

ALGORITHMES MÉTAHEURISTIQUES HYBRIDES POUR LA SÉLECTION DE GÈNES ET LA CLASSIFICATION DE DONNÉES DE BIOPUCES

THÈSE DE DOCTORAT

Spécialité : Informatique

ÉCOLE DOCTORALE STIM

Présentée et soutenue publiquement

Le 14 novembre 2008

À Angers

Par **José Crispín HERNÁNDEZ HERNÁNDEZ**

Devant le jury ci-dessous :

<i>Rapporteurs :</i>	Christel VRAIN, Alexandre CAMINADA,	Professeur à l'Université d'Orléans Professeur à l'Université de Technologie de Belfort-Montbéliard
<i>Examineurs :</i>	Frédéric SAUBION, Carlos A. REYES GARCIA,	Professeur à l'Université d'Angers Professeur au INAOEP, Mexique
<i>Directeur de thèse :</i> <i>Co-encadrant :</i>	Jin-Kao HAO, Béatrice DUVAL,	Professeur à l'Université d'Angers Maître de conférences à l'Université d'Angers

Remerciements

Je tiens tout d'abord à remercier à Jin-Kao HAO, mon directeur de thèse, pour m'avoir accueilli, encadré, soutenu et prodigué de nombreux conseils tout au long de ces quatre années de thèse. Je lui suis particulièrement reconnaissant de m'avoir laissé une grande liberté scientifique, ce qui m'a permis de recevoir un apprentissage idéal et privilégié de la recherche.

Je remercie vivement Béatrice DUVAL, co-encadrante de cette thèse, qui m'a beaucoup appris et beaucoup apportée, sur le plan scientifique. Qu'elle soit consciente que ce travail ne serait pas là sans elle.

J'adresse mes sincères remerciements à Alexandre CAMINADA et Christel VRAIN, les deux rapporteurs de ma thèse, pour avoir accepté d'apporter leurs avis éclairés sur mes travaux de recherche. Je sais également gré à Frédéric SAUBION et Carlos Alberto REYES GARCIA, qui ont accepté de participer au jury en tant qu'examineurs.

Je remercie tout particulièrement Jean-Michel RICHER, Frédéric LARDEUX et Anne et Bertrand RAIMBAULT pour leur disponibilité, leur aide, leurs remarques pertinentes et leur participation à la tâche laborieuse que constitue la relecture du présent manuscrit.

Mes remerciements vont aussi à tous les membres permanents du LERIA et du département informatique. Je remercie particulièrement les personnes qui m'ont permis de résoudre des problèmes aussi bien techniques qu'administratifs.

Je ne peux qu'adresser un remerciement particulier et amical à Frédéric LARDEUX, Eduardo RODRIGUEZ et Lisset sa femme, pour toute l'aide que vous m'avez apportée au moment de mon arrivée en France.

Je suis reconnaissant envers mes collègues thésards : Edmundo BONILLA HUERTA et Giglia GOMEZ VILLOUTA. Je remercie par la même occasion mes amis : la famille RAIMBAULT, la famille CHEVALLEREAU et la famille GOMEZ-MARQUEZ pour les bons moments de détente passés en leur compagnie. Merci à vous tous, car vous avez fait de mon séjour en France une expérience inoubliable.

J'aimerais remercier du fond du coeur mes parents pour leur soutien moral et matériel, ma famille et belle famille qui ont toujours porté un intérêt à ce que je faisais. Qu'ils trouvent dans ce travail le témoignage de mon affection.

Mes remerciements ne seraient pas complets sans y avoir associé Maria Luisa et Ivan David, ma femme et mon fils, que je remercie du fond du cœur pour m'avoir accompagné dans cette aventure qu'est d'effectuer un doctorat à l'étranger avec tous les doutes et les satisfactions que cette expérience a pu nous procurer. Maria Luisa et Ivan David, qui ont su m'encourager, me soutenir et faire des concessions afin que ma thèse se déroule dans les meilleures conditions, je vous serai toujours très reconnaissant.

Enfin, je remercie la Direction Générale d'Éducation Supérieure Technologique (DGEST) et l'Institut Technologique d'Apizaco pour avoir mis à ma disposition tous les moyens financiers pour le développement de cette thèse.

*Je dédie cette thèse à María Luisa. J'ai trouvé en toi une fée qui illumine mon cœur
Ainsi à Iván David, le plus précieux dans la vie.
José Crispín*

Sommaire

Introduction générale	1
1 Classification des données de puces à ADN	5
1.1 Technologie des puces à ADN	6
1.2 Jeux de données utilisés dans cette thèse	9
1.2.1 Leucémie	9
1.2.2 Cancer du colon	10
1.2.3 Tumeurs du cerveau	11
1.2.4 Lymphome	11
1.2.5 Cancer du Poumon	12
1.2.6 Cancer ovarien	12
1.2.7 Cancer du sein	12
1.2.8 Cancer de la prostate	12
1.3 La classification supervisée	13
1.3.1 Notions de classification	13
1.3.2 Le problème de la généralisation	14
1.3.3 Évaluation d'une hypothèse de classification	15
1.4 Méthodes de classification supervisée	16
1.4.1 Méthodes à noyau et SVM	16
1.4.2 Classifieur bayésien naïf	16
1.4.3 k-PPV	17
1.4.4 Réseaux de neurones	18
1.4.5 Arbres de décision	18
1.4.6 Bagging	18
1.4.7 Boosting	19
2 Sélection d'attributs	21
2.1 Présentation informelle du problème	22
2.2 Notions de pertinence, non pertinence et redondance	23
2.2.1 Pertinence d'attributs	23
2.2.2 Redondance d'attributs	24
2.3 Sélection d'attributs	26
2.3.1 La sélection vue comme un problème d'optimisation	26
2.3.2 Schéma général de la sélection d'attributs	27

2.3.3	Génération de sous-ensembles et procédures de recherche	29
2.4	Les approches pour la sélection d'attributs	30
2.4.1	Méthodes de Filtres	30
2.4.2	Méthodes Enveloppées	31
2.4.3	Méthodes Intégrées	32
2.5	Validation des méthodes	32
3	Recherche locale pour la sélection de gènes	35
3.1	SVMs	37
3.1.1	Classifieur SVM linéaire	37
3.1.2	Coefficients de classement d'un SVM	41
3.2	Recherche locale	42
3.2.1	Méthode de descente	43
3.2.2	Recherche tabou	44
3.2.3	Recherche locale itérée	45
3.3	Application de la recherche locale au problème de la sélection de gènes . .	47
3.3.1	Espace de recherche et représentation	47
3.3.2	Fonction d'évaluation	48
3.3.3	Mouvement et voisinage	48
3.3.4	Solution initiale	49
3.3.5	Schéma général	49
3.4	Comparaison d'algorithmes de recherche locale	50
3.4.1	Conditions expérimentales	50
3.4.2	Stratégie de descente spécifique	51
3.4.3	Recherche Tabou spécifique	52
3.4.4	Recherche locale itérée spécifique	54
3.5	Comparaison avec d'autres approches de sélection de gènes	57
3.5.1	Comparaison avec SVM-RFE	57
3.5.2	Comparaison avec d'autres approches de l'état d'art	58
3.6	Conclusion	58
4	Algorithme génétique spécifique pour la sélection de gènes	61
4.1	Algorithmes génétiques	63
4.1.1	Représentation des solutions	63
4.1.2	Fonction d'évaluation	63
4.1.3	Opérateurs génétiques	64
4.1.4	Phase de remplacement	68
4.1.5	Schéma général d'un algorithme génétique	68
4.2	Sélection de gènes par algorithme génétique	69
4.2.1	Représentation	69
4.2.2	Fonction d'évaluation	70
4.2.3	Opérateur de croisement spécifique	70
4.2.4	Opérateur de mutation spécifique	73
4.2.5	Initialisation	74

4.2.6	Sélection et Remplacement	75
4.2.7	Résultats comparatifs	75
4.2.8	Comparaison des opérateurs de croisement	75
4.2.9	Comparaison avec d'autres approches avec un nombre de gènes fixe	77
4.2.10	Comparaison avec d'autres approches	77
4.3	Conclusion	80
5	Un algorithme hybride pour la sélection de gènes	83
5.1	Algorithmes mémétiques	85
5.2	Hybridation entre l'algorithme génétique et la recherche locale itérée pour la sélection de gènes	87
5.2.1	Représentation	88
5.2.2	Fonction d'aptitude	88
5.2.3	Espace de recherche	88
5.2.4	Initialisation	89
5.2.5	Sélection	89
5.2.6	Opérateur de croisement	89
5.2.7	Opérateur de recherche locale	89
5.2.8	Nouvelle population	89
5.2.9	Procédure générale	89
5.2.10	Conditions expérimentales	90
5.3	Résultats Expérimentaux	90
5.3.1	Comparaison de MASEL avec nos autres modèles	91
5.3.2	Comparaison avec d'autres méthodes de sélection	91
5.4	Conclusion	92
	Conclusion générale	95
	Index	99
	Références bibliographiques	101
	Liste des publications personnelles	115
	Résumé / Abstract	118

Liste des figures

1.1	Etapes d'une analyse par puces à ADN.	6
1.2	Processus d'acquisition du image.	8
2.1	Catégorisation d'attributs.	25
2.2	Processus de la sélection d'attributs.	28
2.3	Schéma de validation croisée pour l'évaluation d'un processus de sélection-classification.	34
3.1	Hyperplan optimal de séparation	38
3.2	Hyperplans séparateurs dans le cas des exemples non linéairement séparables	40
3.3	Recherche locale pour la sélection de gènes des données de puces à ADN.	51
3.4	Méthode de la meilleure amélioration	52
3.5	Exemple de fonctionnement de la liste tabou.	53
3.6	Exemple de l'ensemble des mouvements possibles considérés.	53
3.7	solution optimum locale.	54
3.8	Parcours de $s - 1\ 2\ 6\ 8$ à $s - 1\ 2\ 7\ 8$	55
3.9	Exemple d'une perturbation.	55
3.10	Parcours de $s - 1\ 2\ 6\ 8$ à $s - 1\ 2\ 8\ 9$	56
4.1	Éléments d'un algorithme génétique	64
4.2	Illustration des opérateurs de croisement.	68
4.3	Exemple d'une mutation typique.	68
4.4	Un individu et ses composants.	70
4.5	Une paire des individus sélectionnés.	72
4.6	Résultat obtenu à partir de l'étape 1.	72
4.7	Processus pour obtenir les valeurs de AI_i et de AJ_i respectivement.	73
4.8	Obtention du fils K^x	73
4.9	Un individu sélectionné I.	74
4.10	Obtention de K^x	74
4.11	Comparaison de performance entre trois opérateurs de croisement sur trois jeux de données.	78
5.1	Comportement schématique d'un algorithme génétique, d'une recherche locale et d'un algorithme mémétique.	85

5.2	Schéma des algorithmes mémétiques.	86
5.3	Modèle de sélection de gènes par l'algorithme mémétique avec validation croisée.	88

Liste des tables

1.1	Matrice d'expression des gènes.	9
3.1	Comparaison entre les algorithmes de recherche locale. Taux de classification.	57
3.2	Comparaison entre les algorithmes de recherche locale. Nombre de gènes sélectionnés.	57
3.3	Résultats de l'algorithme SVM-RFE	58
3.4	Comparaison avec d'autres méthodes de référence	59
4.1	Comparaison entre quatre approches de sélection et la notre. Le tableau contient : d'une part, le nombre de gènes sélectionnés et le taux de classification rapporté par chaque auteur (<i>Rapporté</i>) et, d'autre part, le taux de classification obtenu par notre approche (<i>GeSeX</i>) en respectant la valeur du nombre de gènes rapporté dans le papier correspondant.	79
4.2	Comparaison avec les méthodes de référence	80
5.1	Évolution de nos résultats. TC indique le taux de classification et NG indique le nombre de gènes sélectionnés (tous les deux en montrant la moyenne et l'écart-type).	91
5.2	Comparaison avec les méthodes de référence	93

Liste des algorithmes

3.1	Méthode de la première amélioration	44
3.2	Méthode de la meilleure amélioration	44
3.3	Méthode de recherche tabou	45
3.4	Méthode de recherche locale itérée	46
3.5	Procédure générale <i>SdG</i>	50
4.1	Schéma d'un algorithme génétique	69
5.1	Procédure générale d'un algorithme mémétique	90

Introduction générale

Contexte de travail

Le domaine de la bioinformatique suscite depuis déjà plusieurs années un intérêt très grand dans la communauté scientifique car il ouvre des perspectives très riches pour la compréhension des phénomènes biologiques. Les problèmes abordés concernent par exemple le séquençage du génome, la modélisation de la structure des protéines, ou la reconstruction d'arbres phylogénétiques (phylogénie). Ces problèmes nécessitent la collaboration entre biologistes et informaticiens car les problèmes à traiter posent souvent de grandes difficultés algorithmiques.

Notre laboratoire est depuis quelques années intéressé à des problèmes de bioinformatique qui s'expriment comme des problèmes d'optimisation. Pour l'alignement de séquences ou la reconstruction d'arbres phylogénétiques les méthodes métaheuristiques se sont révélées très efficaces.

Dans cette thèse nous abordons un problème de bioinformatique qui est celui de la classification de données de biopuces. La technologie des puces à ADN permet de différencier des tissus tumoraux et des tissus sains à partir de la mesure simultanée d'un grand nombre de gènes au sein d'un échantillon biologique. Pour cette tâche de classification, on dispose d'un faible nombre d'échantillons alors que chaque échantillon est décrit par un très grand nombre de gènes.

Le traitement de ces données nécessite donc de réduire le nombre de gènes pour proposer un sous-ensemble de gènes pertinents et de construire un classifieur prédisant le type de tumeur qui caractérise un échantillon cellulaire. Il s'agit d'un problème de sélection d'attributs

La sélection d'attributs est un problème complexe qui a déjà été largement étudié, mais les dimensions des données des biopuces nécessitent des approches spécifiques (plusieurs milliers de gènes).

Dans cette thèse nous nous intéressons au développement d'algorithmes métaheuristiques spécialisés pour la sélection de gènes. Les méthodes que nous proposons utilisent des informations spécifiques au problème de la sélection et de la classification pour proposer des solutions efficaces.

Organisation de la thèse

Cette thèse est composée de cinq chapitres dont nous présentons une brève description dans les paragraphes suivants :

- Le **Chapitre 1** aborde les concepts et les principes de base de la technologie de puces à ADN. En particulier, on traite les différentes étapes de l'analyse de puces ADN. Dans ce chapitre, on s'intéresse également à la classification supervisée, car le diagnostic à partir des données d'expression est un problème de classification supervisée. Ensuite, les méthodes de classification supervisée, les plus utilisées pour classer les puces à ADN, sont décrites brièvement. Enfin, une description de chaque jeu de données utilisé dans cette thèse est détaillée.
- Le **Chapitre 2** introduit de façon informelle le problème de sélection d'attributs et présente ses concepts de base. Les éléments clés d'un processus de sélection sont définis en détail. Ensuite, les principales approches de sélection, comme les méthodes de filtrage, les méthodes d'enveloppe et les méthodes intégrées, sont décrites. Enfin, nous présentons quels protocoles expérimentaux doivent être réalisés pour l'évaluation d'une méthode de sélection afin d'éviter le "biais de sélection" qui est apparu dans les premières publications de ce domaine.
- Le **Chapitre 3** est consacré aux méthodes de recherche locale que nous proposons d'appliquer pour la sélection de gènes. Au début du chapitre, nous rappelons le principe de base d'un classifieur SVM linéaire et les informations utiles qu'apporte ce classifieur pour la sélection d'attributs. Cette information sera utilisée par tous les modèles de sélection de gènes proposés. De plus, les concepts d'optimisation combinatoire et de recherche locale sont abordés, et les principes généraux des méthodes de recherche locale que nous utilisons dans ce chapitre sont brièvement décrits. La suite du chapitre présente les méthodes de recherche locale proposées pour faire la sélection de gènes et une étude détaillée des différentes stratégies. Enfin, deux études comparatives avec d'autres algorithmes de l'état de l'art sont montrées.
- Le **Chapitre 4** présente un algorithme génétique pour la sélection de gènes où nous proposons des opérateurs bien adaptés pour la sélection. Nous proposons notamment un opérateur de croisement spécifique qui utilise des informations fournies par le classifieur SVM pour guider l'exploration génétique. Au début de ce chapitre, nous exposons le principe de base d'un algorithme génétique. Ensuite, l'adaptation de cet algorithme au problème de sélection de gènes est présentée, et illustrée par un exemple. Par ailleurs, une étude comparative est réalisée entre deux opérateurs de croisement standard et notre opérateur de croisement spécifique. Finalement, notre modèle génétique est comparé avec d'autres travaux du domaine.
- Le **Chapitre 5** est consacré à la présentation de notre modèle hybride MASEL pour la sélection de gènes. Nous combinons notre meilleure méthode de recherche locale avec notre opérateur de croisement spécifique. Dans ce chapitre nous abor-

dons d'abord, les éléments de base d'un algorithme mémétique. Après avoir décrit dans les chapitres précédents l'aspect génétique incluant l'opérateur de croisement original et les méthodes de recherche locale et ses composants, nous présenterons l'algorithme mémétique en tant que tel ainsi que des résultats expérimentaux comparés aux autres travaux de l'état de l'art.

La thèse se termine sur une synthèse de nos différentes contributions, et donne quelques perspectives de recherche.

Chapitre 1

Classification des données de puces à ADN

Sommaire

1.1	Technologie des puces à ADN	6
1.2	Jeux de données utilisés dans cette thèse	9
1.2.1	Leucémie	9
1.2.2	Cancer du colon	10
1.2.3	Tumeurs du cerveau	11
1.2.4	Lymphome	11
1.2.5	Cancer du Poumon	12
1.2.6	Cancer ovarien	12
1.2.7	Cancer du sein	12
1.2.8	Cancer de la prostate	12
1.3	La classification supervisée	13
1.3.1	Notions de classification	13
1.3.2	Le problème de la généralisation	14
1.3.3	Évaluation d'une hypothèse de classification	15
1.4	Méthodes de classification supervisée	16
1.4.1	Méthodes à noyau et SVM	16
1.4.2	Classifieur bayésien naïf	16
1.4.3	k-PPV	17
1.4.4	Réseaux de neurones	18
1.4.5	Arbres de décision	18
1.4.6	Bagging	18
1.4.7	Boosting	19

1.1 Technologie des puces à ADN

La technologie des puces à ADN ou biopuces, connaît à l'heure actuelle un essor exceptionnel et suscite un formidable intérêt dans la communauté scientifique. Cette technologie a été développée au début des années 1990 et permet la mesure simultanée des niveaux d'expression de plusieurs milliers de gènes, voire d'un génome entier, dans des dizaines de conditions différentes, physiologiques ou pathologiques. L'utilité de ces informations est scientifiquement incontestable car la connaissance du niveau d'expression d'un gène dans ces différentes situations constitue une avancée vers sa fonction, mais également vers le criblage de nouvelles molécules et l'identification de nouveaux médicaments et de nouveaux outils de diagnostic. [Lander, 1999; Speed, 2000; Baldi et Brunak, 2001; Godoy, 2001; Grody, 2001; Dudoit *et al.*, 2002; Dubitzky *et al.*, 2003; Stekel, 2003; Knudsen, 2004; Schoemaker et Lin, 2005].

Le fonctionnement des puces à ADN repose sur le principe de complémentarité des brins de la double hélice d'ADN et la propriété d'hybridation entre deux séquences complémentaires d'acides nucléiques [Lander, 1999; Southern, 2001a; Soularue et Gidrol, 2002; Dubitzky *et al.*, 2003]: une séquence d'ADN ou d'ARN peut donc servir de **sonde** pour capturer son complémentaire (**cible**) dans un mélange d'acides nucléiques

Une puce ADN (appelée *DNA microarray* en anglais) est constituée de fragments d'ADN immobilisés sur un support solide, de manière ordonnée. Chaque emplacement de séquence est soigneusement repéré: la position (x_i, y_i) correspond au gène i . Un emplacement est souvent appelé *spot* ou *sonde*. L'hybridation de la puce avec un échantillon biologique qui a été marqué par une substance radioactive ou fluorescente permet de quantifier l'ensemble des *cibles* qu'il contient; l'intensité du signal émis est proportionnel à la quantité de gènes cibles qu'il contient.

Les différentes phases d'une analyse par puces ADN sont indiquées dans la figure 1.1.



Figure 1.1 – Etapes d'une analyse par puces à ADN.

La préparation des cibles et l'hybridation : pour comparer les niveaux d'expression dans deux échantillons biologiques ou deux conditions (référence et pathologique), la première étape consiste en la préparation du génome exprimé dans ces deux échantillons. Il s'agit d'extraire les ARNm d'un échantillon biologique à analyser et la qualité de l'extraction est bien sûr primordiale pour la réussite de l'hybridation qui va suivre. Une mauvaise purification peut conduire à une augmentation des bruits de fond sur la lame.

La deuxième étape consiste à marquer les deux échantillons pour ensuite les hybrider en utilisant un four et à les nettoyer en utilisant une station de lavage. Les échantillons

sont marqués par des substances fluorescentes (Cy3 et Cy5), c'est-à-dire qu'une culture est marquée avec un fluorochrome vert, tandis que la seconde est marquée avec un fluorochrome rouge.

L'hybridation est ensuite réalisée sur une seule puce (simple marquage) ou sur deux puces (double marquage : un échantillon sur chaque puce). Les ADN marqués sont mélangés (cible) et placés sur la puce à ADN (sonde). Ce processus d'hybridation est réalisé dans une station fluïdique (four) pour favoriser les liaisons entre séquences complémentaires [Southern, 2001b; Soularue et Gidrol, 2002; Stekel, 2003]. La durée oscille entre 10 à 17 heures en milieu liquide à 60 degrés, en fait à cette température un fragment d'ADN simple brin ou d'ARN messenger reconnaît son brin complémentaire (ADNc) parmi des milliers d'autres pour former un ADN de double brin (duplex ou double hélice).

L'étape de nettoyage ou lavage des puces a pour but d'ôter de la puce des cibles non hybridées. La puce est lavée à plusieurs reprises afin qu'il ne reste sur la lame que les brins parfaitement appariés.

Acquisition et analyse des images : suite à l'hybridation, une étape de lecture de la puce permet de repérer les sondes ayant réagi avec l'échantillon testé. Cette lecture est une étape clé [Southern, 2001a]. En effet, sa qualité conditionne de façon importante la précision des données et donc, la pertinence des interprétations. L'obtention des images est réalisée par lecture des puces sur des scanners de haute précision, adaptés aux marqueurs utilisés. Le procédé de détection combine deux lasers, pour exciter les fluorochromes *Cy3* et *Cy5*. On obtient alors deux images dont le niveau de gris représente l'intensité de la fluorescence lue. Si on remplace les niveaux de gris par des niveaux de vert pour la première image et des niveaux de rouge pour la seconde, on obtient en les superposant une image en fausses couleurs composée de spots allant du vert au rouge quand un des fluorophores domine, en passant par le jaune (même intensité pour les deux fluorophores). Le noir symbolise l'absence de signal. L'intensité du signal de fluorescence pour chaque couple (gène,spot) est proportionnel à l'intensité d'hybridation donc à l'expression du gène ciblé (voir figure 1.2). Les images sont traitées par des logiciels d'analyse qui permettent de mesurer la fluorescence de chaque spot sur la lame (estimant les niveaux d'expression pour chacun des gènes présents sur la puce), mais aussi de relier chaque sonde à l'annotation correspondante (nom de gène, numéro de l'ADNc utilisé, séquence de l'oligonucléotide, etc.). Ainsi, pour chaque spot, l'intensité de chaque marqueur est calculée puis comparée au bruit de fond.

Transformation des données : les rapports des intensités de fluorescences en rouge et vert sont généralement utilisés pour mesurer une variation d'expression d'un gène entre deux conditions (référence et pathologique, par exemple). Les données d'intensité sont rarement manipulées sans transformation et la transformation la plus couramment employée est celle qui utilise le logarithme à base deux. Il existe plusieurs raisons pour justifier cette transformation. D'une part, la variation du logarithme des intensités est moins dépendante de la grandeur des intensités, et d'autre part, cette transformation permet de se rapprocher d'une distribution symétrique et d'obtenir une meilleure dispersion avec moins de valeurs extrêmes, dans [Dudoit *et al.*, 2002] les auteurs utilisent cette transformation.

La normalisation consiste à ajuster l'intensité globale des images acquises sur chacun

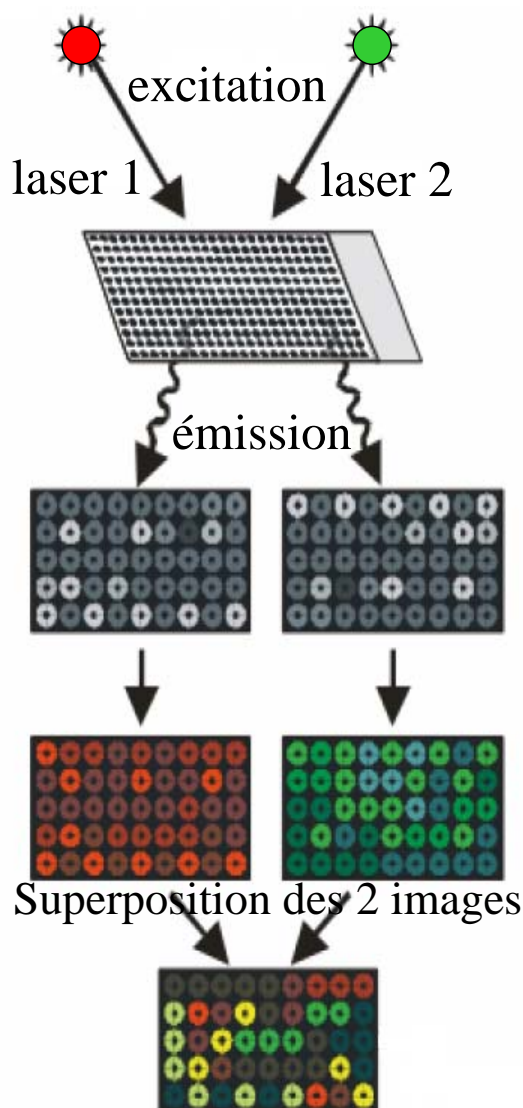


Figure 1.2 – Processus d’acquisition du image.

des deux canaux rouge et vert, de manière à corriger les différences systématiques entre les échantillons sur la même lame, qui ne représentent pas de variations biologiques entre les échantillons et qui tendent à déséquilibrer le signal de l’un des canaux par rapport à l’autre. Cette procédure de normalisation est définie par les gènes de référence. Les gènes de référence en moyenne ne doivent pas changer d’expression entre deux conditions.

La normalisation est effectuée à partir de toutes les sondes présentes sur le support pour éliminer les différences entre les différentes puces liées aux variations de quantité de départ, aux biais de marquage ou d’hybridation et aux variations du bruit de fond [Southern, 2001a; Yang *et al.*, 2002; Stekel, 2003; Jiang *et al.*, 2008; Stafford, 2008].

1.2 Jeux de données utilisés dans cette thèse

Présentation des données de puces à ADN : après les transformations décrites ci-dessus, les données recueillies pour l'étude d'un problème donné sont regroupées sous forme de matrice avec une ligne par couple (gène, sonde) et une colonne par échantillon (voir table 1.1). Chaque valeur de m_{ij} est la mesure du niveau d'expression du i -ième gène dans le j -ième échantillon, où $i = 1, \dots, M$ et $j = 1, \dots, N$ [Dudoit *et al.*, 2002; Dubitzky *et al.*, 2003].

$G\grave{e}ne_{id}$	$\acute{E}chantillon_1$	$\acute{E}chantillon_2$	\dots	$\acute{E}chantillon_M$
$G\grave{e}ne_1$	m_{11}	m_{12}	\dots	m_{1N}
$G\grave{e}ne_2$	m_{21}	m_{22}	\dots	m_{2N}
$G\grave{e}ne_3$	m_{31}	m_{32}	\dots	m_{3N}
\vdots	\vdots	\vdots	\ddots	\vdots
$G\grave{e}ne_N$	m_{M1}	m_{M2}	\dots	m_{MN}

Table 1.1 – Matrice d'expression des gènes.

Dans les étapes que l'on vient de voir, plusieurs d'entre elles peuvent être source d'imprécision ou d'erreurs dans les mesures obtenues. De plus, le coût d'une puce à ADN et le coût d'une analyse étant très élevé l'on ne dispose à l'heure actuelle que de quelques dizaines d'expériences pour l'étude d'un problème donné (une pathologie par exemple). Pourtant chaque expérience a permis de relever le niveau d'expression pour plusieurs milliers de gènes. Les matrices de données qui sont actuellement disponibles ont donc les caractéristiques suivantes:

1. Grande dimensionnalité due au nombre élevé de descripteurs (gènes)
2. Nombre limité d'échantillons

1.2 Jeux de données utilisés dans cette thèse

Dans la suite de ce document, nous allons présenter des méthodes de sélection d'attributs que nous avons développées sur un certain nombre de jeux de données. Nous avons utilisé des jeux de données publics, facilement accessibles et qui sont utilisés dans de nombreux travaux concernant la classification des données de puces à ADN. Ces jeux constituent en quelque sorte des jeux tests qui permettent de comparer les méthodes proposées depuis quelques années dans le domaine de la bioinformatique. Nous donnons ci-dessous les caractéristiques de ces jeux de données qui concernent tous des problèmes de reconnaissance de cancers et ou de prévision de diagnostic en oncologie.

1.2.1 Leucémie

Le jeu de données appelé dans la suite "données de la leucémie" ou "leucémie" est constitué de 72 échantillons représentant deux types de leucémie aigüe. 47 tissus sont du type Leucémie lymphoblastique aigüe (ALL) et 25 sont du type Leucémie myéloïde aigüe

(AML). Pour chaque échantillon, on a relevé les niveaux d'expression de 7129 gènes. Une description plus complète de ce jeu de données peut être trouvée dans l'article [Golub *et al.*, 1999] et les données peuvent être téléchargées à partir de :

<http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi>

sous la rubrique "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression".

L'article [Golub *et al.*, 1999], publié en 1999, a été le premier à démontrer la possibilité d'un diagnostic à partir de données de puces à ADN et a donc été à l'origine d'un intérêt particulier pour cette problématique de la classification à partir des données de biopuces. C'est pourquoi nous résumons ici les principes de cette étude.

Dans cette recherche, Golub et ses collègues ont d'abord considéré 38 échantillons de moëlle osseuse pour 27 patients atteints de la forme ALL et 11 patients atteints de la forme AML de la leucémie. La distinction entre ces deux formes de la maladie est bien connue mais nécessite un ensemble de tests de hauts niveaux (morphologie de la tumeur, histologie, ...) qui doivent être réalisés dans différents laboratoires spécialisés. L'objectif de ce travail était de voir si les seules données d'expression de gènes pouvaient conduire à une classification précise des leucémies. Pour cela, Golub et son équipe ont d'abord identifié un ensemble de gènes corrélés à la distinction de classe ALL/AML. Ils ont retenu les 50 gènes les plus pertinents pour construire un système de prédiction permettant d'affecter un nouvel échantillon à la classe ALL ou AML. Le classifieur proposé était en fait la combinaison de 50 votes, chaque gène votant, d'après son niveau d'expression, pour l'une ou l'autre classe. Ce classifieur a été appliqué à un jeu de test de 34 échantillons issus de prélèvement de moëlle osseuse ou de sang. Pour 29 de ces 34 cas, le classifieur a proposé des prédictions "fortes" (convergence des votes des différents gènes) et ces 29 prédictions étaient correctes. Cette étude a donc montré que l'information issue des puces à ADN était suffisante pour construire un système de diagnostic précis, et que de plus un petit nombre de gènes suffisait à construire un prédicteur fiable.

Signalons enfin que généralement, avant toute analyse statistique, les données "brutes" de ce jeu de données sont pré-traitées selon un protocole comportant une étape de seuillage, de filtrage et de transformation logarithmique [Dudoit *et al.*, 2002]. Le seuillage consiste à ne conserver que les valeurs comprises entre 100 et 16000. Le filtrage a pour but d'éliminer avant tout traitement les gènes dont l'expression est trop uniforme et dont la variation n'est pas significative par rapport à la de mesure des niveaux d'expression. Pour chaque gène, on relève la valeur minimale s_{min} et la valeur maximale s_{max} du niveau d'expression parmi les tissus disponibles et on ne garde que les gènes tels que $s_{max}/s_{min} > 5$ et $s_{max}-s_{min} > 500$. Et on effectue une transformation logarithmique en base 10 des données.

