**, **AP⁽³⁾** respectivement

IV.7 Conclusion

Dans ce chapitre ,nous avons décrit deux modèles d'optimisation a base de *ProSadm* (**P**rogrammation d'un **S**ystème d'**A**ide au **D**iagnostic **M**édical , notre ancien système développé voir chapitre **III**) et la logique floue de Zadah, en se référent a une méthode méta-heuristique *HBA* (**H**omogeneity **B**ased-**A**lgorithm) pour la résolution du problème d'optimisation des classifieurs .Nous avons exprimé le problème étudié de façon formelle .Cette démarche de formalisation nous a aidé a la clarification du problème .Elle permet également au lecteur de ce mémoire de magister de se familiariser plus avec les concepts de surgénéralisation, surapprentissage et d'interprétabilité, les principaux problèmes des classifieurs et de le préparer a nos deux contributions : les modèles *ProSadm-HBA* & *F-HBA* , deux modèles d'optimisation métaheuristique pour l'aide au diagnostic médicale.

En effet, dans ce chapitre, nous avons détaillé ces modèles, nous avons commencé par une description générale des architectures des démarches proposées .En suite, nous avons décrit d'une part les choix des paramètres des modèles proposés et d'autre part, l'implémentation et la présentation des interfaces des applications développées.

Afin d'évaluer ces modèles, nous proposons dans le chapitre suivant une étude empirique comparative selon divers paramètres présentés dans ce chapitre **IV**.

Chapitre V :

Validation Expérimentale

Ce chapitre valide nos approches proposées (*ProSadm-HBA* & *F-HBA*) en procédant à une étude détaillée des résultats obtenus, une analyse de nos expérimentations et une étude comparative de nos résultats.

V.1 Introduction

Afin de pouvoir tester la validité de nos systèmes *ProSadm*, *ProSadm-HBA*, *modèle Flou*, *F-HBA*, nous avons décidé de les implémenter (voir chapitre III et IV). Cette implémentation s'est déroulée en deux temps : dans un premier temps, nous avons implémenté les noyaux de base *ProSadm* & le modèle *Flou*. Puis, dans un deuxième temps, nous avons étendu ces modèles par l'ajout de la métaheuristique *HBA* (Homogeneity Based-Algorithm). Notre objectif est d'améliorer les performances des classifieurs des données médicales et de proposer des modèles *ProSadm-HBA*, *F-HBA* comme des solutions envisageables dans le domaine d'optimisation des classifieurs.

Ce chapitre s'articule autour de trois sections : la première présente la méthodologie expérimentale envisagée pour une étude détaillée des résultats obtenus, la deuxième expose une analyse détaillée de nos expérimentations qui sont faites sur les trois bases de données médicales présentées dans le chapitre IV. Et la troisième présente une étude comparative de nos expérimentations. Les principales conclusions de l'étude comparative sont divisées en deux parties. La première partie est consacrée aux différences que présentent les quatre approches implémentées. La seconde partie porte sur la comparaison de nos résultats à ceux disponibles dans la littérature.

V.2 Méthodologie Expérimentale

Notons que L'algorithme *HBA* (Homogeneity-Based Algorithm) divise la base de données utilisée en deux ensembles : des données d'apprentissage et de test tel que décrit dans le tableau IV.1 (Caractéristiques des bases de données). La procédure à suivre pour réaliser les expérimentations dans cette étude est décrites ci-après :

- *Etape 1* : Dans un premier temps, nous calculons la valeur du cout totale CT_1 (total misclassification cost) en appliquant les approches classiques *ProSadm* ou le modèle *Flou* sur la base d'apprentissage T . Et, dans un deuxième temps nous validons le modèle généré en utilisant la base de test.
- *Etape 2* : nous calculons la valeur de CT_2 . Cette valeur est obtenue en appliquant les approches *ProSadm* ou le modèle *Flou* en conjonction avec l'algorithme *HBA* (Homogeneity Based-Algorithm) sur la base d'apprentissage T et en testant le modèle sur la base de teste comme décrit a l'étape 1. ($(\alpha^-, \alpha^+) \in [0, 20]$ & $(\beta^-, \beta^+) \in [0, 2]$) .[Pham 2009]
- *Etape 3* : Comparaison des valeurs CT_1 et CT_2 . (CT_1 et CT_2 est en fonction de C_{FP} ; C_{FN} ; C_{NC}).

V.3 Résultats et Analyse Expérimentales

Dans cette section nous présentons les résultats des différentes considérations proposées pour les valeurs de la fonction objective CT (voir figure V.1) et ceci est pour chaque base de données utilisée.

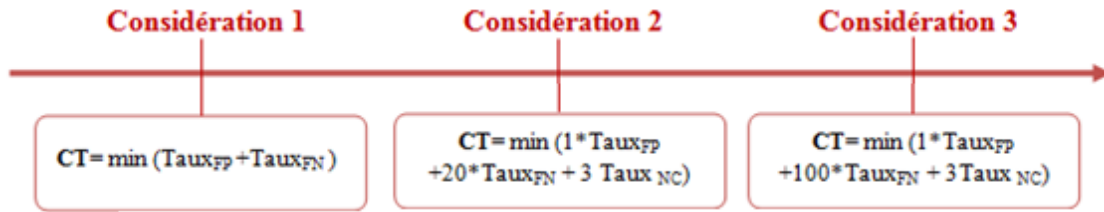


Figure V.1 : Schéma représentatif des fonctions objectives utilisées

V.3.1 Considération 1 : Cette considération permet un cout identique égal à 1 (pour FP & FN). Cependant, le test ne pénalise pas les cas non classifiés (i.e. $C_{NC} = 0$). Ce scénario est le même employé dans la plupart des méthodes de classification actuelles [Pham 2009]. Ainsi, la fonction objective dans ce test est:

$$CT = \min (Taux_{FP} + Taux_{FN}). \quad (5.1)$$

Les performances des modèles proposés : **ProSadm, Flou, ProSadm-HBA & F-HBA** sont présentés dans le tableau **V.1** suivant :

Base de Données	Algorithme	FP (%)	FN (%)	Nc (%)	CT (%)	Taux d'Amélioration
PID	ProSadm	0.52	32.81	66.66	33.33	
	Modèle flou	67.18	0	32.18	67.18	
	ProSadm-HBA	0	32.81	0	32.81	1.56
	F-HBA	0	8.33	55.2	8.33	87.60
T.H	ProSadm	0	37.68	60.86	37.68	
	Modèle-flou	52.17	0	44.92	52.17	
	ProSadm-HBA	0	31.9	0	31.9	15.33
	F-HBA	0	0	77.9	0	100
AP	ProSadm	0	4.8	95.2	4.8	
	Modèle-flou	0	19.04	80.95	19.4	
	ProSadm-HBA	0	0	29.62	0	100
	F-HBA	0	0	76.19	0	100

Tab V.1 Résultats de minimisation pour $CT = \min (Taux_{FP} + Taux_{FN})$

Notons que le taux d'amélioration de x par rapport y est donné par la formule suivante

$$\text{Taux d'amélioration } (x,y) = [(y-x) / y] * 100 \quad (5.2)$$

Par exemple **Tab V.1** montre que le taux d'amélioration du modèle **ProSadm-HBA** par rapport au modèle **ProSadm** égal a 15.33% pour la deuxième base de données médicale (T.H) et égal a 100% pour la troisième base de données AP (Appendicite).

V.3.1.1 Analyse des performances de classification

Les résultats de cette considération sont présentés dans **Tab V.1**. Ce tableau montre les trois taux d'échecs (Taux du **Fp**, Taux du **Fn**, Taux des cas non classifiés

N_c) et leurs cout total CT correspondant ,obtenues en utilisant les approches proposées (*ProSadm, Modèle Floue, ProSadm-HBA, F-HBA*) . Notons que *ProSadm-HBA* et *F-HBA* signifie que l'algorithme *ABH* est utilisé en conjonction avec le modèle *ProSadm* et le modèle *flou* (Neuro-Flou) respectivement. Après plusieurs générations de l'algorithme génétique *AG*, nos approches (*ProSadm-HBA, F-HBA*) appliquées sur les bases de données médicales (**PID** (Diabète), **T.H** (Troubles hépatiques) et **A.P** (Appendicite)), trouvent des valeurs optimales (Globales) du paramètre CT . Les valeurs moyenne du cout totale (CT) obtenue sont 20.57, 15.95 , 0.0 pour les bases de données (*PID, TH, AP*) respectivement. Ces valeurs sont plus améliorées (minimale) par rapport aux valeurs moyennes achevées par les approche classique (*ProSadm* et modèle *Flou* (Neuro-Flou)) par 44.58%, 57.66% ,100% pour les bases de données médicales *PID, TH, AP* respectivement.

Notons que la méthode *ABH* utilise l'algorithme génétique *AG* pour calculer les valeurs des paramètres (α^- , α^+) et (β^- , β^+) utilisé pour étendre ou réduire les ensembles homogènes positive et négatives. Le **Tab V.2** suivant montre les valeurs finales des paramètres (α^- , α^+) et (β^- , β^+) et ceci pour chaque base données médicales .Les valeurs initiales des paramètres (α^- , α^+) et (β^- , β^+) étant généralement initialisées par 0.

	α^-_{Final}	α^+_{Final}	β^-_{Final}	β^+_{Final}
PID (ProSadm-HBA)	17.06	2.13	0.16	0.35
PID (F-HBA)	13.36	1.82	0.6	0.62
T.H(ProSadm-HBA)	14.78	0.04	0.63	0.51
T.H (F-HBA)	19.26	11.9	0.21	0.93
AP (ProSadm-HBA)	18.71	13.71	0.56	0.11
AP (F-HBA)	15.03	8.77	0.66	0.81

Tab V.2 : Valeurs finales des paramètres (α^- , α^+) et (β^- , β^+) pour les bases de données médicales *PID, T.H, A.P* (Considération n°1).

V.3.1.2 Analyse des règles floues

Dans ce qui suit, nous calculons le degré de sollicitation pour chaque règle par rapport aux nombre d'exemples. Sachant que [**Settouti 2011**]:

$$\text{Degré de sollicitation} = \frac{\text{Nombre de personnes qui activent cette règle a plus de 50\%}}{\text{Le nombre total des personnes de chaque classe}} \quad (5.3)$$

Les règles extraits des systèmes implémentés a base flou (Modèle *Flou, F-HBA*) et citées dans les tableaux qui présentent le calcul des degrés de sollicitation des règles activées par les exemples (Classifiés ou non classifiés) sont présentés ci-dessous :

Nous avons utilisé pour chaque variable linguistique deux fonctions d'appartenances :

- Grande: **gr**
- Petite : **pt**
- Pour la base de Diabète:

- . **R1** : Si (Npreg est **pt**)&(Glu est **pt**)&(BP est **pt**)&(Skin est **pt**)&(Insuline est **pt**)&(BMI est **pt**)&(PED est **pt**)&(Age est **pt**) Alors C1.
- . **R33** : Si (Npreg est **pt**)&(Glu est **pt**)&(BP est **gr**)&(Skin est **pt**)&(Insuline est **pt**)&(BMI est **pt**)&(PED est **pt**)&(Age est **pt**) Alors C33.
- . **R65** : Si (Npreg est **pt**)&(Glu est **gr**)&(BP est **pt**)&(Skin est **pt**)&(Insuline est **pt**)&(BMI est **pt**)&(PED est **pt**)&(Age est **pt**) Alors C65.
- . **R66** : Si (Npreg est **pt**)&(Glu est **gr**)&(BP est **pt**)&(Skin est **pt**)&(Insuline est **pt**)&(BMI est **pt**)&(PED est **pt**)&(Age est **gr**) Alors C66.
- . **R97** : Si (Npreg est **pt**)&(Glu est **gr**)&(BP est **gr**)&(Skin est **pt**)&(Insuline est **pt**)&(BMI est **pt**)&(PED est **pt**)&(Age est **pt**) Alors C97.
- . **R101** : Si (Npreg est **pt**)&(Glu est **gr**)&(BP est **gr**)&(Skin est **pt**)&(Insuline est **pt**)&(BMI est **gr**)&(PED est **pt**)&(Age est **pt**) Alors C101.
- . **R109** : Si (Npreg est **pt**)&(Glu est **gr**)&(BP est **gr**)&(Skin est **pt**)&(Insuline est **gr**)&(BMI est **gr**)&(PED est **pt**)&(Age est **pt**) Alors C109.

- Pour la base du Troubles Hépatites

- . **R1** : Si (VGM est **pt**)&(Alkphos est **pt**)&(SGPT est **pt**)&(SGOT est **pt**)&(Gammagt est **pt**)&(Drinks est **pt**) Alors C1.
- . **R33** : Si (VGM est **gr**)&(Alkphos est **pt**)&(SGPT est **pt**)&(SGOT est **pt**)&(Gammagt est **pt**)&(Drinks est **pt**) Alors C33.
- . **R45** : Si (VGM est **gr**)&(Alkphos est **pt**)&(SGPT est **gr**)&(SGOT est **gr**)&(Gammagt est **pt**)&(Drinks est **pt**) Alors C45.
- . **R49** : Si (VGM est **gr**)&(Alkphos est **gr**)&(SGPT est **pt**)&(SGOT est **pt**)&(Gammagt est **pt**)&(Drinks est **pt**) Alors C49.

- Pour la base d'Appendicite

- . **R1** : Si (WBC1 est **pt**)&(MNEP est **pt**)&(MNEA est **pt**)&(MBAP est **pt**)&(MBAA est **pt**)&(HNEP est **pt**)&(HNEA est **pt**) Alors C1.

a. La base de données de diabète PID :

a.1 Analyse des règles générées par le modèle Flou :

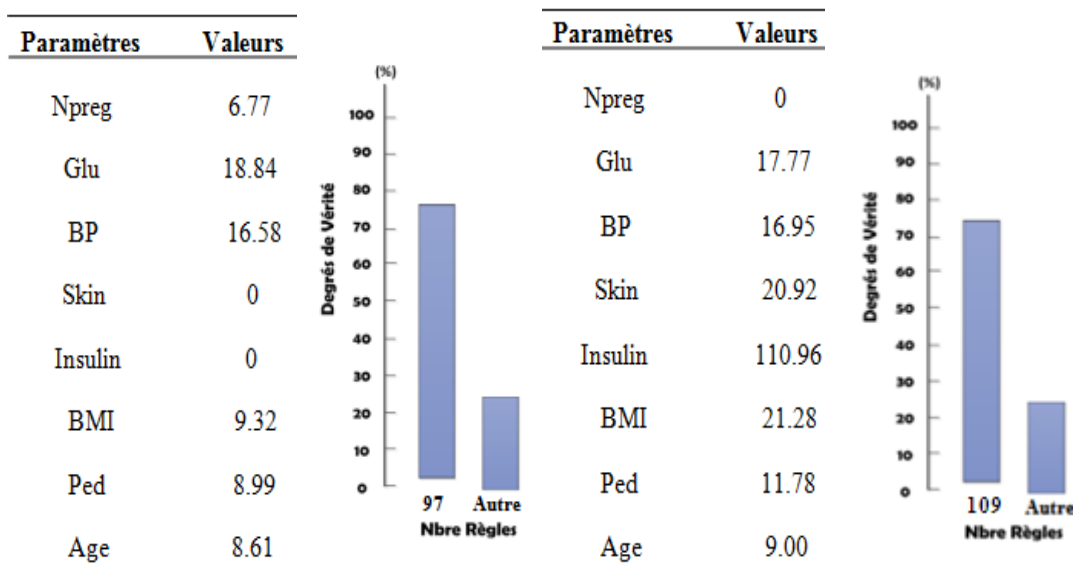
- *Cas non diabétiques prédit comme diabétiques (Fp)*

Le tableau **Tab V.3** suivant présente les degrés de sollicitation des règles activé par des exemples non diabétiques reconnus comme diabétiques.

Règle :	Degré-Sollicitation
[1]	0.008
[33]	0.008
[65]	0.008
[66]	0.008
[97]	0.03
[101]	0.008
[109]	0.008

Tab V.3 : Degré de sollicitation pour les règles des cas **FP**

Deux exemples des cas de **FP** sont présentés au-dessous dans **Tab V.4** et **Tab V.5**. Le premier exemple a activé la règle [97] (cas non diabétique) par un degré de 78.22%. Le deuxième exemple a activé la règle [109] par un degré de 74.75%.



Tab V.4 1^{ère} Exemple des cas de FP

Tab V.5 2^{ème} Exemple des cas de FP

Ces deux exemples sont classés comme non-diabétiques dans la base d'origine, mais ils ont des caractéristiques des cas diabétiques (ex ; Glu >1.4 g/l). C'est pour cette raison que ces exemples sont mal classés par le modèle **Flou**.