1.2.2 Cancer du colon

Ce jeu de données [Alon *et al.*, 1999], qui concerne le cancer du colon, est constitué de 62 échantillons dont 40 sont des tissus tumoraux et 22 sont des tissus sains ou normaux. Les expériences ont été menées avec des puces relevant les valeurs d'expression

1.2 Jeux de données utilisés dans cette thèse

pour plus de 6500 gènes humains mais seuls les 2000 gènes ayant les plus fortes intensités minimales ont été retenus. La matrice des niveaux d'expression comporte donc 2000 colonnes et 62 lignes. Dans le papier [Alon *et al.*, 1999], les auteurs utilisaient un jeu d'apprentissage de 40 échantillons (26 tumeurs, 14 normaux) et un jeu de test de 22 cas (14 tumeurs, 8 normaux). Ce jeu de données peut être téléchargé à l'adresse:

<http://microarray.princeton.edu/oncology/affydata/index.html>

1.2.3 Tumeurs du cerveau

Cette base de données relative aux tumeurs embryonnaires du système nerveux central ou CNS a été utilisée par Pomeroy et al [Pomeroy *et al.*, 2002]. Ce jeu contient les échantillons obtenus de 60 patients qui ont été traités pour des médulloblastomes. Parmi ces 60 patients, 39 sont survivants et pour les 21 autres le traitement est un échec. Chaque échantillon est décrit par 7129 gènes. Le jeu de données est accessible à :

<http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi>

sous la rubrique "Gene Expression-Based Classification and Outcome Prediction of Central Nervous System Embryonal Tumors"

1.2.4 Lymphome

Ce jeu de données provient du site <http://llmpp.nih.gov/lymphoma/data.shtml>, associé à la publication de Alizadeh et al. [Alizadeh *et al.*, 2000]. Il contient les informations médicales de 96 échantillons de lymphocytes relatifs à différents types de lymphome. Dans la littérature, on trouve des études regroupant les échantillons de deux manières différentes.

1. 62 échantillons de lymphocytes malins (dont 42 proviennent de lymphomes diffus à grandes cellules B ou DLBCL, 9 sont des lymphomes folliculaires ou FL et 11 sont des lymphomes diffus lymphocytiques de type leucémie lymphoïde chronique ou CLL) et 34 échantillons de lymphocytes normaux. Ce groupement a été proposé par Weston et al. et également utilisé par Rakotomamonjy [Weston *et al.*, 2002; Rakotomamonjy, 2003]. Il contient donc 96 échantillons et sera noté Lymphome⁹⁶ dans la suite. Il est disponible sur le site :
<http://www.kyb.tuebingen.mpg.de/bs/people/weston/10/>
2. 24 échantillons de DLBCL de cellules de centre germinatif (*GC B-like*) et 23 échantillons de DLBCL activés (*activated B-like*). Ce groupement a été identifié par Alizadeh et al. [Alizadeh *et al.*, 2000]. Il contient donc 47 échantillons et sera noté Lymphome⁴⁷ dans la suite.

Ce jeu contient des données manquantes. Pour le traiter, nous avons remplacé les valeurs manquantes selon la méthode suggérée par [Troyanskaya *et al.*, 2001].

1.2.5 Cancer du Poumon

Le jeu de données concernant le cancer du poumon, il a été traité par [Gordon *et al.*, 2002]. Ce jeu décrit deux types de pathologie du cancer du poumon: le cancer de type adénocarcinome *ADCA* et le cancer du mésothéliome malin de la plèvre *MPM*. Le jeu contient 181 instances décrites pour 12533 gènes. Les instances sont généralement divisées en 16 *MPM* et 16 *ADCM* pour l'apprentissage, et 15 *MPM* et 134 *ADCA* pour le test.

<http://www-genome.wi.mit.edu/cgi-bin/cancer/datasets.cgi>

1.2.6 Cancer ovarien

Le jeu de données du cancer ovarien comporte 253 échantillons dont 91 dits normaux (sans maladie) et 162 avec cancer. Ces données ont été obtenues par spectrométrie à partir d'échantillons de sérum de femmes saines et de sérum de femmes atteintes d'un cancer de l'ovaire. Chaque spectre représente l'expression de 15154 peptides. Pour une description complète consulter [Petricoin *et al.*, 2002]. Lorsque nous avons téléchargé ce jeu de données, il était disponible à l'adresse suivante :

<http://research.i2r.a-star.edu.sg/rp/>

1.2.7 Cancer du sein

L'équipe de Van't Veer a abordé le problème du sur-traitement des patientes par la chimiothérapie adjuvante du cancer du sein [Veer *et al.*, 2002]. Il existe des patientes ayant une tumeur du sein sans envahissement ganglionnaire axillaire (N^-) et traitées par chimiothérapie, qui survivraient avec un traitement loco-régional seul. Cependant, l'imperfection des facteurs pronostiques actuels et le bénéfice apporté par cette chimiothérapie font qu'il faut sur-traiter de nombreuses patientes. L'objectif est de différencier, parmi les tumeurs N^- , celles qui ne vont pas rechuter de celles qui vont rechuter. La base d'apprentissage contient 78 échantillons de patientes, dont 34 avaient développé des métastases dans les 5 ans (rechute) et 44 étaient indemnes au-delà de 5 ans (non-rechute). La base de test contient 12 patientes avec rechute et 7 patientes avec non rechute. Le nombre de gènes utilisés dans cette étude étaient 24481. Nous remarquons qu'il existe 283 gènes avec bruit qui ont été enlevés. Ce jeu de données est disponible à l'adresse :

<http://homes.esat.kuleuven.be/~npochet/Bioinformatics/>

1.2.8 Cancer de la prostate

Dans ce jeu de données, le niveau d'expression de 12600 gènes est mesuré sur 102 tissus. L'objectif initial est de distinguer les tissus normaux (52) des tissus cancéreux (50). Pour une description complète de ce jeu de données consulter [Singh *et al.*, 2002a]. Ce jeu de données est disponible à l'adresse :

<http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi>

sous la rubrique “Gene Expression Correlates of Clinical Prostate Cancer Behavior”.

Nous venons de décrire les jeux de données sur lesquelles nous allons travailler. L’analyse de ces données a pour but de mettre en évidence l’importance de certains gènes dans les pathologies considérées et de construire un système de diagnostic s’appuyant uniquement sur les données de puces à ADN. Il s’agit là d’un problème de *classification supervisée* dont nous allons rappeler dans la section suivante les points utiles pour notre étude.

1.3 La classification supervisée

Le terme de classification peut désigner deux approches distinctes : la classification supervisée et la classification non-supervisée (automatic classification et clustering en anglais). Les méthodes non supervisées ont pour but de constituer des groupes d’exemples (ou des groupes d’attributs) en fonction des données observées, sans connaissance a priori. En revanche les méthodes supervisées utilisent la connaissance a priori sur l’appartenance d’un exemple à une classe pour construire un système de reconnaissance de ces classes.

L’objectif de la classification supervisée est d’apprendre à l’aide d’un ensemble d’entraînement une procédure de classification qui permet de prédire l’appartenance d’un nouvel exemple à une classe. Les systèmes d’apprentissage permettant d’obtenir une telle procédure peuvent être basés sur des hypothèses probabilistes (classifieur naïf de Bayes), sur des notions de proximité (plus proches voisins) ou sur des recherches dans des espaces d’hypothèses (arbres de décisions, ...) Nous décrivons dans la suite quelques méthodes de classification supervisée bien connues dans la littérature. Nous n’aborderons ici que les méthodes de classification utilisées dans les travaux concernant la classification des données de puces à ADN.

1.3.1 Notions de classification

Nous rappelons tout d’abord une définition du problème de classification supervisée ou reconnaissances de formes. L’objectif de la classification est d’identifier les classes auxquelles appartiennent des objets à partir de leurs caractéristiques ou attributs descriptifs. Cette section emprunte des notations et différents éléments aux livres [Cristianini et Shawe-Taylor, 2000; Cornuéjols et Miclet, 2002] auxquels nous invitons le lecteur à se reporter.

Dans le cadre de la classification supervisée, les classes sont connues et l’on dispose d’exemples de chaque classe.

Définition 1.1 (Exemple) *Un exemple est un couple (x, y) , où $x \in \mathcal{X}$ est la description ou la représentation de l’objet et $y \in \mathcal{Y}$ représente la supervision de x .*

Dans un problème de classification, \mathbf{y} s'appelle la classe de \mathbf{x} . Pour la classification binaire nous utilisons typiquement \mathcal{X} pour dénoter l'espace d'entrées tel que $\mathcal{X} \subseteq \mathbb{R}^p$ et \mathcal{Y} l'espace de sortie tel que $\mathcal{Y} = \{-1, 1\}$ ¹.

Définition 1.2 (Classification supervisée) Soit un ensemble d'exemples de n données étiquetées : $\mathcal{S} = \{(x_1, y_1), \dots, (x_n, y_n)\}$. Chaque donnée x_i est caractérisée par p attributs et par sa classe y_i . On cherche une hypothèse h telle que :

1. h satisfait les échantillons

$$\forall_i \in \{1, \dots, n\} h(x_i) = y_i$$

2. h possède de bonnes propriétés de généralisation.

Le problème de la classification consiste donc, en s'appuyant sur l'ensemble d'exemples à prédire la classe de toute nouvelle donnée $\mathbf{x} \in \mathbb{R}^p$.

1.3.2 Le problème de la généralisation

L'objectif de la classification est de fournir une procédure ayant un bon pouvoir prédictif c'est-à-dire garantissant des prédictions fiables sur les nouveaux exemples qui seront soumis au système. La qualité prédictive d'un modèle peut être évaluée par le risque réel ou espérance du risque, qui mesure la probabilité de mauvaise classification d'un hypothèse h .

Définition 1.3 (Risque réel) Soit h une hypothèse apprise à partir d'un échantillon \mathcal{S} d'exemples de $\mathcal{X} \times \mathcal{Y}$

$$R(h) = \int_{x \in \mathcal{X}, Y} l[h(x_i), y_i] dF(x, y) \quad (1.1)$$

où l est une fonction de perte ou de coût associé aux mauvaises classifications et où l'intégrale prend en compte la distribution F de l'ensemble des exemples sur le produit cartésien de $\mathcal{X} \times \mathcal{Y}$.

La fonction de perte la plus simple utilisée en classification est définie par :

$$l[h(x_i), y_i] = \begin{cases} 0 & \text{si } y_i = h(x_i), \\ 1 & \text{si } y_i \neq h(x_i), \end{cases}$$

La distribution des exemples est inconnue, ce qui rend impossible le calcul du risque réel. Le système d'apprentissage n'a en fait accès qu'à l'erreur apparente (ou empirique) qui est mesurée sur l'échantillon d'apprentissage.

Définition 1.4 (Risque empirique) Soit un ensemble d'apprentissage $\mathcal{S} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ de taille n et une hypothèse h . Le risque empirique de h calculé sur \mathcal{S} est défini par:

$$R_{emp} = \frac{1}{n} \sum_{i=1}^n l[h(x_i), y_i] \quad (1.2)$$

¹Parfois $\mathcal{Y} = \{+, -\}$, $\mathcal{Y} = \{1, 0\}$, $\mathcal{Y} = \{1, 2\}$

Avec la fonction de perte présentée ci-dessus, le risque empirique ou apparent est simplement le nombre d'exemples de \mathcal{S} qui sont mal classés.

On peut montrer que, lorsque la taille de l'échantillon tend vers l'infini, le risque apparent converge -en probabilité si les éléments de \mathcal{S} sont tirés aléatoirement- vers le risque réel. Malheureusement on ne dispose que d'un échantillon limité d'exemples; le risque empirique est très optimiste et n'est pas un bon indicateur des performances prédictives de l'hypothèse h .

1.3.3 Évaluation d'une hypothèse de classification

Pour avoir une estimation non optimiste de l'erreur de classification, il faut recourir à une base d'exemples qui n'ont pas servi pour l'apprentissage : il s'agit de la base de test. La base de test contient elle aussi des exemples étiquetés qui permettent de comparer les prédictions d'une hypothèse h avec la valeur réelle de la classe. Cette base de test est généralement obtenue en réservant une partie des exemples supervisés initiaux et en ne les utilisant pas pour la phase d'apprentissage. Lorsqu'on dispose de peu d'exemples, comme c'est le cas dans le traitement des données d'expression de gènes, il est pénalisant de laisser de côté une partie des exemples pendant la phase d'apprentissage. On peut alors utiliser le processus de *validation croisée* pour proposer une estimation du risque réel.

L'algorithme de validation croisée à k blocs (*k-fold cross-validation*) consiste à découper l'ensemble initial d'exemples \mathcal{D} en k blocs. On répète alors k phases d'apprentissage-évaluation où une hypothèse h est obtenue par apprentissage sur $k - 1$ blocs de données et testée sur le bloc restant. L'estimateur de l'erreur est obtenu comme la moyenne des k erreurs empiriques ainsi obtenues. L'algorithme est alors :

1. Partitionner l'ensemble d'exemples \mathcal{D} en k sous-ensembles disjoints: $\mathcal{D}_1, \dots, \mathcal{D}_k$
2. Pour tout i de 1 à k
 - Appliquer l'algorithme d'apprentissage sur le jeu d'apprentissage $\mathcal{D} - \mathcal{D}_i$ pour obtenir une hypothèse h_i
 - Calculer R_i l'erreur de h_i sur \mathcal{D}_i
3. Retourner $R = \frac{\sum_{1 \leq i \leq k} R_i}{k}$ comme estimation de l'erreur

Même s'il n'existe pas pour cela de justifications théoriques claires, l'usage montre que l'évaluation par validation croisée fournit de bons résultats pour $k=10$. Il faut noter que lorsque le nombre d'échantillons dont on dispose est limité on peut également appliquer le processus appelé *leave-one-out cross validation* où la validation croisée est appliquée avec $k = n$ le nombre d'échantillons.

Nous utiliserons la procédure de validation croisée dans la suite de cette thèse pour évaluer les classifieurs que nous mettrons en oeuvre sur les jeux de données présentés plus haut.

Nous préciserons à la fin du chapitre suivant le protocole d'expérimentation à respecter pour évaluer à la fois une méthode de sélection d'attributs et un classifieur.

1.4 Méthodes de classification supervisée

1.4.1 Méthodes à noyau et SVM

Les Séparateurs à Vaste Marge ou Support Vector Machines (SVMs) ont été introduits dès 1992 [Boser *et al.*, 1992; Cortes et Vapnik, 1995; Vapnik, 1998]. Les SVMs sont des classifieurs qui reposent sur deux idées clés : la notion de marge maximale et la notion de fonction noyau.

La première idée clé est la notion de marge maximale. On cherche l'hyperplan qui sépare les exemples positifs des exemples négatifs, en garantissant que la distance entre la frontière de séparation et les échantillons les plus proches (marge) soit maximale. Ces derniers sont appelés vecteurs supports. Ceci est fait en formulant le problème comme un problème d'optimisation quadratique, pour lequel il existe des algorithmes connus.

Afin de pouvoir traiter des cas où les données ne sont pas linéairement séparables, la deuxième idée clé des SVM est de transformer l'espace de représentation des données d'entrées en un espace de plus grande dimension appelée espace de caractéristiques (possiblement de dimension infinie), dans lequel il est probable qu'il existe une séparatrice linéaire. Ceci est réalisé grâce à une fonction noyau, qui doit respecter certaines conditions, et qui a l'avantage de ne pas nécessiter la connaissance explicite de la transformation à appliquer pour le changement d'espace. Les fonctions noyau permettent de transformer un produit scalaire dans un espace de grande dimension, ce qui est coûteux, en une simple évaluation ponctuelle d'une fonction. Pour plus détails voir notamment dans [Scholkopf et Smola, 2001; Cristianini et Shawe-Taylor, 2000; Burges, 1998]

nous reviendrons dans le chapitre 3 sur le classifieur SVM.

1.4.2 Classifieur bayésien naïf

Le classifieur bayésien naïf est une méthode qui trouve son fondement théorique dans le théorème de Bayes [Mitchell, 1997; Cornuéjols et Miclet, 2002; Wu *et al.*, 2008]. Il permet les inférences probabilistes et repose sur l'hypothèse que les solutions recherchées peuvent être trouvées à partir de distributions de probabilité dans les données et dans les hypothèses. Cette méthode permet de déterminer la classification d'un exemple quelconque spécifiée en termes d'attributs en supposant que les attributs $\{x_1, \dots, x_n\}$ de l'espace d'entrée \mathcal{X} sont indépendants les uns des autres et $y \in \mathcal{Y}$ tel que $\mathcal{Y} = \{0, 1\}$ pour la classification binaire. La règle de classification de Bayes s'écrit :

$$C_{NBayes}(x) = \operatorname{argmax}_{y \in \mathcal{Y}} P(x_1, \dots, x_n | y) \times P(y) \quad (1.3)$$

On peut remplacer $P(x_1, \dots, x_n)$ et $P(y)$ par des estimations faites sur l'ensemble d'échantillons \mathcal{S} . Pour toute classe y on estime $P(y)$ par la proportion d'éléments de la classe y dans \mathcal{S} . Étant donné que l'estimation des $P(x_1, \dots, x_n)$ n'est pas évidente car le nombre de descriptions possibles peut être grand, il faudrait un échantillon \mathcal{S} de taille trop importante pour pouvoir estimer correctement ces quantités. Pour cela on utilise l'hypothèse suivante : les valeurs des attributs sont indépendantes connaissant la classe.

Cette hypothèse permet d'utiliser l'égalité suivante :

$$P(x_1, \dots, x_n|y) = \prod_{y \in \mathcal{Y}} P(x_i|y) \quad (1.4)$$

Il suffit alors d'estimer, pour tout i et toute classe y , $P'(x_i|y)$ par la proportion d'éléments de classe y ayant la valeur x_i pour le i -ème attribut. Finalement, le classifieur naïf de Bayes associe à toute description x la classe :

$$C_{NBayes}(x) = \operatorname{argmax}_i \prod_i P'(x_1, \dots, x_n|y) \times P(y) \quad (1.5)$$

Ce classifieur est simple, facile à programmer, et souvent efficace. Mais il est très sensible à la présence d'attributs corrélés.

1.4.3 k-PPV

L'algorithme des k plus proches voisins (noté k-PPV) [Duda *et al.*, 2000; Bishop, 2006; Wu *et al.*, 2008] est une méthode basée sur la notion de proximité (voisinage) entre exemples et sur l'idée de raisonner à partir de cas similaires pour prendre une décision. Autrement dit des entrées x_i semblables devraient avoir des valeurs y_i semblables. Le principe est le suivant : on note x un nouvel exemple décrit par un vecteur de p attributs. On trouve alors, parmi l'ensemble d'exemples d'apprentissage, les k plus proches voisins de x et on associe à x la classe majoritaire parmi ses k voisins lui ressemblant le plus dans la base d'apprentissage. Cette méthode dépend donc des trois éléments suivants:

1. Le nombre de voisins retenus.
2. La mesure de distance entre exemple.
3. La combinaison des classes.

Le résultat dépend du réglage de ces paramètres. Pour le premier critère, on utilise généralement un nombre de voisins compris entre 1 et 7. Pour le deuxième paramètre, la méthode nécessite une métrique pour mesurer la proximité entre l'exemple à classer x et chacun des exemples de l'ensemble d'apprentissage. Lorsque les attributs sont numériques la distance euclidienne est généralement utilisée. Le troisième paramètre indique de quelle manière on combine les valeurs associées aux voisins pour obtenir la valeur associée à x . Pour la classification, la classe retenue pour x est la classe majoritaire chez ses voisins.

Les points positifs de cette méthode sont qu'elle ne pose aucune hypothèse sur la forme des classes à apprendre. La méthode est simple puisqu'il n'y a pas besoin d'apprentissage d'un modèle de classification et son pouvoir prédictif est souvent bon.

Mais la performance de cette méthode diminue lorsque la dimension augmente, puisque pour chaque nouvelle classification, il est nécessaire de calculer toutes les distances de x à chacun des exemples d'apprentissage. De plus, la performance dépend fortement de k , le nombre de voisins choisi et il est nécessaire d'avoir un grand nombre d'observations pour obtenir une bonne précision des résultats.

1.4.4 Réseaux de neurones

Un réseau de neurones est constitué d'un graphe pondéré orienté dont les noeuds symbolisent les neurones. Ces neurones possèdent une fonction d'activation qui permet d'influencer les autres neurones du réseau; les fonctions les plus souvent utilisées, sont la fonction signe ou la fonction sigmoïde.

Les connexions entre les neurones, que l'on nomme liens synaptiques, propagent l'activité des neurones avec une pondération caractéristique de la connexion. On appelle poids synaptique la pondération des liens synaptiques.

Les neurones peuvent être organisés de différentes manières, c'est ce qui définit l'architecture et le modèle du réseau. L'architecture la plus courante est celle dite du perceptron multicouches. L'interconnexion entre les unités de calcul élémentaires, qui sont les neurones, permet un calcul global complexe, qui en classification se traduit par des frontières de décision aux formes complexes.

La phase d'apprentissage ou d'entraînement du réseau consiste à régler les poids synaptiques grâce à l'ensemble d'apprentissage.

1.4.5 Arbres de décision

Les arbres de décision sont une des méthodes les plus connues en classification. Le principe des arbres de décision est de réaliser la classification d'un exemple par une suite de tests sur les attributs qui le décrivent. Concrètement, dans la représentation graphique d'un arbre,

1. Un noeud interne correspond à un test sur la valeur d'un attribut.
2. Une branche part d'un noeud et correspond à une ou plusieurs valeurs de ce test.
3. Une feuille est un noeud d'où ne part aucune branche et correspond à une classe.

Une règle de décision (de la forme si ... alors ...) est créée pour chaque chemin partant de la racine de l'arbre et parcourant les tests (en faisant des conjonctions) jusqu'à la feuille qui est l'étiquette de la classe. Les arbres de décision sont particulièrement appréciés car ils permettent une compréhension aisée, mais lors d'une tâche de classification relativement complexe l'utilisateur n'est plus capable d'explorer efficacement les résultats obtenus sous forme textuelle.

Les principaux algorithmes de construction d'un arbre de décision sont : C4.5 [Quinlan, 1993], et CART [Breimann *et al.*, 1984]. Dans ces algorithmes, à chaque étape de création d'un noeud, un critère de séparation entre les classes est utilisé pour décider quel attribut est le plus pertinent pour la classification

1.4.6 Bagging

Cette méthode a été proposée par L. Breiman [Breiman, 1996]. Le principe du bagging est d'entraîner un algorithme d'apprentissage sur plusieurs bases d'apprentissage obtenues par tirage avec remise (ou bootstrap) de m' exemples d'apprentissage dans l'échantillon d'apprentissage S . Pour chaque tirage b , une hypothèse h_b est obtenue. L'hypothèse

finale est basée sur les moyennes des hypothèses obtenues. Son avantage est qu'on améliore la performance des classifieurs instables en calculant la moyenne de leurs réponses. Ainsi, si les hypothèses h_b calculées pour chaque tirage b ont une variance importante, alors l'hypothèse finale aura une variance réduite. Un classifieur est dit instable si un petit changement dans les données d'apprentissage provoque un gros changement dans le comportement du classifieur. Le but du bagging est d'atténuer l'instabilité inhérente à certains classifieurs.

1.4.7 Boosting

Le boosting est une méthode d'agrégation de classifieurs faibles pour obtenir un classifieur performant [Schapire, 1990]. Son principe consiste à assigner un poids égal à chaque exemple d'apprentissage ($1/n$) (appelé pondération) où n est le nombre d'échantillons. Ce poids indique la difficulté de prédire la classe de cet exemple. L'algorithme s'exécute T fois construisant T classifieurs sur les exemples d'apprentissage pondérés. Chaque fois qu'on produit un classifieur on change les poids des nouveaux exemples utilisés pour le classifieur suivant, on augmente le poids de ceux dont la classe est mal prédite et on diminue le poids des autres. Ensuite, on calcule l'erreur (e) du modèle sur l'ensemble pondéré. Si e est égale à zéro ou e est supérieure à 0.5 on termine la construction du classifieur. L'idée est de forcer le nouveau classifieur à diminuer l'erreur attendue. Le classifieur final se forme en utilisant un schéma de vote.

Syntaxe Les différentes techniques de classification que nous venons de décrire ont été utilisées dans différents travaux concernant le diagnostic à partir de données de puces à ADN et notamment sur les jeux de données particuliers que nous avons décrits.

Chapitre 2

Sélection d'attributs

Sommaire

2.1	Présentation informelle du problème	22
2.2	Notions de pertinence, non pertinence et redondance	23
2.2.1	Pertinence d'attributs	23
2.2.2	Redondance d'attributs	24
2.3	Sélection d'attributs	26
2.3.1	La sélection vue comme un problème d'optimisation	26
2.3.2	Schéma général de la sélection d'attributs	27
2.3.3	Génération de sous-ensembles et procédures de recherche	29
2.4	Les approches pour la sélection d'attributs	30
2.4.1	Méthodes de Filtrés	30
2.4.2	Méthodes Enveloppes	31
2.4.3	Méthodes Intégrées	32
2.5	Validation des méthodes	32

LA sélection d'attributs est devenue un sujet de recherche très actif depuis une dizaine d'années dans les domaines de l'apprentissage artificiel [Jain et Zongker, 1997; Dash et Liu, 1997; Kohavi et John, 1997; Blum et Langley, 1997; Duch, 2006], de la fouille de données, du traitement d'images [Singh *et al.*, 2002b; Fukunaga, 1990], et de l'analyse de données en bioinformatique. Dans tous ces domaines, les applications nécessitent de traiter des données décrites par un très grand nombre d'attributs. Ainsi on peut avoir à traiter des pages web décrites par plusieurs centaines de descripteurs, des images décrites par plusieurs milliers de pixels ou des données en bioinformatique donnant les niveaux d'expression de plusieurs milliers de gènes [Guyon et Elisseeff, 2003; Dai *et al.*, 2006]. [Guyon et Elisseeff, 2003; Dai *et al.*, 2006].

Dans ce chapitre, nous présentons d'abord de manière informelle le problème de la sélection d'attributs pour situer le travail et l'intérêt de la thèse. Dans la section 1, nous abordons les notions de pertinence, non-pertinence et redondance autour desquelles s'articulent la sélection d'attributs. Dans la section 2, nous présentons différents points de vue du processus de sélection d'attributs et la section 3 rappelle les différents types de méthodes pour la sélection d'attributs. Enfin dans la section 4, nous présentons le protocole expérimental qui doit être utilisé pour obtenir une estimation non biaisée d'une méthode de sélection.

2.1 Présentation informelle du problème

La sélection d'attributs consiste à choisir parmi un ensemble d'attributs de grande taille un sous-ensemble d'attributs intéressants pour le problème étudié. Cette problématique peut concerner différentes tâches d'apprentissage ou de fouille de données mais nous parlerons seulement ici de la sélection d'attributs réalisée pour la classification supervisée. Dans ce contexte, les principales motivations de la sélection d'attributs sont les suivantes [Jain et Zongker, 1997; Yang et Honovar, 1998] :

1. Utiliser un sous-ensemble plus petit permet d'améliorer la classification si l'on élimine les attributs qui sont source de bruit. Cela permet aussi une meilleure compréhension des phénomènes étudiés.
2. Des petits sous-ensembles d'attributs permettent une meilleure généralisation des données en évitant le sur-apprentissage.
3. Une fois que les meilleurs attributs sont identifiés, les temps d'apprentissage et d'exécution sont réduits et en conséquence l'apprentissage est moins coûteux.

En présence de centaines voire de milliers d'attributs il y a beaucoup de chances pour que des attributs soient corrélés et expriment des informations similaires, on dira alors qu'ils sont redondants. D'un autre côté, les attributs qui fournissent le plus d'information pour la classification seront dits pertinents. L'objectif de la sélection est donc de trouver un sous-ensemble optimal d'attributs qui ait les propriétés suivantes : il doit être composé d'attributs pertinents et il doit chercher à éviter les attributs redondants. De plus cet ensemble doit permettre de satisfaire au mieux l'objectif fixé c'est-à-dire la précision

de l'apprentissage, la rapidité de l'apprentissage ou bien encore l'explicabilité du classifieur proposé [Dash et Liu, 1997; Kohavi et John, 1997; Blum et Langley, 1997; Guyon *et al.*, 2002; Yu et Liu, 2004b; Yu et Liu, 2004a; Molina *et al.*, 2002b; Molina *et al.*, 2002a; Liu et Motoda, 2008a].

On peut illustrer cette problématique par le petit exemple suivant: [Liu et Yu, 2005]
Exemple : Considérons un problème de classification, où les descripteurs sont 5 attributs booléens F_1, \dots, F_5 . Supposons que la fonction de classification s'écrit $C = g(F_1, F_2)$ où g est une fonction booléenne. De plus, nous supposons qu'il existe les relations suivantes entre les attributs: $F_2 = \bar{F}_3$ et $F_4 = \bar{F}_5$ (notre problème concerne donc huit exemples possibles). Afin de déterminer la fonction cible, F_1 est indispensable; un des attributs F_2 ou F_3 peut être enlevé, car on remarque que C peut également s'écrire $g(F_1, \bar{F}_3)$, mais un des attributs F_2 ou F_3 doit être utilisé pour définir C . Quant à F_4 et F_5 , ils peuvent être éliminés car ils ne sont pas utiles pour définir C . On peut donc sélectionner pour ce problème deux sous-ensembles d'attributs optimaux: $\{F_1, F_2\}$ ou $\{F_1, F_3\}$.

Nous voyons que les notions de pertinence et redondance jouent un rôle fondamental dans la sélection d'attributs et nous allons donc les détailler dans la section suivante.

2.2 Notions de pertinence, non pertinence et redondance

Nous présentons d'abord une formalisation de la pertinence des attributs qui permet de distinguer les différentes contributions d'un attribut à la définition d'une fonction de classification. De même, nous présentons une définition de la redondance qui permet de proposer un mécanisme d'élimination des attributs redondants.

2.2.1 Pertinence d'attributs

Dans [Liu et Yu, 2005], les auteurs définissent la pertinence dans le cas d'attributs et de fonctions booléennes, et en supposant que les données sont non bruitées. Une définition plus large proposée par [Kohavi et John, 1997] définit les attributs pertinents comme ceux dont les valeurs varient systématiquement avec les valeurs de classe. Autrement dit, un attribut F_i est pertinent si connaître sa valeur change les probabilités sur les valeurs de la classe C . Mais cette définition peut être précisée pour distinguer les attributs fortement pertinents et les attributs faiblement pertinents grâce aux définitions suivantes. Soit F un ensemble complet d'attributs, F_i un attribut, et $S_i = F - \{F_i\}$. On suppose que l'on travaille avec un espace probabilisé où la probabilité est notée P . $P(C|S)$ est la probabilité de la classe C connaissant les attributs de l'ensemble S .

Définition 2.1 Un attribut F_i est fortement pertinent ssi :

$$P(C|F_i, S_i) \neq P(C|S_i) \quad (2.1)$$

Définition 2.2 Un attribut F_i est faiblement pertinent ssi :

$$P(C|F_i, S_i) = P(C|S_i) \text{ et } \exists S'_i \subset S_i \text{ tel que } P(C|F_i, S'_i) \neq P(C|S'_i) \quad (2.2)$$

Définition 2.3 Un attribut F_i est non pertinent ssi :

$$P(C|F_i, S_i) = P(C|S_i) \text{ et } \forall S'_i \subseteq P(C|F_i, S'_i) = P(C|S'_i) \quad (2.3)$$

D'après ces définitions, les attributs fortement pertinents sont donc indispensables et devraient figurer dans tout sous-ensemble optimal sélectionné, car leur absence devrait conduire à un défaut de reconnaissance de la fonction cible.

La faible pertinence suggère que l'attribut n'est pas toujours important, mais il peut devenir nécessaire pour un sous-ensemble optimal dans certaines conditions.

La non-pertinence d'un attribut se définit simplement par rapport à 2.1 et 2.2 et indique qu'un attribut n'est pas du tout nécessaire dans un sous-ensemble optimal d'attributs. Si l'on revient sur l'exemple 2.1, on peut dire que l'attribut F_1 est fortement pertinent, que F_2, F_3 sont faiblement pertinents, et que F_4, F_5 sont non pertinents. Un sous-ensemble d'attributs optimal ne devrait inclure que les attributs fortement pertinents, aucun attribut non pertinent, et un sous-ensemble d'attributs faiblement pertinents.