- Cas des exemples non classifiés Nc (Unclassifiable)

Le degré de sollicitation pour la règle qui représente les cas non classifiés (correctement reconnus de la base **PID**) est présenté dans le tableau **V.6** suivant :

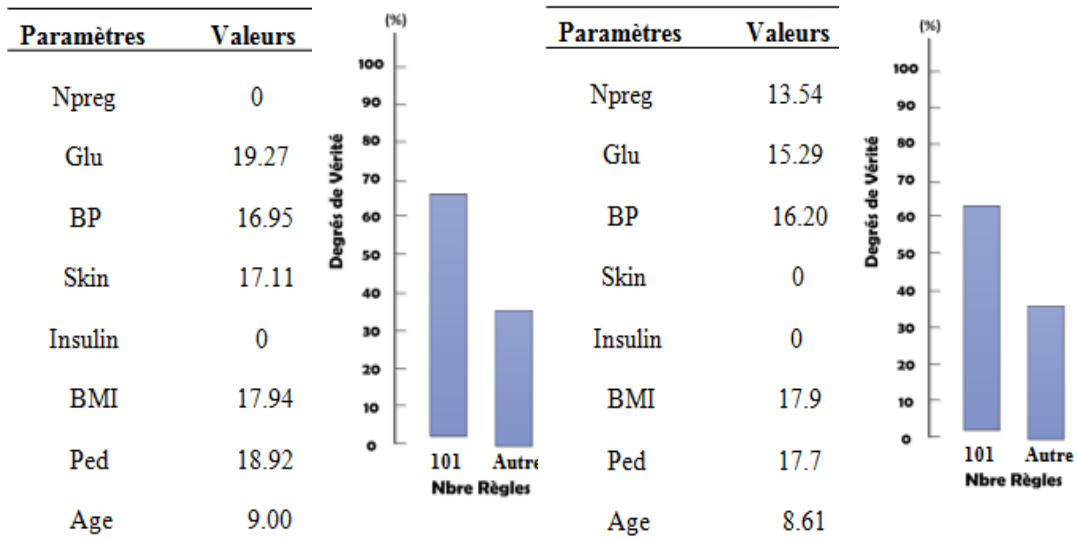
Règle :	Degré-Sollicitation
[101]	0.079

Tab V.6 : Degré de sollicitation pour les règles des cas **Nc**

Le tableau **Tab V.7** suivant présente deux exemples des paramètres de deux patients tirés de la base de teste et qui ont été reconnu correctement par le modèle Flou (Neuro-Flou).

Le premier exemple a activé la règle **[101]** par un degré de 67.92%.

Le deuxième exemple a activé la règle **[101]** par un degré de 63.18%



Tab V.7 1^{ère} Exemple et 2^{ème} Exemple des cas Nc

Ces deux exemples qui sont correctement reconnus par le modèle **Flou** possèdent :

- (BMI > 30k/m² (obèse)) et (Gly ≥ 1.4 g/l). (caractéristiques des cas diabétiques).

a.2 Analyse des règles générées par le modèle F-HBA

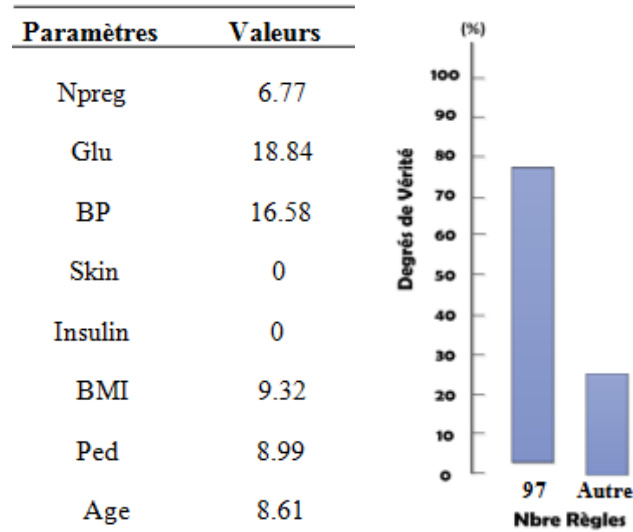
- Cas non diabétiques correctement reconnus (Vn)

Les degrés de sollicitation pour les règles qui représentent les cas non diabétiques correctement reconnus sont illustrés par **Tab V.8**

Règle :	Degré-Sollicitation
[101]	0.008
[97]	0.02
[66]	0.008

Tab V.8 Degré de sollicitation pour les règles des cas Vn

L'exemple ayant les caractéristiques suivantes a activé la règle **[97]** avec un degré de sollicitation =78.22%. Cet exemple qui a été mal reconnue par le système Flou, a été correctement reconnu par le modèle F-HBA.



Tab V.9 Exemple des cas de VN

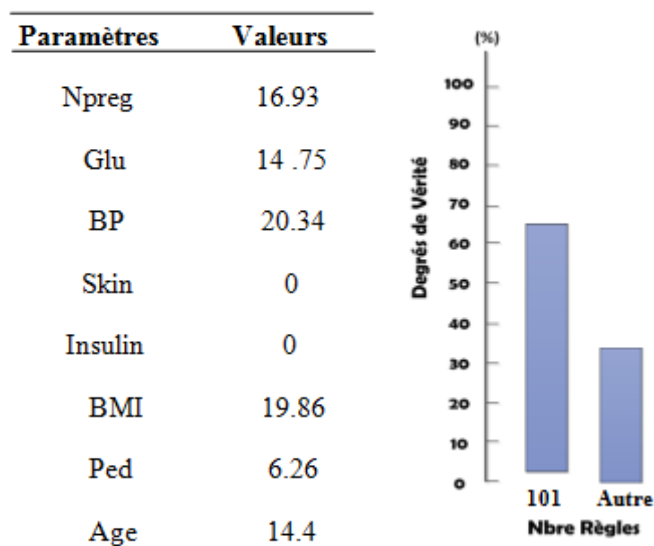
- *Cas diabétiques prédit comme non diabétiques (Fn)*

Le tableau **Tab V.10** cité au-dessus illustre le degré de sollicitation des règles activé par des exemples diabétiques reconnus comme non diabétiques.

Règle :	Degré-Sollicitation
[101]	0.015

Tab V.10 : Degré de sollicitation pour les règles des cas FN

L'exemple ayant les caractéristiques suivantes a activé la règle [101] avec un degré de sollicitation =66.02%. Cet exemple a des caractéristiques des cas non diabétiques (ex Gly <1.4g/l) mais il est classés dans la base originale comme des cas diabétiques. C'est pour cette raison le modèle **F-HBA** a reconnu cet exemple comme des cas non diabétique.



Tab V.11 Exemple des cas de FN

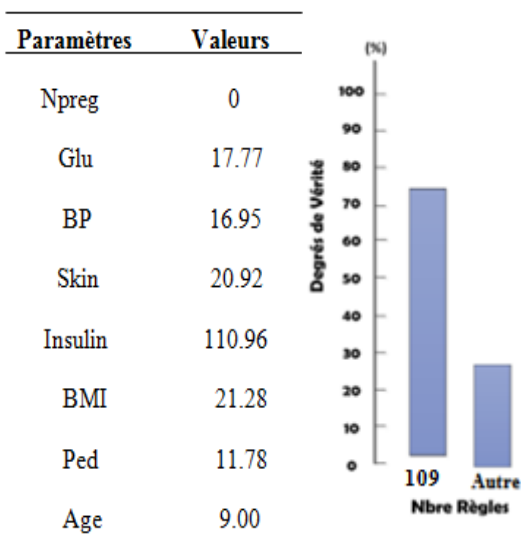
- Cas des exemples non classifiés Nc (Unclassifiable)

Les degrés de sollicitation pour les règles qui représentent les cas non classifiés (correctement reconnus) sont présentés dans le tableau **Tab V.12** suivant :

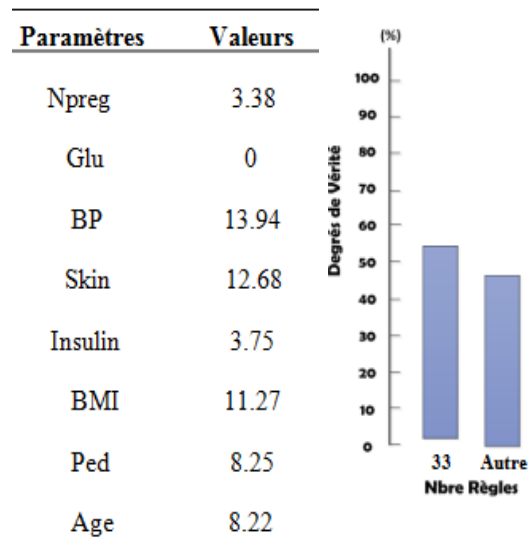
Règle :	Degré-Sollicitation
[101]	0.06
[109]	0.008
[97]	0.008
[1]	0.008
[65]	0.008
[33]	0.008

Tab V.12 : Degré de sollicitation pour les règles des cas Nc

Nous remarquons que la règle [101] possède le degré de sollicitation le plus important. Les deux exemples suivants présentent des cas typiques des exemples non classifiés Nc. Le premier exemple a été correctement classé par l'approche F-HBA et mal classé par le modèle Flou.



Tab V.13 1^{ère} Exemple des cas de Nc (non classifié)



Tab V.14 2^{ème} Exemple des cas de Nc (non classifié)

Le premier exemple (**Tab V.13**) a activé la règle [109] par un degré de 74.45%. Le deuxième exemple (**Tab V.14**) a activé la règle [33] par un degré de 53.30%.

b. La base de troubles hépatiques (T.H) :

b.1 Analyse des règles pour le modèle Flou :

- *Cas des patients atteints d'hépatite correctement reconnus (Vp)*

Les degrés de sollicitation pour les règles qui représentent les cas des patients atteints d'hépatite correctement reconnus par le modèle *Flou* sont illustrés par **Tab V.15** suivant :

Règle :	Degré-Sollicitation
[1]	0.5
[33]	0.5

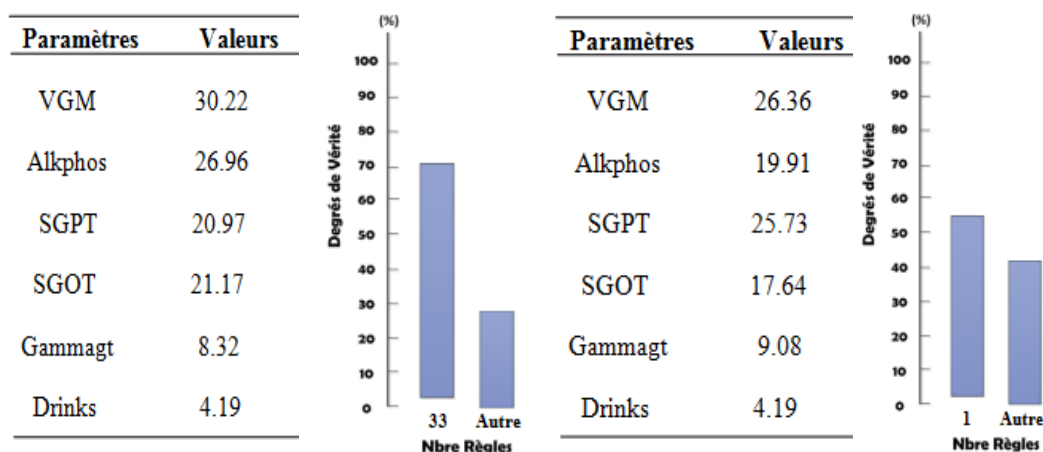
Tab V.15 : Degré de sollicitation pour les règles des cas **VP**

Ci-dessous deux exemples de la base de teste qui représentent des cas du **VP**.

Le premier exemple a activé la règle [33] avec un degré de sollicitation = 71.34%

Le deuxième exemple a activé la règle [1] avec un degré de sollicitation = 55.67 %

Ces deux patients ont été correctement classifiés par le système Flou.



Tab V.16 1^{ère} Exemple et 2^{ème} Exemple des cas de **Vp**

- *Cas des patients sans hépatite prédit comme hépatiques (Fp)*

Le tableau **V.17** suivant présente le degré de sollicitation des règles activé par des exemples sans hépatite reconnus comme hépatique.

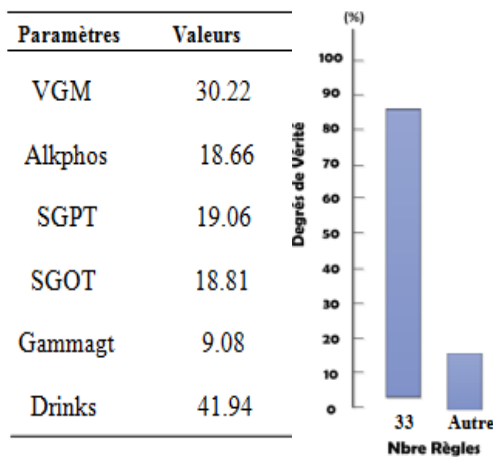
Règle :	Degré-Sollicitation
[33]	0.20
[49]	0.07

Tab V.17 : Degré de sollicitation pour les règles des cas **FP**

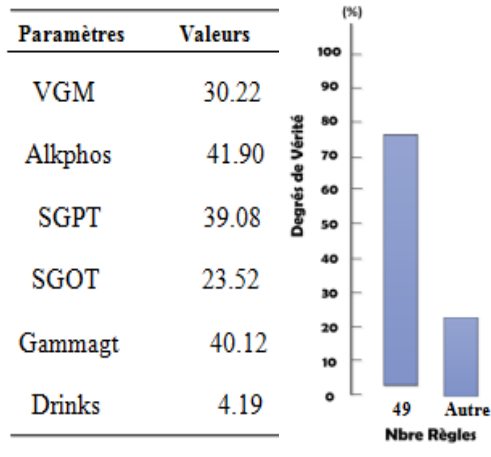
Ci-dessous deux exemples du cas de FP. Le **Tab V.18** présente le premier exemple, il a activé la règle [33] avec un degré de sollicitation = 87.64% . **Tab V.19** présente le deuxième exemple qui a activé la règle [49] avec un degré de sollicitation = 78.39% .

Ces deux exemples sont classés comme sans hépatite dans la base d'origine, mais ils ont des caractéristiques des patients hépatiques (ex : le tableau **Tab V.18** présente une

augmentation du volume globulaire moyen (94 fl) chez une personne consommant régulièrement une bonne quantité de l'alcool, et le tableau **Tab V.19** présente une élévation des enzymes SGPT et Gamma GT) .C'est pour cette raison que ces deux exemples sont mal classés par le modèle **Flou**.



Tab V.18 1^{ère} Exemple des cas de **FP**



Tab V.19 2^{ème} Exemple des cas de **FP**

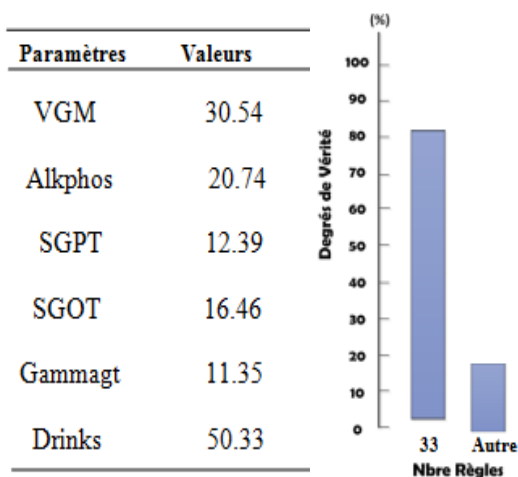
- *Cas des exemples non classifiés Nc (Unclassifiable)*

Les degrés de sollicitation pour les règles qui représentent les cas non classifiés (correctement reconnus) sont présentés dans le tableau **Tab V.20** suivant :

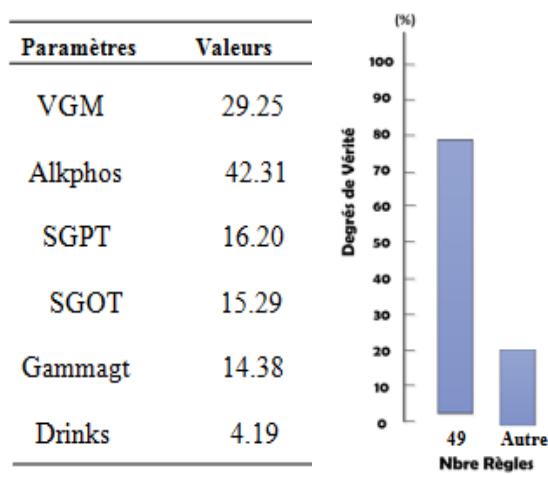
Règle :	Degré-Sollicitation
[33]	0.16
[49]	0.06

Tab V.20 : Degré de sollicitation pour les règles des cas **Nc**

Les tableaux **Tab V.21**, **V.22** représentent des exemples des cas non classifiés par le système Flou. Le premier exemple a activé la règle **[33]** avec un degré de sollicitation = 81.57% et le deuxième exemple a activé la règle **[49]** avec un degré de sollicitation = 79.77%



Tab V.21 1^{ère} Exemple des cas Nc



Tab V.22 2^{ème} Exemple des cas Nc

Ces deux exemples présents des valeurs élevées de l'enzyme VGM (Volume Globulaire moyen). De plus le Tableau V.21 présente un cas d'une personne alcoolique.

b.2 Analyse des règles floues générées par le modèle F-HBA

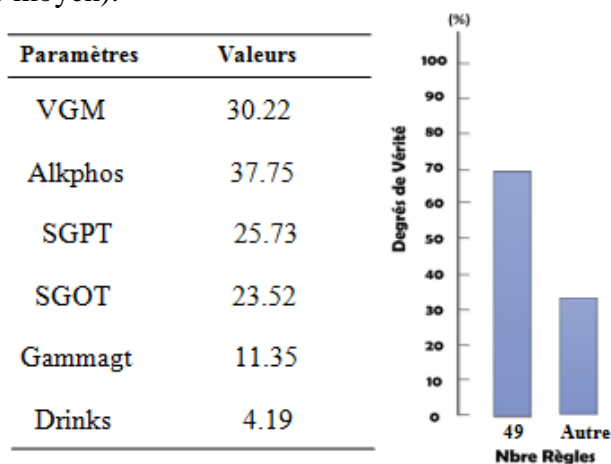
- Cas des patients atteints d'hépatite correctement reconnus (Vp)

Les degrés de sollicitation pour les règles qui représentent les cas des patients atteints d'hépatite correctement reconnus par le modèle *Flou* sont illustrés par Tab V.23 suivant :

Règle :	Degré-Sollicitation
[1]	0.03
[33]	0.12
[49]	0.03

Tab V.23 : Degré de sollicitation pour les règles des cas VP

Le tableau V.24 montre un exemple des cas du vrai positif. Cet exemple a activé la règle [49] avec un degré de sollicitation = 68.61 %. Ce patient hépatite a été correctement classifié par l'approche *F_HBA*. Il présente une valeur élevée de l'enzyme VGM (Volume Globulaire moyen).