Pour savoir comment choisir quels attributs faiblement pertinents on doit sélectionner, il faut mettre en évidence la notion de redondance.

2.2.2 Redondance d'attributs

La notion de la redondance d'attributs se comprend intuitivement et elle est généralement exprimée en termes de corrélation entre attributs. On peut dire que deux attributs sont redondants (entre eux) si leurs valeurs sont complètement corrélées (par exemple, les attributs F_2 et F_3 de l'exemple 1). Cette définition ne se généralise pas directement pour un sous-ensemble d'attributs. On trouve dans [Koller et Sahami, 1996], une définition formelle de la redondance qui permet de concevoir une approche pour identifier et éliminer les attributs redondants. Cette formalisation repose sur la notion de couverture de Markov (Markov blanket) d'un attribut qui permet d'identifier les attributs non pertinents et redondants [Koller et Sahami, 1996; Yu et Liu, 2004a].

Définition 2.4 . Soit F l'ensemble total d'attributs et C la classe. Soit F_i un attribut, et M_i un sous-ensemble d'attributs qui ne contient pas F_i , c'est-à-dire : $M_i \subseteq F$ et $F_i \notin M_i$.

M_i est une couverture de Markov pour F_i ssi

$$P(F - M_i - \{F_i\}, C|F_i, M_i) = P(F - M_i - \{F_i\}, C|M_i) \quad (2.4)$$

La définition de couverture de Markov impose que M_i subsume non seulement l'information que F_i apporte sur C mais aussi l'information qu'il apporte sur tous les autres attributs. Dans [Koller et Sahami, 1996], il est montré qu'un sous-ensemble d'attributs optimal peut être obtenu par une procédure d'élimination descendante, appelée filtrage par couverture de Markov (Markov blanket Filtering) et définie comme suit :

Filtrage par couverture de Markov :. Soit G l'ensemble d'attributs courant ($G = F$ au départ). A chaque étape de la procédure, s'il existe une couverture de Markov pour l'attribut F_i dans l'ensemble G courant, F_i est enlevé de G .

On peut montrer que ce processus garantit qu'un attribut enlevé dans une étape précédente peut trouver une couverture de Markov dans une étape postérieure.

2.2 Notions de pertinence, non pertinence et redondance

Selon les définitions précédentes de la pertinence d'attributs, on peut également montrer que les attributs fortement pertinents ne peuvent trouver aucune couverture de Markov. Par contre, les attributs non pertinents doivent être enlevés de toute façon, et il n'est donc pas nécessaire de s'y intéresser dans la définition des attributs redondants. Cela conduit à la définition suivante de la redondance [Yu et Liu, 2004a].

Définition 2.5 [Yu et Liu, 2004a]. Soit G l'ensemble d'attributs courant, un attribut F_i est redondant et par conséquent peut être enlevé de G ssi il est faiblement pertinent et qu'il possède une couverture de Markov dans G .

Afin de synthétiser les différentes notions de pertinence et redondance que l'on vient de présenter, on peut proposer une catégorisation des attributs présentée dans la Figure 2.1. Dans ce schéma, un ensemble initial d'attributs peut être partitionné en quatre catégories: attributs non pertinents (partie I), attributs redondants (partie II) (qui sont faiblement pertinents comme on l'a vu), attributs faiblement pertinents et non-redondants (partie III), et attributs fortement pertinents (partie IV). Un sous-ensemble d'attributs optimal contient essentiellement tous les attributs des parties III et IV. Il est important de préciser que pour un ensemble initial donné, le processus de filtrage par couverture de Markov peut conduire à différents découpages donnant les parties II et III (qui sont disjointes). Ainsi, dans l'exemple précédent 2.1, l'attribut F_2 ou l'attribut F_3 devraient être enlevés comme attributs redondants et donc figurer dans (II) mais pas les deux à la fois.

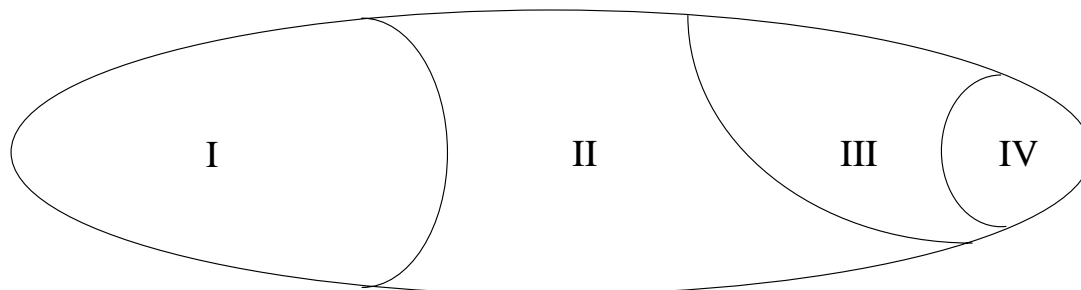


Figure 2.1 – Catégorisation d'attributs.

Une discussion plus générale au sujet des notions de pertinence, et de redondance d'attributs peut être trouvée dans : [John *et al.*, 1994; Jaeger *et al.*, 2003; Guyon et Elisseeff, 2003; Yu et Liu, 2004b; Yu et Liu, 2004a; Dash et Liu, 2006; Mundra et Rajapaks, 2007; Mao et Tang, 2007; Liu et Motoda, 2008b].

Les formalisations précédentes sont intéressantes pour mieux cerner les notions de redondance et de pertinence. Néanmoins les définitions probabilistiques que nous avons vues ne permettent pas de proposer un processus de sélection d'attributs applicable sur des données de grande dimension. Dans la suite de ce chapitre nous allons donc présenter différents points de vue permettant de comprendre le processus de la sélection d'attributs, d'un point de vue plus opérationnel.

2.3 Sélection d'attributs

2.3.1 La sélection vue comme un problème d'optimisation

Le problème de la sélection d'un sous-ensemble d'attributs peut être vu comme une recherche dans un espace d'hypothèses (appelé ensemble de solutions possibles) [Blum et Langley, 1997]. Étant donné un ensemble initial \mathcal{X} de n attributs, la sélection d'un "bon" sous-ensemble d'attributs nécessite d'examiner potentiellement $2^n - 1$ sous-ensembles possibles. La qualité d'un sous-ensemble sélectionné est évaluée selon un critère de performance que l'on notera \mathcal{J} . Dans le cas d'un problème de classification supervisée, ce critère est très souvent la précision d'un classifieur construit à partir de l'ensemble d'attributs sélectionnés.

La recherche d'un sous-ensemble d'attributs, optimal pour le critère \mathcal{J} que l'on s'est donné, est alors un problème NP -difficile [Davies et Russell, 1994; Cotta et Moscato, 2003]. Plusieurs approches peuvent être envisagées pour contourner cette difficulté. Elles sont formalisées dans la définition suivante:

Définition 2.6 (Sélection d'attributs) [Molina et al., 2002a] Soit X un ensemble d'attributs. Soit \mathcal{J} une mesure d'évaluation qui attribue à tout sous-ensemble de X un score: $\mathcal{J} : \bar{X} \subseteq X \rightarrow \mathbb{R}$. \mathcal{J} doit être optimisée (maximisée ou minimisée suivant la nature de \mathcal{J}), on supposera dans la suite que \mathcal{J} doit être maximisée .

La sélection d'un sous-ensemble d'attributs peut se faire suivant un des schémas suivants :

- Nombre d'attributs fixé: Pour un nombre m fixé, avec $m < n$, on cherche à trouver $\bar{X} \subset X$ tel que $|\bar{X}| = m$ et que $\mathcal{J}(\bar{X})$ soit maximum.
- Seuil de performance fixé: On se donne une valeur seuil \mathcal{J}_{opt} , c'est-à-dire, le minimum acceptable pour \mathcal{J} , et on cherche à trouver $\bar{X} \subseteq X$ tel que le cardinal de \bar{X} soit le plus petit possible et que $\mathcal{J}(\bar{X}) \geq \mathcal{J}_{opt}$.
- Compromis performance et nombre d'attributs. Trouver un compromis entre le fait de minimiser le nombre d'attributs $|\bar{X}|$ et le fait d'optimiser $\mathcal{J}(\bar{X})$.

La première stratégie consiste à passer d'un ensemble initial de n attributs à un sous-ensemble de m attributs sélectionnés qui donne une performance au moins égale ou meilleure à celle obtenue avec l'ensemble complet. Cela suppose qu'on connaît le nombre optimal des attributs à sélectionner. La première difficulté est de définir a priori ce nombre m . Ce nombre dépend de la taille, de la quantité et de la qualité de l'information disponible. Si m est fixé, une deuxième difficulté consiste alors à examiner toutes les combinaisons possibles. La recherche d'un sous-ensemble de m attributs parmi n donne un nombre de combinaisons $\binom{m}{n}$. A titre indicatif, le nombre de combinaisons sans répétition C_{49}^6 vaut 13983816 (il s'agit là du nombre de combinaisons possibles pour un tirage du loto). La croissance exponentielle de $\binom{m}{n}$ rend la recherche très coûteuse et une exploration exhaustive n'est pas envisageable, même pour des valeurs modérées de m et n .

Dans le deuxième cas on fixe un seuil de performance \mathcal{J}_{opt} à respecter. On cherche donc un sous-ensemble de cardinalité minimale dont la performance soit meilleure que \mathcal{J}_{opt} . La valeur \mathcal{J}_{opt} peut être une valeur observée avec une certaine représentation du

problème et on se fixe l'objectif de trouver une représentation utilisant un nombre minimum d'attributs mais garantissant une performance au moins égale à \mathcal{J}_{opt} . Nous verrons dans les chapitres suivants que des méthodes évolutionnaires comme les algorithmes génétiques peuvent être utilisés pour cet objectif.

Dans le troisième cas, on considère un problème d'optimisation bi-critère où l'on cherche à la fois à maximiser la fonction \mathcal{J}_{opt} tout en minimisant le nombre d'attributs retenus.

Dans le cadre de la sélection d'attributs de puces à ADN, il faut considérer le bon compromis entre la performance et la taille du sous-ensemble final en prenant les critères précédemment cités.

Nous verrons dans les chapitres suivants que dans les méthodes que nous proposons ou dans les expériences réalisées, nous utilisons chacun de ces trois cas de figures.

2.3.2 Schéma général de la sélection d'attributs

Les différentes méthodes proposées dans la littérature pour la sélection d'attributs peuvent être décrites par un schéma général [Dash et Liu, 1997] (voir Figure 2.2) dans lequel on trouve les éléments clés suivants :

1. Une procédure de génération de sous-ensembles candidats qui détermine l'exploration de l'espace de recherche.
2. Une fonction d'évaluation donnant la qualité des sous-ensembles candidats.
3. Une condition d'arrêt.
4. Un processus de validation pour vérifier si l'objectif souhaité est atteint.

Nous détaillons ci-dessous les étapes importantes de ce schéma.

Initialisation

Pour former un ensemble de départ dans le processus de sélection d'attributs, on peut envisager différents cas qui sont fortement liés à la direction de recherche qu'on va utiliser. L'ensemble considéré comme point de départ peut être a) l'ensemble de tous les attributs disponibles, b) l'ensemble vide, c) un ensemble d'attributs tirés de manière aléatoire ou e) une population de sous-ensembles d'attributs lorsqu'on travaillera avec des méthodes à base de population.

Critère d'évaluation

On distingue deux grandes approches pour la sélection d'attributs suivant que l'on va évaluer les attributs individuellement ou évaluer des sous-ensembles d'attributs.

a) L'évaluation individuelle

Dans l'évaluation individuelle, chaque attribut est évaluée indépendamment des autres à partir des seules informations fournies par les données et on lui attribue un poids (score) selon son degré de pertinence. Ce score permet d'établir une liste

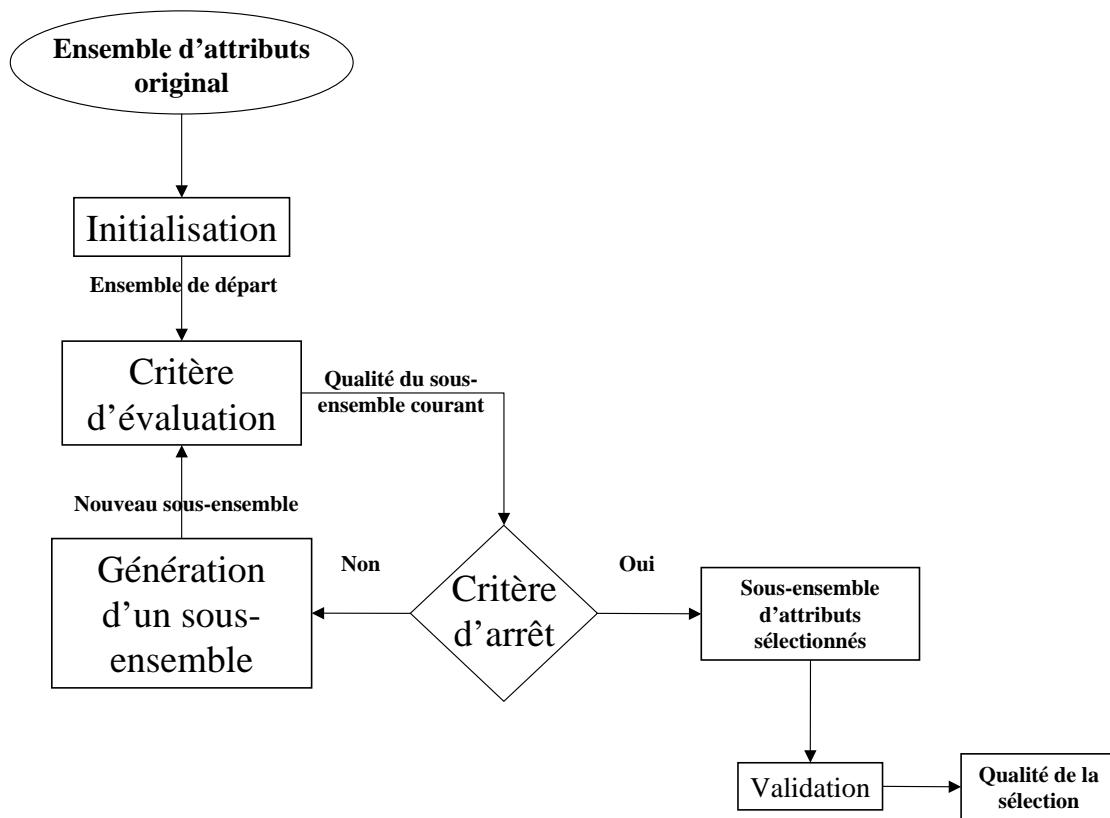


Figure 2.2 – Processus de la sélection d'attributs.

classée des attributs qui donnera un ensemble de m attributs en sélectionnant les m premiers. On garantit ainsi d'avoir des attributs pertinents (voir les catégories II, III et IV de la figure 2.1) mais cette approche ne peut enlever les attributs redondants car ces attributs ont probablement des scores similaires. Pour les données de grande dimension contenant un grand nombre d'attributs corrélés, cette approche produit des résultats éloignés des optimaux car elle ne gère pas du tout la redondance. Le principal intérêt de cette méthode est sa complexité linéaire qui lui permet de traiter des données de grande dimension. On peut remarquer que cette approche peut être utilisée dans un premier temps pour réduire la taille de l'ensemble de départ.

b) L'évaluation d'un sous-ensemble

Beaucoup de méthodes de sélection s'appuient sur l'évaluation des sous-ensembles pour gérer à la fois la redondance et la pertinence. Dans la figure 2.2, la génération de sous-ensembles produit des sous-ensembles candidats suivant une stratégie de recherche [Liu et Yu, 2005]. Chaque sous-ensemble d'attributs candidat est évalué par une certaine mesure d'évaluation et comparé avec le meilleur sous-ensemble d'attributs obtenu précédemment par rapport à cette mesure. Si le sous-ensemble

courant est meilleur, il remplace le meilleur sous-ensemble d'attributs mémorisé. Le processus de génération et d'évaluation d'un sous-ensemble est répété jusqu'à ce qu'un critère d'arrêt soit satisfait. On peut noter que les mesures d'évaluation utilisées par cette approche prennent en compte l'existence et l'effet d'attributs redondants à la différence des méthodes d'évaluation individuelle. Un sous-ensemble d'attributs sélectionné par cette approche se rapproche d'un sous-ensemble optimal (voir les catégories III et IV dans la figure 2.1). Plusieurs méthodes se sont montrées performantes en enlevant les attributs non pertinents et les attributs redondants [Koller et Sahami, 1996]. Néanmoins, le problème de ces méthodes est qu'elles doivent explorer l'espace des sous-ensembles d'attributs.

2.3.3 Génération de sous-ensembles et procédures de recherche

L'espace de recherche comporte 2^n sous-ensembles candidats si n est le nombre d'attributs initial. Plusieurs stratégies de recherche heuristiques peuvent être envisagées. Dans une stratégie de recherche séquentielle ascendante (forward selection), l'ensemble de départ est l'ensemble vide et chaque étape de génération de sous-ensembles considère tous les attributs qui ne sont pas encore sélectionnés pour retenir le meilleur d'entre eux et l'ajouter à l'ensemble courant. De la même façon, une recherche descendante part de l'ensemble complet d'attributs et retire à chaque étape un attribut.

Ces méthodes de nature gloutonne ne font pas de retour arrière (et ne sont donc pas complètes). Leur principal avantage est qu'elles sont de complexité quadratique. Sur des données de taille modérée, elles peuvent donc être appliquées efficacement. Pour les données de très grande dimension comme celles issues de la bioinformatique, il faut adapter ces procédures. On peut par exemple proposer une procédure de recherche descendante où dans les premières étapes de la recherche, lorsqu'on travaille avec un grand nombre d'attributs, on élimine non pas un seul attribut à la fois mais plusieurs attributs. Il faut pour cela pouvoir estimer efficacement (en un seul calcul) quels sont les attributs à retirer. On trouvera une illustration de ce schéma dans la méthode SVM-RFE [Guyon *et al.*, 2002].

La recherche peut aussi se faire grâce à des procédures évolutionnaires ou métaheuristiques. C'est ce que nous présenterons dans les chapitres suivants. Dans ces approches, la génération de sous-ensembles candidats se fait généralement suivant un processus aléatoire (croisement par exemple).

Pour appliquer ces stratégies sur des données de grande dimension, il sera nécessaire de guider efficacement la recherche et un des apports de notre travail sera de proposer des informations utiles pour guider au mieux les stratégies de recherche.

Critère d'arrêt

L'initialisation et le critère d'arrêt définissent les bornes de la recherche. Dans le cas de méthodes basées sur un critère d'évaluation individuelle la condition d'arrêt peut être un nombre fixé d'attributs à retenir. Dans le cas des méthodes d'évaluation de sous-ensembles, le critère d'arrêt peut être un temps de calcul fixé, un nombre d'itérations fixé,

l'absence de gain de performance par rapport aux solutions déjà trouvées ou encore le fait que les sous-ensembles candidats deviennent trop homogènes (dans le cas d'algorithmes à base de populations).

2.4 Les approches pour la sélection d'attributs

Le schéma général présenté dans la section précédente se retrouve dans les 3 grands types d'approches que nous décrivons ci-dessous. Ces approches se distinguent par la manière dont la sélection d'attributs interagit (ou non) avec le mécanisme de classification.

2.4.1 Méthodes de Filtres

Dans les méthodes traditionnelles appelées méthodes filtres ou de filtrage, le principe consiste à évaluer chaque attribut (cas univarié) pour lui assigner un score de pertinence. Ce score permet un classement des attributs et in fine la sélection des attributs les mieux classés c'est-à-dire les plus pertinents. Notons que l'on trouve aussi quelques méthodes multivariées qui attribuent des scores à des groupes d'attributs.

L'avantage des méthodes de filtrage est qu'elles peuvent être utilisées lorsqu'on travaille avec un très grand nombre d'attributs car elles sont de complexité raisonnable. Elles ne tiennent compte que des informations présentes dans les données et sont indépendantes du processus de classification.

Le principal point négatif de ces méthodes est qu'elles évaluent les attributs individuellement en négligeant les interactions possibles avec les autres attributs. Ces approches ont donc du mal à éliminer les attributs qui sont redondants. Enfin, ces méthodes reposent généralement sur le choix d'un seuil pour le critère de pertinence choisi ou d'un nombre d'attributs à sélectionner qui doit être fixé a priori. Le choix de ces paramètres n'est pas facile à réaliser.

Nous détaillons dans la suite de cette section les critères de filtre qui ont été utilisés dans le domaine de la bio-informatique pour la sélection de gènes. La mesure de pertinence utilisée dans une méthode filtre peut être une mesure statistique classique telle que la t-statistique, le test de Fisher ou le test de Wilcoxon. Certaines mesures de filtrage ont été proposées spécifiquement pour la sélection de gènes telles que B/W [Dudoit *et al.*, 2002] ou SNR [Golub *et al.*, 1999].

a) t-statistique

Le critère t-statistique est utilisé dans [Nguyen et Rocke, 2002]:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/n_1 + s_2^2/n_2}} \quad (2.5)$$

où n_k , \bar{x}_k et s_k^2 sont la taille, la moyenne la variance des classes $k = 1, 2$. Pour chaque gène une t - valeur est calculée et si on souhaite sélectionner p gènes, on retient $p/2$ gènes avec les plus grandes valeurs positives (gènes fortement exprimés

2.4 Les approches pour la sélection d'attributs

dans la classe 1) et les $p/2$ gènes avec les plus "grandes" valeurs négatives (gènes fortement exprimés dans la classe 2).

b) test de Wilcoxon

La statistique de la somme des rangs de Wilcoxon est utilisée pour déterminer si un gène est différentiellement exprimé dans deux classes [Deng *et al.*, 2004; Chu *et al.*, 2005]. Pour un gène donné, les valeurs d'expression sont triés dans l'ordre croissant et on attribue un rang à chaque échantillon. La valeur W est définie comme la somme des rangs de la classe 1, et on associe à cette valeur une p -valeur indiquant si l'hypothèse que les expressions des deux classes sont différentes peut être retenue.

c) Fisher

Le test de Fisher [Duda *et al.*, 2000] est défini comme suit :

$$P = \frac{(\bar{x}_1 - \bar{x}_2)^2}{(s_1^2 + s_2^2)} \quad (2.6)$$

où $\bar{x}_k, (s_k)^2$ sont la moyenne et l'écart-type de l'attribut pour la classe $k = 1, 2$. Un score important indique donc que les moyennes des 2 classes sont significativement différentes.

d) BW

Le score discriminant BW a été proposé par [Dudoit *et al.*, 2002]. Il est basé sur le rapport entre dispersion entre classes et dispersion intra-classes. Pour un attribut j , ce rapport est obtenu comme suit :

$$BW(j) = \frac{\sum_i \sum_k I(y_i = k)(\bar{x}_{kj} - \bar{x}_j)^2}{\sum_i \sum_k I(y_i = k)(x_{ij} - \bar{x}_{kj})^2} \quad (2.7)$$

où \bar{x}_j et \bar{x}_{kj} dénotent respectivement la moyenne d'un attribut j à travers tous les échantillons et à travers les échantillons appartenant à la classe k seulement.

e) SNR ou S/N

Ce critère a été proposé par [Golub *et al.*, 1999] et est défini comme suit :

$$P(j) = \frac{\bar{x}_{1j} - \bar{x}_{2j}}{s_{1j} + s_{2j}} \quad (2.8)$$

où \bar{x}_{kj}, s_{kj} dénotent la moyenne et l'écart-type de l'attribut j pour les échantillons de classes $k = 1, 2$. De grandes valeurs de $|P(j)|$ indiquent une forte corrélation entre les valeurs de l'attribut et la distinction de classes.

2.4.2 Méthodes Enveloppées

Dans le deuxième type d'approche pour la sélection d'attributs, le mécanisme de sélection interagit avec un classifieur pour trouver un sous-ensemble d'attributs qui sera optimal pour ce modèle d'apprentissage [Kohavi et John, 1997]. Comme on l'a présenté

auparavant, la sélection peut être vue comme une exploration de sous-ensembles candidat et dans les méthodes enveloppes, un algorithme de recherche "enveloppe" le processus de classification. Le processus d'apprentissage effectue la tâche de l'évaluation des sous-ensembles d'attributs .

L'intérêt de ces méthodes est que le sous-ensemble choisi est parfaitement adapté au classifieur .

En revanche, un risque de sur-apprentissage existe. De plus, ces méthodes enveloppes sont beaucoup plus coûteuses en temps de calcul puisqu'un classifieur doit être construit chaque fois que l'on doit évaluer un sous-ensemble candidat. Leur complexité de calcul est donc dépendante de la complexité du modèle d'apprentissage utilisé.

2.4.3 Méthodes Intégrées

Ces méthodes sont proches des méthodes d'enveloppe, car elles combinent le processus d'exploration avec un algorithme d'apprentissage [Navin *et al.*, 2006]. La différence avec les méthodes enveloppes est que le classifieur sert non seulement à évaluer un sous-ensemble candidats mais aussi à guider le mécanisme de sélection. Selon cette définition, les méthodes de construction d'arbres de décision comme C4.5 de Quinlan [Quinlan, 1993] ou CART (Classification and Regression Tree) de Breiman [Breiman *et al.*, 1984] relèvent de cette approche puisque la sélection des attributs se fait en même temps que la construction du modèle. Parmi les derniers travaux qui utilisent cette approche, on peut citer Weighted naive Bayes [Duda *et al.*, 2000] et les travaux dans lesquels le mécanisme d'apprentissage fournit des poids aux attributs pour faire la sélection. On peut citer notamment RFE-SVM [Guyon *et al.*, 2002; Rakotomamonjy, 2003], RFE+PKLR [Zhu et Hastie, 2004] et A-ROM [Weston *et al.*, 2002].

L'avantage de ces méthodes est que le processus de recherche est guidé par des informations intéressantes fournies par le classifieur, ce qui rend ces méthodes plus efficaces que les méthodes enveloppes [Saeys *et al.*, 2007].

2.5 Validation des méthodes

Pour évaluer la sélection d'attributs, on aura besoin d'appliquer un classifieur et d'examiner les performances de la classification. On a vu dans le chapitre précédent comment un classifieur évalue le risque empirique soit sur un ensemble de test, soit par validation croisée. L'idée la plus naturelle pour évaluer une méthode de sélection d'attributs est donc d'appliquer cette méthode sur l'ensemble complet des échantillons disponibles ce qui conduit à un sous-ensemble de gènes sélectionnés. Ensuite on retient ce sous-ensemble de gènes comme représentation du problème et on évalue par validation croisée par exemple la performance d'une règle de classification entraînée sur cette représentation.

Ce schéma a été appliqué par exemple dans le travail de [Guyon *et al.*, 2002] où de très bonnes performances sont obtenues pour les jeux de la Leucémie et du cancer du Colon. Ainsi, dans le cas de la Leucémie, la méthode SVM-RFE sélectionne 2 gènes à partir des-

quels on obtient pour un classifieur SVM un taux de bonne classification de 100%. Mais ces résultats ainsi que ceux que l'on trouve dans d'autres travaux de la même époque sont très optimistes et souffrent d'une erreur d'estimation appelée "biais de sélection" et mise en évidence par [Ambroise et McLachlan, 2002].

Les auteurs de [Ambroise et McLachlan, 2002] ont étudié le protocole expérimental généralement employé pour évaluer les méthodes de sélection de gènes pour les données de puces à ADN et ils ont observé que la procédure appliquée pour ces travaux pouvait fournir des conclusions erronées concernant la performance des algorithmes. Le biais de sélection se produit lorsque le processus de sélection est effectué à partir de tous les échantillons disponibles et que l'on applique ensuite un processus de validation pour estimer l'erreur en classification.

L'information de tous les échantillons est utilisée pour la sélection, et la sélection est donc mise à l'écart du processus de validation croisée généralement employé. Pour corriger ce problème de biais de sélection les auteurs proposent que les méthodes de sélection d'attributs utilisent un schéma de validation croisée externe à la procédure de sélection. Ainsi on doit d'abord découper le jeu de données en k blocs. $(k - 1)$ blocs sont utilisés pour sélectionner un ensemble de gènes et construire un classifieur et on évalue cet ensemble sélectionné et le classifieur appris sur le bloc de données restant. (voir Figure 2.3).

Le modèle de validation pour la sélection et classification d'attributs est présenté dans la Figure 2.3, qui montre la partition de données en un ensemble d'échantillons d'apprentissage et un ensemble d'échantillons de test. Cette méthode peut être utilisée pour évaluer n'importe quel type de méthode de sélection, soit de filtrage, soit d'enveloppe ou soit d'approche intégrée.

Notons tout de même que cette validation exige de recommencer le processus de la sélection à chaque itération de la validation croisée ce qui peut être coûteux en temps de calcul pour une méthode enveloppe ou intégrée.

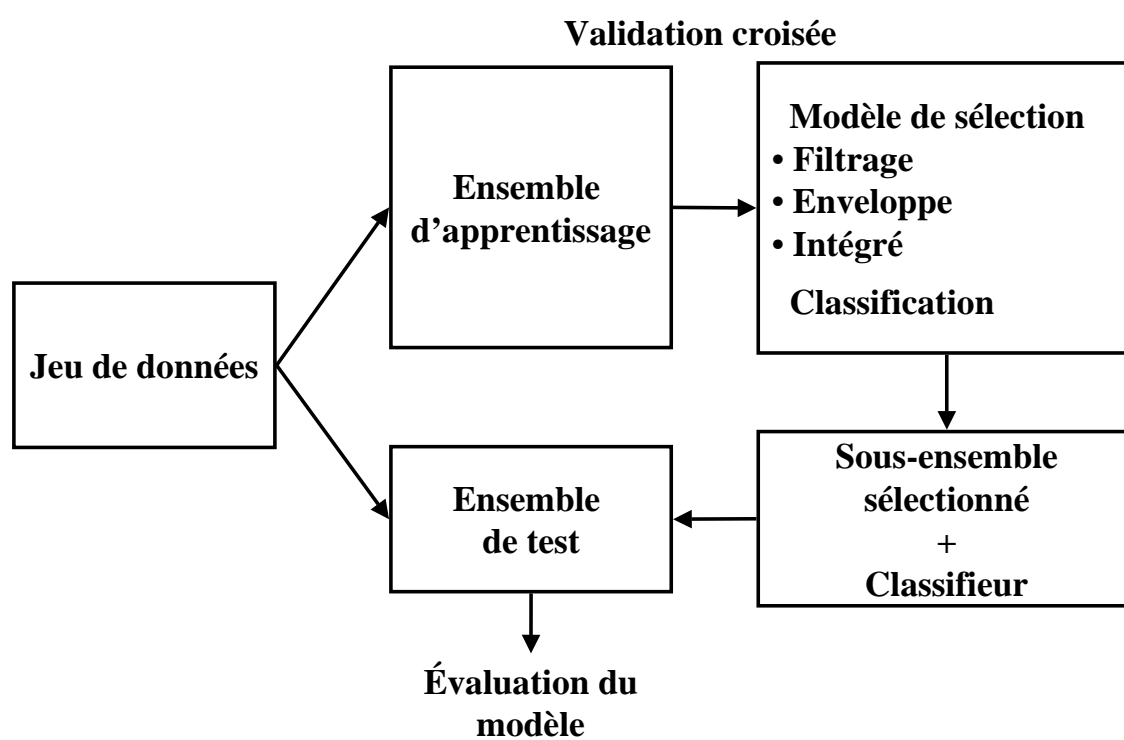


Figure 2.3 – Schéma de validation croisée pour l'évaluation d'un processus de sélection-classification.

Chapitre 3

Recherche locale pour la sélection de gènes

Sommaire

3.1 SVMs	37
3.1.1 Classifieur SVM linéaire	37
3.1.2 Coefficients de classement d'un SVM	41
3.2 Recherche locale	42
3.2.1 Méthode de descente	43
3.2.2 Recherche tabou	44
3.2.3 Recherche locale itérée	45
3.3 Application de la recherche locale au problème de la sélection de gènes 47	
3.3.1 Espace de recherche et représentation	47
3.3.2 Fonction d'évaluation	48
3.3.3 Mouvement et voisinage	48
3.3.4 Solution initiale	49
3.3.5 Schéma général	49
3.4 Comparaison d'algorithmes de recherche locale	50
3.4.1 Conditions expérimentales	50
3.4.2 Stratégie de descente spécifique	51
3.4.3 Recherche Tabou spécifique	52
3.4.4 Recherche locale itérée spécifique	54
3.5 Comparaison avec d'autres approches de sélection de gènes	57
3.5.1 Comparaison avec SVM-RFE	57
3.5.2 Comparaison avec d'autres approches de l'état d'art	58
3.6 Conclusion	58

DANS CE chapitre, nous présentons une approche de type intégrée pour faire face au problème de sélection de gènes pour la classification des données de puces à ADN. Notre approche sélectionne un sous-ensemble de gènes grâce à une méthode de recherche locale (RL). D'abord, nous avons comparé les différents algorithmes de recherche locale (RL). Il s'agit des algorithmes de la descente avec la stratégie de première amélioration et la stratégie de meilleure amélioration, de la recherche tabou combinée avec les deux descentes, et de la recherche locale itérée combinée avec la recherche tabou.

Dans ces algorithmes, la qualité d'un sous-ensemble de gènes sélectionnés est évaluée grâce à un SVM construit sur ce sous-ensemble. Les coefficients d'un SVM linéaire permettent de caractériser la pertinence des gènes [Guyon *et al.*, 2002]. Ici, nous utilisons l'information apportée par ces coefficients pour éliminer des gènes et pour améliorer la recherche locale. De plus, la fonction d'évaluation de nos algorithmes prend en compte non seulement le taux de classification mais encore la marge du SVM.

Afin d'évaluer l'efficacité pratique de ces algorithmes, nous avons effectué des expérimentations poussées sur neuf jeux de données de la littérature. Lors de ces expériences, la recherche locale itérée combinée avec la recherche tabou a obtenu les meilleurs résultats.

Enfin, la connaissance acquise au cours de cette étude expérimentale nous a conduit à comparer l'algorithme de recherche locale itérée combinée avec la recherche tabou à huit autres méthodes de sélection issues de l'état de l'art. Les résultats obtenus démontrent une bonne amélioration en efficacité de la recherche locale itérée combinée avec la recherche tabou par rapport à ces méthodes : en effet les taux de classification obtenus sont aussi bons pour des nombres de gènes comparables.

Une partie de cette étude a fait l'objet d'une publication dans la conférence internationale *BIRD-ALBIO'08* [Hernandez-Hernandez *et al.*, 2008].

3.1 SVMs

Il est usuel dans l'approche de sélection de type intégrée d'utiliser un classifieur afin d'évaluer la qualité d'un sous-ensemble d'attributs sélectionnés. Les séparateurs à vastes marges (ou classifieur SVM) peuvent être utilisés pour ce but.

Dans ce chapitre nous proposons une approche de type intégrée qui réalise la sélection d'un sous-ensemble de gènes par des méthodes de recherche locale. Ainsi, la qualité d'un sous-ensemble de gènes sélectionnés est évaluée grâce à un classifieur SVM construit sur ce sous-ensemble.

Dans cette thèse, nous n'utilisons que des SVM à noyaux linéaires car ces modèles se sont révélés bien adaptés aux données que nous souhaitons traiter. La présentation qui suit se limite donc à rappeler les caractéristiques des SVM linéaires utiles pour notre travail.

Les coefficients d'un classifieur SVM linéaire permettent de caractériser la pertinence de gènes [Guyon *et al.*, 2002; Rakotomamonjy, 2003]. Ici nous utilisons l'information apportée par ces coefficients pour éliminer des gènes sélectionnés et pour informer la recherche locale. C'est pourquoi nous rappelons ci-dessous les caractéristiques principales d'un classifieur SVM et expliquons comment un processus de sélection de gènes par la recherche locale peut être guidé par l'information utile apportée par un classifieur SVM. Cette section emprunte des notations et différents éléments aux livres [Vapnik, 1998; Cristianini et Shawe-Taylor, 2000; Scholkopf et Smola, 2001; Kecman, 2001; Herbrich, 2002; Cornuéjols et Miclet, 2002; Abe, 2005; Wang *et al.*, 2005a] et aux articles [Boser *et al.*, 1992; Cortes et Vapnik, 1995; Burges, 1998; Guyon *et al.*, 2002] auxquels nous invitons le lecteur à se reporter.

3.1.1 Classifieur SVM linéaire

Cas linéairement séparable : soit $\mathcal{S} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ avec $x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}$ un ensemble d'exemples d'apprentissage.

On suppose dans un premier temps qu'il existe un hyperplan séparateur permettant de distinguer les exemples positifs (de la classe 1) des exemples négatifs (de la classe -1). La recherche d'un hyperplan séparateur revient à chercher une fonction de décision de la forme $H(x) = w^T x + b = 0$ où w est le vecteur normal à l'hyperplan, $|b| / \|w\|$ est la distance perpendiculaire de l'hyperplan à l'origine, et $\|w\|$ est la norme euclidienne du vecteur w . Les points de l'échantillon d'apprentissage sont correctement classés par l'hyperplan si, $\forall 1 \leq i \leq n$

$$\begin{cases} w^T x_i + b > 0 & \text{si } y_i = 1 \\ w^T x_i + b < 0 & \text{si } y_i = -1 \end{cases}$$

Ces 2 contraintes peuvent se résumer en:

$$\forall 1 \leq i \leq n \quad y_i (w^T x_i + b) > 0$$

Si les exemples sont linéairement séparables, on peut trouver plusieurs hyperplans satisfaisant la condition ci-dessus (voir figure 3.1). On peut alors chercher l'hyperplan

optimal c'est-à-dire celui qui va être le plus éloigné des exemples positifs et des exemples négatifs. La distance de l'hyperplan à chacune des classes est appelée *marge*. On cherche donc l'hyperplan qui assure la plus grande marge.

On peut normaliser les paramètres cherchés w et b pour que les points positifs les plus proches de l'hyperplan (hyperplan H_1) vérifient l'équation $w^T x + b = 1$ et de même les points négatifs les plus proches de l'hyperplan (hyperplan H_2) vérifient l'équation $w^T x + b = -1$. La distance entre H_1 et l'hyperplan optimal vaut alors $1/\|w\|$ comme la distance entre H_2 et cet hyperplan optimal. La marge est donc égale à

$$M = \frac{2}{\|w\|} \quad (3.1)$$

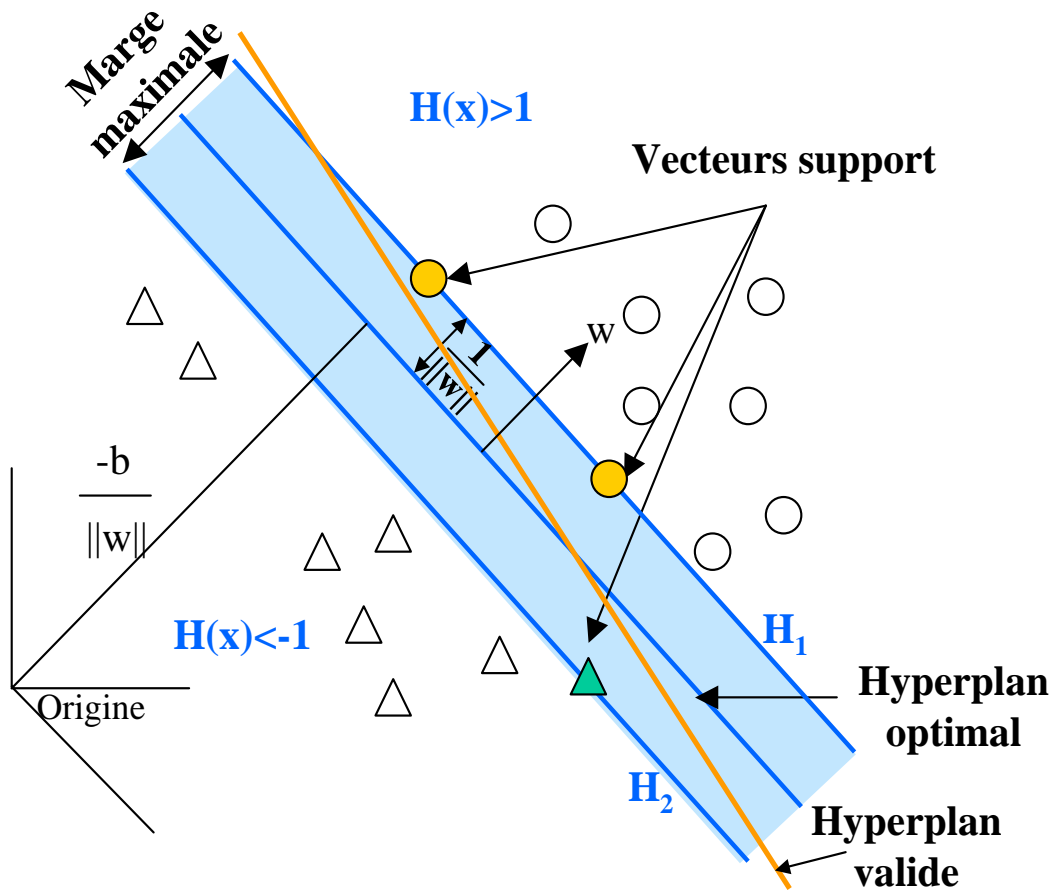


Figure 3.1 – Hyperplan optimal de séparation

Pour garantir la marge maximale, la recherche de l'hyperplan optimal revient donc à minimiser $\|w\|$, soit à résoudre le problème d'optimisation suivant qui porte sur les paramètres w et b

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad (3.2)$$

avec la contrainte $y_i (w \cdot x_i + b) \geq 1, i = 1, \dots, n$.

Cette écriture du problème, appelée *formulation primale* implique le réglage de $p + 1$ paramètres, p étant la dimension de l'espace des entrées \mathcal{X} . Cela est possible avec des méthodes de programmation quadratique pour des valeurs de p assez petites, mais devient inenvisageable pour des valeurs de p dépassant quelque centaines. Heureusement, il existe une transformation de ce problème dans une formulation duale [Boser *et al.*, 1992] qu'on peut résoudre en pratique.

Ceci peut se résoudre par la fonction Lagrangienne qui incorpore des informations sur la fonction objectif et sur les contraintes et donc le caractère stationnaire peut être utilisé pour détecter des solutions. Plus précisément, le Lagrangien est défini comme étant la somme de la fonction objectif et d'une combinaison linéaire des contraintes dont les coefficients $\alpha_i \geq 0$ sont appelés *multiplieurs de Lagrange* ou encore *variables duales*.

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i [y_i (w^T x_i + b) - 1] \quad (3.3)$$

Le Lagrangien doit être minimisé par rapport à w et b , et maximisé par rapport à α . Au point-selle, la dérivée du Lagrangien par rapport aux variables primaires doit s'annuler, selon les conditions de Karush-Kuhn-Tucker. Ceci s'écrit:

$$\frac{\partial}{\partial w} L(w, b, \alpha) = 0, \quad \frac{\partial}{\partial b} L(w, b, \alpha) = 0 \quad (3.4)$$

et conduit à:

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (3.5)$$

et à:

$$w = \sum_{i=1}^n \alpha_i y_i x_i \quad (3.6)$$

On peut ainsi transformer le problème initial en son dual consistant à maximiser par rapport à α la forme suivante du lagrangien :

$$Max_{\alpha} = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \quad (3.7)$$

sous les contraintes $\sum_{i=1}^n y_i \alpha_i = 0$ et $\alpha_i \geq 0, i = 1, \dots, n$.

La fonction de décision d'un classifieur SVM linéaire avec un vecteur d'entrée \mathcal{X} est donnée par $H(\mathcal{X}) = w \cdot \mathcal{X} + b$ avec $w = \sum_{i=1}^n \alpha_i y_i \mathcal{X}_i$ et $b = y_i - w \cdot \mathcal{X}_i$.

Le vecteur de poids w est une combinaison linéaire d'exemples d'apprentissage. Les α_i sont tous nuls exceptés ceux correspondant aux *vecteurs support*.

Cas non linéairement séparable : on considère ici le cas où des exemples sont mal classés par l'hyperplan optimal. Cela peut résulter du bruit dans les données. Dans ce cas, l'hyperplan optimal est celui qui satisfait les contraintes suivantes : d'une part, la distance entre les vecteurs support et l'hyperplan optimal doit être maximale, et d'autre part, la distance entre les exemples mal classés et l'hyperplan optimal doit être minimale (voir la figure 3.2). Pour cela, une technique dite des variables ressort (*slack variables*) permet de chercher un hyperplan séparateur faisant le moins d'erreurs possible [Cortes et Vapnik, 1995]. Pour formuler tout cela, on utilise les variables ressort ξ_i . L'objectif est de résoudre le problème d'optimisation suivant:

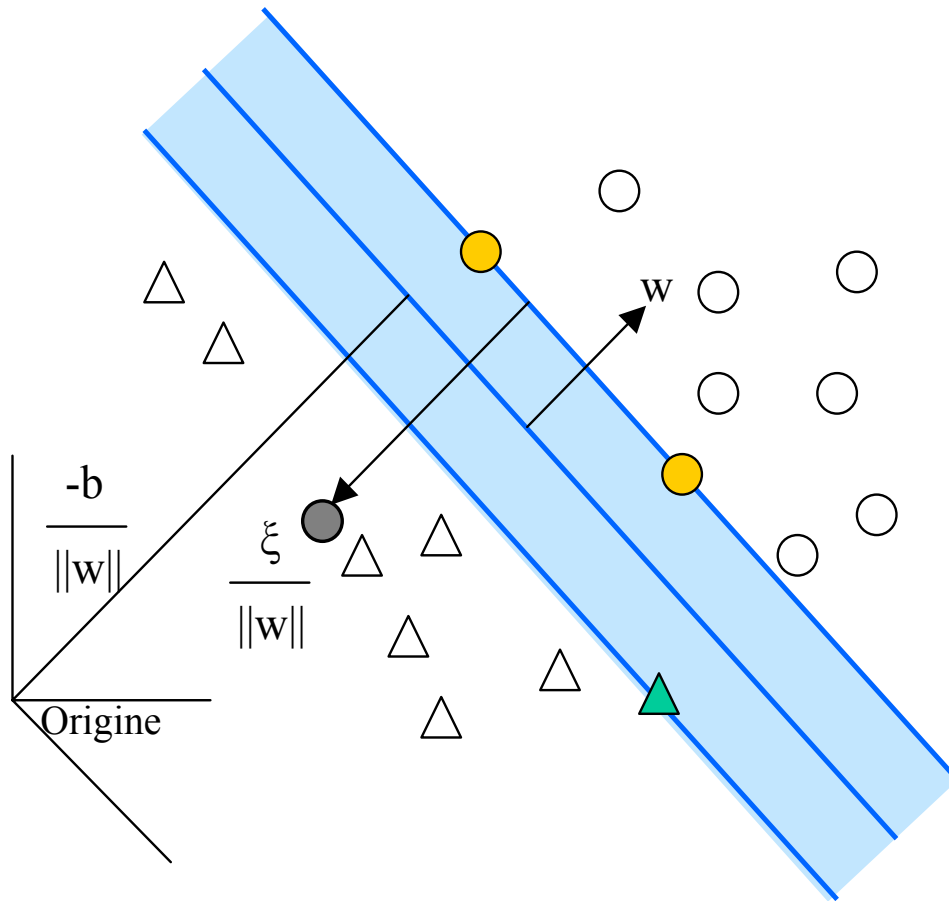


Figure 3.2 – Hyperplans séparateurs dans le cas des exemples non linéairement séparables

$$\min_{w,b,\xi_i} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad (3.8)$$

avec la contrainte $y_i (w^T x_i + b) \geq 1 - \xi_i, i = 1, \dots, n$.

Dans cette formulation, w est le vecteur de poids qui détermine l'hyperplan, ξ_i est la variable de relâchement d'une contrainte et C un coefficient de pénalisation du relâchement. Cela mène à un problème dual légèrement différent de celui du cas des exemples linéairement séparables : maximiser le lagrangien donné dans l'équation 3.3 par rapport à α sous les contraintes :

$$\sum_{i=1}^n \alpha_i y_i = 0 \text{ avec } 0 \leq \alpha_i \leq C \text{ pour } i = 1, \dots, n$$

Le problème d'optimisation se transcrit comme suit:

$$Max_{\alpha} = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \quad (3.9)$$

Le calcul du vecteur w , le biais b et la fonction de décision restent exactement le même que pour le cas linéairement séparable.

3.1.2 Coefficients de classement d'un SVM

Lorsqu'une fonction de décision s'exprime comme une fonction linéaire des entrées, les coefficients de cette combinaison représentent l'influence de chaque variable d'entrée sur la classification. Dans un classifieur SVM linéaire, la valeur absolue des coordonnées du vecteur w indique l'importance de chaque variable d'entrée pour le calcul de l'hyperplan séparateur. Par exemple, si l'hyperplan est orthogonal à une direction correspondant à un gène g_i , alors ce gène est important pour la classification et sera affecté d'un poids fort. Ces coefficients peuvent donc être utilisés pour classer les différentes variables selon leur pertinence pour la classification.

Pour un classifieur SVM linéaire construit sur un ensemble d'attributs donné, et dont la fonction de décision s'écrit $w^T x + b = 0$, on appellera **vecteur des coefficients de classement des attributs** le vecteur c tel que

$$\forall i, c_i = (w_i)^2 \quad (3.10)$$

Cette idée est utilisée dans [Guyon *et al.*, 2002] qui propose une méthode d'élimination récursive des variables (SVM-RFE). Plus exactement, dans cette sélection séquentielle qui part de l'ensemble complet de gènes, on élimine progressivement le gène le moins pertinent. Pour déterminer le gène à enlever à chaque itération, on considère le gène qui a l'influence la plus petite sur la fonction de coût du processus de classification, c'est-à-dire, le gène qui a le plus petit coefficient c_i

Dans la suite de ce chapitre, nous utiliserons ce vecteur des coefficients de classement pour développer des méthodes de recherche locale spécifiques.

De plus, pour traiter le fait que les exemples ne sont pas linéairement séparables nous utilisons la technique de *variables ressort*.

3.2 Recherche locale

Avant d'aborder les méthodes de recherche locale, nous donnerons quelques notions de base.

Définition 3.1 (Optimisation combinatoire) *Un problème d'optimisation combinatoire est soit un problème de minimisation, soit un problème de maximisation d'une fonction donnée, et est composé par un ensemble d'instances.*

Définition 3.2 (Instance) *Une Instance I d'un problème de minimisation est un couple (S, f) . S est l'espace de recherche et $f : S \rightarrow \mathbb{R}$ est la fonction objectif, qui à chaque solution potentielle $s \in S$ associe un coût réel $f(s)$ à minimiser. Le problème est de trouver $s' \in S$ tel que $f(s') \leq f(s)$ pour toute solution $s \in S$.*

Remarquons que d'une manière similaire, on peut également définir les problèmes de maximisation en remplaçant simplement \leq par \geq .

Définition 3.3 (Heuristique) *Une heuristique est une méthode, conçue pour un problème d'optimisation donné, qui produit une solution non nécessairement optimale lorsqu'on lui fournit une instance de ce problème.*

Définition 3.4 (Métaheuristique) *Une métaheuristique est définie comme un processus itératif qui guide une heuristique subordonnée en combinant intelligemment différents concepts pour explorer et exploiter l'espace de recherche, des stratégies d'apprentissage sont utilisées pour structurer des renseignements pour trouver des solutions efficaces presque optimales.*

On peut maintenant situer et comprendre les méthodes de recherche locale.

Les algorithmes de recherche locale sont souvent présentés comme des méthodes qui peuvent être à la fois intuitives et très efficaces pour résoudre de manière approchée des instances de problèmes combinatoires lorsque ces instances possèdent des solutions. Ils trouvent leurs origines à la fin des années 50 et au début des années 60 sur le problème du voyageur de commerce [Aarts et Lenstra, 1997].

Notons que dans la littérature, la notion de recherche locale est de plus en plus employée pour désigner la classe des méthodes fondées sur l'idée de parcours (descente, recherche tabou, recherche locale itérée,...).

Toutes les méthodes de recherche locale, appelées aussi métaheuristicques à base de voisinage s'appuient sur le même principe. Partant d'une solution quelconque s , qui peut être prise au hasard ou peut être construite par un algorithme dans l'espace de recherche S , ces méthodes cherchent à améliorer de façon itérative la solution existante en explorant le voisinage de celle-ci. Plus précisément, la recherche locale passe itérativement d'une solution à une autre solution en appliquant des transformations ou mouvements $(s^0, s^1, \dots, s^i, s^{i+1}, \dots)$ tel que s^0 est la solution initiale et $\forall i, s^{i+1} \in \mathcal{N}(s^i)$, où $\mathcal{N}(s^i)$ est le voisinage de s^i . La recherche locale s'arrête généralement quand un nombre d'itérations pré-défini est atteint ou quand une solution de qualité est obtenue et on retourne la meilleure solution trouvée durant cette exploration.

La recherche locale se caractérise par trois éléments : une fonction d'évaluation, une fonction de voisinage et une stratégie de mouvement.

Définition 3.5 (Fonction d'évaluation) Soit $f : S \rightarrow \mathbb{R}$ une application ou une fonction d'évaluation qui à chaque solution s associe une valeur $f(s) \in \mathbb{R}$.

Définition 3.6 (Fonction de voisinage) Soit $\mathcal{N} : S \rightarrow 2^S$ une application ou une fonction qui associe un sous-ensemble de S à toute solution $s \in S$. On appelle $\mathcal{N}(s)$ le voisinage de s et une solution $s' \in \mathcal{N}(s)$ est dite voisine de s .

Définition 3.7 (Stratégie de mouvement) Une stratégie de mouvement correspond aux mouvements ou règles de passage d'une solution courante s à une de ses solutions voisines s' .

On peut maintenant définir plus exactement les notions d'optimum local et optimum global.

Définition 3.8 (Optimum local) Une solution $s \in S$ est un optimum local relatif à son voisinage \mathcal{N} si et seulement s'il n'existe pas de solution $s' \in \mathcal{N}(s)$ dont l'évaluation est de meilleure qualité que s , soit

$$\forall s' \in \mathcal{N}(s), \begin{cases} f(s) \leq f(s') & \text{dans le cas d'un problème de minimisation} \\ f(s) \geq f(s') & \text{dans le cas d'un problème de maximisation} \end{cases}$$

Définition 3.9 (Optimum global) Une solution $s \in S$ qui vérifie la propriété précédente pour tous les voisinages du problème est appelée un optimum global.

Nous évoquons trois méthodes de recherche locale que nous avons utilisées pour traiter le problème de la sélection de gènes.

3.2.1 Méthode de descente

La méthode de recherche locale la plus élémentaire est la méthode de la descente, appelée descente du gradient ou la descente *Hillclimbing*. Elle peut être décrite comme suit. À partir d'une solution quelconque, on cherche une solution dans le voisinage et on accepte cette solution si elle améliore la solution courante [Hoos et Stützle, 2004].

Nous présentons dans l'algorithme 3.1 une méthode de descente simple (méthode de la première amélioration). Partant d'une solution initiale s , on choisit une solution s' dans le voisinage $\mathcal{N}(s)$. Si la solution s' a un coût meilleur que s , alors on accepte cette solution comme nouvelle solution s et on recommence le processus jusqu'à ce qu'un optimum local soit atteint.

Une version plus efficace de la méthode de la première amélioration est la méthode de la meilleure amélioration. Au lieu de choisir la première solution voisine s' de qualité supérieure à s , on choisit toujours la meilleure solution s' du voisinage s . L'algorithme 3.2 illustre cette méthode.

Algorithme 3.1 : Méthode de la première amélioration

Entrées : S, \mathcal{N}, f
Sorties : une solution s correspondant à un optimum local

```

1 début
2   initialiser : générer une solution initiale  $s \in S$ 
3   répéter
4     rechercher dans le voisinage : choisir une solution  $s' \in \mathcal{N}(s)$ 
5     si  $f(s') > f(s)$  alors
6        $s \leftarrow s'$ 
7     fin
8   jusqu'à  $f(s') \leq f(s), \forall s' \in \mathcal{N}(s)$ 
9   retourner  $s$ 
10 fin

```

Algorithme 3.2 : Méthode de la meilleure amélioration

Entrées : S, \mathcal{N}, f
Sorties : une solution s correspondant à un optimum local

```

1 début
2   initialiser : générer une solution initiale  $s \in S$ 
3   répéter
4     rechercher dans le voisinage : choisir une solution  $s' \in \mathcal{N}(s)$  tel que  $f(s') > f(s)$  et
5      $\forall s'' \in \mathcal{N}(s), f(s'') \leq f(s')$ 
6      $s \leftarrow s'$ 
7   jusqu'à  $f(s') \leq f(s), \forall s' \in \mathcal{N}(s)$ 
8   retourner  $s$ 
9 fin

```

3.2.2 Recherche tabou

Les racines de la recherche tabou remontent aux années 70 [Glover, 1977]. Cette méthode a d'abord été présentée par Glover [Glover, 1986]; les idées de base ont aussi été esquissées par [Hansen, 1986]. D'autres efforts supplémentaires de la formalisation de cette méthode sont présentés dans [Glover, 1989; Glover, 1990a; Glover et Laguna, 1998].

La recherche tabou se base sur l'utilisation de structures de mémoire. Le principe de base consiste à choisir la meilleure solution du voisinage de la solution courante, parfois moins bonne que la solution courante. Par conséquent, la recherche ne s'arrête pas au premier optimum local trouvé. Le danger est alors de revenir à des solutions déjà explorées. Pour éviter des cycles, on mémorise les dernières solutions visitées dans une liste tabou et on interdit tout mouvement qui conduit à une solution de cette liste. La liste tabou est donc une sorte de mémoire à court terme. Tout mouvement qui nous mène de la solution courante à une solution de la liste tabou est appelé mouvement tabou.

Lorsque la liste tabou est trop complexe ou occupe une grande place de mémoire, on mémorise plutôt des attributs des solutions ou des modifications des solutions que des solutions entières. La mémorisation de ces attributs ou modifications permet certes un gain en place mémoire, mais elle n'a pas que des avantages.

3.2 Recherche locale

Le stockage des attributs ou des modifications des solutions, permet d'identifier une solution de bien meilleure qualité avec un statut tabou. Accepter tout de même cette solution revient à dépasser son statut tabou, c'est l'application du *critère d'aspiration*. Ce mécanisme permet alors de lever le statut tabou d'une solution, sans pour autant introduire un risque de cycles dans le processus de recherche. La fonction d'aspiration la plus répandue est celle qui consiste à révoquer le statut tabou d'un mouvement tabou si ce dernier permet d'atteindre une solution de qualité supérieure à celle de la meilleure solution trouvée jusqu'alors.

Un facteur critique est la taille de la liste tabou. Si sa valeur est trop petite, des cycles peuvent se produire dans le processus de recherche. Au contraire, si sa valeur est trop grande, des mouvements de bonne qualité peuvent être interdits et mènent à l'exploration de solutions de qualité plus basse. Certains auteurs, comme [Taillard, 1991], proposent l'utilisation d'une liste tabou de longueur variable. D'autres auteurs, comme [Battiti et Techioiii, 1994], préfèrent les listes tabou réactives qui varient selon les résultats de la recherche.

La recherche tabou est décrite dans l'algorithme 3.3.

Algorithme 3.3 : Méthode de recherche tabou

```
Entrées :  $S, \mathcal{N}, f$ 
Sorties : une solution  $s^*$  correspondant à un optimum
1 début
2   initialiser : générer une solution initiale  $s \in S$ 
3    $s^* \leftarrow s$ 
4    $\mathcal{T} \leftarrow \emptyset$ 
5   répéter
6     construire  $\mathcal{N}^*(s) = \{s'' \in \mathcal{N}(s) \setminus \mathcal{T}\}$ 
7     choisir une solution  $s' \in \mathcal{N}^*(s) : f(s') \geq f(s''), \forall s'' \in \mathcal{N}^*(s)$ 
8     mettre à jour  $\mathcal{T}$  avec  $s$ 
9     si  $f(s') > f(s^*)$  alors  $s^* \leftarrow s', s \leftarrow s^*$ 
10  jusqu'à satisfaire un critère d'arrêt
11  retourner  $s^*$ 
12 fin
```

Dans cette méthode, on peut utiliser comme critère d'arrêt par exemple un nombre maximum d'itérations sans amélioration de la solution courante ou la meilleure solution trouvée, ou un temps d'exécution limité.

Nous invitons le lecteur intéressé par plus de détails sur la recherche tabou à consulter, par exemple, les articles de synthèse [Glover, 1990b; Glover *et al.*, 1993; Glover et Laguna, 1993; Hertz et Werra, 2003; Gendreau, 2003] et livres suivants [Reeves, 1993; Glover et Laguna, 1998].

3.2.3 Recherche locale itérée

La méthode de recherche locale itérée est une variante des méthodes de descente qui remédie au problème de l'arrêt de ces dernières dans un optimum local. Dans l'ordre chronologique, il existe des travaux tels que [Baxter, 1981; Baum, 1986; Martin *et al.*, 1991]

qui présentent des approches proches. Cependant, le nom de la méthode pourrait revenir sans aucun doute à [Stutzle, 1999; Lourenço *et al.*, 2002b].

Lorsque la méthode de descente reste bloquée dans un optimum local, il existe plusieurs manières de s'enfuir afin de poursuivre la recherche, sans obligatoirement changer de voisinage.

La recherche locale itérée est décrite dans l'algorithme 3.4.

Algorithme 3.4 : Méthode de recherche locale itérée

```

Entrées :  $S, N, f$ 
Sorties : une solution  $s'$  correspondant à un optimum
1 début
2   initialiser : générer une solution initiale  $s \in S$ 
3    $s' \leftarrow \text{RechercheLocale}(s)$ 
4   répéter
5      $s^* = \text{Perturbation}(s', \text{histoire})$ 
6      $s'' = \text{RechercheLocale}(s^*)$ 
7      $s' = \text{CritèreAcceptation}(s', s'', \text{histoire})$ 
8   jusqu'à satisfaire un critère d'arrêt
9   retourner  $s'$ 
10 fin

```

On distingue quatre étapes importantes: la génération de la solution initiale, la recherche locale (partielle ou interne), la perturbation sur une solution donnée et l'exécution d'un critère d'acceptation.

La solution initiale peut être générée de deux manières différentes : soit tirage au hasard ou soit par une heuristique constructive.

Idéalement, la *recherche locale* qui constitue la part substantielle de la recherche locale itérée devrait toujours rendre un optimum local de qualité. En plus, puisqu'elle s'exécute à chaque itération de la recherche locale itérée, une recherche locale rapide et performante est préférable.

L'étape de perturbation prend une solution localement optimale s' et produit une autre solution s^* pour recommencer la recherche locale dans l'itération suivante.

L'optimum local résultant s^* doit passer le critère d'acceptation pour être désigné comme la nouvelle solution courante s' . Le critère d'acceptation qui favorise la diversification accepte toutes les solutions perturbées. Un autre choix est qu'elle doit accepter seulement les solutions qui sont des améliorations à la valeur généralement optimale.

De même que d'autres méthodes de recherche locale, la recherche locale itérée s'arrête lorsqu'un nombre maximum de cycles est atteint ou que la solution ne s'améliore plus.

Le lecteur souhaitant plus de détails peut consulter les articles [Besten *et al.*, 2001; Chiarandini et Stützle, 2002] et les livres [Hoos et Stützle, 2004; Lourenço *et al.*, 2002a].

3.3 Application de la recherche locale au problème de la sélection de gènes

Le problème de la sélection de gènes est un problème très difficile. C'est pourquoi, les métaheuristiques à base de voisinage ou méthodes de recherche locale semblent bien appropriées pour traiter ce problème. Ainsi, nous traitons le problème de la sélection de gènes comme un problème combinatoire de maximisation (S, f) , c'est-à-dire qu'on cherche un sous-ensemble de gènes afin de construire un classifieur avec la performance maximum prédisant le type de tumeur qui caractérise un échantillon.

Plus exactement, nous formulons ce problème de maximisation (S, f) , tel que :

- l'espace de recherche S est défini par les sous-ensembles de gènes sélectionnés possibles et,
- la fonction objectif $f : S \rightarrow \mathbb{R}$ est telle que $\forall s \in S$, $f(s)$ représente le taux de classification d'un classifieur construit avec le sous-ensemble de gènes s .

Dans cette section nous présentons les adaptations des différents éléments qui constituent les méthodes de recherche locale décrites ci-dessus pour aborder ce problème.