Tab V.24 Exemple des cas de VP

- Cas des patients sans hépatite correctement reconnus (Vn)

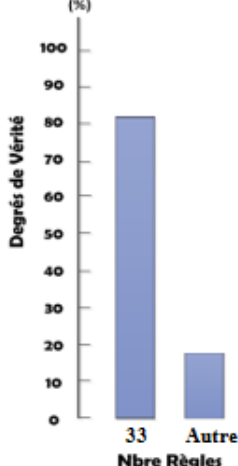
Le degré de sollicitation pour la règle qui représente le cas d'un patients sans hépatite correctement reconnus est illustré par **Tab V.25**.

Règle :	Degré-Sollicitation
[33]	0.13

Tab V.25 : Degré de sollicitation pour les règles des cas VN

Le tableau **V.26** suivant montre un exemple des cas du Vrai Négatif qui a activé la règle [33] avec un degré de sollicitation = 81.68 % .Cet exemple a été correctement classé sans hépatite avec des valeurs des enzymes (SGPT, SGOT, Gamma GT, Alkphos) normales. D'où cette personne est proche de la classe des patients n'ayant pas une hépatite.

Paramètres	Valeurs
VGM	30.22
Alkphos	24.06
SGPT	20.01
SGOT	21.17
Gammagt	19.68
Drinks	16.77



Nbre Rèales	Degrés de Vérité (%)
33	81.68
Autre	18.32

Tab V.26 Exemple des cas de VN

- Cas des exemples non classifiés Nc (Unclassifiable)

Les degrés de sollicitation pour les règles qui représentent les cas non classifiés (correctement reconnus) sont présentés dans le tableau **Tab V.27** suivant :

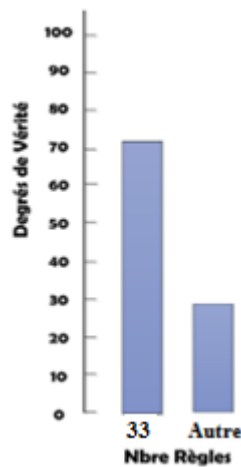
Règle :	Degré -Sollicitation
[33]	0.25
[45]	0.01
[49]	0.10

Tab V.27 : Degré de sollicitation pour les règles des cas Nc (non classifiés)

Le **Tab V.28** représente un exemple des cas non classifiés qui a activé la règle [33] avec un degré de sollicitation = 70.38 % .Le deuxième exemple cité dans le tableau **Tab V.29** a activé la règle [45] avec un degré de sollicitation = 66.08 % .Ces deux patients

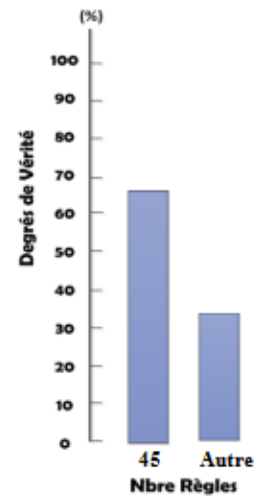
alcooliques présentes des valeurs normales du VGM et Alkphos . Le tableau V.28 présente aussi de bonnes valeurs pour les enzymes SGPT, SGOT et Gamma GT.

Paramètres	Valeurs
VGM	28.61
Alkphos	19.91
SGPT	30.50
SGOT	25.87
Gammagt	10.60
Drinks	33.55



Tab V.28 1^{ère} Exemple des cas de Nc (non classifié)

Paramètres	Valeurs
VGM	30.22
Alkphos	17.83
SGPT	146.81
SGOT	96.44
Gammagt	91.61
Drinks	33.55



Tab V.29 2^{ème} Exemple des cas de Nc (non classifié)

c. La base d'appendicite (AP)

c.1 Analyse des règles pour le modèle Flou

- Cas des exemples mal classifiés (Fn).

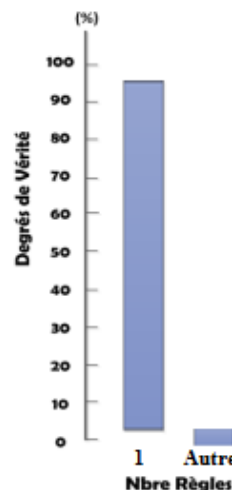
Le tableau V.30 suivant présente le degré de sollicitation des règles activé par des exemples malades reconnus comme non malades (Appendicite).

Règle :	Degré-Sollicitation
[1]	0.2

Tab V.30 : Degré de sollicitation pour les règles des cas FN

La règle 1 est un **prototype** .L'exemple cité dans le tableau V.31 a activé cette règle par un degré de sollicitation ≈100%.

Paramètres	Valeurs
1	45.46
2	48.83
3	22.97
4	0
5	0
6	57.25
7	27.00



Tab V.31 Exemple des cas de FN

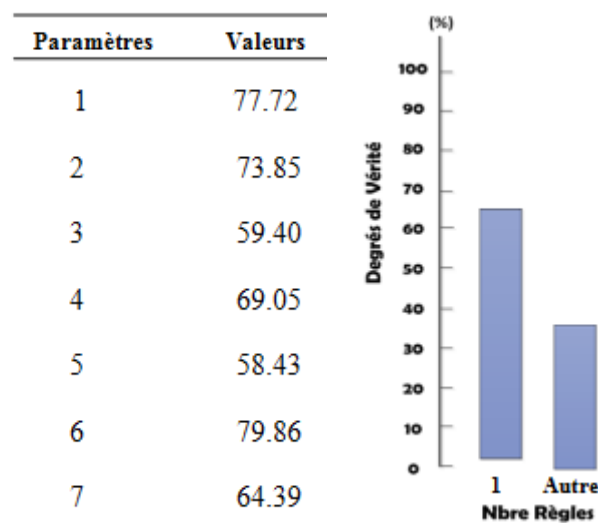
- Cas des exemples non classifiés (Unclassifiable Nc)

Le degré de sollicitation pour la règle qui représente les cas non classifiés (correctement reconnus) est présenté dans le tableau **Tab V.32** suivant :

Règle :	Degré-Sollicitation
[1]	0.2

Tab V.32 : Degré de sollicitation pour les règles des cas Nc

Le **Tab V.33** montre un exemple des cas non classifiés correctement reconnus. Cet exemple a activé la règle [1] avec un degré de sollicitation = 63.67 %



Tab V.33 Exemple des cas de Nc

c.2 Analyse des règles floues générées par le modèle F-HBA

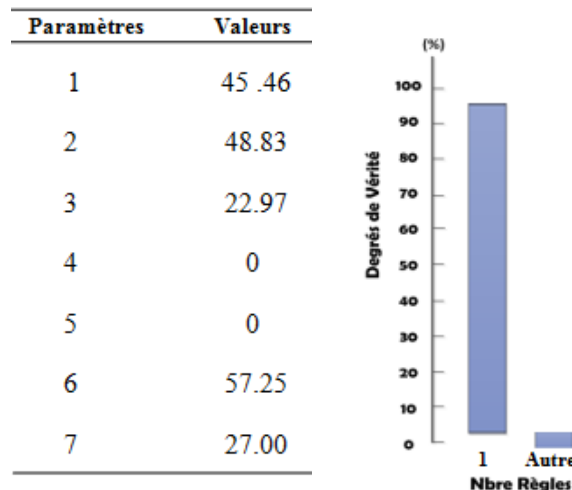
- Cas des exemples non classifiés (Unclassifiable Nc)

Le degré de sollicitation pour la règle qui représente les cas non classifiés (correctement reconnus) est présenté dans le tableau **Tab V.34** suivant :

Règle :	Degré-Sollicitation
[1]	0.12

Tab V.34 : Degré de sollicitation pour les règles des cas Nc

La règle 1 est un prototype. L'exemple suivant a activé cette règle par un degré de sollicitation ≈ 100 %



Tab V.35 Exemple des cas de Nc

V.3.2 Considération 2

Nous supposons maintenant dans ce test que le cout du Faux Négatif est 20 fois plus que la considération 1 et le cout du Faux Positif et du Nc (objets non classifiés) serait égal a 1 et 3 respectivement. Ce scénario est plus réaliste [Pham 2009]. Il est a espéré que plus le coût de pénalité du faux-négatif augmente, moins le cas du faux-négatif sera trouvée [Pham 2009]. Ainsi La fonction objective s'écrit :

$$CT = \min (Taux_{FP} + 20 Taux_{FN} + 3 Taux_{NC}). \quad (5.4)$$

Les performances de nos approches proposés : **ProSadm, Flou, ProSadm-HBA & F-HBA** sont présentés dans le tableau V.36 suivant :

Base de données	Algorithme	FP (%)	FN (%)	Nc (%)	CT (%)	Taux Amélioration
PID	ProSadm	0.52	32.81	66.66	856.72	
	Modèle flou	67.18	0	32.18	163.72	
	ProSadm-HBA	0	26.56	14.58	574.99	32.88
	F-HBA	2.08	8.33	54.68	332.72	pas d'amélioration
T.H	ProSadm	1.16	38.37	59.30	946.46	
	Modèle-flou	52.17	0	44.92	186.93	
	ProSadm-HBA	0	38.4	1.16	771.48	18.48
	F-HBA	0	1.44	71.01	241.83	pas d'amélioration
AP	ProSadm	0	14.28	85.71	542.73	
	Modèle-flou	0	19.04	80.95	623.65	
	ProSadm-HBA	0	0	25.9	77.7	85.68
	F-HBA	0	0	76.19	228.57	63.34

Tab V.36 Résultats de minimisation pour $CT = \min (Taux_{FP} + 20Taux_{FN} + 3 Taux_{NC})$

Le tableau **Tab V.36** montre que le taux d'amélioration du modèle *ProSadm-HBA* par rapport au modèle *ProSadm* égal a 32.88% pour la première base de données médicale (*PID*) et égal a 85.68% pour la troisième base de données *AP* (Appendicite).

V.3.2.1 Analyse des performances de classification

Les résultats de cette considération sont présentés dans **Tab V.36**. Après plusieurs générations de l'algorithme génétique **AG**, nos approches proposées trouvent des valeurs minimales pour le paramètre **CT**. Les valeurs moyennes du cout total (**CT**) obtenue sont 453.85, 506.65, 153.13 pour les bases de données (*PID*, *TH*, *AP*) respectivement. Ces valeurs sont plus améliorées (minimale) par rapport aux valeurs moyennes achevées par les approches classiques (*ProSadm* et modèle *Flou* (Neuro-Flou)) par 16.44% , 9.24% ,74.51% pour les bases de données médicales *PID*, *TH*, *AP* respectivement . Le Tableau **Tab V.36** montre qu'il n'y a pas d'amélioration du modèle *F-HBA* par rapport au modèle *Flou* (Neuro-Flou) dans les bases de données **PID** et **T.H** (Troubles hépatiques) .La raison d'avoir un **CT** (pour *F-HBA*) inferieur au cout total (**CT**) obtenus par le modèle Flou est que le modèle classique a trouvé des valeurs de **CT** optimale (proche de l'optimal). Aussi la base d'apprentissage utilisée par les méthodes classiques et les méthodes avec *ABH* ne sont pas équivalentes (voir chapitre **III**).

La méthode *ABH* utilise l'algorithme génétique *AG* pour calculer les valeurs des paramètres (α^- , α^+) et (β^- , β^+) utilisées pour étendre ou réduire les ensembles homogènes positive et négatives. Le **Tab V.37** suivant montre les valeurs finales des paramètres (α^- , α^+) et (β^- , β^+) et ceci pour chaque base données médicales utilisées (*PID*,*T.H*,*AP*).Les valeurs initiales des paramètres (α^- , α^+) et (β^- , β^+) étant généralement initialisées a 0.

	α^-_{Final}	α^+_{Final}	β^-_{Final}	β^+_{Final}
PID (<i>ProSadm-HBA</i>)	8.34	16.35	0.86	0.68
PID (<i>F-HBA</i>)	12.32	11.71	0.95	0.63
T.H (<i>ProSadm-HBA</i>)	10.87	12.05	0.40	0.81
T.H (<i>F-HBA</i>)	18.65	4.96	0.71	0.96
AP (<i>ProSadm-HBA</i>)	4.36	18.02	0.43	0.84
AP (<i>F-HBA</i>)	15.03	8.77	0.66	0.81

Tab V.37 : Valeurs finales des paramètres (α^- , α^+) et (β^- , β^+) pour les bases de données médicales *PID*, *T.H*, *A.P* (Considération n° 2)

V.3.2.2 Analyse des règles floues

a. La base de données de diabète **PID** :

a.1 Analyse des règles générées par le modèle Flou :

Même résultats obtenues en considération 1 (Voir page 117)

a.2 Analyse des règles générées par le modèle **F-HBA**

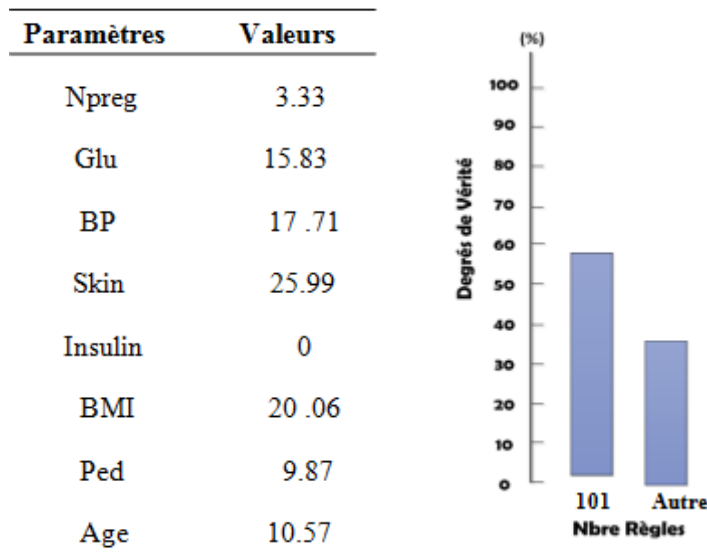
- Cas de diabétiques correctement reconnus (*Vp*)

Le degré de sollicitation pour la règle qui représente les cas diabétiques correctement reconnus est illustré par **Tab V.38**

Règle :	Degré-Sollicitation
[101]	0.015

Tab V.38 Exemple des cas de VP

L'exemple ayant les caractéristiques suivantes a activé la règle [101] avec un degré de sollicitation =57.59%



Tab V.39 Exemple des cas de VP

Cet exemple a été correctement classé comme diabétique avec l'BMI dépasse 30 kg/m² (ce que signifie que cette femme est obèse [Ammar 2008]) et Glu >1.4 g/l.

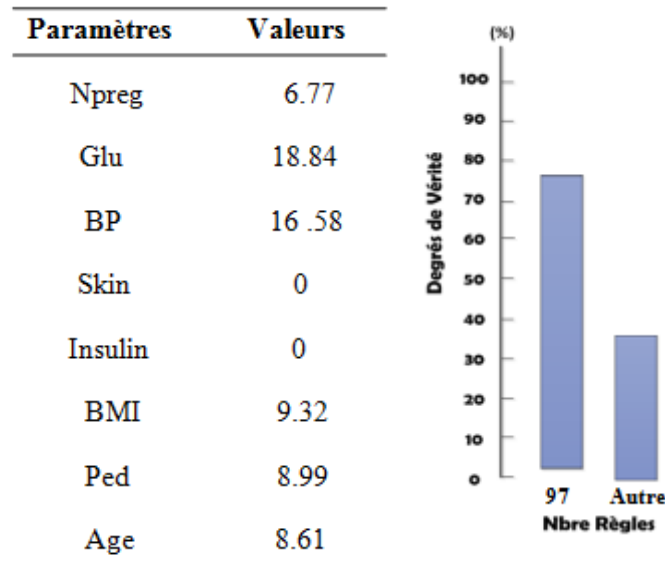
- Cas de non diabétiques correctement reconnus (Vn)

Les degrés de sollicitation pour les règles qui représentent les cas non diabétiques correctement reconnus sont illustrés par **Tab V.40**

Règle :	Degré-Sollicitation
[101]	0.007
[97]	0.031
[66]	0.007

Tab V.40 : Degré de sollicitation pour les règles des cas VN

Le tableau **Tab V.41** suivant présente un exemple des cas du vrai négatif .Cet exemple a activé la règle [97] avec un degré de sollicitation = 78.22%



Tab V.41 Exemple des cas de VN

Cet exemple a été mal classé par le modèle *Flou* et correctement classé par le modèle *F-HBA* dans les considérations 1 et 2. Il s'agit d'une hyperglycémie (i.e. le taux de glycémie à jeun est supérieur à 1,26 g/l).