D'autre part, on remarque que le problème de la sélection de gènes est de grande taille. Afin d'explorer et exploiter des zones intéressantes de l'espace de recherche nous avons envisagé de limiter cet espace. Pour cela, notre modèle débute par une phase de pré-sélection réalisée grâce à la méthode de filtrage BW (voir sous-section 2.4.1). Cette pré-sélection permet de restreindre l'espace de recherche et nous fournit un groupe initial G_p des gènes sur lequel nos algorithmes de recherche locale réalisent la sélection d'un sous-ensemble de gènes pertinents pour la classification. Le filtrage BW a été choisi, car il a donné des résultats très encourageants pour la construction de classifieurs très performants. Le nombre de gènes pré-sélectionnés p a été fixé à 75.

3.3.1 Espace de recherche et représentation

Pour le groupe G_p de p gènes pré-sélectionnés, l'espace de recherche est défini par l'ensemble $\Omega = 2^p$ qui contient tous les ensembles possibles de G_p .

Une solution $s = \langle s^g, s^c \rangle$ est composée de deux parties s^g et s^c appelées respectivement : vecteur de sous-ensemble de gènes et vecteur des coefficients du classement. La première partie $s^g = (g_1, g_2 \dots g_p)$ est un vecteur binaire de longueur fixe p . Chaque position g_i représente un gène particulier, g_i est un nombre binaire qui indique si le gène est sélectionné (valeur 1) ou non (valeur 0). La seconde partie $s^c = (c_1, c_2 \dots c_p)$ est un vecteur réel positif de longueur fixe p et correspond au vecteur des coefficients du classement c issu du SVM linéaire construit sur le sous-ensemble sélectionné s^g . s^c indique la contribution de chaque gène à la fonction de classification.

Par conséquent, une solution représente un sous-ensemble de gènes candidat avec des informations de classement supplémentaires sur chaque gène sélectionné. Un sous-ensemble de gènes est évalué par un classifieur SVM linéaire et les coefficients du classement obtenus pendant cette évaluation seront utilisés dans nos stratégies de recherche locale spécialisées.

3.3.2 Fonction d'évaluation

Étant donné une solution candidate $s = \langle s^g, s^c \rangle$, la qualité de s (plus précisément, de la partie sous-ensemble s^g) est évaluée par une fonction d'évaluation f selon deux critères : la possibilité d'obtenir une bonne classification avec ce sous-ensemble de gènes (C) et la marge maximale (M) donnée par le classifieur SVM construit sur s^g (équation 3.1). D'une manière formelle, la fonction d'évaluation peut être écrite comme suit :

$$f(s) = \langle f_C(s), f_M(s) \rangle \quad (3.11)$$

où

- $f_C(s)$ est la précision de classification du classifieur SVM construit sur l'ensemble de s^g et appliqué aux données d'apprentissage,

- $f_M(s)$ est la marge maximale de ce classifieur SVM, donnée par l'équation 3.11.

Étant donné deux solutions candidates s et s' , il est possible de les comparer de la façon suivante: $f(s)$ est meilleure que $f(s')$, dénoté par $f(s) > f(s')$, si la précision de la classification est meilleure pour s ou si les deux solutions donnent le même taux de classification correcte avec une marge supérieure pour s .

$$f(s) > f(s') \Leftrightarrow (f_C(s) > f_C(s')) \text{ ou } (f_C(s) = f_C(s') \wedge f_M(s) > f_M(s')) \quad (3.12)$$

Ainsi, le critère dominant est la précision de classification, les égalités sont départagées en comparant les marges maximales, avec une préférence pour une plus grande valeur (une plus grande marge indique une meilleure discrimination entre les deux classes).

3.3.3 Mouvement et voisinage

Une des caractéristiques les plus importantes d'un algorithme de recherche locale est son voisinage. Dans un algorithme de recherche locale, en appliquant un opérateur de mouvement mv à une solution candidate s , cela mène à une nouvelle solution s' , dénotée par $s' = s \oplus mv$. Soit $\Gamma(s)$ l'ensemble de tous les mouvements possibles qui peuvent être appliqués à s , alors le voisinage $\mathcal{N}(s)$ de s est défini par :

$$\mathcal{N}(s) = \{s \oplus mv | mv \in \Gamma(s)\} \quad (3.13)$$

Dans notre cas, le mouvement est basé sur l'opération enlever/ajouter qui enlève un gène g_i de la solution s et ajoute un autre gène g_j . De plus, l'opérateur de mouvement est défini de telle façon qu'il intègre les connaissances sémantiques de la sélection de gènes et du problème de classification. Plus formellement, soit $s = \langle s^g, s^c \rangle$ avec $s^g = (g_1, g_2 \dots g_p)$ et $s^c = (c_1, c_2 \dots c_p)$, et de plus :

- $i = \text{ArgMin}_j \{c_j | c_j \in s^c \wedge c_j \neq 0\}$, i.e. i identifie le gène g_i qui a le coefficient de classement le plus faible c_i et il est ainsi le gène le moins influent pour la classification,
- $O_s = \{j \in [0, \dots, p] | g_j = 0\}$, i.e. O_s est l'ensemble de gènes non sélectionnés dans la solution courante s

3.3 Application de la recherche locale au problème de la sélection de gènes

Alors notre opérateur de mouvement enlève, de la solution courante, g_i (identifié par l'indice i ci-dessus) qui est le gène le moins influent parmi les gènes sélectionnés et ajoute un gène non sélectionné g_j ($j \in O_s$). Cela peut être formellement écrit comme :

$$mv(i, j) = (g_i : 1 \rightarrow 0; g_j : 0 \rightarrow 1) \quad (3.14)$$

Dans le voisinage de s on peut remarquer que pour deux solutions voisines $s = \langle s^g, s^c \rangle$ et $s' = \langle s'^g, s'^c \rangle$, la distance de hamming entre s^g et s'^g est exactement deux. On voit que la taille de ce voisinage est égale à $|O_s|$ et bornée par p , la longueur de s . De plus, toutes les solutions comportent le même nombre de gènes sélectionnés. L'exploration d'un voisinage va donc considérer différents ensembles de gènes de même taille.

3.3.4 Solution initiale

La solution initiale peut être créée au hasard au risque d'avoir une solution de mauvaise qualité. C'est pourquoi, nous proposons une façon simple d'obtenir une solution initiale qui ne soit pas trop mauvaise. Nous générons au hasard l solutions telles que le nombre de gènes dans chaque solution varie entre $p * 0.9$ et $p * 0.6$, où p est le nombre de gènes pré-sélectionnés. Parmi ces l solutions, nous retenons la meilleure solution selon la fonction d'évaluation (voir l'équation 3.11).

La solution initiale fournit l'ensemble de gènes initial s^g utilisé par la méthode de recherche locale. Le vecteur s^c de cette solution initiale est calculé en construisant un classifieur SVM avec s^c pour représentation des données.

3.3.5 Schéma général

La procédure générale *SdG* pour la sélection de gènes est montrée dans l'algorithme 3.5. Elle est composée de deux phases principales répétées : la phase de recherche locale proprement dite qui explore le voisinage de la solution courante pour la sélection de gènes et la phase de réduction de gènes. À la ligne 6, un algorithme RL_{SVM} (avec n'importe lequel des algorithmes de recherche locale ci-dessus) est utilisé pour rechercher un bon sous-ensemble de gènes d'une taille donnée. Après chaque phase de recherche locale, à la ligne 7, la réduction d'un gène est réalisée en éliminant le gène le moins pertinent pour la classification (i.e, le gène qui a le plus faible coefficient de classement) du bon sous-ensemble de gènes donné par la phase RL_{SVM} . À partir de ce sous-ensemble de gènes, une nouvelle recherche de RL_{SVM} est réappliquée.

Ce processus en deux étapes s'arrête lorsque l'élimination d'un gène donne un sous-ensemble de moins bonne qualité, c'est-à-dire pour lequel la précision de classification sur les données d'apprentissage est plus faible.

Algorithme 3.5 : Procédure générale *SdG*

Entrées : G_p , i.e. un groupe de p gènes pré-sélectionnés par le filtrage *BW*
Sorties : s^g , un ensemble de gènes sélectionnés

```

1  début
2  |   générer un ensemble de gènes initial  $s^g$ 
3  |   répéter
4  |   |   évaluer la qualité de  $s^g$  par le classifieur SVM sur les données d'apprentissage et remplir  $s^c$ 
5  |   |    $s \leftarrow (s^g, s^c)$  /*  $s$  est la solution courante */
6  |   |    $s \leftarrow RL_{SVM}(s)$  /* phase RL : appliquer un algorithme de recherche locale basé sur un SVM
7  |   |   |   pour améliorer la solution courante  $s \leftarrow (s^g, s^c)$  */
8  |   |   |    $s^g = s^g - \{g_i\}$  /* phase de réduction d'un gène : élimine le gène moins pertinent pour la
9  |   |   |   |   classification de la meilleure solution trouvée par  $RL_{SVM}$  */
10 |   |   jusqu'à ce qu'un critère d'arrêt soit vérifié
11 |   retourner  $s$ 
12 fin

```

3.4 Comparaison d'algorithmes de recherche locale

3.4.1 Conditions expérimentales

La façon sans doute la plus répandue de tenir compte du biais de sélection est le modèle en deux étapes (voir section 2.5). Dans la première étape, il est nécessaire d'appliquer l'algorithme d'apprentissage pour la sélection de gènes sur un ensemble d'apprentissage afin de construire un classifieur prédisant le type de tumeur qui caractérise un échantillon cellulaire et seulement ensuite, on passe à la deuxième étape pour exécuter une validation sur l'ensemble de test qui n'avait pas été vu pendant l'étape de sélection sur un ensemble de données d'entraînement. De ce fait, nous avons adopté ce modèle dans nos expériences.

Pour chaque algorithme de recherche locale et chaque jeu de données (voir la section 1.2), nous avons réalisé 50 expériences. Le diagramme de la figure 3.3 schématise le fonctionnement de notre méthode proposée pour la sélection de gènes pendant une expérience. D'abord, les deux ensembles d'apprentissage (A) et de test (T), de taille fixée, sont tirés au hasard, tout en s'assurant que les proportions de positifs et de négatifs sont respectées dans les deux ensembles. Notre méthode débute par une phase de pré-sélection réalisée grâce au critère de filtrage, dans notre cas, le ratio *BW* obtient un groupe G_p de p gènes (typiquement, $p \geq 75$). L'ensemble de gènes initial s^g (obtenu à partir de la solution initiale) est alors utilisé par l'algorithme RL_{SVM} pour déterminer un bon sous-ensemble de gènes (de taille optimale et avec un bon taux de classification sur les échantillons de A). Enfin, le sous-ensemble de gènes sélectionnés est évalué sur les échantillons de test, T , en utilisant le classifieur SVM. Les résultats du taux de classification et du nombre de gènes sélectionnés sont utilisés pour calculer les statistiques moyennes.

Les comparaisons s'appuient sur deux critères : la moyenne et l'écart-type du taux de classification (*moy* et *e.t.*) sur les échantillons de test et la moyenne et écart-type du nombre de gènes sélectionnés pour ces 50 expériences. Le temps de calcul n'est pas exposé, cependant, notez qu'une expérience prend environ 20 minutes sur un ordinateur

3.4 Comparaison d'algorithmes de recherche locale

de type Pentium Centrino Duo, 1.2Ghz. Notre algorithme a été implémenté en langage MATLAB, en utilisant le classifieur SVM linéaire développé par Steven Gunn ¹.

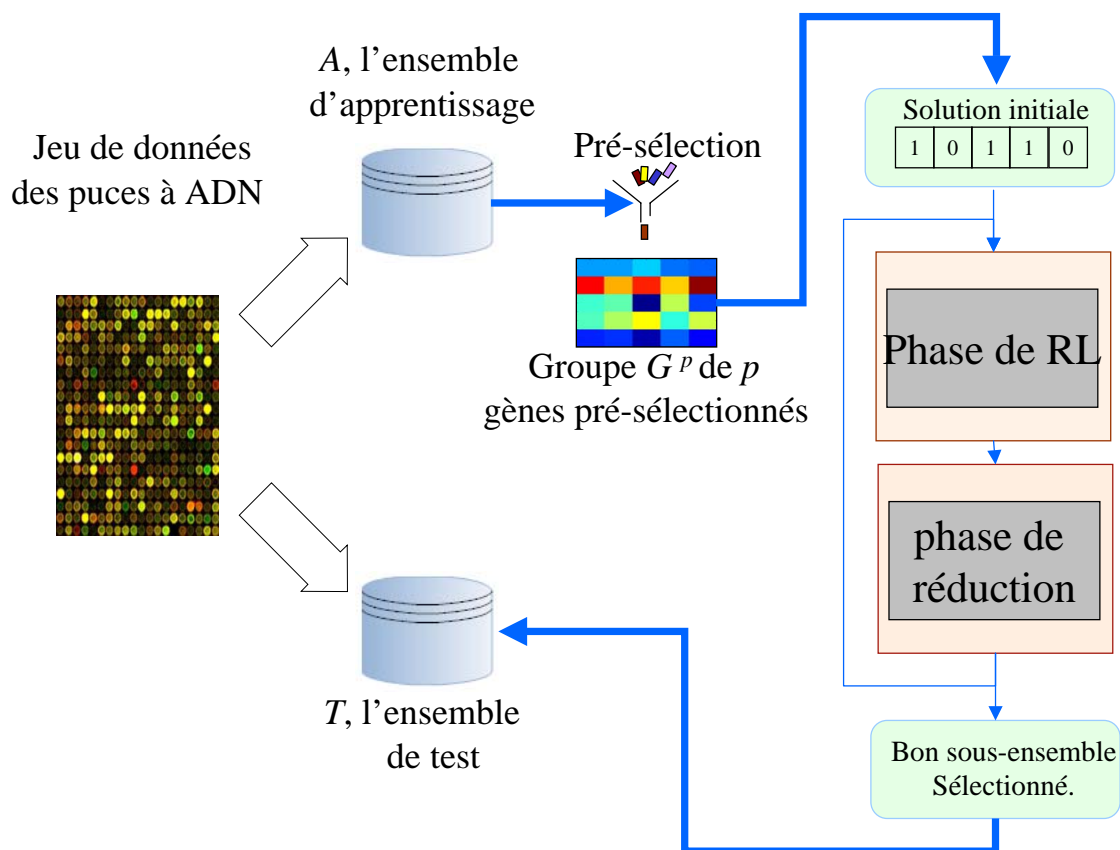


Figure 3.3 – Recherche locale pour la sélection de gènes des données de puces à ADN.

3.4.2 Stratégie de descente spécifique

Typiquement, la descente pure accepte seulement des solutions voisines de meilleur coût et s'arrête lorsqu'un optimum est atteint. Pour choisir la solution voisine qui sera acceptée, nous avons implémenté les deux stratégies de descente. Par la suite, une solution quelconque se rapporte à un ensemble de gènes sélectionnés à un moment donné.

- la méthode de la première amélioration notée par D^p , sélectionne à chaque itération la première solution voisine trouvée s' de qualité supérieure par rapport à la solution courante s , c'est-à-dire $s' \in \mathcal{N}(s)$ tel que $f(s') > f(s)$
- la méthode de la meilleure amélioration notée par D^m , évalue à chaque itération toutes les solutions voisines pour sélectionner la meilleure solution voisine s' , c'est-à-dire $s' \in \mathcal{N}(s)$ tel que $f(s') > f(s)$ et $\forall s'' \in \mathcal{N}(s), f(s'') \leq f(s')$

¹<http://www.isis.ecs.soton.ac.uk/resources/svminfo/>

Dans la figure 3.4 nous montrons avec un exemple l'application de la méthode de la meilleure amélioration. Observez dans la figure 3.4.a que le gène le moins influent se trouve à la position 6. Le mouvement consiste donc à supprimer ce gène et à ajouter un des gènes de O_s , c'est-à-dire un des gènes 3, 4, 5 ou 9. Le tableau de la figure 3.4.b montre les résultats d'un SVM construit sur $\Gamma(s)$, l'ensemble de tous mouvements possibles qui peuvent être appliqués à s^g . Pour le mouvement $mv(6, 3)$, si le gène 3 est ajouté, on travaille donc avec les gènes sélectionnés 1 2 3 7 8 10 et on a un taux de classification $f_C = 98\%$ et une marge f_M de 0.9251. Le mouvement $mv(6, 4)$ donne la meilleure solution parmi l'ensemble de tous les mouvements possibles $\Gamma(s)$ qui peuvent être appliqués. Le voisin retenu est illustré dans la figure 3.4.c.

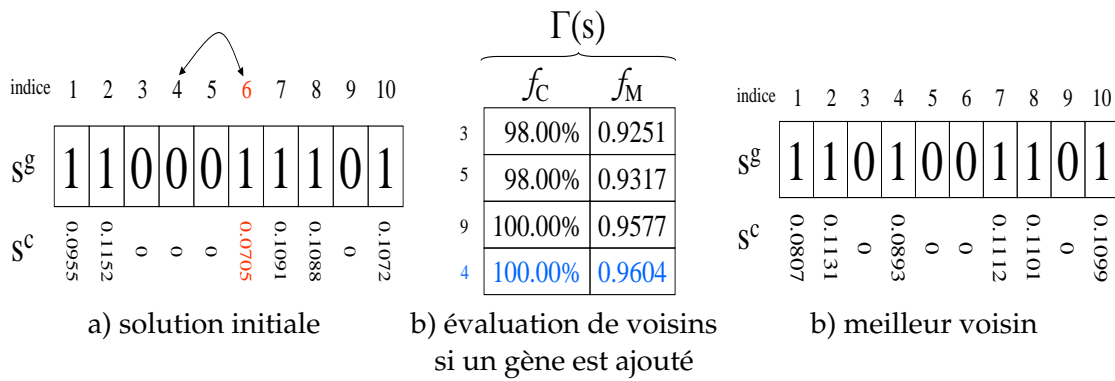


Figure 3.4 – Méthode de la meilleure amélioration

Les tableaux de résultats 3.1 et 3.2 ci-après montrent dans la colonne **Descente** les résultats obtenus par chaque algorithme (D^p et D^m) par rapport aux neuf jeux de données. D'un côté, le premier tableau indique la moyenne et l'écart-type du taux de classification obtenus sur l'ensemble de test sur les 50 expériences. D'un autre côté, le deuxième tableau expose la moyenne et l'écart-type du nombre de gènes sélectionnés obtenus sur l'ensemble de test sur les mêmes expériences. Dans le premier tableau on a indiqué le nombre d'échantillons dans l'ensemble d'apprentissage et dans l'ensemble de test.

On constate sur les tableaux 3.1 et 3.2 que l'algorithme D^m a obtenu les meilleurs résultats par rapport à l'algorithme D^p sur tous les jeux de données.

Néanmoins, nous nous sommes aperçus que dans certaines expériences, l'algorithme D^p a obtenu des résultats tout à fait compétitifs. Nous avons donc décidé d'ajouter une liste tabou à ces deux algorithmes, en cherchant à améliorer les résultats déjà obtenus.

3.4.3 Recherche Tabou spécifique

À partir des algorithmes de descente (D^p , D^m), nous obtenons les algorithmes de recherche tabou de base (RT^p et RT^m) en ajoutant la liste tabou. Les détails de ces algorithmes sont présentés dans la suite de cette sous-section. À chaque itération, la solution courante s est remplacée par la solution voisine s' fournie par une stratégie de descente

3.4 Comparaison d'algorithmes de recherche locale

si elle n'appartient pas à la liste tabou i.e.

$$s' \in \mathcal{N}(s) \text{ tel que } \forall s'' \in \mathcal{N}(s), f(s'') \leq f(s') \text{ et } s' \notin \bar{S} \quad (3.15)$$

où \bar{S} est l'ensemble de solutions qui appartiennent à la liste tabou. Remarquez que contrairement aux algorithmes de descente, la solution voisine sélectionnée s' peut ne pas être meilleure que s .

Un tel choix peut occasionner des cycles. Le rôle essentiel de la liste tabou est de prévenir le bouclage de l'algorithme. Dans notre cas, la liste tabou est mise en oeuvre comme suit. Chaque fois qu'un mouvement $mv(i, j)$ est effectué, i.e. le gène g_i est enlevé et le gène g_j est sélectionné, g_i est enregistré dans la liste tabou pour les k itérations suivantes. Par conséquent, g_i ne peut pas être resélectionné pendant cette période. La valeur de k a été déterminée expérimentalement et varie typiquement de $k_{min} = 3$ à $k_{max} = 10$.

Remarquez qu'une telle liste tabou n'interdit pas à un gène récemment sélectionné g_j d'être enlevé juste après sa sélection si son coefficient de classement est très faible.

On reprend l'exemple déjà donné figure 3.4 en ajoutant une liste tabou. On suppose que cette liste tabou contient déjà la valeur 3. Pour la solution s le mouvement choisi est $mv(6, 4)$: 6 est le gène le moins pertinent et il est supprimé. Il est donc ajouté en tête de la liste tabou qui devient (6, 3) (voir figure 3.5).

$$\begin{array}{ccc} \text{Liste tabou : (3)} & & \text{Liste tabou : (6, 3)} \\ \mathbf{S} - 1 \ 2 \ \mathbf{6} \ 7 \ 8 \ 10 & \xrightarrow{mv(6,4)} & \mathbf{S} - 1 \ 2 \ \mathbf{4} \ 7 \ 8 \ 10 \end{array}$$

Figure 3.5 – Exemple de fonctionnement de la liste tabou.

Pour le choix du gène ajouté à l'ensemble de gènes sélectionnés, remarquons ici que le gène 3 qui figure dans la liste tabou n'est pas considéré (figure 3.6). On a donc seulement 3 mouvements possibles.

		$\Gamma(s)$	
gène		f_C	f_M
3		-	-
5		98.00%	0.9317
9		100.00%	0.9577
4		100.00%	0.9604

Figure 3.6 – Exemple de l'ensemble des mouvements possibles considérés.

Les résultats comparatifs sur tous les jeux de données entre ces deux algorithmes de

recherche tabou sont aussi présentés dans les tableaux 3.1 et 3.2 à la colonne Recherche tabou. Notez que RT^p et RT^m ont amélioré les résultats par rapport aux algorithmes de descente. De plus RT^m donne de meilleures moyennes que RT^p .

3.4.4 Recherche locale itérée spécifique

Notre recherche locale itérée utilise la recherche tabou RT^m jusqu' à ce qu'elle reste bloquée dans un optimum local; alors, la recherche applique un opérateur de perturbation à la solution optimum locale pour s'en extraire afin de relancer la recherche. Nous considérons la combinaison avec la RT^m , dénotée par RLI^{RT} car c'est la meilleure combinaison que nous avons trouvée grâce aux expériences précédentes. Plus exactement, RLI^{RT} itère deux phases : une phase de recherche locale pour atteindre un optimum local et une perturbation pour diversifier la recherche. Notre opérateur de perturbation change le meilleur optimum local d'une façon contrôlée et est basé sur la fonction d'évaluation pour continuer le processus de recherche. Autrement, la recherche s'arrête.

Nous allons présenter un cas pour lequel notre opérateur de perturbation est appliqué (voir la figure 3.7). Avant d'effectuer un mouvement, nous vérifions que la nouvelle solution améliore la solution courante, dans le cas contraire, à partir de cette solution obtenue par la recherche tabou RT^m , nous appliquons une perturbation à la solution courante. Plus exactement, pour la solution s le mouvement évalué est $mv(6, 7)$, car il correspond à la meilleure solution trouvée parmi l'ensemble de tous les mouvements possibles à appliquer (figure 3.8.a). Cependant, constatez dans la figure 3.8.b que si on ajoute 7, toutes les solutions voisines que l'on peut avoir sont inférieures, donc, cette solution la plus performante est un optimum local.

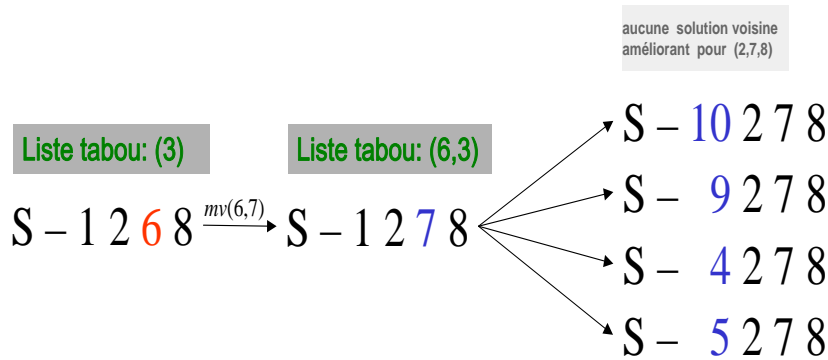


Figure 3.7 – solution optimum locale.

Ainsi, nous appliquons l'opérateur de perturbation, voir la figure 3.9. Encore une fois pour la solution s , on évalue un nouveau mouvement possible (relance), le $mv(6, 9)$ qui est la deuxième meilleure solution trouvée parmi l'ensemble de tous les mouvements possibles à appliquer (voir figure 3.10.a). Observez dans la figure 3.10.b. que si on ajoute 9, les deux premières solutions voisines que l'on peut obtenir sont supérieures ou égales, donc, cette solution permet de s'extraire de l'optimum local.

$\Gamma(s)$			$\Gamma(s)$		
gène	f_C	f_M	gène	f_C	f_M
3	-	-	6	-	-
5	96.00%	0.9317	3	-	-
4	98.00%	0.9577	5	96.00%	0.9533
10	98.00%	0.9604	4	98.00%	0.9721
9	100.00%	0.9767	9	98.00%	0.9756
7	100.00%	0.9804	10	98.00%	0.9784

a) solutions pour 1,2,8 b) solutions pour 2,7,8

Figure 3.8 – Parcours de $s - 1\ 2\ 6\ 8$ à $s - 1\ 2\ 7\ 8$

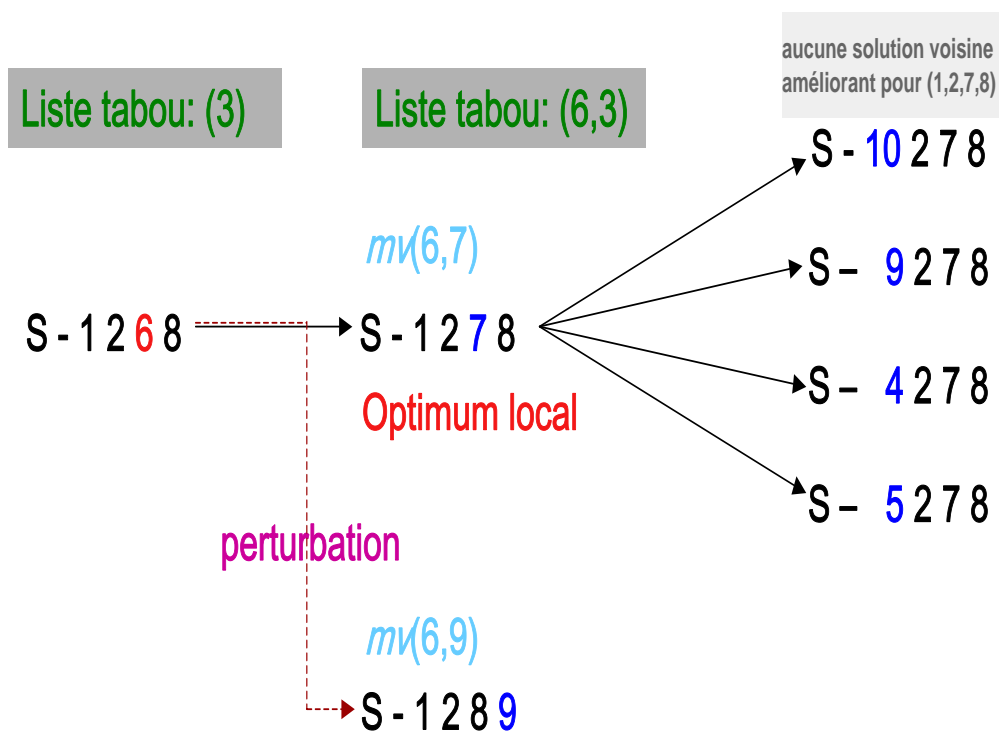


Figure 3.9 – Exemple d'une perturbation.

Le mouvement effectué est contrôlé, car on a considéré le deuxième meilleur mouvement, par rapport à la valeur donnée de la fonction d'évaluation. Dans une structure

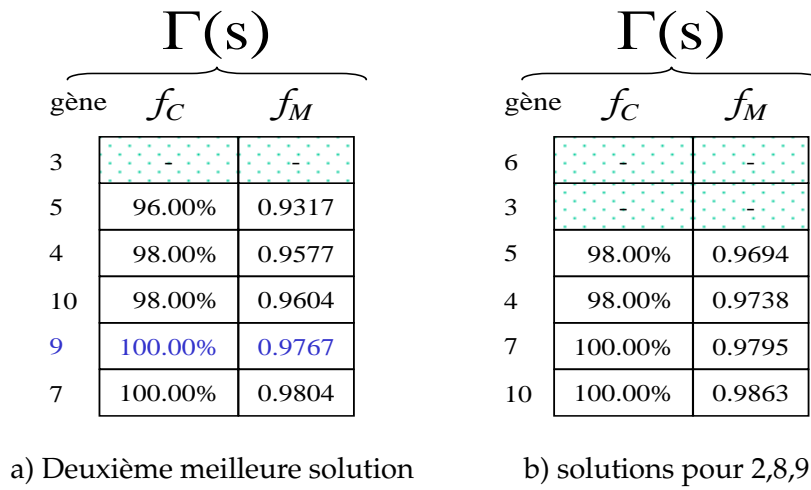


Figure 3.10 – Parcours de $s - 1\ 2\ 6\ 8$ à $s - 1\ 2\ 8\ 9$

de mémoire on stocke tous les mouvements possibles de s afin d'évaluer 5 relances en essayant de sortir du minimum local. Grâce à cette structure de mémoire, la perturbation génère un bon point de départ à partir d'une bonne solution. Il faut rappeler que la perturbation a une influence capitale sur le fonctionnement de la recherche locale itérée. Une trop grande perturbation effectuerait une relance aléatoire de la recherche et ne permettrait pas de converger. Nous utilisons l'information apportée par la fonction d'évaluation pour décider quelles nouvelles zones intéressantes on pourrait explorer et exploiter.

Nous illustrons également les résultats obtenus par la recherche locale itérée dans les tableaux 3.1 et 3.2 à la colonne RLI^{RT} , où la moyenne et l'écart type sur cinquante expériences du taux de classification et du nombre de gènes sélectionnés sur les neuf jeux de données sont présentés. Ces deux tableaux montrent clairement que le meilleur résultat obtenu entre toutes les méthodes de recherche locale est RLI^{RT} . Notez que pour le cas du jeu de la leucémie le taux moyen de classification est inférieur par rapport à D^m et RT^m . Néanmoins, le nombre moyen de gènes sélectionnés est meilleur. De plus, nous avons observé dans certains cas que RLI^{RT} a obtenu des meilleurs taux de classification avec un nombre de gènes sélectionnés plus petit par rapport aux autres méthodes de recherche locale (voir l'écart-type du taux de classification).

Après avoir comparé les méthodes de recherche locale, ci-dessus, tant en taux de classification qu'en nombre de gènes sélectionnés sur les neuf jeux de données, nous avons retenu la méthode RLI^{RT} . Ce choix est justifié par le fait que cette méthode atteint des meilleurs résultats.

3.5 Comparaison avec d'autres approches de sélection de gènes

Table 3.1 – Comparaison entre les algorithmes de recherche locale. Taux de classification.

Jeu de données	Découpage d'échantillons		Descente				Recherche tabou				Recherche locale itérée RLI^{RT}	
			D^P		D^m		RT^P		RT^m		moy.	e.t.
			moy.	e.t.	moy.	e.t.	moy.	e.t.	moy.	e.t.		
Colon	50	12	82.16%	9.81%	84.83%	9.17%	83.83%	9.13%	85.50%	8.21%	87.00%	7.36%
Leucémie	60	36	91.58%	3.94%	92.52%	3.42%	91.82%	2.54%	92.47%	3.36%	91.94%	4.06%
Lymphome ⁹⁶	38	34	90.88%	3.02%	92.11%	2.19%	91.27%	2.80%	92.44%	1.86%	95.44%	2.15%
Sein	78	19	78.53%	4.74%	79.37%	4.61%	82.84%	4.10%	84.21%	4.25%	89.58%	2.49%
Poumon	32	149	99.09%	0.70%	99.19%	0.58%	99.66%	0.39%	99.68%	0.39%	99.93%	0.20%
Prostate	102	34	92.65%	2.99%	93.71%	2.52%	95.29%	2.38%	95.88%	2.36%	98.41%	1.48%
Ovaire	135	118	99.25%	0.96%	99.80%	0.56%	99.95%	0.27%	99.98%	0.12%	100.00%	0.00%
Cerveau	30	30	75.27%	3.69%	76.53%	3.23%	78.33%	2.88%	80.20%	2.73%	84.00%	1.65%
Lymphome ⁴⁷	24	23	92.00%	3.33%	92.78%	3.12%	95.65%	3.04%	95.91%	2.97%	100.00%	0.00%

Table 3.2 – Comparaison entre les algorithmes de recherche locale. Nombre de gènes sélectionnés.

Jeu de données	Descente				Recherche tabou				Recherche locale itérée RLI^{RT}	
	D^P		D^m		RT^P		RT^m		moy.	e.t.
	moy.	e.t.	moy.	e.t.	moy.	e.t.	moy.	e.t.		
Colon	19.94	1.01	15.32	1.83	15.80	1.35	11.16	2.81	8.20	2.09
Leucémie	6.22	1.54	6.04	1.38	5.96	1.35	4.74	1.32	3.14	1.08
Lymphome ⁹⁶	17.58	2.36	17.04	2.44	17.34	1.61	14.32	2.21	12.46	1.58
Sein	13.20	9.94	11.56	9.49	11.36	8.01	9.68	7.82	7.90	6.27
Poumon	11.54	6.07	9.78	5.13	9.02	3.57	7.12	2.97	4.66	2.25
Prostate	7.60	7.38	7.44	6.69	6.86	6.17	5.82	4.24	5.08	4.10
Ovaire	5.48	3.50	4.86	2.21	4.68	1.79	3.54	0.91	2.52	0.54
Cerveau	17.48	8.62	13.90	6.84	13.08	5.95	9.12	5.16	9.06	4.24
Lymphome ⁴⁷	12.54	6.75	11.96	6.02	9.22	5.87	7.94	5.57	7.34	4.16

3.5 Comparaison avec d'autres approches de sélection de gènes

3.5.1 Comparaison avec SVM-RFE

Nous nous intéressons ici à la validation expérimentale des algorithmes de recherche locale présentés ci-dessus. L'approche proposée est un peu rattachée à l'approche connue de SVM-RFE [Guyon *et al.*, 2002]. Notez que SVM-RFE est complètement glouton, donc un gène incorrectement éliminé ne peut jamais être resélectionné ensuite. Le tableau 3.3 montre le résultat de SVM-RFE obtenu sous les mêmes conditions expérimentales (voir sous-section 3.4.1). En comparant les tableaux 3.1, et 3.2 contre 3.3, on observe que SVM-RFE fait mieux que les algorithmes D^P , D^m et RT^P , mais il est surpassé par les algorithmes

RT^m et RLI^{RT} . Cela confirme l'intérêt d'utiliser un algorithme de recherche locale pour explorer l'espace de recherche d'une taille fixe avant l'élimination d'un gène.

Table 3.3 – Résultats de l'algorithme SVM-RFE

Jeu de données	Taux de classification		Nombre de gènes sélectionnés	
	moy.	e.t.	moy.	e.t.
	Colon	85.16%	8.11%	18.32
Leucémie	92.35%	3.25%	4.82	2.39
Lymphome ⁹⁶	92.33%	3.96%	16.40	2.51
Sein	83.79%	4.11%	10.40	7.78
Poumon	99.67%	0.38%	8.10	3.14
Prostate	95.65%	2.47%	6.14	4.29
Ovaire	99.97%	0.17%	4.04	1.17
Cerveau	79.53%	3.09%	10.04	4.82
Lymphome ⁴⁷	95.74%	2.99%	9.06	5.55

3.5.2 Comparaison avec d'autres approches de l'état d'art

Le tableau 3.4 montre les résultats de huit autres algorithmes de sélection proposés récemment de la littérature. Nous avons choisi ces références car elles utilisent le même protocole expérimental ou semblable pour éviter le biais de sélection. Encore une fois, nous avons observé que notre approche de recherche locale basée sur un SVM (particulièrement, RT^m et RLI^{RT}) est très compétitive puisque ses résultats dominent souvent ces méthodes de référence avec un taux de classification plus haut et un sous-ensemble de gènes sélectionnés plus petit.

3.6 Conclusion

Nous avons présenté dans ce chapitre une approche de recherche locale basée sur un SVM, pour la sélection de sous-ensemble de gènes avec deux caractéristiques originales. En premier lieu, la fonction d'évaluation de nos algorithmes de recherche locale est basée non seulement sur l'exactitude de classification donnée par le classifieur SVM, mais aussi sur sa marge maximale. En second lieu, l'information du classement fournie par le classifieur SVM est explicitement exploitée par les algorithmes de recherche locale. Ces deux caractéristiques garantissent que l'approche de recherche locale basée sur un SVM est complètement adaptée au problème visé et constitue son fondement basique.

En utilisant un protocole expérimental qui évite le problème de biais de sélection, dans un premier temps, les différents algorithmes de recherche locale (D^p , D^m , RT^p , RT^m et RLI^{RT}) sont comparés sur neuf jeux de données souvent utilisés dans ce domaine et

3.6 Conclusion

Jeu de données	RLI^{RT}	Références							
		[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Colon	87.00±7.36	85.83±2.0	82.33±9.0	85.71	87.10±4.47	82.03±7.49	-	-	95.16
	8.20±2.09	20	20	30	-	-	-	-	-
Leucémie	91.94±4.06	-	-	-	94.86±3.10	94.40±3.84	-	-	91.18
	3.14±1.08	-	-	-	-	-	-	-	-
Lymphome ⁹⁶	95.44±2.15	91.57±0.9	92.28±4.0	-	-	-	-	-	-
	12.46±1.58	20	20	-	-	-	-	-	-
Sein	89.58±2.49	-	-	-	-	68.42±7.62	-	-	89.47
	7.90±6.27	-	-	-	-	-	-	-	-
Poumon	99.93±0.20	-	-	-	99.16±0.75	-	98.66	-	93.29
	4.66±2.25	-	-	-	-	-	8	-	-
Prostate	98.41±1.48	-	-	94.11	93.57±4.10	88.10±4.93	94.12	-	73.53
	5.08±4.10	-	-	20	-	-	22	-	-
Ovaire	100.00±0.0	-	-	94.82	99.56±0.92	-	97.46	-	-
	2.52±0.54	-	-	20	-	-	14	-	-
Cerveau	84.00±1.65	-	-	-	83.98±6.36	-	-	-	88.33
	9.06±4.24	-	-	-	-	-	-	-	-
Lymphome ⁴⁷	100.00±0.0	-	-	-	-	-	-	84.61	-
	7.34±4.16	-	-	-	-	-	-	50	-

1. [Weston *et al.*, 2002]
2. [Rakotomamonjy, 2003]
3. [Orsenigo, 2008]
4. [Xiong *et al.*, 2007]
5. [Pochet *et al.*, 2004]
6. [Shah et Kusiak, 2007]
7. [Li *et al.*, 2001]
8. [Tan et Gilbert, 2003]

Table 3.4 – Comparaison avec d'autres méthodes de référence

qui concernent le cancer du colon, de la leucémie, du sein, du poumon, de la prostate, de l'ovaire, du cerveau et deux lymphomes. Les résultats obtenus de ces expérimentations montrent que RLI^{RT} est le meilleur algorithme de recherche locale car il a remporté les meilleurs taux de classification et les meilleurs nombres de gènes sélectionnés (moyennes sur cinquante expériences).

Dans un second temps, RLI^{RT} a été comparé avec huit autres algorithmes de sélection de gènes de l'état d'art. Les résultats expérimentaux indiquent clairement que RLI^{RT} rivalise très bien avec les méthodes de référence du point de vue du taux de classification et du nombre de gènes sélectionnés. L'approche proposée a un avantage supplémentaire et important sur les méthodes de filtre et SVM-RFE. En effet, l'approche de recherche locale basée sur un SVM nous permet de générer de multiples sous-ensembles de gènes de haute qualité, qui peuvent être utilisés pour une nouvelle analyse et dans le but de fouille de données.

Cette étude montre que la recherche locale constitue une approche simple, cependant puissante pour la sélection de gènes et la classification de données de puces à ADN. Son

efficacité dépend fortement de la manière dont les informations sémantiques du problème donné sont intégrées dans ses opérateurs fondamentaux comme la fonction d'évaluation et le voisinage. Finalement, il est clair que l'approche proposée peut facilement être combinée avec d'autres méthodes de classification et des informations du classement.

Chapitre 4

Algorithme génétique spécifique pour la sélection de gènes

Sommaire

4.1 Algorithmes génétiques	63
4.1.1 Représentation des solutions	63
4.1.2 Fonction d'évaluation	63
4.1.3 Opérateurs génétiques	64
4.1.4 Phase de remplacement	68
4.1.5 Schéma général d'un algorithme génétique	68
4.2 Sélection de gènes par algorithme génétique	69
4.2.1 Représentation	69
4.2.2 Fonction d'évaluation	70
4.2.3 Opérateur de croisement spécifique	70
4.2.4 Opérateur de mutation spécifique	73
4.2.5 Initialisation	74
4.2.6 Sélection et Remplacement	75
4.2.7 Résultats comparatifs	75
4.2.8 Comparaison des opérateurs de croisement	75
4.2.9 Comparaison avec d'autres approches avec un nombre de gènes fixe	77
4.2.10 Comparaison avec d'autres approches	77
4.3 Conclusion	80

DANS CE chapitre, nous proposons une approche intégrée qui réalise la sélection d'un sous-ensemble de gènes par un algorithme génétique. Les caractéristiques principales de l'approche proposée concernent les opérateurs de croisement et de mutation spécialisés qui prennent en compte l'information de classement fourni par un classifieur SVM.

Nous comparons, d'une part la qualité des solutions générées par notre opérateur que nous avons appelé *GeSeX* (GENe SElection Crossover) avec des opérateurs des croisement connus et, d'autre part, la performance de notre algorithme génétique avec d'autres approches de sélection de gènes.

Les travaux présentés dans cette partie ont fait l'objet de deux publications dans les conférences *EvoBio'07* et *EA'07* [Hernandez-Hernandez *et al.*, 2007a; Hernandez-Hernandez *et al.*, 2007b].

4.1 Algorithmes génétiques

Les algorithmes génétiques sont des algorithmes de recherche inspirés des mécanismes de l'évolution naturelle des êtres vivants et de la génétique.

John H. Holland a exposé ses premiers travaux sur les algorithmes génétiques en 1962 [Holland, 1962]. L'ouvrage de David Goldberg [Goldberg, 1989] a largement contribué à les vulgariser. En Allemagne, entre les années soixante et soixante-dix, sont apparues des méthodes appelées stratégies d'évolution [Rechenberg, 1971].

Les algorithmes génétiques partent de l'idée d'utiliser les principes des processus d'évolution naturelle en tant que technique d'optimisation globale. Dans l'évolution naturelle, le problème auquel chaque espèce est confrontée est de chercher à s'adapter à un environnement complexe et généralement non statique. Très schématiquement, la connaissance acquise par chaque espèce est codée dans les chromosomes de ses membres. Lors des reproductions sexuelles, les contenus des chromosomes sont mélangés, modifiés et transmis aux descendants par un certain nombre d'opérateurs génétiques : la mutation, qui se traduit par l'inversion d'une faible partie du matériel génétique, et le croisement qui échange certaines parties des chromosomes des parents. Cette particularité de l'évolution naturelle : la capacité d'une population à explorer son environnement en parallèle et à recombinaison les meilleurs individus entre eux, est empruntée par les algorithmes génétiques.

Pour un problème d'optimisation donné, un individu représente un point de l'espace de recherche, une solution potentielle. On lui associe la valeur du critère à optimiser, son adaptation. On génère ensuite de façon itérative des populations d'individus sur lesquelles on applique des processus de sélection, de croisement et de mutation. La sélection a pour but de favoriser les meilleurs éléments de la population pour le critère considéré (les mieux adaptés), le croisement et la mutation assurent l'exploration et exploitation de l'espace de recherche. Ces sont les éléments principaux d'un algorithme génétique (voir figure 4.1).

Dans la suite de cette section, nous présentons en détail chacun d'eux.

4.1.1 Représentation des solutions

Un aspect important des algorithmes génétiques est la façon dont sont codées toutes les solutions. Les algorithmes génétiques établissent une analogie entre l'ensemble de solutions d'un problème et l'ensemble d'individus d'une population naturelle, en codant l'information sur chaque solution. Une solution s est fréquemment codée par une chaîne de bits de longueur n i.e. $s[i] \in \{0, 1\}, \forall i = 1, \dots, n$. Plus de détails et de références sur ce codage et d'autres schémas de codage sont donnés dans [Michalewicz, 1998; Mitchell, 1997; Goldberg, 1991; Wright, 1991].

4.1.2 Fonction d'évaluation

La fonction d'évaluation, aussi appelée fonction d'aptitude, est un facteur important des algorithmes génétiques [Michalewicz, 1995]. Elle évalue chaque individu d'une po-

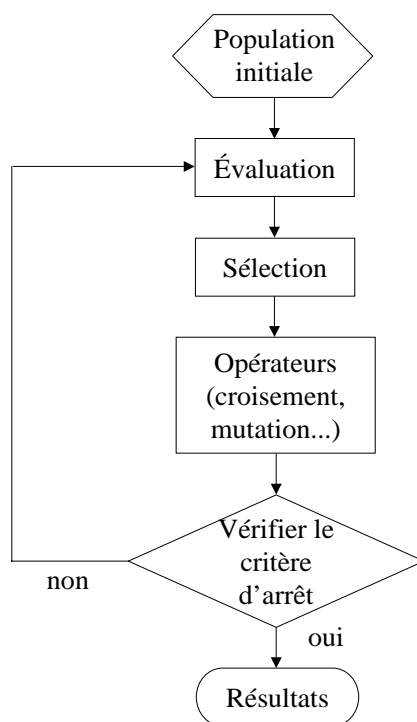


Figure 4.1 – Éléments d'un algorithme génétique

pulation et donne la qualité de chaque individu par rapport au problème posé. Celle-ci doit prendre en compte les bons paramètres du problème, par exemple, il faut une fonction d'évaluation qui augmente ou diminue progressivement à mesure qu'on s'approche de la bonne solution pour guider le processus de recherche.

C'est au cours de la phase d'évaluation, où l'ensemble des individus d'une population sont évalués (notamment ceux ayant subi une mutation ou un croisement), que l'on peut quantifier leur degré d'aptitude.

4.1.3 Opérateurs génétiques

A chaque génération, les opérateurs génétiques travaillent sur les individus formant la population. On différencie quatre opérateurs: opérateur d'initialisation, opérateur de sélection, opérateur de croisement et opérateur de mutation.

L'opérateur d'initialisation

Habituellement, cet opérateur génère un ensemble de plusieurs solutions. Cet ensemble constitue ce qui est appelé la population initiale. Souvent, la population initiale est générée de manière aléatoire afin de couvrir le mieux possible l'espace des solutions.

L'opérateur de sélection

La sélection joue un rôle très important dans les algorithmes génétiques: d'une part, pour diriger les recherches vers les meilleurs individus et d'autre part, pour maintenir la diversité des individus dans la population. Elle est liée au compromis entre la vitesse de convergence élevée et une forte probabilité de trouver un optimum global dans le cas d'un problème d'optimisation. Si la sélection choisit seulement le meilleur individu, la population convergera rapidement vers cet individu [Goldberg et Deb, 1991; Hancock, 1994; Dawid, 1999]. La sélection doit donc s'intéresser aux meilleurs individus tout en acceptant certains individus de moins bonne qualité. Plusieurs formes de sélection sont possibles, les plus connues sont [Chambers, 2001]:

- sélection linéaire par rapport au rang
- sélection uniforme par rapport au rang
- sélection proportionnelle
- sélection proportionnelle à reste stochastique
- sélection stochastique universelle
- sélection par tournoi

Dans la **sélection linéaire par rapport au rang**, le rang i de chaque individu I_i dans la population est $\forall_i \in 1, \dots, N: rang(I_i) = i$. Alors, un individu est choisi aléatoirement avec une probabilité proportionnelle à son rang [Baker, 1985]:

$$p_s(I_i) = \frac{1}{N} \left[\eta_+ - (\eta_+ - \eta_-) \frac{i-1}{N-1} \right]$$

avec $\eta_- = 2 - \eta_+$ et $1 \leq \eta_+ \leq 2$. Cette sélection n'utilise pas directement la performance des individus et donc un réajustement d'adaptation n'est pas nécessaire

La **sélection uniforme par rapport au rang** consiste à choisir de façon équiprobable les individus de rang inférieur ou égal à μ avec $\mu \leq N$. Les autres individus sont exclus de la population et ne peuvent participer à la reproduction [Bäck et Hoffmeister, 1991]. La probabilité de sélection s'exprime par:

$$p_s(I_i) = \begin{cases} \frac{1}{\mu} & \text{si } 1 \leq i \leq \mu \\ 0 & \text{si } \mu < i \leq N \end{cases}$$

La **sélection proportionnelle**, appelée aussi roulette (RWS) ou roue de la fortune [Goldberg, 1989], consiste à dupliquer chaque individu proportionnellement à sa valeur d'adaptation. On effectue, en quelque sorte, autant de tirage avec remise qu'il y a d'éléments dans la population. Ainsi, dans le cas d'un codage binaire, la qualité d'adaptation d'un individu particulier étant $f(I_i)$, la probabilité avec laquelle il sera réintroduit dans la nouvelle population de taille N est :

$$p_s(I_i) = \frac{f(I_i)}{\sum_{j=1}^N f(I_j)}$$

Les individus ayant une grande qualité ont donc plus de chance d'être sélectionnés. On parle alors de sélection proportionnelle. L'inconvénient majeur de cette méthode repose sur le fait qu'un individu n'étant pas le meilleur peut tout de même dominer la

sélection. Elle peut aussi engendrer une perte de diversité par la domination d'un super individu. Un autre inconvénient est sa faible performance vers la fin de la recherche quand l'ensemble des individus se ressemblent.

Dans la **sélection proportionnelle à reste stochastique** (SRS), le nombre de copies attendu pour chaque individu I_i est directement fixé par le rapport de sa performance avec la performance moyenne de la population [Baker, 1987] :

$$n(I_i) = \text{partie entière} \left[\frac{f(I_i)}{f_{moy}} \right]$$

Dans un premier temps, on n'obtient que $\alpha = \sum_{j=1}^N n(I_j)$ individus et il manque $(N - \alpha)$. On complète la population en associant à chaque individu I_i une probabilité d'être sélectionné égal à :

$$p_s(I_i) = \frac{1}{N-\alpha} \left[\frac{f(I_i)}{f_{moy}} - n(I_i) \right]$$

Le nombre de copies à compléter pour chaque individu est :

$$n_{sup}(I_i) = \frac{1}{N-\alpha} \cdot p_s(I_i)$$

Comme pour la sélection précédente RWS, un réajustement préalable de la fonction d'adaptation est également indispensable pour cette sélection.

Contrairement à la sélection proportionnelle RWS où il faut N tirages aléatoires pour sélectionner N individus, la **sélection stochastique universelle** (SUS) ne nécessite qu'un seul tirage pour choisir tous les parents d'une génération [Baker, 1987]. À partir d'une variable aléatoire θ , prise uniformément dans l'intervalle $[0, f_{moy}]$, on définit deux séries de pointeurs p_u et p_v de la manière suivante :

$$\begin{cases} p_u = \theta + (u - 1)f_{moy} & u = 1, 2, \dots, N \\ p_v = \sum_{j=1}^v f(I_j) & v = 1, 2, \dots, N \end{cases}$$

le pseudo code de la sélection stochastique universelle est donné ci-après :

```

1 u=1
2 v=1
3 pour i=1,...,N faire
4   tant que  $p_u < p_v$  faire
5     Sélectionner  $I_i$ 
6     Incrémenter  $u$ 
7   fin
8 fin
```

Cette sélection est également précédée d'un réajustement de la fonction d'adaptation.

Lors de la **sélection par tournoi**, k individus de la population sont choisis aléatoirement et celui ayant la performance la plus élevée sera retenu pour participer à la reproduction. L'opération est répétée autant de fois qu'il y a d'individus à sélectionner.

La probabilité qu'un individu de rang i soit sélectionné après compétition est donnée ci-après :

$$p_s(I_i) = \begin{cases} \frac{C_{N-i}^{k-1}}{C_N^k} & \text{si } 1 \leq i \leq N - k + 1 \\ 0 & \text{si } N - k + 2 \leq i \leq N \end{cases}$$

où C_m^p désigne la combinaison de m individus p à p sans répétitions, soit le nombre de groupes de p individus différents qu'on peut former avec m individus sans tenir compte de l'ordre des individus.

Dans le cas particulier du tournoi de deux individus ($k = 2$), qualifié de tournoi binaire probabiliste, la probabilité de sélection citée précédemment se réduit à

$$p_s(I_i) = \frac{2}{N} \cdot \frac{N-i}{N-1}$$

La sélection par tournoi de deux individus est donc équivalente à la sélection linéaire par rapport au rang avec $\eta_+ = 2$.

L'opérateur de croisement

L'opérateur de croisement, appelé aussi recombinaison, est l'instrument majeur des innovations dans l'algorithme génétique [Holland, 1975]. Les individus potentiels existant au sein d'une population génétique se croisent. Cette opération génère un (ou plusieurs) nouvel individu qui peut se rapprocher de la solution optimum. Les opérateurs de croisement les plus connus sont:

- opérateur de croisement à un point. Dans cet opérateur deux individus se croisent et s'échangent des portions de leur information en un seul point.
- opérateur de croisement à multiples points. Contrairement à l'opérateur précédent, il y a au moins deux points de croisement (dans tous les deux opérateurs, un point de croisement est aléatoirement choisi).
- opérateur de croisement uniforme. Cet opérateur décide pour chaque bit/gène avec probabilité fixée, indépendamment, si on prendra celui de l'un ou l'autre parent.

Un exemple de chaque croisement est illustré dans la figure 4.2. Dans le cas des deux premiers opérateurs de croisement, un point de croisement est marqué graphiquement par |.

L'opérateur de mutation

L'opérateur de mutation sur un individu échange aléatoirement un bit pour son complément. La mutation vise à modifier de façon aléatoire une partie de la population, elle provoque l'auto-adaptation des individus. Le taux de mutation est généralement faible [Fogarty, 1989]. Ce taux faible permet d'éviter une dispersion aléatoire de la population et n'entraîne que quelques modifications sur un nombre limité d'individus. Un exemple est montré dans la figure 4.3.

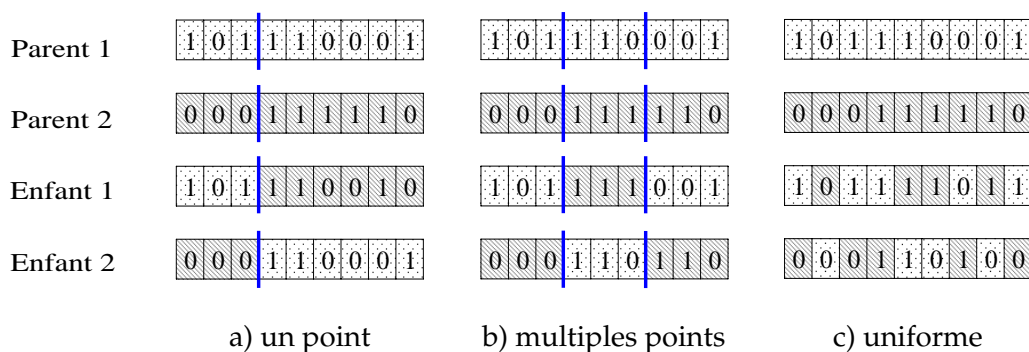


Figure 4.2 – Illustration des opérateurs de croisement.

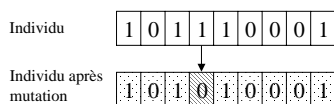


Figure 4.3 – Exemple d'une mutation typique.

4.1.4 Phase de remplacement

Une fois qu'on a généré des nouveaux individus par croisement et par mutation, il faut décider de la constitution de la nouvelle population [Reeves, 1995]. Le remplacement décide quels individus conserver. Il existe différents schémas de remplacement. Après avoir évalué les nouveaux individus générés, on applique un de ces schémas. Nous allons présenter brièvement ces schémas, les plus communs [Whitley, 1989; Jong, 1975].

- remplacement élitiste
- remplacement du plus mauvais
- remplacement par descendance

Le **remplacement élitiste** classique, couramment employé, garantit la survie du meilleur parent à chaque génération.

Le **remplacement du plus mauvais** présente des variantes qui consiste à remplacer les individus anciens les plus mauvais par des nouveaux individus.

Dans le **remplacement par descendance**, appelé aussi générationnelle, il n'y a aucune compétition entre le parents et les enfants. La population de la nouvelle génération est obtenue par descendance, c'est-à-dire, les enfants remplacent automatiquement leurs parents, quelque soit leur adaptation.

4.1.5 Schéma général d'un algorithme génétique

La procédure générale du fonctionnement d'un algorithme génétique est représenté dans l'algorithme ci-dessous.

Nous invitons le lecteur intéressé vers des livres [Whitley, 1995; Gen et Cheng, 1997;

4.2 Sélection de gènes par algorithme génétique

Algorithme 4.1 : Schéma d'un algorithme génétique

```
1 début
2   générer la population initiale  $P$ 
3    $nbIter \leftarrow 1$ 
4   tant que  $nbIter < maxIter$  faire
5     phase d'évaluation de la population courante  $P(nbIter)$  /* calcul de la qualité de chaque
6     individu de la population courante par la fonction d'évaluation  $f(i)$  */
7     sélection des meilleurs individus de  $P(nbIter)$  /* qui vont devenir procréateurs de la
8     génération suivante  $P(nbIter + 1)$  */
9     application des opérateurs génétiques /* qui modifient les procréateurs et obtiennent les
10    descendants */
11    remplacement d'une partie de la population par les descendants pour obtenir  $P(nbIter) + 1$ 
12     $nbIter \leftarrow nbIter + 1$ 
13 fin
```

[Mitchell, 1999; Haupt et Haupt, 2004; Eiben et Smith, 2007]

4.2 Sélection de gènes par algorithme génétique

Rappelons que la classification de données de puces à ADN nécessite la sélection d'un sous-ensemble de gènes pertinents afin d'obtenir une bonne performance de classification. La sélection de gènes peut se modéliser comme un problème d'optimisation combinatoire (voir la section 3.3).

Les algorithmes génétiques sont bien adaptés à ce type de problème. Bien qu'ils aient été utilisés dans de nombreux travaux dans ce domaine, notre algorithme contient des opérateurs de croisement et de mutation spécifiques au problème. Remarquons que notre fonction de coût cherche à optimiser le taux de classification et la taille sous-ensemble en même temps.

Nous présentons dans les paragraphes suivants les détails de notre algorithme génétique spécifique proposé pour la sélection et la classification de gènes de données de puces à ADN.

Tout comme l'approche de la recherche locale présentée dans le chapitre 3, notre modèle génétique débute par une phase de pré-sélection à l'aide d'une méthode de filtre (BW dans notre cas). Ainsi, nous restreignons l'espace de recherche de notre algorithme génétique en utilisant cette méthode de filtre. Cette pré-sélection nous fournit un groupe initial G des gènes sur lequel notre algorithme génétique réalise la sélection d'un sous-ensemble de gènes pertinents pour la classification. Le nombre de gènes pré-sélectionnés p a été fixé à 75.

4.2.1 Représentation

Un individu $I = \langle I^x, I^y \rangle$ est composé de deux parties I^x et I^y appelées respectivement : vecteur de sous-ensemble de gènes et vecteur des coefficients de classement.

La première partie I^x est une chaîne de bits de longueur fixe p . Chaque bit I_i^x ($i = 1 \dots p$) correspond à un gène particulier et indique si le gène est sélectionné ou non. La seconde partie I^y est un vecteur réel positif de longueur fixe p et correspond au vecteur des coefficients de classement c issus du SVM linéaire. I^y indique la contribution de chaque gène à la fonction de classification.

De ce fait, un individu représente un sous-ensemble de gènes candidats avec des informations de classement supplémentaires sur chaque gène sélectionné. Le vecteur d'un sous-ensemble de gènes d'un individu sera évalué par un classifieur SVM linéaire et les coefficients de classement obtenus pendant cette évaluation seront utilisés dans nos opérateurs de croisement et de mutation spécifiques. On observe un exemple de la représentation d'un individu dans la figure 4.4.

I^x	1	1	0	0	1	1	1	1	0	1	1
I^y	0.0955	0.1152	0	0	0.0988	0.0705	0.1091	0.1088	0	0.1072	0.1059

Figure 4.4 – Un individu et ses composants.

4.2.2 Fonction d'évaluation

Étant donné un individu $I = \langle I^x, I^y \rangle$, la qualité de I (plus exactement, du vecteur de gènes I^x) est évaluée par une fonction d'évaluation f selon deux critères : la capacité d'obtenir une bonne classification avec ce sous-ensemble de gènes et le nombre de gènes contenus dans ce sous-ensemble. D'une manière formelle, la fonction d'évaluation est définie comme suit :

$$f(I) = \frac{CA_{SVM}(I^x) + \left(1 - \frac{|I^x|}{p}\right)}{2} \quad (4.1)$$

Le premier terme de la fonction d'évaluation $CA_{SVM}(I^x)$ est le taux de classification obtenu avec un classifieur linéaire construit sur ce sous-ensemble (calculé par validation croisée sur 10 expériences). Le second terme assure que pour deux sous-ensembles de gènes ayant un taux de classification égal, le plus petit est préféré.

Pour un individu donné I , cette fonction d'évaluation produit une valeur d'aptitude réelle positive $f(I)$ (f est une fonction à valeur dans $[0,1]$ et une valeur de f élevée indique un individu de bonne qualité). En même temps, le vecteur des coefficients de classement obtenu du classifieur SVM est calculé et copié dans I^y .

4.2.3 Opérateur de croisement spécifique

Le croisement est un des opérateurs d'évolution clé pour n'importe quel algorithme génétique efficace et a besoin d'une conception attentive. Pour notre problème de recherche, nous voulons sélectionner des sous-ensembles de gènes de petite taille avec une

4.2 Sélection de gènes par algorithme génétique

grande performance en classification. Nous avons donc conçu un opérateur de croisement hautement spécialisé, appelé *GeSex* (Genetic Selection Crossover). *GeSex* est basé sur deux principes fondamentaux :

- 1) Conserver les gènes partagés par les deux parents
- 2) Préserver des gènes "de grande qualité" de chaque parent, même s'ils ne sont pas partagés par les deux parents.

La notion de "qualité" d'un gène est ici définie par son coefficient de classement correspondant stocké dans I^y . Remarquez que l'application du premier principe aura comme effet principal d'obtenir des sous-ensembles de gènes de plus en plus petits alors que l'application du deuxième principe nous permet de garder de bons gènes au cours du processus de recherche.

Plus précisément, soit $I = \langle I^x, I^y \rangle$ et $J = \langle J^x, J^y \rangle$ deux individus (parents) sélectionnés, nous combinons I et J pour obtenir un fils unique $K = \langle K^x, K^y \rangle$ en réalisant les étapes suivantes :

- 1. Nous utilisons l'opérateur logique booléen ET (\otimes) pour extraire le sous-ensemble de gènes partagés par les deux parents et les arranger dans un vecteur de sous-ensemble de gènes intermédiaire F .

$$F = I^x \otimes J^x$$

- 2. Pour le sous-ensemble de gènes obtenu à l'étape 1, on extrait les coefficients maximaux max_I et max_J des vecteurs de coefficients de classement origine I^y et J^y .

$$max_I = \max \{ I_i^y \mid i \text{ t.q. } F_i = 1 \}$$

et

$$max_J = \max \{ J_i^y \mid i \text{ t.q. } F_i = 1 \}$$

- 3. Cette étape vise à transmettre des gènes de bonne qualité de chaque parent I et J qui n'ont pas été conservés par l'opérateur logique booléen ET appliqué dans la première étape. Il s'agit des gènes avec un coefficient de classement plus grand que max_I et max_J . Les gènes sélectionnés de I et de J sont stockés dans deux vecteurs intermédiaires AI et AJ .

$$AI_i = \begin{cases} 1 & \text{Si } I_i^x = 1 \text{ et } F_i = 0 \text{ et } I_i^y > max_I \\ 0 & \text{sinon} \end{cases}$$

et

$$AJ_i = \begin{cases} 1 & \text{Si } J_i^x = 1 \text{ et } F_i = 0 \text{ et } J_i^y > max_J \\ 0 & \text{sinon} \end{cases}$$

- 4. Le vecteur sous-ensemble de gènes K^x du descendant K est alors obtenu en groupant tous les gènes de F , AI et AJ avec l'utilisation de l'opérateur logique booléen OU (\oplus).

$$K^x = F \oplus AI \oplus AJ$$

Le vecteur des coefficients de classement K^y sera rempli quand l'individu K sera considéré par le classifieur SVM lors la phase d'évaluation.