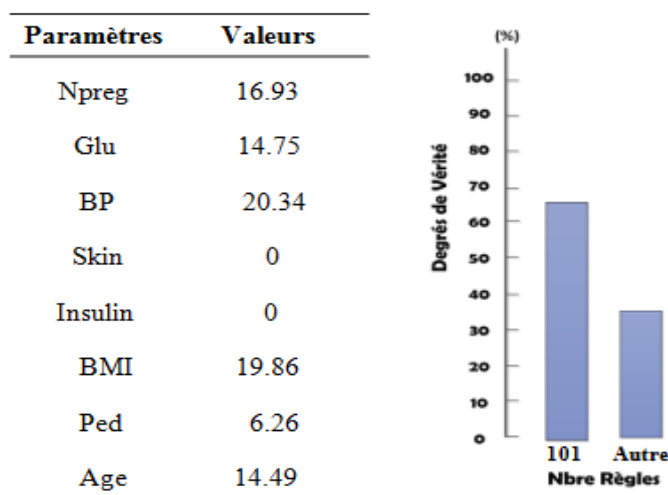
- Cas de non diabétiques prédit comme diabétiques (Fn)

Le tableau **Tab V.42** cité au-dessus illustre le degré de sollicitation des règles activé par des exemples diabétiques reconnus comme non diabétiques

Règle :	Degré-Sollicitation
[101]	0.007

Tab V.42 Degré de sollicitation pour les règles des cas Fn

L'exemple ayant les caractéristiques suivantes a activé la règle [101] avec un degré de sollicitation =66.02%



Tab V.43 Exemple des cas de FN

Cet exemple a des caractéristiques des cas non diabétiques (ex Gly <1.4g/l) mais il est classés dans la base originale comme des cas diabétiques. C'est pour cette raison le modèle *F-HBA* a reconnu cet exemple comme des cas non diabétiques.

- *Cas des exemples non classifiés Nc (Unclassifiable)*

Les degrés de sollicitation pour les règles qui représentent les cas non classifiés (correctement reconnus) sont présentés dans le tableau **Tab V.44** suivant :

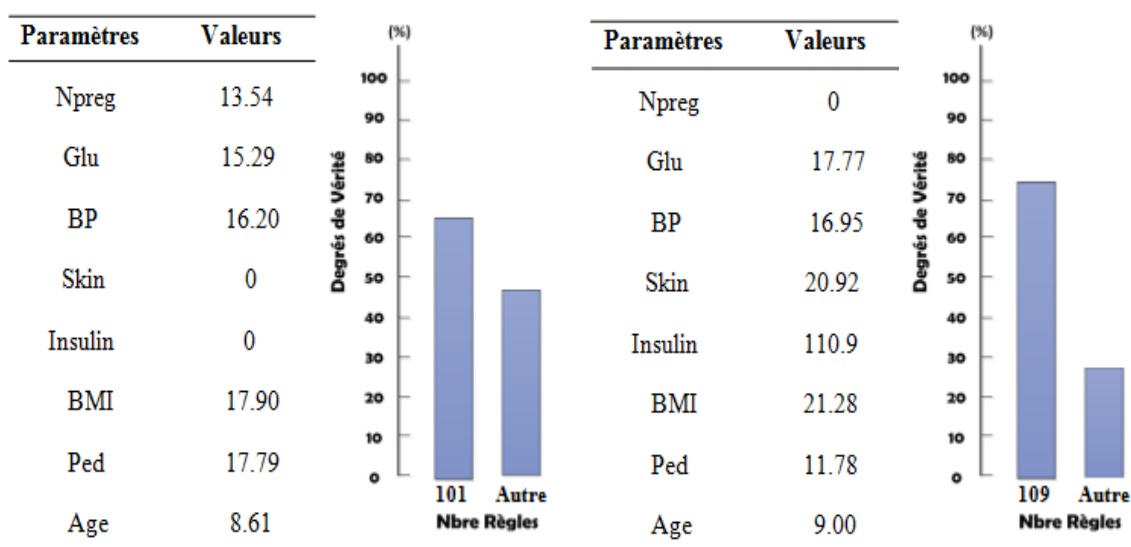
Règle :	Degré-Sollicitation
[109]	0.007
[101]	0.05
[65]	0.007
[33]	0.007
[1]	0.007

Tab V.44 : Degré de sollicitation pour les règles des cas NC

Les deux tableaux **Tab V.45** & **Tab V.46** suivants sont des exemples des paramètres de deux patient tirés de la base de teste et qui ont été reconnu correctement par le modèle *F-HBA*

Le premier exemple a activé la règle **[101]** par un degré de 63.15%

Le deuxième exemple a activé la règle **[109]** par un degré de 74.45%



Tab V.45 1^{ère} Exemple des cas de Nc (non classifié)

Tab V.46 2^{ème} Exemple des cas de Nc (non classifié)

Le premier exemple a été correctement classé par l'approche **F-HBA** dans la 2^{ème} considération et par l'approche *Flou* dans la 1^{ère} considération.

Le deuxième exemple a été correctement classé par l'approche *F-HBA* dans la considération 1 et 2 et mal classé par l'approche *Flou* dans la 1^{ère} considération.

b. La base de données T.H (Troubles Hépatiques) :

b.1 Analyse des règles générées par le modèle Flou :

Même résultats obtenus en considération 1 (Voir page 121).

b.2 Analyse des règles générées par le modèle F-HBA

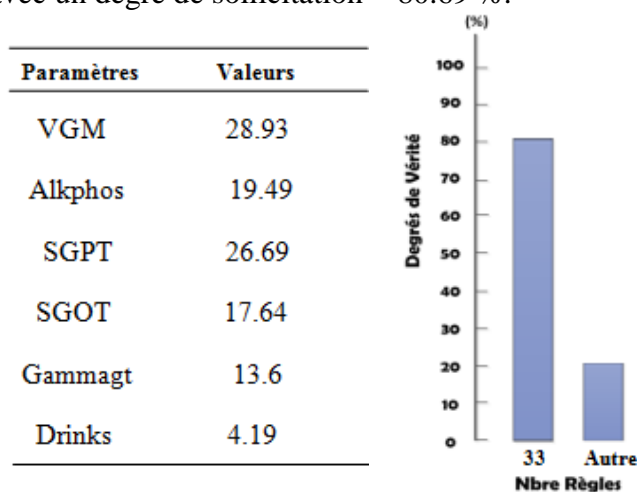
- Cas des patients atteints d'hépatite correctement reconnus (*Vp*)

Les degrés de sollicitation pour les règles qui représentent les cas des patients atteints d'hépatite correctement reconnus par le modèle *Flou* sont illustrés par **Tab V.47** suivant :

Règle :	Degrés-Sollicitation
[1]	0.03
[33]	0.15
[49]	0.03

Tab V.47 : Degré de sollicitation pour les règles des cas **VP**

Le tableau **V.48** cité au-dessus illustre un exemple des cas de Vrai Positif. Cet exemple a activé la règle [33] avec un degré de sollicitation = 80.69 %.



Tab V.48 Exemple des cas de **VP**

Ce patient hépatite (présente une valeur élevée de l'enzyme VGM a été correctement classifié par le système *F-HBA*).

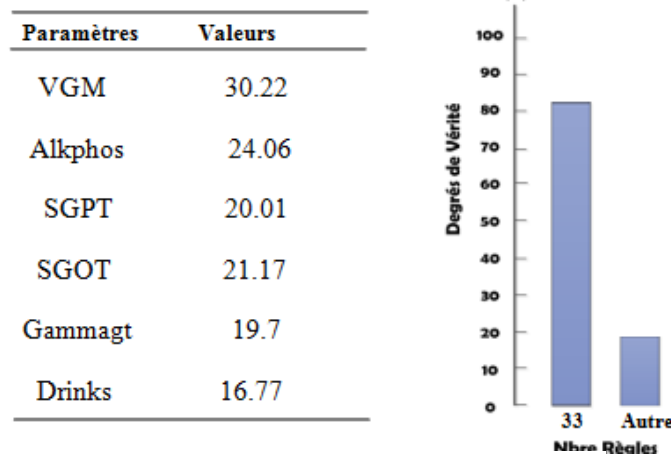
- Cas des patients sans hépatite correctement reconnus (*Vn*)

Le degré de sollicitation pour la règle qui représente le cas des patients sans hépatite correctement reconnus est illustré par le **Tab V.49** :

Règle :	Degrés-Sollicitation
[33]	0.19

Tab V.49 : Degré de sollicitation pour les règles des cas **VN**

Le tableau **V.50** présente un exemple des cas du VN .Cet exemple a activé la règle [33] avec un degré de sollicitation = 81.68 % .Cet personne présente des caractéristiques proches de la classe sans hépatite (ex : bonnes valeurs du VGM, Alkphos, SGPT, SGOT et Gamma GT) .



Tab V.50 Exemple des cas de VN

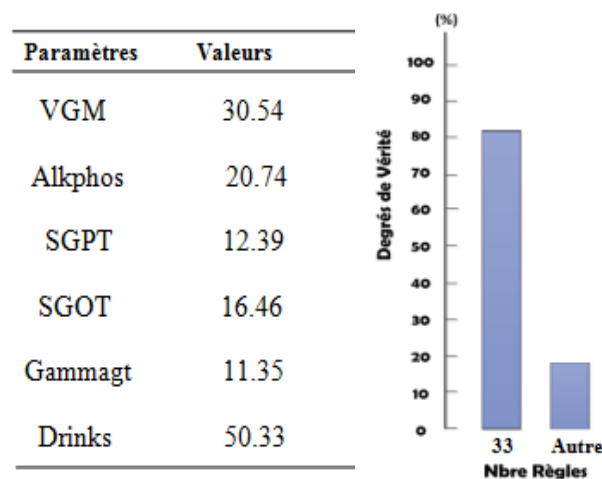
- *Cas des patients hépatiques prédit comme non hépatiques (Fn)*

Le tableau **V.51** cité au-dessus illustre le degré de sollicitation de règle activé par des exemples hépatique reconnus comme des cas sans hépatite :

Règle :	Degré-Sollicitation
[33]	0.03

Tab V.51 : Degré de sollicitation pour les règles des cas FN

L'exemple ayant les caractéristiques suivantes **Tab V.52** a activé la règle [33] avec un degré de sollicitation = 81.57 % .Cet personne alcoolique présente des caractéristiques des personnes sans hépatite (valeurs normales du VGM, Alkphos, SGPT, SGOT, Gamma GT) mais il est classé dans la base d'origine comme hépatite.



Tab V.52 Exemple des cas de FN

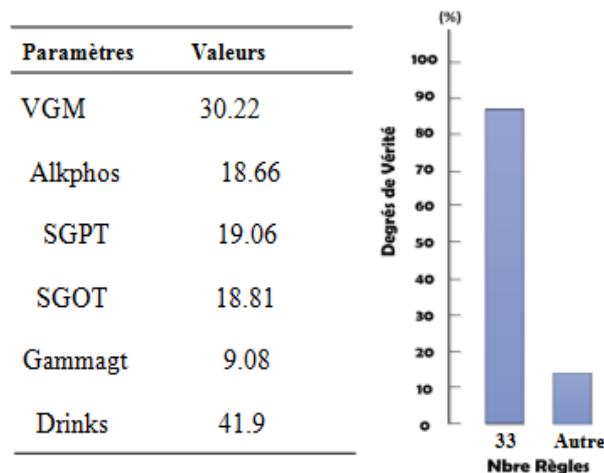
- *Cas des exemples non classifiés Nc (Unclassifiable)*

Les degrés de sollicitation pour les règles qui représentent les cas non classifiés (correctement reconnus) sont présentés dans le tableau **Tab V.53** suivant :

Règle :	Degré-Sollicitation
[33]	0.18
[49]	0.11

Tab V.53: Degré de sollicitation pour les règles des cas Nc

L'exemple ayant les caractéristiques suivantes **Tab V.54** a activé la règle [33] avec un degré de sollicitation = 87.64 %. Cette personne alcoolique a été classifié correctement par le système *F-HBA* comme sans Hépatite, car il présente des caractéristiques des patients hépatites (ex : bonne valeurs du VGM et des enzymes (gamma GT, SGPT, SGOT, Alkphos).



Tab V.54 Exemple des cas de Nc

c. La base de données AP (Appendicite) :

c.1 Analyse des règles générées par le modèle Flou :

Même résultats obtenus en considération 1 (Voir page 126).

c.2 Analyse des règles générées par le modèle F-HBA :

Même résultats obtenus en considération 1 (Voir page 127)

V.3.3 Considération 3

Dans cette considération, nous supposons aussi que le cout du faux négatif sera plus pénalisé .Ainsi, la fonction objectif s'écrit :

$$CT = \min (Taux_{FP} + 100 Taux_{FN} + 3 Taux_{NC}). \quad (5.5)$$

Les performances de nos approches proposés : **ProSadm, Flou, ProSadm-HBA & F-HBA** sont présentés dans le tableau **V.55** suivant :

Base de Données	Algorithme	FP (%)	FN (%)	Nc (%)	CT (%)	Taux Amélioration
PID	ProSadm	0	32.81	67.18	3482.56	
	Modèle flou	67.18	0	32.81	165.61	
	ProSadm-HBA	0	26.56	13.02	2695.06	22.61
	F-HBA	3.64	13.54	42.7	1485.74	pas d'amélioration
T.H	ProSadm	1.16	37.20	60.5	3902.66	
	Modèle-flou	52.17	0	44.92	186.93	
	ProSadm-HBA	0	38.37	0	3837	1.68
	F-HBA	0	1.44	71.01	357.03	pas d'amélioration
AP	ProSadm	0	9.52	90.5	1223.5	
	Modèle-flou	0	19.04	80.95	2146.85	
	ProSadm-HBA	0	0	25.9	77.7	93.64
	F-HBA	0	0	76.19	228.57	89.35

Tab V.55 Résultats de minimisation pour $CT = \min (Taux_{FP} + 100 Taux_{FN} + 3Taux_{Nc})$

Le tableau **Tab V.55** montre que le taux d'amélioration du modèle *ProSadm-HBA* par rapport au modèle *ProSadm* égal a 22.61% pour la première base de données médicale (PID) et égal a 93.64% pour la troisième base de données AP (Appendicite).

V.3.3.1 Analyse des performances de classification

Les résultats de cette considération sont présentés dans **Tab V.55**. Après plusieurs générations de l'algorithme génétique *AG*, nos approches proposées trouvent des valeurs minimales pour le paramètre *CT*. Les valeurs moyennes du cout total (*CT*) obtenue sont 2090.4, 2097.01, 153.13 pour les bases de données (*PID*, *TH*, *AP*) respectivement. Ces valeurs sont plus améliorées (minimale) par rapport aux valeurs moyennes achevées par les approches classique (*ProSadm* et modèle *Flou* (Neuro-Flou)) par 11.3%, 0.84% ,91.49% pour les bases de données médicales *PID*, *TH*, *AP* respectivement. Le Tableau **Tab V.55** montre qu'il n'y a pas d'amélioration du modèle *F-HBA* par rapport au modèle *Flou* (Neuro-Flou) dans les bases de données *PID* et *T.H* (Troubles hépatiques) .La raison d'avoir un *CT* (pour *F-HBA*) inferieur au cout total (*CT*) obtenue par le modèle Flou est que le modèle classique a trouvé des valeurs de *CT* optimale (proche de l'optimal). Aussi la base d'apprentissage utilisée par les méthodes classiques et les méthodes avec *ABH* ne sont pas équivalente (voir chapitre III).

La méthode *ABH* utilise l'algorithme génétique *AG* pour calculer les valeurs des paramètres (α^- , α^+) et (β^- , β^+) utilisé pour étendre ou réduire les ensembles homogènes positive et négatives. Le **Tab V.56** suivant montre les valeurs finales des paramètres (α^- , α^+) et (β^- , β^+) et ceci pour chaque base données médicales utilisées (PID,T.H,AP).Les valeurs initiales des paramètres (α^- , α^+) et (β^- , β^+) étant généralement initialisées a 0 .

	α^- Final	α^+ Final	β^- Final	β^+ Final
PID (ProSadm-HBA)	12.32	11.71	0.95	0.79
PID (F-HBA)	11.57	6.38	0.55	0.63
T.H(ProSadm-HBA)	6.89	15.89	0.53	0.03
T.H (F-HBA)	18.65	4.96	0.71	0.96
AP (ProSadm-HBA)	4.36	18.36	0.43	0.84
AP (F-HBA)	15.03	8.77	0.66	0.81

Tab V.56 : Valeurs finales des paramètres (α^- , α^+) et (β^- , β^+) pour les bases de données médicales PID, T.H, A.P (Considération n° 3)

V.3.3.2 Analyse des règles floues

a. La base de données de diabète PID :

a.1 Analyse des règles générées par le modèle Flou :

Même résultats obtenus en considération 1 et 2 (Voir page 117).

a.2 Analyse des règles générées par le modèle F-HBA :

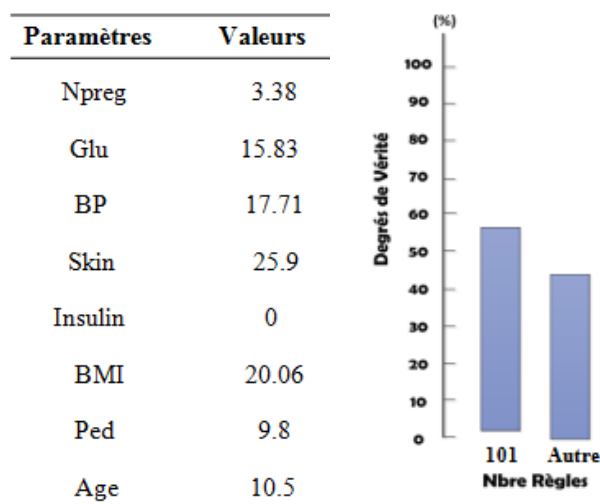
- Cas de diabétiques correctement reconnus (Vp)

Le degré de sollicitation pour les règles qui représentent les cas diabétiques correctement reconnus sont illustrés par **Tab V.57**

Règle :	Degré-Sollicitation
[101]	0.015

Tab V.57 : Degré de sollicitation pour les règles des cas **VP**

L'exemple ayant les caractéristiques suivantes (**Tab V.58**) a activé la règle [101] avec un degré de sollicitation =57.59%. Cet exemple présente des caractéristiques d'un(e)



Tab V.58 Exemple des cas de **VP**

patient(e) diabétique (ayant une glycémie supérieur à 1.4g/l et BMI >30 kg/m²) et correctement classifié par l'approche *F-HBA* .

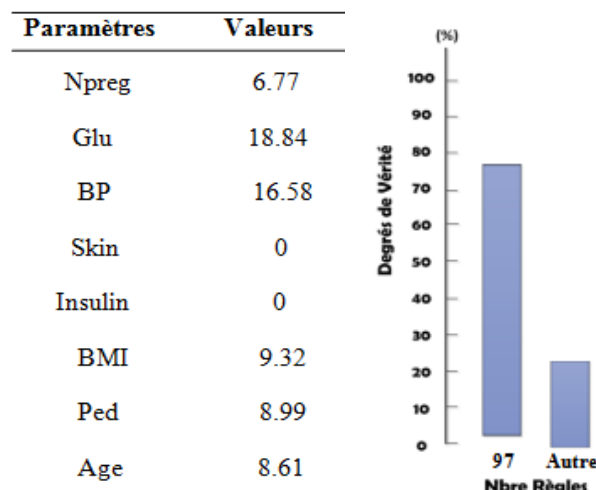
- *Cas de non diabétiques correctement reconnus (Vn)*

Les degrés de sollicitation pour les règles qui représentent les cas non diabétiques correctement reconnus sont illustrés par **Tab V.59**

Règle :	Degré-Sollicitation
[66]	0.007
[97]	0.02
[101]	0.007

Tab V.59 : Degré de sollicitation pour les règles des cas VN

Le tableau **Tab V.60** suivant présente un exemple des cas du VN qui a activé la règle [97] avec un degré de sollicitation = 78.22 %. Cet exemple qui a été mal par le système Flou et correctement reconnue par le modèle *F-HBA* dans la 1^{ere}, 2^{eme} et 3^{eme} considération.



Tab V.60 Exemple des cas de VN

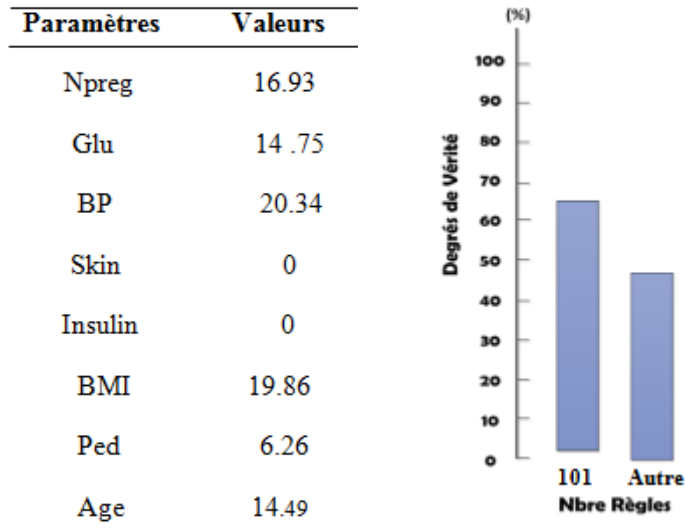
- *Cas de diabétiques prédit comme non diabétiques (Fn)*

Le tableau **Tab V.61** cité au-dessus illustre le degré de sollicitation des règles activé par des exemples diabétiques reconnus comme non diabétiques

Règle :	Degré-Sollicitation
[101]	0.01

Tab V.61 : Degré de sollicitation pour les règles des cas FN

L'exemple ayant les caractéristiques suivantes (**Tab V.62**) a activé la règle [101] avec un degré de sollicitation = 66.02 %. Cet exemple a des caractéristiques des cas non diabétiques (ex Gly <1.4g/l) mais il est classés dans la base originale comme des cas diabétiques. C'est pour cette raison le modèle *F-HBA* a reconnu cet exemple comme des cas non diabétiques.



Tab V.62 Exemple des cas de FN

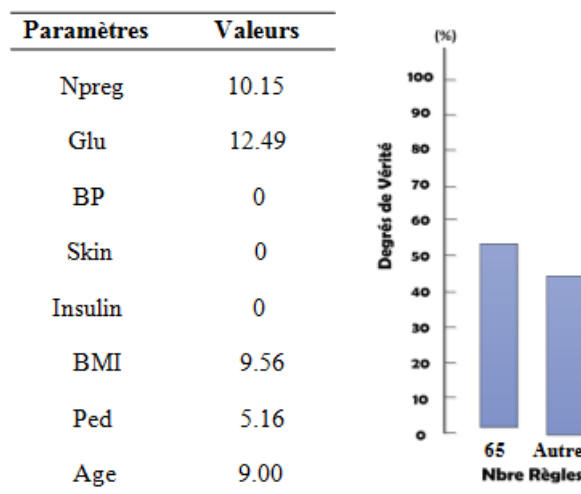
- *Cas non diabétiques prédit comme diabétiques (Fp)*

Le tableau **Tab V.63** cité au-dessus illustre le degré de sollicitation des règles activé par des exemples non diabétiques reconnus comme diabétiques

Règle :	Degré-Sollicitation
[65]	0.015

Tab V.63 : Degré de sollicitation pour les règles des cas **FP**

L'exemple ayant les caractéristiques suivantes (**Tab V.64**) a activé la règle [65] avec un degré de sollicitation = 53.81 %



Tab V.64 Exemple des cas de **FP**

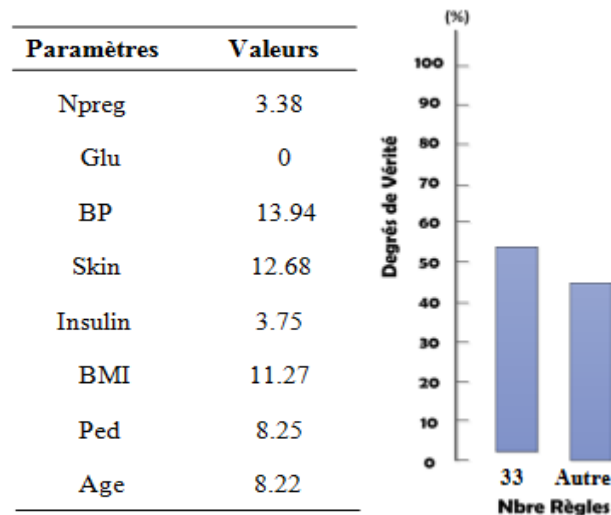
- *Cas des exemples non classifiés Nc (Unclassifiable)*

Les degrés de sollicitation pour les règles qui représentent les cas non classifiés (correctement reconnus) sont présentés dans le tableau **Tab V.65** suivant :

Règle :	Degré-Sollicitation
[1]	0.007
[33]	0.007
[97]	0.007
[101]	0.031
[109]	0.007

Tab V.65 : Degré de sollicitation pour les règles des cas Nc

Le tableau suivant (**Tab V.66**) présente un cas des exemples non classifiés (Nc) .Cet exemple a activé la règle [33] avec un degré de sollicitation égale a 53.29%



Tab V.66 Exemple des cas de Nc

Cet exemple a été correctement classifié par l'approche *F-HBA* dans la 1^{re} et 3^{eme} considération. Il présente une hypoglycémie (un taux bas de sucre dans le sang (moins de 0,5 g/l)).

b. La base de données T.H (Troubles Hépatiques) :

b.1 Analyse des règles générées par le modèle Flou :

Même résultats obtenus en considération 1.2 (Voir page 121).

b.2 Analyse des règles générées par le modèle F-HBA :

Même résultats obtenus en considération 2 (Voir page 133).

c. La base de données AP (Appendicite) :

c.1 Analyse des règles générées par le modèle Flou :

Même résultats obtenus en considération 1.2 (Voir page 126).

c.2 Analyse des règles générées par le modèle F-HBA

Même résultats obtenus en considération 1.2 (Voir page 127).

V.4 Etude Comparative

V.4.1 Avec Métaheuristique VS Sans Métaheuristique

Le tableau **Tab V.67** présente les performances de classification des différentes approches implémentées. Ce tableau montre que les deux systèmes *ProSadm-HBA* et *F-HBA* ont donné des résultats meilleurs ou égal par rapport aux résultats obtenus par le système *ProSadm* et le modèle *Flou* pour les trois bases de données médicales utilisées. A titre d'exemple, le système *ProSadm-HBA* a obtenu un Taux de classification = 100% pour la base d'Appendicite avec un moyen d'amélioration qui égale à 4.8% (100-95.2) par rapport au résultat obtenu par l'approche classique *ProSadm*.

Base De données	ProSadm	Flou	ProSadm-HBA		F-HBA	
	Taux Classification	Taux Classification	Taux Classification	Moyen Amélioration	Taux Classification	Moyen Amélioration
PID (Diabètes)	66.67 %	32.82 %	67.19 %	0.52 %	91.7 %	58.88 %
Troubles Hépatiques	62.32 %	47.83 %	68.1 %	5.78 %	100 %	52.17 %
Appendicite	95.2 %	80.6 %	100 %	4.8 %	100 %	19.4 %

Tab V.67 Résultats expérimentaux des bases de données utilisées

Ces résultats montrent clairement que les approches proposées, qui utilise le classifieur (*ProSadm* ou *Flou*) comme méthode d'apprentissage ou d'extraction de connaissance et la méta-heuristique *ABH* pour le problème d'optimisation (en termes de taux d'erreur et du cout de pénalité associé) sont des approches valides.

V.4.2 Sensibilité et Spécificité

En analyse médicales, il y a trois mesures de performance qui sont importantes : Taux de classification, Sensibilité (Se), Spécificité (Sp). La **figure V.2** et le tableau **Tab V.68** indiquent les meilleurs résultats obtenus par les modèles *ProSadm-HBA* et *F-HBA* en termes des trois critères d'évaluations (Taux de classification, Sensibilité et Spécificité).

Il est intéressant de remarquer que dans la plupart des bases de données et les fonctions objectives **CT** (cout total (**figure V.1**)) utilisées, les valeurs de spécificité obtenues sont meilleurs que les valeurs du Sensibilité. Par conséquent, il est suggéré que les approches développées (**ProSadm** et **Flou**) utilisées en conjonction avec l'algorithme **ABH** sont favorisées pour prédire les patients non malades.

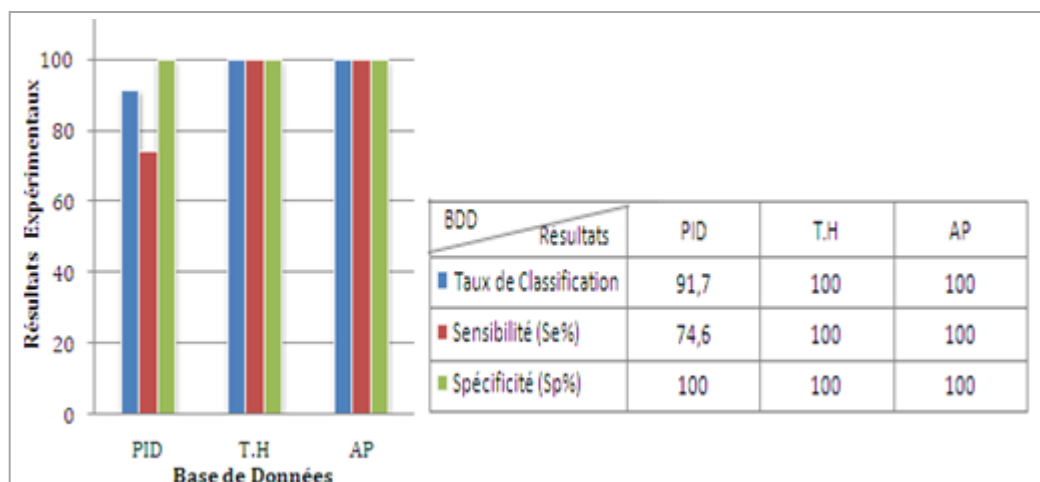


Figure V.2 Meilleur Performances de classification des méthodes proposées

Tab V.68 Évaluation des approches proposées sur les bases des données du Diabète, Troubles Hépatiques et Appendicite.

D'après le **Tableau V.68** et la **figure V.2**, les meilleures performances du Taux de classification obtenues sont égale à 91.7%, 100%, 100% en utilisant les approches *ProSadm-HBA* et *F-HBA* avec une Sensibilité égale à 74,6%, 100%, 100% et une Spécificité égale à 100%, 100%, 100% pour les bases de données médicales PID(Diabète), Trouble Hépatique et Appendicite respectivement.

D'après les tableaux **Tab V.67, 68, 69, 70,71**, nous constatons que les résultats des performances obtenues pour les bases de données médicales utilisées sont satisfaisants et encourageants en les comparant avec les résultats cités dans la littérature. Nous remarquons aussi que nos approches proposées en particulier les modèles hybrides avec la métaheuristique *HBA* (**H**omogeneity **B**ased-**A**lgorithm) ont données les meilleur résultats.

V.4.3 Comparaison avec les résultats de la littérature

Nous avons comparé nos résultats à ceux disponibles dans la littérature sur les mêmes jeux de données et présentées dans le chapitre **IV**. Nous avons regroupé ces différents résultats dans des tableaux. D'après les tableaux **Tab V.69, 70,71**, nous remarquons que les modèles implémentées *ProSadm-HBA* et *F-HBA* ont donné de bon performances qui sont acceptables en les comparant avec celles obtenues par les travaux cités dans les tableaux cités au-dessous.

Référence	Méthode	Taux-Classification	Moyen-d'Amélioration
[Au 2001]	Approche Flou	77.6 %	
[Blachnik 2006]	NefClass	73.83 %	
	PTDL	70.43 %	
	C4.5	74.48 %	
[Pham 2009]	SVM-HBA	94.8 %	
	ANN-HBA	94.8 %	
	DT-HBA	91.7 %	
[Settouti 2012]	FCM-ANFIS	83.85 %	
Nos-approches	F_HBA	91.7 %	9.01 %

Tab V.69 Performance de classification pour la base de données de diabète PID

Référence	Méthode	Taux-Classification	Moyen-d'Amélioration
[Pham 2009]	SVM-HBA	100 %	
	ANN-HBA	98.6 %	
	DT-HBA	95.7 %	
[Rajeswari 2010]	Réseau Bayésien Naïf	96.52 %	
[Ramana 2011]	Réseau Bayésien Naïf	56.52 %	
	C4.5	68.69 %	
	Rétro-Propagation	71.59 %	
	K plus proche voisin	62.89 %	
	SVM	58.26 %	
Nos-approches	F-HBA	100 %	21.24

Tab V.70 Performance de classification pour la base de données TH(Troubles Hépatiques)

Référence	Méthode	Taux-Classification	Moyen-d'Amélioration
[Weiss 89]	(Cart,K+proche voisin,R.N.A..)	89.6 %	
[Nakashima 2003]	Classification Floue	84.0 %	
[Blachnik 2006]	NefClass	87.73 %	
	PTDL	85.77 %	
	C4.5	85.82 %	
[Pham 2009]	SVM-HBA	100 %	
	ANN-HBA	96.4 %	
	DT-HBA	89.3 %	
Nos-approches	ProSadm-HBA	100 %	10.17 %
	F-HBA	100 %	10.17 %

Tab V.71 Performance de classification pour la base de données d'appendicite

Pour le même jeu de données, nous remarquons que l'approche *F-HBA* obtient des résultats de performance (Taux de classification) équivalente ou meilleur a celle de la plupart des autre méthodes .De plus, cette approche augmente la transparence du système développé.