Nous illustrons ci-dessous le fonctionnement de notre opérateur $GeSeX$ pas à pas en suivant un exemple. D'abord, on sélectionne deux individus quelconques, soient I et J respectivement. Les deux parents sont montrés dans la figure 4.5. On observe les deux composants de chaque individu, le vecteur sous-ensemble de gènes sélectionnés et le vecteur des coefficients de classement. En appliquant l'opérateur logique booléen ET sur ces deux individus, on extrait les gènes communs à I et J . Ils sont encerclés sur la figure 4.5.

position	1	2	3	4	5	6	7	8	9	10	11	
I	0.0955	0.1152	0	0	0.0988	0.0705	0.1091	0.1088	0	0.1072	0.1059	I^y
I	①	1	0	0	1	①	①	①	0	1	①	I^x
J	①	0	0	1	0	①	①	①	0	0	①	J^x
J	0.1041	0	0	0.1081	0	0.0879	0.0839	0.0911	0	0	0.1035	J^y

Figure 4.5 – Une paire des individus sélectionnés.

Le résultat de cette première opération est enregistré dans le vecteur intermédiaire F , comme le montre la figure 4.6.

position	1	2	3	4	5	6	7	8	9	10	11
F	1	0	0	0	0	1	1	1	0	0	1

Figure 4.6 – Résultat obtenu à partir de l'étape 1.

Les étapes suivantes vont chercher les gènes de bonne qualité qui n'ont pas été préservés par l'opérateur booléen ET.

Étant donné F , on calcule max_I (respectivement max_J), le coefficient maximal dans I_y (respectivement dans J_y) des gènes de F . Pour l'exemple traité, on voit sur la figure 4.7 que $max_I = 0.1091$ (gène en position 7 cercle grisé) et $max_J = 0.1041$ (gène en position 1 cercle grisé).

Pour les gènes de I (respectivement de J) qui ne sont pas retenus dans l'étape 1 (ceux qui ne sont pas encerclés), on examine le coefficient et si celui-ci est supérieur à max_I (respectivement max_J), le bit correspondant est mis à 1 dans le vecteur AI (respectivement AJ).

4.2 Sélection de gènes par algorithme génétique

On observe sur le figure 4.7 qu'un seul gène de coefficient de classement $0.1152 > \max_I = 0.1091$ est ainsi retenu dans AI . De même, le gène de coefficient de classement 0.1081 est retenu dans AJ .

Finalement la figure 4.8 montre que le vecteur de gènes de l'enfant K^x est obtenu par l'opérateur booléen OU sur les vecteurs F , AI , et AJ .

position	1	2	3	4	5	6	7	8	9	10	11	position	1	2	3	4	5	6	7	8	9	10	11
I^y	$\textcircled{0.0953}$	0.1152	0	0	0.0988	$\textcircled{0.0705}$	$\textcircled{0.1091}$	$\textcircled{0.1088}$	0	0.1072	$\textcircled{0.1059}$	J^y	$\textcircled{0.1041}$	0	0	0.1081	0	$\textcircled{0.0879}$	$\textcircled{0.0839}$	$\textcircled{0.0917}$	0	0	$\textcircled{0.1035}$
I^x	1	1	0	0	1	1	1	1	0	1	$\textcircled{1}$	J^x	1	0	0	1	0	1	1	1	0	0	1
F	1	0	0	0	0	1	1	1	0	0	1	F	1	0	0	0	0	1	1	1	0	0	1
$\max_I=0.1091$												$\max_J=0.1041$											
AI	0	1	0	0	0	0	0	0	0	0	0	AJ	0	0	0	1	0	0	0	0	0	0	0

Figure 4.7 – Processus pour obtenir les valeurs de AI_i et de AJ_i respectivement.

F	1	0	0	0	0	1	1	1	0	0	1
AI	0	1	0	0	0	0	0	0	0	0	0
AJ	0	0	0	1	0	0	0	0	0	0	0
K^x	1	1	0	1	0	1	1	1	0	0	1

Figure 4.8 – Obtention du fils K^x .

4.2.4 Opérateur de mutation spécifique

Comme pour l'opérateur de croisement ci-dessus, nous concevons un opérateur de mutation qui est sémantiquement significatif en ce qui concerne notre problème de sélection de gènes et de classification. L'idée de base est d'éliminer quelques gènes "médiocres" et d'introduire en même temps aléatoirement d'autres gènes pour garder un certain degré de diversité dans la population de l'algorithme génétique.

Étant donné un individu $I = \langle I^x, I^y \rangle$, l'application de l'opérateur de mutation à I consiste à réaliser les étapes suivants :

- 1. La première étape calcule la moyenne \bar{c} des coefficients de classement des gènes dans l'individu I .
- 2. La deuxième étape élimine (avec une certaine probabilité) des gènes médiocres (i.e. inférieurs à la moyenne) et pour chaque gène éliminé on introduit aléatoirement un nouveau gène. Tout gène i tel que $I_i^x = 1$ et $I_i^y < \bar{c}$ ($i = 1 \dots p$) est soumis à une mutation avec une probabilité p^m . Si une mutation se produit, on prend aléatoirement un I_j^x telle que $I_j^x = 0$ et on met I_j^x à 1.

Encore une fois, nous considérons un exemple pour mieux expliquer le fonctionnement de notre opérateur de mutation. Soit un individu I et ses deux composant : I^x le vecteur de sous-ensemble de gènes et I^y le vecteur des coefficients de classement (voir la figure 4.9). En premier lieu, nous calculons la moyenne du vecteur des coefficients de classement \bar{c} . Dans notre exemple, le résultat obtenu est à 0.0964. Ce processus continue en mutant des gènes qui sont "médiocres" parmi des gènes non sélectionnés selon une probabilité fixée p^m . Nous appelons gènes médiocres, des gènes dont le coefficient de classement est inférieur à \bar{c} . Dans la même figure 4.9, ces coefficients sont encerclés et correspondent aux positions 6, 7 et 8.

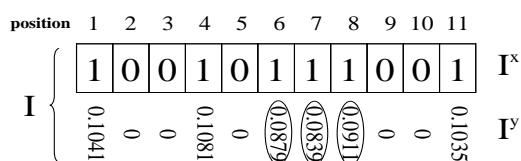


Figure 4.9 – Un individu sélectionné I .

En prenant en compte la probabilité de mutation p^m , dans cet exemple, notre opérateur a affecté seulement le gène de la position 7 et le gène non sélectionné choisi au hasard correspond à la position 3. Le résultat de l'opération de la mutation est illustré dans la figure 4.10.

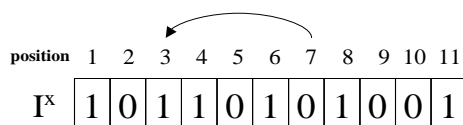


Figure 4.10 – Obtention de K^x .

4.2.5 Initialisation

La population initiale P tirée au hasard est constituée de 100 individus. Chaque individu contient entre $p/2$ et p bits de valeur 1. Rappelons que p est le nombre de gènes pré-sélectionnés à l'aide d'un filtre.

4.2.6 Sélection et Remplacement

On retient 2 individus par élitisme de la population courante et on les transmet dans la nouvelle population. 48 individus sont obtenus par croisement et les autres 48 individus obtenus par mutation.

Nous remarquons que nous avons réalisé différentes expériences avec toutes les méthodes de sélection (voir section 4.1.3), afin de choisir la méthode la plus performante appliquée dans notre algorithme génétique. Cette méthode fournit l'ensemble d'individus participants aux opérations de croisement et mutation.

À la suite des expériences, nous avons choisi, parmi toutes ces méthodes, la méthode de sélection stochastique universelle ou SUS (dû à sa variance faible) pour sélectionner les individus participants. Ainsi, les individus participants sont stockés dans une population temporaire P' .

Dans notre cas, d'une part, par chaque croisement, deux individus (de P') sont sélectionnés de manière aléatoire pour être recombinés grâce à l'opérateur de croisement et d'autre part, par chaque mutation, un individu (de P') est aussi sélectionné de manière aléatoire pour une mutation. Ainsi, les croisements et les mutations ne s'effectueront que sur des individus faisant partie de cette population temporaire.

Enfin, pour chaque nouvel individu (obtenu soit par croisement ou soit par mutation), il faut décider si le nouvel individu fera partie de la prochaine génération. Pour savoir si un nouvel individu doit être conservé, nous le comparons avec le plus mauvais individu de la population courante. Si le nouvel individu est meilleur que le plus mauvais individu de la population courante, alors le nouvel individu est ajouté à la nouvelle population et remplace l'individu le plus anciennement stocké dans celle-ci. Dans les autres cas, le nouvel individu est rejeté.

4.2.7 Résultats comparatifs

Notre modèle de sélection prend en compte le problème de biais de sélection. Nous suivons le même schéma de la figure 3.1, en remplaçant la recherche locale par notre algorithme génétique.

4.2.8 Comparaison des opérateurs de croisement

Nous comparons dans cette sous-section la performance de notre nouvel opérateur de croisement $GeSeX$ par rapport à deux autres opérateurs de croisement bien connus : croisement à un point et croisement uniforme. Le but de cette comparaison est de mieux comprendre les influences du nouvel opérateur de croisement $GeSeX$ sur la performance générale de notre algorithme génétique. Pour cela, l'évaluation prend en compte deux aspects : la capacité à générer des nouveaux individus potentiellement prometteurs et la capacité à garder une population diversifiée.

Le premier critère est mesuré par la qualité du meilleur individu de la population. Pour un individu, c'est-à-dire, un sous-ensemble de gènes, nous mesurons sa qualité comme la performance de classification d'un classifieur SVM linéaire construit sur ce

sous-ensemble de gènes. La performance est évaluée par validation croisée sur dix expériences, donc, pour un individu I , il s'agit de $CA_{SVM}(I^x)$ (formule 4.1).

Le deuxième critère, la diversité de la population, est donnée par le calcul de l'entropie de la population proposée par [Grefenstette, 1987]. Elle est calculée par la fonction 4.2 où n est le nombre de gènes, n_{ij} représente le nombre de fois que le gène i a la valeur j dans la population P . Cette fonction permet d'obtenir une valeur appartenant à l'intervalle $[0, 1]$. Une entropie égale à zéro indique que tous les individus de la population sont identiques alors que 1 indique que tous les individus sont uniformément distribués dans la population.

$$Entropy(P) = \frac{\sum_{i=1}^n \sum_{j=0}^1 \left(\frac{n_{ij}}{|P|}\right) \log\left(\frac{n_{ij}}{|P|}\right)}{n \log 2} \quad (4.2)$$

Pour déterminer quel est le croisement le plus efficace, il convient de caractériser la notion de bon croisement. Un bon croisement n'est pas forcément un croisement qui obtient rapidement le meilleur individu de la population mais de préférence celui qui garantit un compromis mutuel entre qualité (le meilleur individu obtenu à la fin) et diversité (entropie) de la population. Effectivement, la diversité est indispensable pour une meilleure exploration de l'espace de recherche au cours des générations et évite à la population de piétiner dans un optimum local.

Afin de comparer les opérateurs de croisement, nous avons inséré chacun d'eux dans un algorithme génétique simplifié (sans mutation) qui effectue une suite de croisements. Notre algorithme a été implémenté en langage MATLAB, en utilisant le même classifieur SVM linéaire de chapitre précédent. Pour une comparaison impartiale, les opérateurs de croisement ont été testés dans les mêmes conditions sur trois jeux de données : colon, leucémie et lymphome. Les paramètres utilisés sont :

- taille de la population $|P| = 100$
- nombre maximal de générations $maxGen = 100$
- probabilité de croisement $p_c = 0.5$, pour l'opérateur de croisement à un point et uniforme.

Étant donnée la nature non déterministe de l'algorithme, dix exécutions indépendantes ont été effectuées pour chaque jeu de données / opérateur. Les résultats sont montrés sur la figure 4.11.

Dans la figure 4.11.a, l'axe des abscisses exprime le nombre de générations alors que l'axe des ordonnées donne la valeur moyenne du taux de classification du meilleur individu de la population sur dix expériences. Cette figure montre clairement que l'opérateur *GeSeX* nous permet d'obtenir de meilleurs résultats pour les trois jeux de données car il atteint constamment un taux de classification plus haut au cours des générations.

Plus spécifiquement, nous examinons le cas du jeu de données de la leucémie. Avec l'opérateur *GeSeX*, un taux de classification moyen de 98.611% est rapidement atteint par le meilleur individu de la population vers la génération 20. Ce taux signifie que pour chacune des 10 expériences, seulement un des 72 échantillons est mal classé dans le processus de validation croisée. Avec les deux autres opérateurs de croisement, le taux de classification moyen est seulement autour de 96% car dans la plupart des expériences 3

des 72 échantillons sont mal classés et pour une ou deux expériences, deux échantillons sont mal classés. Nous pouvons remarquer aussi qu'après 90 générations, la courbe pour *GeSeX* quitte le pallier de 98.61% pour augmenter un peu car pour une ou deux expériences parmi les 10, le meilleur individu atteint le taux de classification maximum de 100%.

Dans la figure 4.11.b, nous montrons l'évolution de l'entropie de la population sur l'axe des Y par rapport au nombre de générations, l'axe des X. Chaque point représente la moyenne de l'entropie de population sur dix expériences. Observez que *GeSeX* garde une plus haute entropie de la population que les autres opérateurs de croisement. Cette figure montre qu'un meilleur équilibre entre la qualité et la diversification de la population est fourni par notre opérateur de croisement *GeSeX*.

4.2.9 Comparaison avec d'autres approches avec un nombre de gènes fixe

Dans cette sous-section, nous comparons nos résultats avec quatre algorithmes de référence [Ni et Liu, 2004; Xiong *et al.*, 2006; de Souza et de Carvalho, 2004; Nguyen *et al.*, 2005]. Pour ce faire, nous réalisons des expériences en suivant les mêmes conditions expérimentales utilisées par chaque algorithme de référence. Plus exactement, nous fixons le nombre de gènes sélectionnés par notre méthode, c'est-à-dire, pour chaque jeu de donnée, nous déterminons quel taux de classification peut être obtenu par notre algorithme génétique pour le nombre de gènes rapporté dans chaque article. En plus, nous avons répliqué le même nombre d'expériences. Nous utilisons aussi le même nombre d'échantillons de test.

Dans cette expérience, notre algorithme génétique utilise les opérateurs de croisement et de mutation spécifiques qui sont présentés dans les sous-sections 4.2.1 et 4.2.4.

Le résultats du tableau 4.1 résument la comparaison : le nombre de gènes et le taux de classification rapporté dans chaque article, et à côté le taux de classification obtenu par notre méthode. Quelques cellules du tableau sont vides car l'expérience ne rapporte aucun résultat pour ce jeu de données.

Dans le tableau 4.1 nous remarquons que les résultats de notre algorithme génétique sont meilleurs que les résultats publiés, sauf pour le résultat de leucémie annoncé par [de Souza et de Carvalho, 2004]. Comme indiqué, dans cette expérience nous restreignons notre méthode à considérer le même nombre de gènes sélectionnés que dans les travaux cités. En fait, notre méthode est capable d'optimiser deux critères : le nombre de gènes sélectionnés et le taux de classification. Ainsi notre méthode est capable de sélectionner de plus petits sous-ensembles de gènes pertinents avec un haut taux de classification.

4.2.10 Comparaison avec d'autres approches

Dans cette expérience nous effectuons une étude comparative où le nombre de gènes sélectionnés n'était pas fixé. Le taux de classification que nous indiquons est une moyenne sur 50 expériences. Dans chaque expérience, un ensemble d'apprentissage et un ensemble de test, de taille fixée, sont tirés au hasard. En même temps, chaque découpage est stratifié par rapport au nombre d'échantillons de chaque classe.

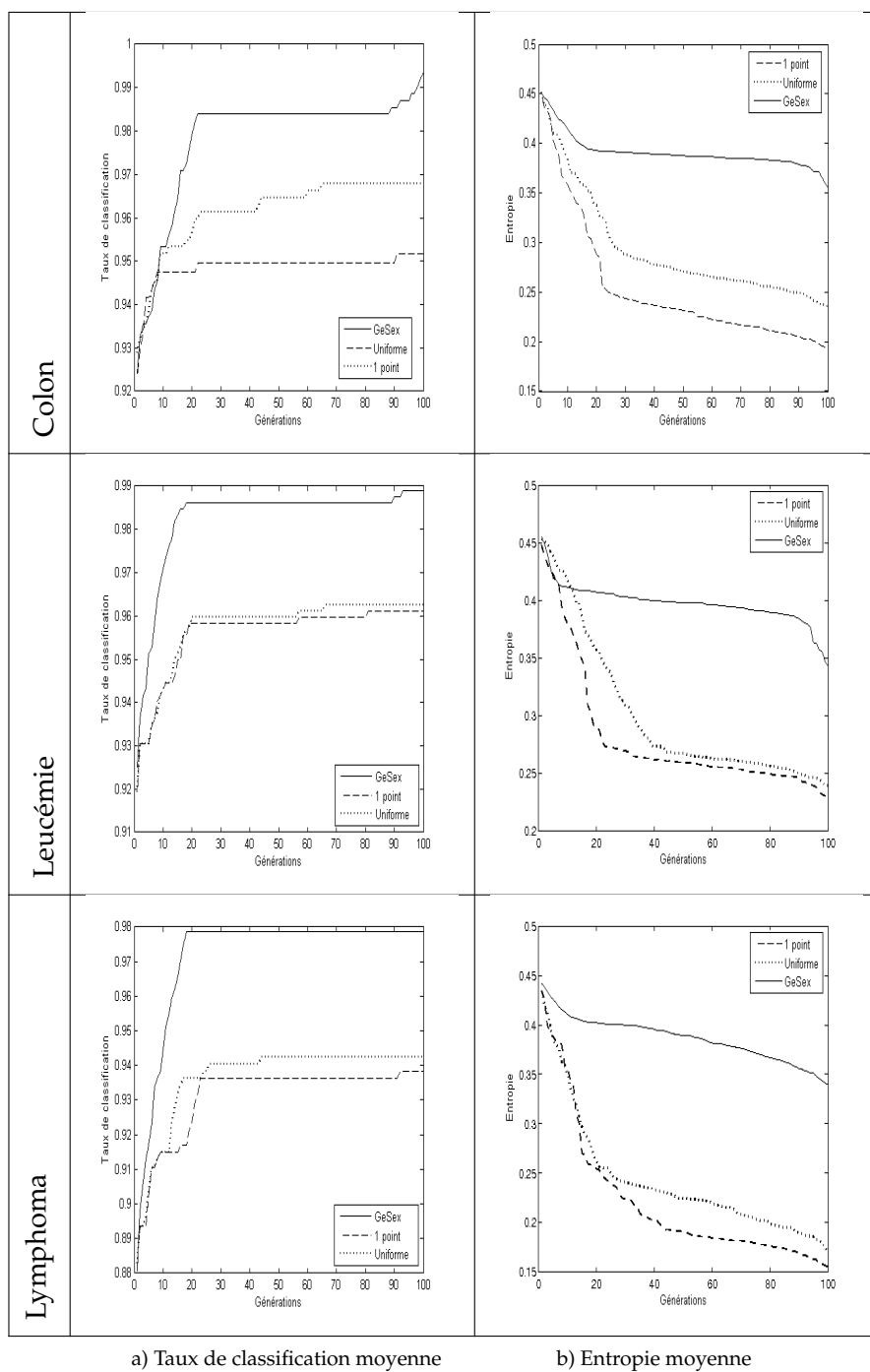


Figure 4.11 – Comparaison de performance entre trois opérateurs de croisement sur trois jeux de données.

4.2 Sélection de gènes par algorithme génétique

Table 4.1 – Comparaison entre quatre approches de sélection et la notre. Le tableau contient : d’une part, le nombre de gènes sélectionnés et le taux de classification rapporté par chaque auteur (*Rapporté*) et, d’autre part, le taux de classification obtenu par notre approche (*GeSeX*) en respectant la valeur du nombre de gènes rapporté dans le papier correspondant.

Jeu de donnée	[1]		[2]		[3]		[4]					
	<i>Rapporté</i>	<i>GeSeX</i>	<i>Rapporté</i>	<i>GeSeX</i>	<i>Rapporté</i>	<i>GeSeX</i>	<i>Rapporté</i>	<i>GeSeX</i>				
Leukemia	8	98.6	100	15	97.1	100	15	99.70	98.82	15	77.06	98.82
Colon	9	95.1	100	20	90.6	93.75	17	77.50	85.9	15	75.33	86.0
Lymphome ⁴⁷	-	-	-	-	-	-	10	96.15	96.92	-	-	-

1. [Ni et Liu, 2004]

2. [Xiong *et al.*, 2006]

3. [de Souza et de Carvalho, 2004]

3. [Nguyen *et al.*, 2005]

Tout d’abord nous indiquons que les auteurs de [Deb et Raji, 2003], qui utilisent un algorithme génétique avec une validation croisée sur N expériences (leave-one-out), optimisent le taux de classification sur l’ensemble d’apprentissage et l’ensemble de test. Donc, les résultats sont biaisés et très optimistes. Par contre, les auteurs de [Zhang *et al.*, 2007; Peng *et al.*, 2003; Wang *et al.*, 2007; Wang *et al.*, 2005b] utilisent également une validation croisée sur N expériences en optimisant seulement l’ensemble d’apprentissage. Cette technique permet d’estimer correctement l’erreur, avec un biais faible, mais une variance plus élevée. Tous les auteurs cités précédemment rapportent le résultat d’une seule exécution.

D’autres variantes sont présentées par les auteurs de [Huang et Liao, 2004; Liu *et al.*, 2002] qui utilisent une validation croisée sur 10 expériences et d’autres comme [Nguyen *et al.*, 2006; Bertoni *et al.*, 2005] qui utilisent une validation croisée sur 5 expériences.

Finalement, les auteurs de [Liu et Iba, 2002; Paul et Iba, 2004; Niiijima et Kuhara, 2006] réalisent plusieurs exécutions (10, 50 et 100, respectivement). Pour chaque exécution, un découpage aléatoire stratifié est réalisé tant pour l’ensemble d’apprentissage que pour l’ensemble de test sur un nombre d’échantillons de la même classe.

D’abord, l’analyse comparative démontre que nos résultats rapportés sont dominants en comparaison de ceux rapportés dans les travaux de [Bertoni *et al.*, 2005; Liu et Iba, 2002].

Par ailleurs, les résultats obtenus par notre modèle génétique sont très encourageants par rapport aux résultats annoncés par les auteurs [Huang et Liao, 2004; Liu *et al.*, 2002; Nguyen *et al.*, 2006; Paul et Iba, 2004; Niiijima et Kuhara, 2006], soit en taux de classification ou soit en nombre de gènes.

Enfin, bien que les résultats rapportés par les auteurs [Zhang *et al.*, 2007; Peng *et al.*, 2003; Wang *et al.*, 2007; Wang *et al.*, 2005b] soient très performants, ils sont obtenus lors d’une seule exécution. Nous remarquons que parmi les 50 exécutions que nous réalisons, nous observons les mêmes ou de meilleurs résultats dans une ou plusieurs exécutions. Nos résultats sont donc très compétitifs.

Table 4.2 – Comparaison avec les méthodes de référence

Auteur \ jeu de données	Colon	Leucémie	Lymphome ⁹⁶	Sein	Poumon	Prostate	Ovaire	Cerveau	Lymphome ⁴⁷
<i>GeSeX</i>	84.6±6.6% 7.05±1.07	91.50±5.90% 3.17±1.16	93.30±3.10% 5.19±1.31	85.68±4.64% 10.70±8.34	99.79±0.32% 6.64±3.57	97.00±2.49% 6.76±7.12	99.98±0.12% 3.10±0.91	79.33±4.71% 13.16±5.83	96.78±3.14% 7.72±6.18
[Zhang <i>et al.</i> , 2007]	90.32 -	- -	- -	90.72 -	100% -	88.97% -	- -	80% -	- -
[Peng <i>et al.</i> , 2003]	93.55% 12	100% 6	- -	- -	- -	- -	- -	- -	- -
[Wang <i>et al.</i> , 2007]	- -	100% 2	- -	- -	- -	97.06 1	- -	71.67 256	- -
[Wang <i>et al.</i> , 2005b]	- -	91.18% 1	- -	- -	- -	- -	- -	- -	88.89% 1
[Deb et Raji, 2003]	96.77% 7	100% 3	100% 5	- -	- -	- -	- -	- -	- -
[Huang et Liao, 2004]	- -	95.20% 10	- -	- -	- -	- -	- -	63.80 10	- -
[Liu <i>et al.</i> , 2002]	- -	- -	- -	- -	- -	- -	100% 17	- -	- -
[Nguyen <i>et al.</i> , 2006]	86.23±6.05% 52	96.07±3.14% 15	- -	- -	100% 30	- -	- -	- -	- -
[Bertoni <i>et al.</i> , 2005]	82.26±10.87% 64	- -	- -	- -	- -	- -	71.67±9.50 17	- -	- -
[Liu et Iba, 2002]	80.00±8.30% 11.40±4.27	90.00±7.00% 15.20±4.54	90.00±3.40 12.90±4.40	- -	- -	- -	- -	- -	- -
[Paul et Iba, 2004]	81.00±8.00% 4.44±1.74	90.00±6.00% 3.16±1.00	93.00±4.00 4.42±2.46	- -	- -	- -	- -	- -	- -
[Niiijima et Kuhara, 2006]	87.80±0.6% 10	94.60±0.60% 10	- -	65.80±0.80% 10	- -	90.40±0.40% 10	- -	64.70±0.90% 10	- -

4.3 Conclusion

Dans ce chapitre, nous avons présenté une approche génétique intégrée pour la sélection et la classification de gènes de données de puces à ADN. L'approche proposée est composée d'une phase de pré-sélection réalisée grâce à la méthode de filtrage *BW* et d'une phase de recherche génétique qui détermine un bon sous-ensemble de gènes pour la classification. Alors que la phase de pré-sélection est conventionnelle, notre algorithme génétique est caractérisé par ses opérateurs de croisement et de mutation hautement spécialisés. En effet, ces opérateurs génétiques sont conçus de telle façon qu'ils intègrent l'information de classement des gènes fournie par le classifieur SVM pendant le processus de calcul d'aptitude. En particulier, l'opérateur de croisement *GeSeX* conserve non seulement les gènes partagés par les deux parents, mais utilise aussi l'information de classement pour préserver des gènes hautement classés même s'ils ne sont pas partagés par les parents. De la même façon, l'information de classement des gènes est incorporée dans l'opérateur de mutation pour éliminer des gènes "médiocres".

En premier lieu, nous avons présenté une étude sur le rôle des opérateurs de croisement pour la sélection de gènes. Notre analyse expérimentale démontre que cet opérateur de croisement se comporte plus efficacement que deux autres opérateurs de croisement

4.3 Conclusion

traditionnels et qu'il garantit un bon équilibre entre l'exploration et l'exploitation de l'espace de recherche.

Ensuite, les résultats expérimentaux montrent que notre approche rivalise très favorablement avec les méthodes de référence du point de vue des taux de classification et du nombre de gènes sélectionnés.

Cette étude confirme de nouveau qu'un algorithme génétique constitue une approche générale et de valeur pour la sélection et la classification de gènes de données de puces à ADN. Son efficacité dépend fortement de la manière dont les informations sémantiques du problème donné sont intégrées dans les opérateurs génétiques tels que le croisement et la mutation.

Chapitre 5

Un algorithme hybride pour la sélection de gènes

Sommaire

5.1	Algorithmes mémétiques	85
5.2	Hybridation entre l'algorithme génétique et la recherche locale itérée pour la sélection de gènes	87
5.2.1	Représentation	88
5.2.2	Fonction d'aptitude	88
5.2.3	Espace de recherche	88
5.2.4	Initialisation	89
5.2.5	Sélection	89
5.2.6	Opérateur de croisement	89
5.2.7	Opérateur de recherche locale	89
5.2.8	Nouvelle population	89
5.2.9	Procédure générale	89
5.2.10	Conditions expérimentales	90
5.3	Résultats Expérimentaux	90
5.3.1	Comparaison de MASEL avec nos autres modèles	91
5.3.2	Comparaison avec d'autres méthodes de sélection	91
5.4	Conclusion	92

DANS CE chapitre, nous présentons un algorithme mémétique pour la sélection de gènes des données de puces à ADN. Cette approche, appelée *MASEL* (Memetic Algorithm for gene SElection), combine une procédure génétique avec un opérateur de recherche locale.

Dans la procédure génétique, notre opérateur de croisement permet d'une part de combiner de bonnes informations et d'autre part de transmettre ces bonnes informations à la solution enfant. Notre opérateur de croisement permet aussi de préserver la diversité des solutions contenues dans la population.

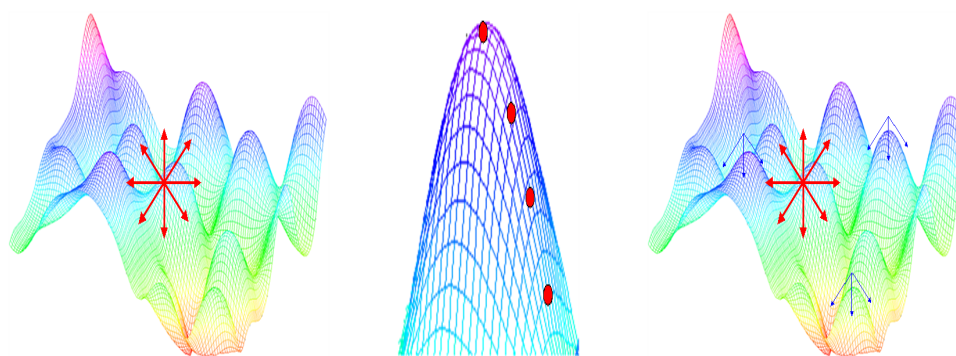
L'opérateur de recherche locale est une méthode de recherche locale itérée qui utilise une stratégie de recherche tabou comme algorithme de base. Cet opérateur permet l'intensification de la recherche dans l'espace de recherche.

MASEL a été testé sur neuf jeux de données et a montré des résultats très compétitifs par rapport à un algorithme mémétique cité ci-après et d'autres algorithmes de l'état de l'art.

5.1 Algorithmes mémétiques

Les algorithmes mémétiques sont introduits pour la première fois par Moscato [Moscato, 1989]. Fondamentalement, ils hybrident les algorithmes génétiques avec des méthodes de recherche locale. Pour cette raison, certains chercheurs les ont vus comme des algorithmes génétiques hybrides [Whitley, 1995; Houck *et al.*, 1996; Fleurent et Ferland, 1996; Lavrac *et al.*, 1996; Reeves, 1996; Cotta et Troya, 1998; Galinier et Hao, 1999]. Pourtant, les combinaisons avec des heuristiques constructives ou les méthodes exactes peuvent aussi appartenir à cette classe d'algorithmes. Comme ils sont les plus conviviaux pour les ordinateurs parallèles (MIMD) et les systèmes distribués (en incluant des systèmes hétérogènes) comme ceux composés par les réseaux de postes de travail, ils ont aussi reçu la dénomination d'algorithmes génétiques parallèles [Chen *et al.*, 1998; Digalakis et Margaritis, 2000]. Finalement, ils sont marginalement appelés algorithmes de recherche locale génétique [Ulder *et al.*, 1991; Kolen et Pesch, 1994].