V.5 Conclusion

Dans ce chapitre, nous avons présenté d'abord la méthodologie expérimentale qui nous a permet une évaluation comparatif de nos résultats a base des quatre approches développées (*ProSadm*, *Flou*, *ProSadm-HBA*, *F-HBA*). Par la suite, nous avons exposé une étude détaillée des résultats obtenues en termes de performance de classification (Cout total C_T) et des règles générées. En fin, nous avons présenté une étude comparative de nos expérimentations.

D'après les meilleures performances obtenues des trois considérations proposées pour les valeurs de la fonction objective C_T , nous constatons que les approches hybrides développées (*F-HBA* et *ProSadm-HBA*) sont des outils puissants et présentent une deuxième opinion pour l'expert humain. Cependant le modèle *F-HBA* augmente l'interprétabilité et la transparence du système d'aide au diagnostic médicale.

Conclusion Générale
&
Perspectives

A l'origine de ce mémoire de magister un intérêt pour les problèmes des classifieurs ; liés a la performance des modèles d'estimation et la transparence des résultats. La réduction des couts associés à ces performances représente un objet de plusieurs travaux de recherche. Nous nous intéressons d'une part au problème du compromis *Sur-apprentissage / Sur-généralisation* qui constitue un problème complexe et ouvert s'inscrivant dans le cadre des problèmes d'amélioration des performances des classifieurs (*PPC*) et d'autre part de l'interprétabilité des systèmes de prédiction a fin d'augmenter la lisibilité du classifieur. En effet, il s'agit d'un coté, de minimiser les couts d'erreurs de classification en terme du faux-positif (F_P), faux-négatif (F_N) et des cas non-classifier (N_C), duquel l'augmentation de l'un implique la diminution des autres, ce qui constitue un problème crucial pour une approche de classification. D'un autre coté, de rechercher des solutions interprétables qui sont exprimées par des termes linguistiques compris par un expert humain.

Il nous est alors nécessaire de proposer des modèles qui résolvent et optimisent la performance des classifieurs. Notre travail a donné d'ores et déjà lieu à des contributions tant au niveau théorique que pratique.

Sur le plan théorique :

Notre première contribution était de proposer un modèle de prédiction dénommé *ProSadm-HBA* pour la résolution et l'optimisation des classifieurs du problème soulevé : *Sur-apprentissage / Sur-généralisation*. Au cours de cela, nous avons cherché à exploiter notre ancien système **ProSadm** (Programmation d'un **S**ystème d'**A**ide au **D**iagnostic **M**édical). Ce travail a été étendu par l'enfoncement des performances du classifieur à travers l'introduction d'une approche d'optimisation : la méta-heuristique **ABH** (Algorithme a **B**ase d'**H**omogénéité).

Notre deuxième contribution dans ce mémoire de magister était de développer un nouveau modèle d'optimisation basé sur la théorie de la logique *floue* (*Neuro-Flou*), pour pallier le problème des boitier-noires caractérisant les réseaux connexionnistes, ce qui a permet d'augmenter l'*interprétabilité* du système de la classification. Ce modèle est appelé : *F-HBA* (*F*uzzy *H*omogeneity *B*ased *A*lgorithm).

Le modèle *F-HBA* est constitué d'un noyau *ANFIS* basé sur un algorithme d'apprentissage neuro-flou, et d'un module d'optimisation mettant en œuvre une méthode métaheuristique *HBA* (**H**omogeneity **B**ased-**A**lgorithm).

La procédure d'exécution des modèles *ProSadm-HBA* et *F-HBA* est la suivante :

Dans un premier temps, nous appliquons des approches classiques (sans méta-heuristique) : *ProSadm* ou *le modèle Flou* (*Neuro-Flou*) sur une base d'apprentissage **T**. Nous calculons par la suite, la valeur du cout totale CT_1 (total misclassification cost).

Dans un deuxième temps nous validons les modèles générés en utilisant une base de test. Nous calculons la valeur de CT_2 . Cette valeur est obtenue en appliquant les approches *ProSadm* ou le modèle *Flou* en conjonction avec l'algorithme *HBA* (**H**omogeneity **B**ased-**A**lgorithm) sur la base d'apprentissage T et en testant le modèle sur la base de teste comme décrit à l'étape 1.

A la fin, nous comparons les valeurs CT_1 et CT_2 . (Notons que : CT_1 et CT_2 est en fonction de C_{FP} ; C_{FN} ; C_{NC} : les couts de pénalités).

$$C_T = \min (C_{FP} \times \text{Taux}_{FP} + C_{FN} \times \text{Taux}_{FN} + C_{NC} \times \text{Taux}_{NC})$$

En résumé, cette étude expérimentale nous a montré que les modèles *ProSadm-HBA* et *F-HBA* répond bien à deux objectifs essentiels que nous nous sommes fixés :

- La modification d'un modèle de classification existant telle que la formule de C_T soit minimisée ou significativement réduite (avec la prise en considération des cas non-classifiés et les différents couts de penalties associés aux erreurs de classification F_P , F_N , N_C).
- L'augmentation de la lisibilité du modèle de classification.

Sur le plan pratique, nous avons appliqué nos approches *ProSadm-HBA* et *F-HBA* à un cas d'étude réel, celui du diagnostic médical. Ce qui nous a permis d'attester que ces deux approches s'avèrent des solutions appropriées surtout dans le cas d'optimisation des classifieurs. Ces solutions présentent l'avantage d'être performants à des contextes d'usage à la fois complexes et à couts inégales (taux du faux positif, négatif et non-classifié).

Notre apport pratique se traduit aussi par la mise au point d'un prototype qui implémente nos modèles *ProSadm-HBA* et *F-HBA*. Nous nous basons pour cela sur notre ancien système *ProSadm* (**P**rogrammation d'un **S**ystème d'**A**ide au **D**iagnostic **M**édical), le modèle *ANFIS* (**A**daptative **N**euro-**F**uzzy **I**nference **S**ystem) et la méta-heuristique *ABH*.

Perspectives

A l'issue de ce travail de magister, nous proposons les perspectives suivantes :

- Adopter les modèles *ProSadm-HBA* et *F-HBA* pour la résolution des problèmes de classification en ligne (online).

- Etendre les modèles *ProSadm-HBA* et *F-HBA* de manière à pouvoir supporter d'autres paramètres comme :
 - Optimisation par colonie de fourmis.
 - K-moyennes flou (fuzzy k-means).
 -
- Lancer des méthodes de réduction de la dimension telle que les approches de sélection dites *wrapper* et *filter* sur l'ensemble des descripteurs des bases de données utilisées.
- Généralisation des modèles *ProSadm-HBA* et *F-HBA* pour la famille des problèmes *NP*.
- Appliquer des méthodes de validation par exemple ceux qui sont citées dans le mémoire, sur l'ensemble des solutions (modèles) obtenues.
- Enrichir les modèles développés par d'autres techniques de classification supervisée telle que les arbres de décision.
- Optimisation de la connaissance en un nombre minimum de règles tout en gardant la performance des résultats.

Annexes

Annexe A

A.1 *Machine de Turing :*

Une machine de Turing déterministe possède une tête de lecture/écriture sur un ruban divisé en cases numérotées. Dans chaque case est écrit un symbole choisi parmi $\{0, 1, b\}$.

La taille du ruban est potentiellement infinie. La machine possède également un état, évoluant parmi un ensemble Γ . La machine fonctionne alors de la manière suivante: elle commence à lire la case 1, et son état initial est un état $e \in \Gamma$ particulier. A chaque fois qu'elle lit un symbole, et en fonction de son état, elle :

- Inscrit un symbole dans la case courante (en effaçant le symbole qui y était inscrit) ;
- Se déplace d'une case vers la droite ou vers la gauche ;
- Change éventuellement d'état.

La machine effectue son calcul (selon le fonctionnement précédent) et s'arrête lorsqu'elle est dans un des deux états f_{oui} et f_{non} , appelés états terminaux. Le temps d'exécution est mesuré par le nombre d'étapes effectuées. L'instance I est acceptée par la machine si le calcul se termine dans l'état f_{oui} , il est rejeté sinon.

Le langage reconnu par la machine est défini comme étant l'ensemble des instances acceptées par cette machine. Nous dirons alors qu'une machine résout le problème de décision $D = (D+, D-)$ si le langage reconnu par la machine est $D+$. L'aspect de *complexité en temps*, représente le nombre d'étapes que va mettre la machine pour déterminer si $I \in D+$.

A.2 *Machine déterministe :*

Les machines déterministes sont telles que chaque action possible est unique, c'est-à-dire que l'action à effectuer est dictée de façon unique par l'état courant de celle-ci. S'il peut y avoir plusieurs choix possibles d'actions à effectuer, la machine est dite non déterministe.

A.3 *Machine non déterministe :*

Pour une machine non déterministe, la fonction régissant le fonctionnement de cette machine est multivaluée. La machine a plusieurs possibilités à chaque itération, et une instance est acceptée s'il existe au moins un ensemble de choix conduisant à l'état d'acceptation f_{oui} .

A.4 Réduction de problème (Réduction de Karp) :

Soient $D_1 = (D_1^+, D_1^-)$ et $D_2 = (D_2^+, D_2^-)$ deux problèmes de décision. D_1 se réduit à D_2 sous une réduction de Karp (ce que nous noterons $D_1 \leq_K D_2$) : S'il existe une fonction f telle que :

- $\forall I \in D_1, f(I) \in D_2$;
- $I \in D_1^+ \Leftrightarrow f(I) \in D_2^+$;
- f est calculable en temps polynomial.

A.5 Problème SAT :

Définissons le problème SAT : une instance est constituée de variables booléennes et d'un ensemble de clauses (disjonctions de littéraux) sur ces variables. Il s'agit alors de déterminer s'il existe une valeur de vérité des variables booléennes satisfaisant toutes les clauses données en entrée.

A.6 Problème NP- Difficile (NP-Hard) :

Les problèmes NP-hard sont des problèmes d'optimisation pour le quel, leur problèmes de décision associé sont NP-complet. La majorité des problèmes d'optimisation du monde réel sont NP-difficile .Ils nécessitent un temps exponentiel pour les résoudre. Les métaheuristiques constituent une alternative importante pour résoudre cette classe de problème. [Talbi 09] Exemple de problèmes NP-Hard : problème de voyageur de commerce.

Annexe B

B.1 Arbre de décision :

Les arbres de décision sont un outil très populaire de classification. Leur principe repose sur la construction d'un arbre de taille limitée. La racine constitue le point de départ de l'arbre et représente l'ensemble des données d'apprentissage. Puis ces données sont segmentées en plusieurs sous-groupes, en fonction d'une variable discriminante (un des attributs). Par exemple dans la *figure B.1* suite, la première variable discriminante est la température corporelle. Elle divise la population en deux sous-groupes : les personnes dont la température est supérieure à 37°C et les autres. Le processus est ensuite réitéré au deuxième niveau de l'arbre, où les sous-populations sont segmentées à leur tour en fonction d'une autre valeur discriminante (dans l'exemple, c'est la toux). Le processus est ainsi réitéré jusqu'à ce que l'on arrive à une feuille de l'arbre, correspondant à la classe des exemples qui ont suivi cette décomposition.

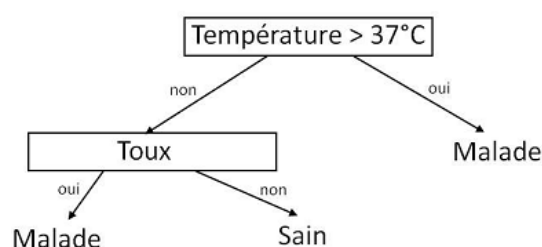


Figure B.1 Schéma d'un arbre de décision

Une fois l'arbre construit à partir des données et d'apprentissage, on peut prédire un nouveau cas en le faisant descendre le long de l'arbre, jusqu'à une feuille. Comme la feuille correspond à une classe, l'exemple sera prédit comme faisant partie de cette classe. Dans l'exemple, on peut en déduire qu'une personne qui a une température < 37°C et qui a de la toux est prédite comme malade, tandis qu'une personne qui a une température < 37°C mais pas de toux est considérée comme saine.

L'intérêt des arbres de décision est en premier lieu leur lisibilité. En effet, il est très simple de comprendre les décisions de l'arbre une fois celui-ci créé, ce qui n'est pas toujours le cas pour les autres classificateurs comme les réseaux de neurones. D'autre part, l'algorithme de création des arbres de décision fait automatiquement la sélection d'attributs jugés pertinents, et ce, même sur des volumes de données importants.

Lors de la création de l'arbre, la première question qui vient à l'esprit est le choix de la variable de segmentation sur un sommet. Autres paramètres importants est la taille limitée que l'on veut fixer à l'arbre. En effet, si la taille est trop grande, on créera une feuille pour chaque exemple et on aura alors un sur-apprentissage des données. Si la taille est trop petite, en revanche, on aura un sous-apprentissage des données.

La figure **B.2** suivante présente un exemple de post-élagage en utilisant les approches : *REP* et *PEP* citées dans [Quinlan 1987]

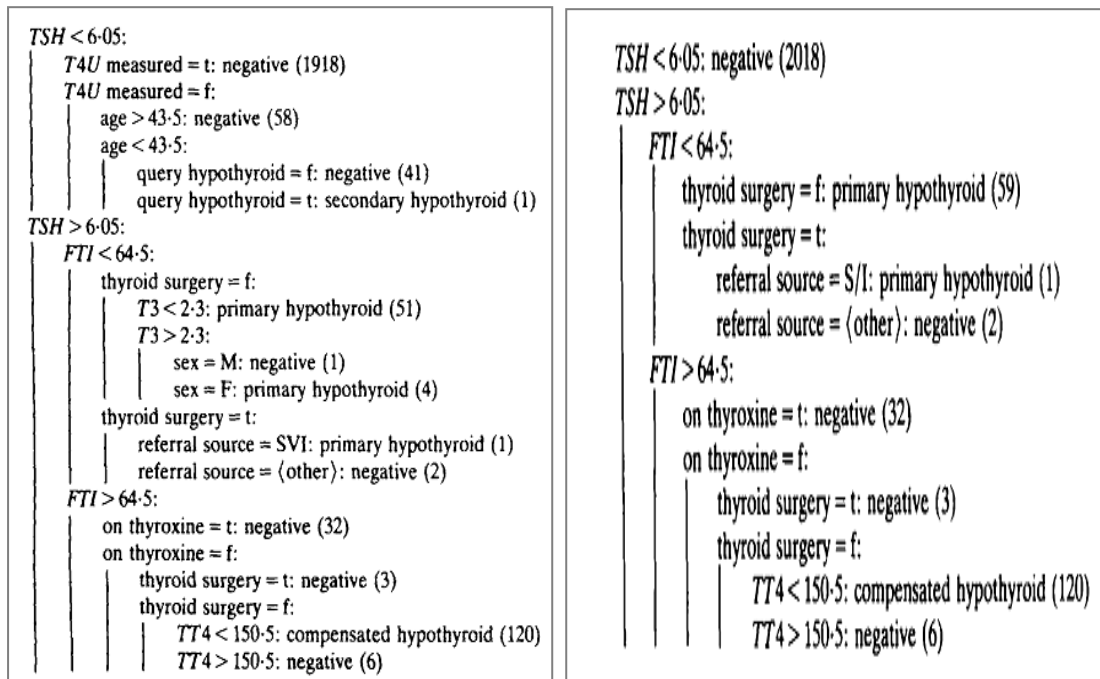


Figure B.2 (a) Arbre de décision originale, (b) A.D après post-élagage (REP, PEP) [Quinlan 87]

B.2 K-plus proches voisins :

En intelligence artificielle, la méthode des *k plus proches voisins* (en anglais *k-nearest neighbor*) est une méthode d'apprentissage supervisé. Le principe de cette méthode est illustré dans L' **Algorithme B.1**). [Cornuéjols 2009]

Algorithme B.1 K-plus proches voisins

- Entrée : Point x ,
 - Sortie : Classe de x .
 - Pour chaque exemple y de l'ensemble d'apprentissage faire
Calculer la distance $D(y, x)$ entre y et x .
 - Fin pour**
 - Dans les K points les plus proches de x .
Compter le nombre d'occurrences de chaque classe.
 - Attribuer à x la classe qui apparait le plus souvent.
 - Fin**
-

L'inconvénient majeur de cette approche est que tous les voisins ont le même poids pour l'affectation d'une classe à un vecteur de données. Aussi, elle ne spécifie pas le degré d'appartenance d'un vecteur à une classe donnée.

Pour palier ce problème, Keller et al. [Keller 1985] ont proposé un classifieur K-ppv flou (Voir **Algorithme B.2**) .

Algorithme B.2 K-plus proches voisins Flou

Entrée : Point x ,

Ensemble K , $1 \leq K \leq n$.

Sortie : Classe de x .

$i = 1$.

- **Faire**

Calculer la distance entre x et x_i

Si ($i \leq K$) alors

Inclure x_i dans l'ensemble des K plus proches voisins.

Sinon Si (x_i est le plus proche élément des voisins de x) alors

Supprimer le plus loin élément des K -plus proches voisins.

Inclure x_i dans l'ensemble des k -plus proches voisins.

Fin si.

Jusqu'à (K -plus proches voisins soient trouvés)

$i = 1$.

- **Faire**

Calculer $\mu_i(x)$.

$i=i+1$;

Jusqu'à (x assigne les degrés d'appartenance pour toutes les classes)

Fin

Avec :

$$\mu_i(x) = \frac{\sum_{j=1}^k \mu_j \left(\frac{1}{\|x - x_j\|^{2/(m-1)}} \right)}{\sum_{j=1}^k \left(\frac{1}{\|x - x_j\|^{2/(m-1)}} \right)}$$

Annexe C

C.1 Les modèles Neuro-Flous

Il existe différents modèles neuro-flous qui font usage d'hybridation de réseaux neuronaux et systèmes d'inférence flou (SIF) [Ammar 2008].

Les figures C.1, C.2 montre les différentes architectures des systèmes neuro-flous.

C.1.1 FALCON : (Fuzzy Adaptive learning Control Network) [Abraham 2001]

Comme le montre la figure C.1 au-dessous, Falcon est un réseau comportant cinq couches. Il possède deux nœuds linguistiques, une pour la sortie désiré et l'autre pour la sortie du Falcon.

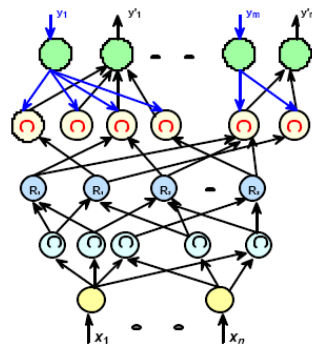


Figure C.1 : Architecture de FALCON

La première couche cachée est responsable de la fuzzification des variables d'entrées tandis que la deuxième couche définit les parties antécédentes des règles floues suivie par les parties conséquences des règles dans la troisième couche cachée. FALCON utilise un apprentissage hybride pour l'ajustement des paramètres des fonctions d'appartenance afin de générer les sorties désirées.

C.1.2 NEFCON : (Neuro-Fuzzy Control) [Abraham 2001]

NEFCON vise à mettre en œuvre le SIF de type Mamdani. Il est composé de trois couches : une couche d'entrée incluant les nœuds d'entrées et les sous-ensembles flous d'antécédant, une couche cachée formée par des règles et un neurone de sortie pour les sous-ensembles flous des conséquences. NEFCON peut être employé pour apprendre des règles initiales, si aucune connaissance du système n'est disponible ou même pour optimiser une base de règles définie. L'architecture de ce modèle est illustrée par la figure C.2 suivante

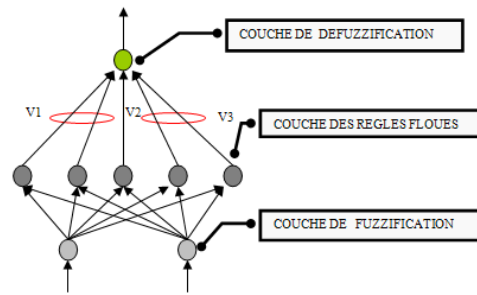


Figure C.2 : Architecture de NEFCON

C. 1.3 FUN : (Fuzzy Net) [Abraham 2001]

Pour FUN , les neurones dans la première couche cachée contient les fonctions d'appartenance pour exercer une fuzzification des valeurs d'entrée. Dans la deuxième couche cachée, les conjonctions (fuzzy-AND) sont calculées. Les fonctions d'appartenance des variables de sortie sont stockées dans la troisième couche cachée. Leur fonction d'activation est un (fuzzy-OR).

A la fin , le neurone de sortie effectue la défuzzification. Le réseau est initialisé avec une base de règles floues et les fonctions d'appartenance correspondant .il utilise une technique d'apprentissage qui change les paramètres des fonctions d'appartenance.

C.2 L'approche K-moyenne (K-means):

L'approche k-means, est l'une des techniques la plus populaire utilisé en clustering. L'objectif de cette approche est de trouver les regroupements (clustering) qui minimise la somme des distances quarrées des éléments de chaque cluster aux centres des clusters. Formellement cet objectif est donné par la formule suivante :

$$J_K = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2 \quad (C.4)$$

L'algorithme procède en deux étapes : dans la première phase, on réassigne tous les objets au centroïde le plus proche, et dans la deuxième phase, on recalcule les centroïdes des classes qui ont été modifiées. L'algorithme de la méthode K-moyenne est décrit au-dessous : [Charu 2013]

- 1- Choisir aléatoirement les K points de données qui seront les centroïdes initiaux des catégories.
- 2- **Faire**
 - Pour chacun des x_i , rattacher au centroïdes le plus proche.
 - Recalculer chacun des centroïdes en faisant la moyenne des points associés.

Tant que changement parmi les centroïdes.

- 3- Retourner les centroïdes .

La méthode de k-moyenne est une approche simple, rapide, facile à comprendre, utilisé pour résoudre des problèmes de classification non supervisé. Cependant, il existe deux facteurs majeurs qui peuvent avoir un impact sur la performance de l'algorithme K-moyenne :

- Le choix des centroïdes initiales.
- L'estimation du nombre des clusters K .

C.3 Expansion Radial :

Dans ce type d'extension, un ensemble homogène F est étendu dans toutes les directions. Soit M : la région étendue à partir de F et R_M, R_F : les radius du M et F respectivement. Dans l'approche d'extension radial, R_F est augmenté par une quantité T (appelé :Step-Size Increase) . Ainsi :

$$R_M = R_F + T \quad (C.5)$$

En suivant la méthodologie de recherche dichotomique, nous supposons aussi qu'il existe une hypersphère G qui couvre l'ensemble homogène F (sans perte de généralité) avec R_G est radius de G :

$$R_G = 2 \times R_F \quad (C.6)$$

En utilisant R_F, R_G, T est dérivé de la différence entre R_G, R_F . Par exemple, T peut être déterminé comme suite :

$$T = \frac{R_G - R_F}{2} \quad (C.7)$$

T doit dépendre de $DH(F)$ à cause de la méthodologie de recherche dichotomique i.e. Si $DH(F)$ augmente alors T diminue. ($DH(F)$ est inversement proportionnelle à T), en utilisant une valeur de seuil (L), on obtient :

$$T = \frac{R_G - R_F}{2} \times \frac{1}{L \times DH(F)} \quad (C.8)$$

Ainsi :

$$R_M = R_F + \frac{R_G - R_F}{2} \times \frac{1}{L \times DH(F)} \quad (C.9)$$

C.3.1 Expansion linéaire (radial vs linéaire) :

L'expansion linéaire étend un ensemble homogène F dans une direction donnée. La différence entre ce type d'expansion et l'expansion cite au-dessus, est qu'un ensemble homogène F est étendu en une hypersphère M en utilisant l'expansion radial. Ensuite, l'hypersphère M est étendue dans une direction donnée en utilisant l'approche d'expansion radial jusqu'à satisfaire les conditions d'arrêts citées au-dessus. La région finale est l'union entre les régions étendues.

C.3.2 Conditions d'arrêts :

Comme a été décrit au-dessus, les conditions d'arrêts de l'approche d'expansion radiale pour un ensemble homogène F de taille n_c , doit satisfaire les deux besoins suivants : [Pham 2007,2008,2009]

- Dépend du degré d'homogénéité. Ceci a été mentionné dans les suppositions fondamentales du *A.B.H* (voire section : III.4 du chapitre III).
- Lorsqu' une région étendue atteint un autre modèle. Afin de répondre au 1^{er} besoin, une borne supérieure R_M doit être directement proportionnel au $DH(F)$, la valeur α^+ et le radius original R_F .

Le 2^{eme} besoin doit être déterminé durant l'expansion. De plus, une borne supérieure pour le nombre des points bruités, doit être directement proportionnelle aux $DH(F)$ et la taille de F (n_f). L'algorithme de l'expansion radial est défini comme suite : [Pham 2007,2008,2009]

Algorithme C.1 Expansion radial

Entrée : Ensemble homogène F , $HD(F)$, R_F , α^+ .

Sortie : Région E expansif.

- Ensemble $M = F$ ($R_F = R_M$).

- Couvrir l'ensemble M par un hyper sphère G avec radius $R_G = 2 * R_M$.

- Répéter :

Ensemble $E = M$ ($R_E = R_M$).

Etendre M en utilisant l'équation (C.9).

Jusqu'à (R_M satisfait les conditions d'arrêts cités aux dessous ou $R_M = R_G$).

- Si (R_M satisfait les conditions d'arrêts) **Alors**

Stop.

Si non

Aller a l'étape 2.

Fin

Fin

C.4 Approche A.G pour trouver les valeurs des seuils :

Rappelons que l'algorithme principale décrit dans (*Algorithme III.1*), utilise quatre valeurs de seuils : α^+ , α^- , β^+ , β^- pour dériver des nouveaux systèmes de classification. Avoir des valeurs de réduction β^+ , β^- élevées amène a un problème de sur-apprentissage. A l'inverse, avoir des valeurs des seuils de réduction petits peut dériver un problème de surgénéralisation. La situation opposée est vrai pour les valeurs des seuils d'expansion α^+ , α^- . [Pham 2007 ,2008, 2009].

Par conséquent, une recherche exhaustive serait impraticable. [Pham 2007], propose d'utiliser l'équation III.23 comme fonction de fitness et la base T2 : comme

base de calibration. L'approche **A.G** est appliquée car l'équation **III.23** n'est pas unimodale. [Pham 2008,2009].

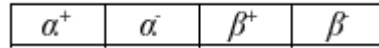


Figure C.3 : Exemple illustratif d'un chromosome composé de 4 gènes

Chaque chromosome consiste de quatre gènes correspondant aux valeurs des seuils α^+ , α^- , β^+ , β^- (voir **figure C.3**). l'algorithme **A.G** crée un enfant (crossver child) en combinant les paires des parents de la population courante. La **figure C.4** présente deux chromosomes **A**, **B** avec les quatre gènes correspondants (α_1^+ , α_1^- , β_1^+ , β_1^-), (α_2^+ , α_2^- , β_2^+ , β_2^-).[Pham 2008,2009].

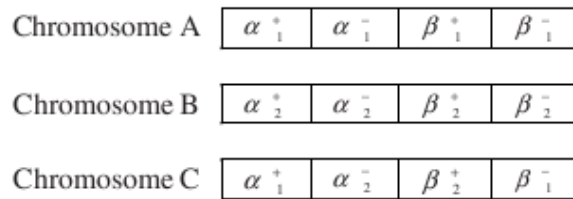


Figure C.4 : Exemple illustratif de la fonction de croisement

Le chromosome **C** (α_1^+ , α_2^- , β_2^+ , β_1^-), est créé en sélectionnant aléatoirement un gène à la même coordonnée de l'un des chromosomes **A**, **B** et l'assigner à l'enfant **C**. Après, la fonction-mutation crée un enfant (**g1**, **g2**, **g3**, **g4**) en changeant aléatoirement les gènes du chromosome parent (α^+ , α^- , β^+ , β^-).

*Références
bibliographiques*

[**Abraham 2001**] Ajith Abraham, Neuro-Fuzzy Systems: State-of-the-Art Modeling Techniques, Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Lecture Notes in Computer Science. Volume. 2084, Springer Verlag Germany, Jose Mira and Alberto Prieto (Eds.), ISBN 3540422358, Spain, pp. 269-276, 2001.

[**Ammar 2008**] Ammar Mohammed. Reconnaissance automatique du diabète et prédiction de la dose d'insuline . Faculté de sciences des ingénieurs. Département d'électronique biomédicale. Université Abou bekr Belkaid .Promotion 2008-2009 .

[**Andreasson 2005**] Niclas Andreasson, Anton Evgrafov, et M. Patriksson. Introduction to Continuous Optimization : Foundations and Fundamental Algorithms. Student-litteratur. ISBN : 978-9144044552 .2005.

[**Arora 09**] S. Arora & B. Barak. Computational complexity : a modern approach. Cambridge University Press, 2009.

[**Asuncion 2007**] Asuncion, A., Newman, D. J. . UCI-Machine Learning Repository. School of Information and Computer Sciences, University of California, Irvine, CA, USA .2007.

[**Au 2001**] Au, W. H., Chan, K. C. C. Classification with degree of membership: a fuzzy approach. In Proceedings of the 1st IEEE international conference on data mining pp. 35–42. San Jose, CA, USA. 2001.

[**Baraldi 1998**] Andrea Baraldi, Palma Blonda, Alfredo Petrosino. Fuzzy Neural Networks for Pattern Recognition. 1998, pp 35-83.

[**Bekaddour 2012.a**] Bekaddour Fatima. M. Amine Chikh . A Neuro-Fuzzy Inference Model for Breast Cancer Recognition. International Journal of Computer Science & Information Technology (IJCSIT) Vol 4, No 5, October 2012 . p.p. 163-173

[**Bekaddour 2012.b**] Bekaddour Fatima. M. Amine Chikh. An Interpretable Fuzzy Classifier For Breast Cancer Diagnosis . BIOMEDICAL ENGINEERING INTERNATIONAL CONFERENCE (BIOMEIC' 12)- OCTOBER 10-11, 2012, TLEMEN (ALGERIA).

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

[**Blachnik, 2006**] Blachnik, M., & Duch, W. Prototype-based threshold rules. LNCS of neural information processing (Vol. 4234). Berlin/Heidelberg: Springer. pp. 1028–1037. 2006

[**Blum 03**] Blum C, Roli A . Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys 35(3): 268–308 .2003

[**Braune 2010**] Michael Bögl ,Günther Zäpfel , Roland Braune Metaheuristic Search Concepts . Springer-Verlag Berlin and Heidelberg GmbH & Co. K; Édition : 1 (février 2010). ISBN: 978-3-642-11342-0. 2010

[**Cahon 2004**] Cahon, S., Melab, N., & Talbi, E.G. (2004, May). Paradis EO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. *Journal of Heuristics*, 10(3), 357–380. <http://dx.doi.org/10.1023/HEUR.0000026900.92269.ec>

[**Charu 2013**] . Charu C. Aggarwal, Chandan K. Reddy (Eds.): Data Clustering: Algorithms and Applications. CRC Press. 2013. ISBN 978-1-46-655821-2

[**Chikh 2005**] Mohammed Amine Chikh . Analyse du signal ECG par les réseaux de neurones et la logique floue.Application a la reconnaissance des battements ventriculaires prématurés. 2005.

[**Cohen 2007**] Cohen S., L. Rokach, O. Maimon, "Decision-tree instance-space decomposition with grouped gain-ratio, ", *Information Science*, Volume 177, Issue 17, pp 3592-3612. 2007

[**Cornuéjols 2003**] Cornujols, A., Miclet, L., Kodratoff, Y., Mitchell, T.: *Apprentissage artificiel : concepts et algorithmes*. Eyrolles (2003) .

[**Demuth 98**] Howard Demuth, Mark Beale. Neural Network Toolbox For Use with MATLAB:Computation ,Visualization, Programming January 1998 .Version 3 by The MathWorks, Inc.

[**Dreyfus 2008**] Gérard *Dreyfus*, Jean-Marc Martinez, Manuel Samuelides, Collectif, "Apprentissage statistique" Eyrolles , 2008 ,ISBN: 2212122292 , 449 pages .

[**Du 2006**] K.-L. Du, M.N.S. Swamy. Neural Networks in a Soft computing Framework. Springer, Aug 2, 2006 - Computers - 616 pages. ISBN-10: 1846283027.

[**Dudani 1976**] Dudani, S., "The Distance-Weighted k-Nearest-Neighbor Rule" .IEEE Transactions on Systems, Man and Cybernetics, vol. 6, no. 4, pp. 325-327. 1976.

[**Dudzinski 87**] K. Dudzinski & S. Walukiewicz. Exact methods for the knapsack problem . *European Journal of Operational Research*, vol. 28, no. 1, pages 321, 1987.

[**Fink 2003**] Andreas Fink, Stefan Voß, David L. Woodruff . Metaheuristic Class Libraries . Handbook of Metaheuristics. International Series in Operations Research & Management Science Volume 57, 2003, pp 515-535

[**Fortnow 09**] L. Fortnow.The status of the P versus NP problem. *Communications of the ACM*, vol. 52, no. 9, pages 78-86, 2009.

[**Freeman 1991**] James A. Freeman , David M. Skapura. Neural Networks Algorithms, Applications, and Programming Techniques .Loral Space Information Systems and

Adjunct Faculty, School of Natural and Applied Sciences. University of Houston at Clear Lake. Addison-Wesley Publishing Company, Inc. 1991

[Fullér 1995] R. Fullér, Neural fuzzy systems, Université de Abo Akademi . ESF Séries A:443, 1995. 249 pages . [ISBN 951-650-624-0, ISSN 0358-5654] .

[Gaspero 2002] Luca Di Gaspero and Andrea Schaerf .Writing local search algorithms using EASY LOCAL ++. In: Stefan Voß and David L. Woodruff (eds.), Optimization Software Class Libraries, OR/CS Interfaces Series. Kluwer Academic Publishers, Boston, pp. 155–175.2002

[Glover 86] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research, vol. 13, no. 5, pages 533-549, 1986.

[Glover 89i] F. Glover. Tabu search - part i. ORSA Journal on Computing, 1:190–206, 1989.

[Glover 89ii] F. Glover. Tabu search - part ii. ORSA Journal on Computing, 2:4–32, 1989.

[Hao 1999] Jin-Kao Hao, Philippe Galinier, Michel Habib. Méthaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. Revue d'Intelligence Artificielle .Vol 13. No.2 .pp : 283-324. 1999.

[Hassoun 96] M. H. Hassoun, Fundamentals of Artificial Neural Networks, MIT PRESS, 1996.

[Haykin 99] Simon Haykin . Neural Networks. A Comprehensive Foundation .Second Edition . McMater University Hamilton, Ontario, Canada. 1999 by Pearson Educaton.Inc

[Hertz 2003] A. Hertz and M. Widmer. Guidelines for the use of meta-heuristics in combinatorial optimization. European Journal of Operational Research, 151:247–252, 2003.

[Hideyuki 1997] Hideyuki TAKAGI . Introduction to Fuzzy Systems, Neural Networks, and Genetic Algorithms. In :Intelligent Systems: Fuzzy Logic,Neural Networks, and Genetic Algorithms, Ch.1, pp.1–33,edited by D. Ruan, Kluwer Academic Publishers (Norwell,Massachusetts, USA), (September, 1997).

[Hosseini 2011] Monireh Sheikh Hosseini, Maryam Zekri. Review of Medical Image Classification using the Adaptive Neuro-Fuzzy Inference System. Journal of Medical Signals & Sensors. Journal of Medical Signals and Sensors. 2011.

[Jang 92] R.Jang, neuro-fuzzy modeling :architecture, analyses and applications, PhD thesis ,Dep. Of electrical Engineering and computer Science, University of California, Berkeley, 1992

- [**Jang 93**] R.Jang, Anfis :adaptative network-based fuzzy inference système ;IEEE Trans.on Systems,Man and Cybernetics,J.S.,1993
- [**Jang 97**] Jyh-Shing Roger Jang ;Chuen-Tsai Sun; Eiji Mizutani. Neuro Fuzzy And Soft Computing. A computational Approach to learning and Machine intelligence. Prentice Hall, 1997
- [**Johnson 85**] D. S. Johnson.The NP-completeness. Journal of Algorithms , vol. 6, no. 3, pages 434-451, 1985.
- [**Kamber 2006**] Jiawei Han, Micheline Kamber, Jain Pei .Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor .Morgan Kaufmann Publishers, March 2006. ISBN 1-55860-901-6 . 2nd edition.
- [**Karp 72**] R. M. Karp.Reducibility among combinatorial problems. Complexity of Computer computations, pages 85-104, 1972.
- [**Keller 1985**] Keller, J. M., M. R. Gray, and J. A. Givens, Jr, , "A Fuzzy K-Nearest Neighbor Algorithm " Journal of IEEE Transactions on Systems, Man, and Cybernetics, vol. 15, no. 4, pp. 580-585. 1985.
- [**Kohavi 1996**] Kohavi. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid, «Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, pp. 202-207. 1996
- [**Konar 2000**] Amit Konar .Artificial Intelligence and Soft Computing .Behavioral and Cognitive Modeling of the Human Brain. Department of Electronics and Telecommunication Engineering Jadavpur University, Calcutta, India. 2000 by CRC Press LLC.
- [**Korte 2002**] B. Korte, J. Vygen, Combinatorial optimization Theory and Algorithms (Algorithms and Combinatorics), 2nd ed., Springer, 2002.
- [**Kwan 1994**] H. Kwan et Y.Cai, A fuzzy neural network and its application to pattern recognition, IEEE Transactions on Fuzzy Systems, pp .1859193. 1994.
- [**Lin 1996**] Lin,C.-T and Lee, G .Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems .Ed. Prentice Hall. 1996.
- [**Luke 2013**] Sean Luke. Essentials of Metaheuristics . An electronic book available for download at <http://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf>. Second Edition Version 2.0.June, (2013). publisher : Lulu . ISBN: 978-1-300-54962-8.
- [**Magàly 2001**] Anne Magàly de Paula Canuto .Combining Neural Networks and Fuzzy Logic for Applications in Character Recognition. A thesis submitted to the University of Kent at Canterbury for the degree of Doctor Philosophy in the subject of Electronic Engineering. May 2001 .

[**Mededjel 2007**] Mansour Mededjel, Hafida Belbachir . Post-élagage Indirect des Arbres de Décision dans le Data Mining . SETIT:4th International Conference: Sciences of Electronic, Technologies of Information and Telecommunications March 25-29, 2007 – TUNISIA.

[**Michalewicz 2004**] Zbigniew Michalewicz • David B. Fogel .How to Solve It: Modern Heuristics . Springer: Second, Revised and Extended Edition. ISBN: 978-3540224945. September 21, 2004

[**Ming 2002**] H. Ming, THE IMPROVEMENT OF A FUZZY NEURAL NETWORK BASED ON BACKPROPAGATION, Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 5 November 2002 .

[**Moody 1992**] Moody, J. E., “The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems, ” Journal of Advances in Neural Information Processing Systems, vol. 4, pp. 847-854. 1992.

[**Nakashima 2003**] Nakashima, T., Nakai, G., Ishibuchi, H. Constructing fuzzy ensembles for pattern classification problems. In Proceedings of the international conference on systems, man and cybernetics (Vol. 4, pp. 3200–3205). Washington, DC, USA (October). 2003

[**Nauck 97**] D. Nauck, F. Klawon; R. Kruse, “Foundations of Neuro-Fuzzy Systems”, J. Wiley & Sons, 1997.

[**Opeyemi 2012**] Obanijesu Opeyemi, Emuoyibofarhe O. Justice .Development of Neuro-fuzzy System for Early Prediction of Heart Attack . I.J. Information Technology and Computer Science, 2012, 9, 22-28

[**Osman 96**] I. H.Osman and G.Laporte. Metaheuristics : A bibliography. Annals of Operations Research, 63:513–623,1996

[**Papadimitriou 82**] C.H. PAPANIMITRIOU, K. STEIGLITZ, Combinatorial optimization - algorithms and complexity. Prentice Hall, 1982.

[**Papadimitriou 03**] C. H. Papadimitriou. Computational complexity. John Wiley and Sons, 2003.

[**Patnaik 2012**] Srikanta Patnaik, Yeon-Mo Yang (Eds.): Soft Computing Techniques in Vision Science. Studies in Computational Intelligence 395, Springer 2012, ISBN 978-3-642-25506-9.

[**Pham 2007**] Pham, H. N. A., & Triantaphyllou, E. The impact of overfitting and overgeneralization on the classification accuracy in data mining. In O. Maimon & L. Rokach (Eds.), Soft computing for knowledge discovery and data mining, part 4(pp. 391–431). New York, NY, USA: Springer.2007

- [**Pham 2008**] Pham, H. N. A., & Triantaphyllou, E. Prediction of diabetes by employing a new data mining approach which balances fitting and generalization. In Roger Yin Lee (Ed.). Studies in computation intelligence (Vol. 131, pp. 11–26). Germany: Springer. 2008
- [**Pham 2009**] Pham HNA, Triantaphyllou E. An application of a new meta-heuristic for optimizing the classification accuracy when analyzing some medical datasets. Expert Systems with Applications 2009;36(5):9240–9
- [**Pham 2011**] Pham, H. N. A., & Triantaphyllou, E. A meta-heuristic approach for improving the accuracy in some classification algorithms. Computers & Operations Research 38 (2011) 174–189.
- [**Prechelt 1998**] Lutz Prechelt. Early Stopping -- but when? in: Neural Networks: Tricks of the trade, pp 55-69, Lecture Notes in Computer Science 1524, Springer Verlag, Heidelberg, 1998.
- [**PREUX 2011**] Ph. PREUX. Fouille de données. Notes de cours. Université de Lille 3 philippe.2011 preux@univ-lille3.fr. <http://www.grappa.univ-lille3.fr/~ppreux/fouille>
- [**Qu 2005**] Youli Qu, Wenqian Shang, Houkuan Huang, Haibin Zhu, Youngmin Lin, Zhihai Wang. An Improved KNN Algorithm – Fuzzy KNN. Lecture Notes in Computer Science Volume 3801, 2005, pp 741-746 . Springer Link.
- [**Quinlan 1987**] Quinlan, J. R., " Simplifying decision trees, " International Journal of Man-Machine Studies, vol. 27, pp. 221-234. 1987.
- [**Rajeswari 2010**] P.Rajeswari, G.Sophia Reena. Analysis of liver disorder using data-mining algorithm. Global journal of computer science and technology . Vol.10 issue 14 (Ver.1.0) November 2010 pp: 48-52. 2010.
- [**Ramana 2011**] Bendi Venkata Ramana, M.Surendra Prasad Babu , N.B. Venkateswarlu . A critical Study of Selected Classification Algorithms for Liver Disease Diagnosis. International Journal of Database Management Systems (IJDMS), Vol 3. No2, May 2011.
- [**Saidi 2012**] , Mohamed Amine Chikh, Nesma Settouti, Diagnosis of Diabetes Diseases Using an Artificial Immune Recognition System2 (AIRS2) with Fuzzy K-nearest Neighbor . Journal of Medical Systems , October 2012, Volume 36, Issue 5, pp 2721-2729
- [**Sakarovitch 1984**] Sakarovitch, M. .Optimisation combinatoire, Méthodes mathématiques et algorithmiques - Programmation discrète. Hermann. ISBN : 2-7056-5976-5.1984
- [**Settouti 2011**] N. Settouti Renforcement de l'Apprentissage Structurel pour la reconnaissance du diabète. 2011. MÉMOIRE pour obtenir le grade de MAGISTER ENÉLECTRONIQUEBIOMÉDICALE. Juin 2011

- [**Settoui 2012**] Nesma Settoui, Meryem Saidi, and Mohamed Amine Chikh. Interpretable Classifier of Diabetes Disease. *International Journal of Computer Theory and Engineering* vol. 4, no. 3, pp 438-442. 2012
- [**Sharma 2011**] Disha Sharma. Designing and Modeling Fuzzy Control Systems . *International Journal of Computer Applications (0975–8887)* Volume 16 – No.1, pp : 46-49 .February 2011.
- [**Sheng 1999**] Sheng Ma Chuanji Ji. Performance and Efficiency: Recent Advances in Supervised Learning . *PROCEEDINGS OF THE IEEE, VOL. 87, NO. 9, SEPTEMBER 1999.*
- [**Sipser 92**] M. Sipser. The history and status of the P versus NP question. In *Proceedings of the 24th annual ACM Symposium on Theory Of Computing (STOC'92)*, pages 603-618, Victoria, BC, Canada, May 4-6, 1992. ACM.
- [**Talbi 09**] El-Ghazali Talbi . *Metaheuristics: From Design to Implementation* . ISBN: 978-0-470-27858-1 .624 pages. July 2009. Wiley
- [**Tan 2005**] Tan, P. N., S. Michael, and K. Vipin, "Introduction to Data Mining ", Addison-Wesley Publisher, pp. 145-315. 2005,
- [**Tano 1996**] S. Tano, T. Oyama et T. Arnould, Deep combination of Fuzzy Inference and Neural Network in Fuzzy Inference, *Fuzzy Sets and Systems*, pp. 1519160. 1996.
- [**Tso 09**] Brand Tso & Paul M. Mather .*Classification Methods For Remotely Sensed Data*. Second Edition. CRC Press 2009 by Taylor & Francis Group, LLC
- [**VIEIRA 2004**] Vieira, J., F. Morgado Dias, and A. Mota . *Neuro-Fuzzy Systems: A Survey*. In *processing International Conference on Neural Networks and Applications* . 2004.
- [**Voß 99**] S.Voß, S.Martello, I.H.Osman, and C.Roucairol, editors. *MetaHeuristics-Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999
- [**Voudouris 2002**] Christos Voudouris and Raphaël Dorne . Integrating heuristic search and one-way constraints in the iOpt toolkit. In: Stefan Voß and David L. Woodruff (eds.), *Optimization Software Class Libraries, OR/CS Interfaces Series*. Kluwer Academic Publishers, Boston, pp. 177–191.2002
- [**Weigend 1993**] Weigend, A., "On overfitting and the effective number of hidden units" *Proceedings of the 1993 Connectionist Models Summer School*, pp. 335-342. 1993.
- [**Weise 2009**] Thomas Weise .*Global Optimization Algorithms – Theory and Application–* .An electronic book available for download at <http://www.it-weise.de/projects/book.pdf>. 2nd edition . self-published: Germany. 2009 .

[Weiss 89] Weiss, S. M., Kapouleas, I. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In Proceedings of the 11th International joint conference on artificial intelligence pp. 781–787. Detroit, MI, USA .1989.

[Wu 2011] Yue Wu , Biaobiao Zhang ; Jiabin Lu ; K. -L. Du . Fuzzy Logic and Neuro-fuzzy Systems: A Systematic Introduction. International Journal of Artificial Intelligence and Expert Systems (IJAE), Volume (2) : Issue (2) : 2011.

[Zadeh 1965] Lotfi A.Zadeh. Fuzzy Sets. Information Control Journal, volume 8. pp:338--353, 1965.

[Zadeh 1996] Lotfi A.Zadeh. Fuzzy Logic = Computing with words. Life follow, IEEE Transactions On Fuzzy Logic Systems. 1996

[Zhu 2009] Xiaojin Zhu and Andrew B. Goldberg. Introduction to Semi-Supervised Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool, 2009.

[Zhou 2002] Zhou Z. and C. Chen, "Hybrid decision tree" Journal of Knowledge-Based Systems, vol. 15, pp. 515 - 528. 2002.

Résumé

La performance d'une méthode de classification est d'un grand intérêt pour le choix, la comparaison et la validation des algorithmes de classification. La lisibilité des résultats et la réduction du cout d'échec total de la classification sont d'une importance cruciale pour l'amélioration de la performance des classifieurs. Dans ce mémoire de magister, nous proposons principalement deux approches de résolution à travers la description d'un modèle d'optimisation métaheuristique baptisé *ProSadm-HBA* (*ProSadm* : Programmation d'un Système d'Aide au Diagnostic Médical, en conjonction avec la métaheuristique *HBA* : Homogeneity Based-Algorithm) et *F-HBA* (Fuzzy Homogeneity Based-Algorithm). Nous avons validé nos résultats expérimentaux sur des bases de données médicales connues : *Pima* (Diabètes), *TH* (Troubles Hépatiques), *AP* (Appendicite). Les approches développées permettent de minimiser le nombre total d'échecs de la classification (*ProSadm-HBA*) tout en respectant la contrainte d'interprétabilité des classifieurs (*F-HBA*). Ces contributions peuvent être d'un grand intérêt pour les experts dans le domaine médical.

Mots clés: Performances, Sur-Apprentissage, Sur-Généralisation, Métaheuristique, ProSadm, ANFIS, HBA, Optimisation, Interprétabilité.

Abstract

Performance of a classification method is of great interest for choosing, comparing and validation of classification algorithms. The transparency of results and the reduction of the total misclassification cost are of vital importance as well as the improvement of the performance of classifiers. In this magister report, we propose two solving approaches through the description of a meta-heuristic optimization model denoted ProSadm-HBA (ProSadm in conjunction with HBA: Homogeneity Based-Algorithm) and F-HBA (Fuzzy Homogeneity Based-Algorithm) .we validated our experimental results based on three well known medical datasets: *PID* (Pima Indian Diabetes), *LD* (Liver Disorders) and *AP* (Appendicitis) . The developed approaches allow minimizing the total number of misclassification cases (ProSadm-HBA) while respecting the interpretability constraint of classifiers (F-HBA). These contributions can be of great interest for experts in the medical field.

Keywords: Performance, Overfitting, Overgeneralization, Metaheuristic, ProSadm, ANFIS, HBA, Optimization, Interpretability.

ملخص

تعد ماهية أداء طرق التصنيف ذات منفعة كبيرة لاختيار, مقارنة و تقييم خوارزميات التصنيف. تعتبر الشفافية و الحد من الأخطاء الناتجة عن التصنيف ذوا أهمية كبيرة لتحسين أداء طرق التصنيف . في رسالة الماجستير هذه, نقترح كحل, نموذجان أساسيان يعتمدان علي نظام تحسين من خلال استعمال *Meta-heuristic* و قد أسميناها *ProSadm - HBA* & (*F-HBA*). لقد قمنا بتقييم النتائج المتحصل عليها وذلك عبر استعمال قواعد بيانات طبية معروفة : السكري, اضطرابات الكبد و التهاب الزائدة الدودية . إن النماذج المطورة تسمح بتقليص أخطاء التصنيف (*ProSadm-HBA*) مع احترام شفافية أنظمة التصنيف (*F-HBA*) . يمكن لمثل هذه المساهمات أن تكون ذات فائدة كبيرة للخبراء في المجال الطبي.

الكلمات المفتاحية : ANFIS, ProSadm, Metaheuristic, Overgeneralization, Overfitting, تحسين , شفافية .