La combinaison qui semble la plus fructueuse est celle qui utilise une méthode de recherche locale dans une algorithme génétique. On remarque que les algorithmes génétiques peuvent être une bonne solution pour résoudre des problèmes d'optimisation combinatoire. Cependant, un inconvénient d'un algorithme génétique est que les opérateurs standard de croisement et de mutation ne permettent pas d'intensifier suffisamment la recherche comme le rappellent [Hoos et Stützle, 2004]. L'opérateur de mutation apporte une légère modification à l'individu. Son rôle est de favoriser la diversification des individus alors que la sélection se charge de conserver les meilleurs. C'est pourquoi les algorithmes génétiques sont souvent hybridés avec des méthodes de recherche locale. Ces deux méthodes sont complémentaires car l'une permet de détecter de bonnes régions dans l'espace de recherche alors que l'autre se concentre de manière intensive à explorer ces zones de l'espace de recherche [Moscato, 1999]. Ainsi, on peut explorer rapidement les zones intéressantes de l'espace de recherche pour les exploiter en détail comme le montre la figure 5.1.



a) Algorithme génétique b) Recherche locale c) Algorithme mémétique

Figure 5.1 – Comportement schématique d'un algorithme génétique, d'une recherche locale et d'un algorithme mémétique.

Bien qu'il existe de multiples façons de concevoir un algorithme génétique, et que les méthodes de recherche locale soient aussi très nombreuses, l'hybridation la plus fréquente est de trouver dans un algorithme mémétique l'opérateur de recherche locale qui remplace ou succède à la mutation [Barichard, 2003]. En même temps, l'opérateur de croisement est adapté au problème traité [Lardeux, 2005; Goëffon, 2006; Rodriguez-Tello, 2007]. Le schéma de la figure 5.2 montre une telle hybridation.

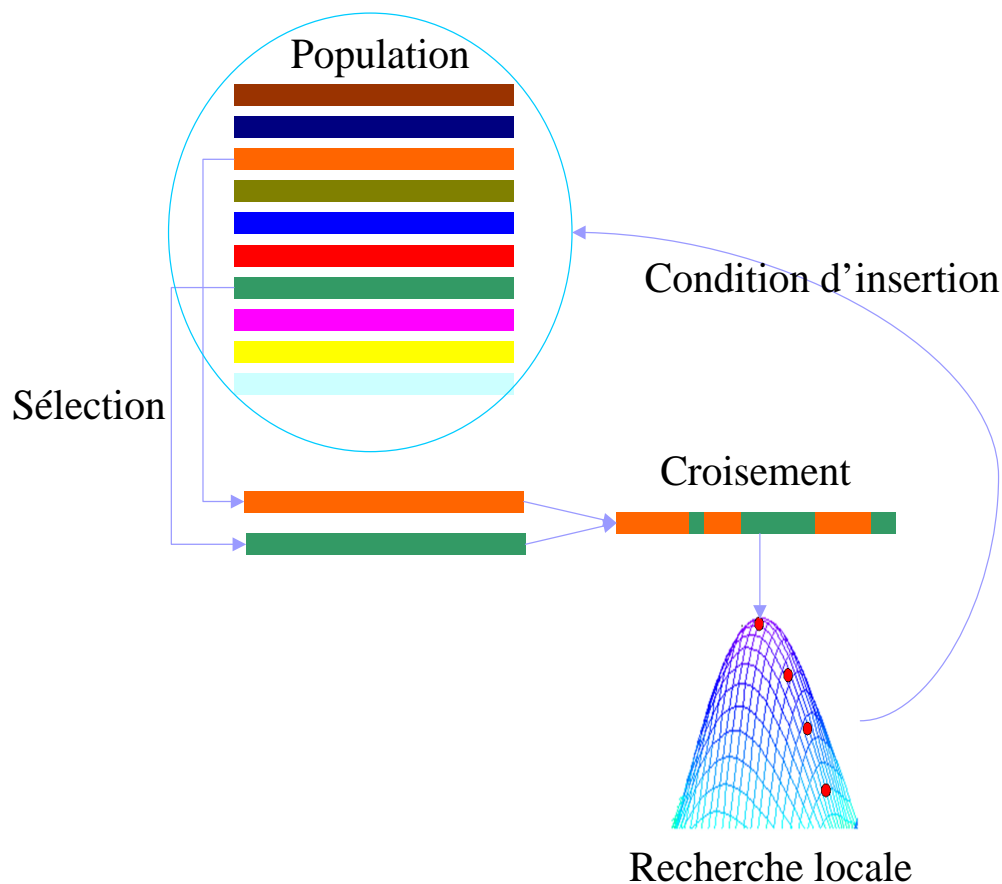


Figure 5.2 – Schéma des algorithmes mémétiques.

Les composants principaux qui jouent un rôle important dans les algorithmes mémétiques sont les suivants :

- La population initiale. Comme dans tout problème d'optimisation, une connaissance de bons points de départ conditionne la rapidité de la convergence vers l'optimum. Si la position de l'optimum dans l'espace de solutions est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux solutions de l'espace de solutions. Si par contre, des informations a priori sur le problème sont disponibles, il paraît bien évidemment naturel de générer les individus dans un sous-domaine

5.2 Hybridation entre l'algorithme génétique et la recherche locale itérée pour la sélection de gènes

particulier afin d'accélérer la convergence.

- La sélection. Elle permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent (voir la section 4.1.3).
- L'opérateur de croisement. Dans un processus itératif, à chaque pas, deux individus (parents) de la population sont sélectionnés et recombinaison afin d'obtenir un nouvel individu (fils).
- L'opérateur de recherche locale. Il est appliqué au nouvel individu afin de l'améliorer.
- La condition d'insertion. On détermine si le fils obtenu est ajouté à la population dans laquelle il remplace alors un individu existant.
- La condition d'arrêt. En général, on fixe un nombre de générations ou un temps d'exécution est imposé. Dans d'autres cas, on arrête lorsque la population cesse d'évoluer ou n'évolue plus suffisamment rapidement.

Les algorithmes mémétiques ont permis de produire d'excellents voire les meilleurs résultats sur des problèmes bien connus. C'est le cas par exemple du problème du voyageur de commerce TSP [Freisleben et Merz, 1996b; Freisleben et Merz, 1996a], du sac à dos [Falkenauer, 1996], de la coloration de graphes [Galinier et Hao, 1999], de la clique maximale [Balas et Niehaus, 1998], de l'affectation quadratique [Merz et Freisleben, 1997], de satisfiabilité [Lardeux, 2005], de maximum de parcimonie [Goëffon, 2006] et d'étiquetage de graphes [Rodriguez-Tello, 2007].

Nous invitons le lecteur intéressé par une description plus détaillée sur les algorithmes mémétiques à se reporter aux références ci-dessus et aux articles suivantes: [Gen et Cheng, 1997; Merz et Freisleben, 1999; Corne *et al.*, 1999; Hart *et al.*, 2005; Eiben et Smith, 2007].

5.2 Hybridation entre l'algorithme génétique et la recherche locale itérée pour la sélection de gènes

L'hybridation entre un algorithme génétique et une recherche locale est maintenant reconnue comme une approche compétitive pour traiter des problèmes combinatoires difficiles. Notre algorithme *MASEL* est une hybridation entre l'algorithme génétique présenté au chapitre 4 et l'algorithme de Recherche locale itérée présenté au chapitre 3. De cette manière, on exploite pleinement la puissance de recherche d'un algorithme de recherche locale qui permet d'intensifier la recherche dans diverses zones pointées par l'opérateur génétique de croisement. Le modèle proposé est montré dans la figure 5.3. Dans la suite de cette section nous présentons les composants principaux de notre algorithme mémétique.

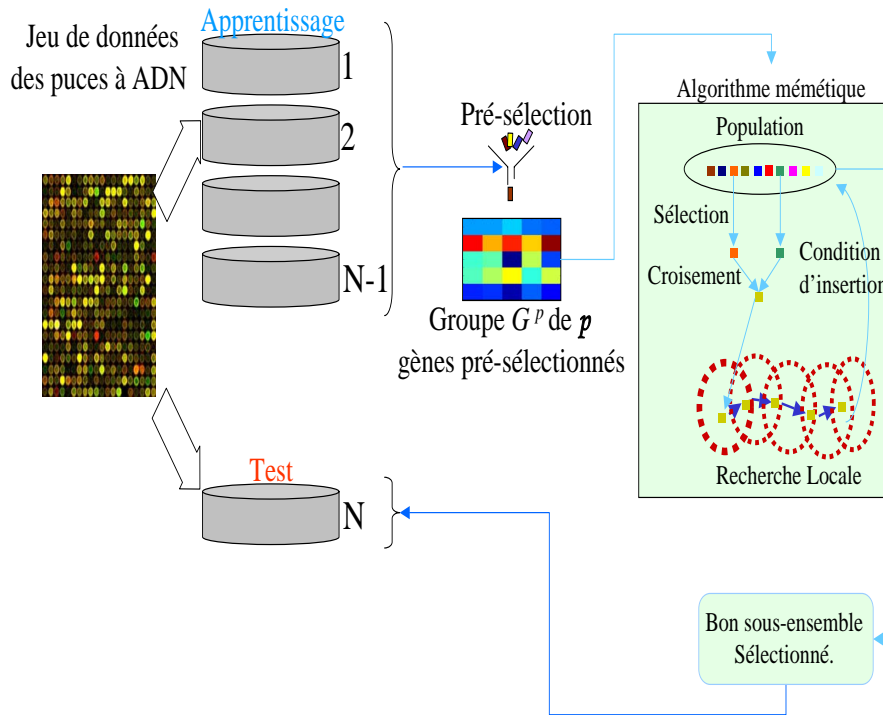


Figure 5.3 – Modèle de sélection de gènes par l’algorithme mémétique avec validation croisée.

5.2.1 Représentation

La représentation d’une solution qui est un ensemble de gènes sélectionnés est la même que celle utilisée par l’algorithme génétique que nous avons présenté dans la section 4.2.1.

5.2.2 Fonction d’aptitude

La fonction d’aptitude est la fonction d’évaluation utilisée par les méthodes de recherche locale que nous avons présentées dans la section 3.3.2.

5.2.3 Espace de recherche

D’abord, notre modèle de sélection réalise une phase de pré-sélection de gènes en utilisant la méthode de filtrage *BW*. De cette façon, on restreint l’espace de recherche de notre algorithme mémétique à un groupe compact de gènes pertinents G . Le nombre de gènes pré-sélectionnés (p) a été fixé à 75.

5.2.4 Initialisation

La première étape dans un algorithme mémétique est de constituer une population initiale servant de départ à l'algorithme. La population P est initialisée avec $|P|$ individus (solutions) générés aléatoirement à partir du groupe compact de gènes pertinents G^p . Grâce au comportement aléatoire de notre procédure d'initialisation, les solutions dans la population initiale sont tout à fait différentes. Ce point est très important pour les algorithmes mémétiques en général parce qu'une population homogène a peu de chance d'évoluer efficacement.

5.2.5 Sélection

Soient $|P|$ le nombre d'individus de la population (taille de la population) et $NbIndEli$ est le nombre d'individus transmis par élitisme. À partir de la population courante P de taille $|P|$, nous créons une sous-population temporaire P' qui contient $|P| - NbIndEli$ paires d'individus (ou parents) qui sont élus grâce à la sélection stochastique universelle décrite dans la section 4.1.3. Dans cette sous-population, les doublons sont interdits. Les parents seront utilisés lors du croisement.

5.2.6 Opérateur de croisement

Nous utilisons l'opérateur de croisement spécifique au problème, $GeSex$, qui a été présenté dans la section 4.2.3 du chapitre 4.

5.2.7 Opérateur de recherche locale

L'opérateur de recherche locale itérée a pour but d'améliorer les individus produits par l'opérateur de croisement. Cette recherche est effectuée avant d'insérer les individus améliorés dans la nouvelle population. Le mouvement et le voisinage utilisés sont les mêmes que ceux présentés à la section 3.3.3. La liste tabou est implémentée de manière classique, et sa taille est fixée à 5 au début de l'algorithme. L'opérateur de recherche locale itérée utilisé pour notre hybridation est en définitive l'algorithme RLI^{RT} présenté à la section 3.4.4 du chapitre 3.

5.2.8 Nouvelle population

La nouvelle population est remplie de la manière suivante : les $NbIndEli$ meilleurs individus de la population courante sont transmis par élitisme dans la nouvelle population. Par ailleurs, les individus issus de l'opérateur de recherche locale complètent la nouvelle population.

5.2.9 Procédure générale

L'algorithme $MASEL$ 5.1 débute avec la génération d'une population initiale à partir d'un groupe de gènes pré-sélectionnés. Ainsi un individu représente un ensemble de

gènes sélectionnés. Chaque individu est évalué par la fonction d'aptitude. L'algorithme entre alors dans un processus itératif. À chaque itération, on crée une population temporaire qui contient des paires de parents qui sont recombinaisonnés par l'opérateur de croisement *GeSex* afin d'obtenir des nouveaux individus. La recherche locale itérée est appliquée pour améliorer ces fils. Les meilleurs individus de la population courante sont transmis par élitisme à la nouvelle population. Les nouveaux individus (fils résultants) sont insérés dans la nouvelle population. Ce processus est répété jusqu'à ce que la condition d'arrêt soit vérifiée, usuellement lorsqu'un nombre maximum de générations ou d'itérations *maxIter* est atteint.

Cette procédure est représentée dans l'algorithme ci-dessous.

Algorithme 5.1 : Procédure générale d'un algorithme mémétique

```

Entrées :  $G, f, n, maxIter$ 
1 début
2   générer la population initiale  $P$  /*  $P = \{x_1, x_2, \dots, x_p\}, P \subseteq G^*$  /
3   Évaluation( $P, f$ )
4    $nbIter \leftarrow 1$ 
5   tant que  $nbIter < maxIter$  faire
6     générer la population temporaire  $P'$ 
7      $P'' \leftarrow$  croisement( $P'$ )
8     rechercheLocaleItérée( $P''$ )
9     transmettre les meilleurs individus de  $P$  par élitisme dans la nouvelle population
10    insérer les nouveaux individus dans la nouvelle population.  $nbIter \leftarrow nbIter + 1$ 
11  fin
12  retourner le meilleur sous-ensemble de gènes retenu
13 fin

```

5.2.10 Conditions expérimentales

L'algorithme mémétique *MASEL* présenté ci-dessus a été codé en langage Octave. Les valeurs des paramètres de *MASEL* ont été fixées empiriquement de manière à ce que l'algorithme soit performant, robuste, et qu'il nécessite des temps de calcul raisonnables. Les valeurs choisies pour *MASEL* sont :

- taille de l'ensemble de gènes pré-sélectionnés de départ par $BW = 75$.
- taille de la population $|P| = 30$
- nombre de fils par génération $nbFils = 25$
- nombre d'individus retenus par élitisme $NbIndEli = 5$
- nombre maximal de générations $maxIter = 30$.

5.3 Résultats Expérimentaux

Pour mener nos expériences, nous utilisons une validation croisée stratifiée sur dix expériences. Ainsi, tout au long d'une expérimentation, d'un côté, on garantit que tous les échantillons participent en tant qu'ensemble d'apprentissage comme l'ensemble de

5.3 Résultats Expérimentaux

test et d'un autre côté, on assure la répartition homogène des deux classes entre les dix blocs. Ce protocole d'expérimentation garantit aussi d'obtenir une sélection de gènes non biaisée. Encore une fois, nous appliquons le modèle *MASEL* sur les 9 jeux de données.

5.3.1 Comparaison de *MASEL* avec nos autres modèles

L'algorithme *MASEL* combine la recherche locale (Chapitre 3) et l'algorithme génétique (chapitre 4). C'est pourquoi, nous réalisons d'abord une étude comparative de ces trois méthodes. Les résultats sont illustrés dans le tableau 5.1.

On observe clairement que par rapport aux taux de classification, *MASEL* domine sur tous les jeux de données. Par ailleurs, quant au nombre de gènes sélectionnés, encore une fois *MASEL* est plus performant, sauf pour le jeu de colon, mais le taux de classification est meilleur.

Nous remarquons aussi la réduction de l'écart-type des taux de classification et des nombres de gènes sélectionnés sur tous les jeux de données. Finalement, on peut observer le compromis entre le taux de classification et le nombre de gènes sélectionnés atteint pour *MASEL*.

Table 5.1 – Évolution de nos résultats. TC indique le taux de classification et NG indique le nombre de gènes sélectionnés (tous les deux en montrant la moyenne et l'écart-type).

Jeu de données	<i>AG - GeSex</i>		<i>RLI^{RT}</i>		<i>MASEL</i>	
	TC	NG	TC	NG	TC	NG
Colon	84.6±6.6%	7.05±1.07	87.00±7.36%	8.20±2.09	98.33±5.27%	7.70±1.95
Leucémie	91.50±5.90%	3.17±1.16	91.94±4.06%	3.14±1.08	100%	6.20±3.19
Lymphome ⁹⁶	93.30±3.10%	5.29±1.31	95.44±2.15%	12.46±1.58	99.00±3.16%	8.05±2.59
Sein	85.68±4.64%	10.70±8.34	89.58±2.49%	7.90±6.27	95.78±5.46%	16.10±4.93
Poumon	99.79±0.32%	6.64±3.57	99.93±0.20%	4.66±2.25	100%	7.30±3.02
Prostate	97.00±2.49%	6.76±7.12	98.41±1.48%	5.08±4.10	97.03±3.83%	25.20±7.27
Ovaire	99.98±0.12%	3.10±0.91	100%	2.52±0.54	100%	3.00±0.00
Cerveau	79.33±4.71%	13.16±5.83	84.00±1.65%	9.06±4.24	95.00±8.05%	20.70±4.55
Lymphome ⁴⁷	96.78±3.14%	7.72±6.18	100%	7.34±4.16	100%	5.70±2.16

5.3.2 Comparaison avec d'autres méthodes de sélection

Nous nous intéressons ici à la comparaison de la performance de *MASEL* avec d'autres travaux de l'état de l'art (voir le tableau 5.2). Pour cela, nous avons choisi les travaux récents de la littérature qui utilisent un protocole d'expérimentation considérant une sélection non biaisée. Dans cette étude, huit travaux sont cités. L'analyse comparative avec chacun des modèles est décrite ci-dessous.

Le travail présenté par [Zhu *et al.*, 2007] utilise un algorithme mémétique basé sur les couvertures de Markov. Il est le seul travail trouvé dans la littérature qui utilise un algorithme mémétique pour faire la sélection de gènes. Ce modèle emploie la technique de 0.632 bootstrap pour valider ses résultats. La comparaison démontre que nos résultats sont dominants pour le taux de classification et pour le nombre de gènes sélectionnés.

Tous les travaux cités ci-après utilisent une validation croisée dans leurs expériences rapportées.

Les auteurs de [Alba *et al.*, 2007] présentent deux modèles de sélection : une optimisation par essaims particulaires et un algorithme génétique. Tous les deux s'appuient sur un classifieur SVM. Pour les jeux de données de la leucémie, du sein, du poumon et de l'ovaire, *MASEL* obtient des taux de classification supérieurs aux résultats rapportés même si les nombres de gènes sélectionnés par notre modèle sont plus grands (sauf pour l'ovaire).

Par rapport aux travaux développés par [Liu *et al.*, 2004; Wang, 2006; Küçükural *et al.*, 2007], on observe l'absence du nombre de gènes sélectionnés. Encore une fois, les résultats produits par *MASEL* sont très performants sur le taux de classification.

Finalement, dans les travaux de [Pang *et al.*, 2007; Yang *et al.*, 2006], on observe que le nombre de gènes retenus par *MASEL* est supérieur et que les taux de classification sont dominants.

5.4 Conclusion

Nous avons présenté, dans ce chapitre, l'algorithme mémétique *MASEL* qui combine une procédure génétique avec notamment l'opérateur de croisement *GeSex* et un opérateur de recherche locale qui utilise une méthode de recherche locale itérée avec une recherche tabou comme algorithme de base. Cette combinaison permet à *MASEL* d'explorer l'espace de recherche tout en intensifiant sa recherche efficacement sur des bonnes régions.

Dans le cadre des expérimentations, nous avons d'abord montré l'amélioration de ce modèle par rapport à nos modèles présentés dans les chapitres 3 et 4. Les résultats montrent clairement que l'algorithme *MASEL* obtient des meilleurs taux de classification avec des ensembles de gènes de petite taille. Ce modèle atteint un bon équilibre entre le taux de classification et le nombre de gènes sélectionnés pour tous les jeux de données.

Enfin, nous avons comparé *MASEL* avec neuf algorithmes de sélection récents de la littérature qui utilisent un protocole d'expérimentation sans biais. Les résultats comparatifs montrent que *MASEL* reste très compétitif en terme de taux de classification et du nombre de gènes sélectionnés.

5.4 Conclusion

Table 5.2 – Comparaison avec les méthodes de référence

Jeu de données	MASEL	Références								
		[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Colon	98.33±05.27	100	100	85.66±5.46	90.32	83.81±10.26	83.87	88.70±01.60	-	98.38
	7.70±1.95	2	3	24.5±7.0	-	23.40±5.03	-	16.83±1.15	-	12
Leucémie	100	97.38	97.27	95.89±2.46	100	-	100	95.08±1.27	96.50±4.80	-
	6.2±3.19	3	4	12.8±4.9	-	-	-	20.76±1.49	30	-
Sein	95.78±5.46	86.35	95.86	80.74±3.45	-	-	95.88	-	-	-
	16.10±4.93	4	4	14.5±4.2	-	-	-	-	-	-
Poumon	100	99.00	99.49	98.96±0.88	100	-	99.45	-	-	-
	7.30±3.02	4	4	14.10±7.00	-	-	-	-	-	-
Prostate	97.03±3.83	98.66	98.65	-	97.06	-	97.06	-	92.10±5.30	96.07
	25.20±7.27	4	4	-	-	-	-	-	30	19
Ovaire	100.00	99.44	98.83	99.71±0.53	98.82	98.80±1.10	-	-	-	100
	3.00±0.00	4	4	9.0±2.06	-	25.60±5.90	-	-	-	12
Cerveau	95.00±8.05	-	-	72.21±5.91	-	65.00±16.02	95.00	-	-	-
	20.70±4.54	-	-	20.50±6.9	-	46.20±5.50	-	-	-	-
Lymphome ⁴⁷	100.00	-	-	-	95.74	-	-	-	-	-
	5.7±2.16	-	-	-	-	-	-	-	-	-

1. [Alba *et al.*, 2007]-PSO/SVM

2. [Alba *et al.*, 2007]-AG/SVM

3. [Zhu *et al.*, 2007]

4. [Liu *et al.*, 2004]

5. [Pang *et al.*, 2007]

6. [Wang, 2006]

7. [Li *et al.*, 2008]

8. [Yang *et al.*, 2006]

9. [Küçükcüral *et al.*, 2007]

Conclusion générale

Principales contributions

Les différents travaux réalisés durant cette thèse apportent plusieurs contributions autour du problème de la sélection de gènes pour la classification de données de biopuces.

Tout d'abord, nous avons développé une approche de type intégrée pour faire face au problème de la sélection. Cette approche utilise un algorithme de recherche locale. Pour cela, différents algorithmes de recherche locale ont été implémentés. Il s'agit des algorithmes de la descente avec la stratégie de première amélioration et la stratégie de meilleure amélioration (D^p et D^m), de la recherche tabou combinée avec les deux descentes (RT^p et RT^m), et de la recherche locale itérée combinée avec la recherche tabou RTI^{RT} . Tous ces algorithmes utilisent un classifieur SVM linéaire, d'une part, pour évaluer la qualité d'un sous-ensemble de gènes sélectionnés, et d'autre part, pour éliminer des gènes et pour effectuer une suite de mouvements dans la recherche locale en utilisant les informations apportées par les coefficients du classifieur. De plus, la fonction d'évaluation de nos algorithmes de recherche locale prend en compte non seulement le taux de classification mais aussi la marge du SVM. Les résultats obtenus lors de notre étude comparative entre les différents algorithmes de recherche locale ont démontré que l'algorithme RTI^{RT} est le plus performant. Ensuite, la comparaison de cet algorithme avec d'autres algorithmes de sélection ont mis en évidence que RLI^{RT} rivalise très bien avec les méthodes de référence du point de vue du taux de classification et du nombre de gènes sélectionnés. Enfin, cette étude montre que la recherche locale constitue une approche simple, cependant puissante pour la sélection de gènes et la classification de données de puces à ADN. Son efficacité dépend fortement de comment les informations sémantiques du problème donné sont intégrées dans ses opérateurs fondamentaux comme la fonction d'évaluation et le voisinage.

Nous avons ensuite proposé une autre approche de sélection de type intégrée pour la sélection de gènes en utilisant un algorithme génétique. Cette approche se sert d'une méthode de filtrage afin de restreindre l'espace de recherche de notre algorithme génétique et la phase de recherche génétique détermine un bon sous-ensemble de gènes pour la classification. Notre algorithme génétique est caractérisé par ses opérateurs de croisement et de mutation hautement spécialisés. En effet, les informations de classement de gènes fournies par le classifieur SVM linéaire sont intégrées dans ces opérateurs. Plus

exactement, l'opérateur de croisement *GeSex* conserve non seulement les gènes partagés par les deux parents, mais utilise aussi l'information de classement pour préserver des gènes hautement classés même s'ils ne sont pas partagés par les parents. De la même façon, l'information de classement des gènes est incorporée dans l'opérateur de mutation pour éliminer des gènes "médiocres". Pour montrer les avantages de notre opérateur de croisement, nous avons réalisé une étude sur le rôle des opérateurs de croisement pour la sélection de gènes. Cette étude démontre que *GeSex* se comporte mieux que deux autres opérateurs de croisement traditionnels et qu'il garantit un bon compromis entre l'exploration et l'exploitation de l'espace de recherche. Ensuite, les résultats expérimentaux montrent que la performance de notre algorithme génétique se compare très favorablement avec d'autres méthodes de référence du point de vue des taux de classification et du nombre de gènes sélectionnés. De plus, cette étude confirme de nouveau qu'un algorithme génétique constitue une approche générale et de valeur pour la sélection et la classification de gènes de données de biopuces. Son efficacité dépend fortement de comment les informations sémantiques du problème donné sont intégrées dans ses opérateurs génétiques tels que le croisement et la mutation.

Finalement, nous avons développé une autre approche de sélection intégrée avec un algorithme mémétique. Cet algorithme mémétique appelé *MASEL* hybride un algorithme génétique et un algorithme de recherche locale. La méthode de recherche locale est la recherche locale itérée RTI^{RT} alors que la phase génétique incorpore l'opérateur de croisement spécialisé *GeSex*. Cette combinaison permet à *MASEL* d'explorer efficacement l'espace de recherche tout en intensifiant sa recherche dans de bonnes régions. L'étude comparative entre les trois modèles développés nous a permis de montrer l'amélioration de ce modèle par rapport aux algorithmes de recherche locale et à l'algorithme génétique décrits ci-dessus. Les résultats montrent clairement que l'algorithme *MASEL* obtient des meilleurs taux de classification avec des ensembles de gènes de petite taille. Ce modèle atteint un bon équilibre entre le taux de classification et le nombre de gènes sélectionnés pour tous les jeux de données. De même, nous avons comparé *MASEL* avec d'autres algorithmes de sélection récents de la littérature qui utilisent un protocole d'expérimentation sans biais. Les résultats comparatifs montrent que *MASEL* reste très compétitif en terme de taux de classification et du nombre de gènes sélectionnés.

Perspectives de recherche

Nous avons proposé différents algorithmes métaheuristiques spécifiques pour la sélection de gènes; bien qu'ils nous aient permis d'obtenir des résultats très compétitifs, nos approches pourraient être encore améliorés.

Tout d'abord, nous pourrions combiner plusieurs méthodes de filtrage pour l'obtention d'ensemble de gènes de départ. Ceci nous permettrait de travailler avec un sous-ensemble initial plus robuste et ne dépendant pas d'une unique métrique.

De même, le classifieur SVM linéaire peut fournir d'autres informations pour la sélection qui peuvent être intégrées dans nos algorithmes. Actuellement nous mesurons l'importance d'un gène en fonction de sa contribution sur la classification. On pourrait

Conclusion générale

utiliser également l'influence d'un gène sur l'une des bornes de risque [Rakotomamonjy, 2003].

Pour ce qui est des algorithmes développés, la recherche locale tabou pourrait être améliorée en considérant l'implémentation d'un critère d'aspiration et d'une liste tabou réactive afin de permettre l'intensification et la diversification de la recherche.

Pour notre algorithme génétique, nous pourrions améliorer les conditions d'insertion des nouveaux individus issus de nos opérateurs génétiques. Par ailleurs, nous pensons que l'algorithme d'optimisation par essais particuliers, qui travaille aussi sur des populations de solutions, pourrait être essayé.

Finalement, la partie hybride pourrait être revue avec ses nouvelles propositions d'implémentations et il serait souhaitable de travailler à la réduction du temps d'exécution en fonction du paramétrage utilisé.

Index

- algorithme
 - génétique, 63
- Attribut redondant, 25
- classification
 - supervisée, 14
- classifieur SVM, 37
- coefficient de pénalisation, 41
- conditions
 - de Karush-Kuhn-Tucker, 39
- espace des entrées, 39
- exemple, 13
- Filtrage par couverture de Markov, 24
- fonction
 - d'évaluation, 63
 - de décision, 37
- fonction d'évaluation, 43
- fonction de voisinage, 43
- formulation primale, 39
- heuristique, 42
- hyperplan, 37
- instance, 42
- métaheuristique, 42
- méthode
 - d'élimination récursive, 41
 - de descente, 43
- méthode de la première amélioration, 43
- méthodes
 - de programmation quadratique, 39
- marge, 38
- Markov blanket, 24
- multiplicateurs de Lagrange, 39
- Non pertinent, 24
- norme euclidienne, 37
- opérateur
 - de croisement, 67
 - opérateur d'initialisation, 64
 - opérateur de sélection, 65
 - opérateurs génétiques, 64
 - optimisation combinatoire, 42
 - optimum global, 43
 - optimum local, 43
- Pertinence faible, 23
- Pertinence forte, 23
- Recherche
 - locale itérée, 46
 - tabou, 44
- Recherche locale, 42
- risque
 - empirique, 14
 - réel, 14
- roue de la fortune, 65
- roulette, 65
- sélection
 - d'attributs, 26
 - linéaire par rapport au rang, 65
 - par tournoi, 66
 - proportionnelle, 65
 - proportionnelle à reste stochastique, 66
 - uniforme par rapport au rang, 65
- stratégie de mouvement, 43
- SVM-RFE, 41
- variable
 - de relâchement, 41
- variables
 - duales, 39
 - variables ressort, 40
- vecteur
 - des coefficients de classement, 41
 - vecteur de poids, 41
 - vecteur normal, 37

Références bibliographiques

- [Aarts et Lenstra, 1997] cité p. 42
E. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization*. Princeton university press, 1997.
- [Abe, 2005] cité p. 37
S. Abe. *Support vector machines for pattern classification*. Springer, 2005.
- [Alba et al., 2007] cité p. 92, 92, 92
E. Alba, J. Garcia-Nieto, L. Jourdan, and E.G. Talbi. Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In IEEE, editor, *CEC-2007*, pages 284–290, 2007.
- [Alizadeh et al., 2000] cité p. 11, 11
A. Alizadeh, M.B. Eisen, R.E. Davis, C.Ma, I.S. Lossos, A. Rosenwald, J.C. Boldrick, H. Sabet, T. Tran, X. Yu, J.I. Powell, L. Yang, G.E. Marti, T. Moore, J.J. Hudson, L. Lu, D.B. Lewis, R. Tibshirani, G. Sherlock, W.C. Chan, T.C. Greiner, D.D. Weisenburger, J.O. Armitage, R. Warnke, R. Levy, W. Wilson, M.R. Grever, J.C. Byrd, D. Botstein, P.O. Brown, and L.M. Staudt. Distinct types of diffuse large (b)–cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, February 2000.
- [Alon et al., 1999] cité p. 10, 10
U. Alon, N. Barkai, and D. Notterman et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Nat. Acad. Sci. USA.*, 96:6745–6750., 1999.
- [Ambroise et McLachlan, 2002] cité p. 32, 33
C. Ambroise and G. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proc. Nat. Acad. Sci. USA*, 99(10):6562–6566, 2002.
- [Baker, 1985] cité p. 65
J.E. Baker. Adaptative selection methods for genetic algorithms. In J.J. Grefenstette, editor, *International conference on genetic algoritms and their applications*, pages 101–111, Hillslade, New jersey, 1985. Lawrence Erlbaum Associates.
- [Baker, 1987] cité p. 66, 66
J.E. Baker. Reducing bias and inefficiency in the selection algorithm. In *The second international conference on genetic algorithms and their application*, 1987.
- [Balas et Niehaus, 1998] cité p. 87
E. Balas and W. Niehaus. Optimized crossover-based genetic algorithm for the maximum cardinality and maximum weight clique problems. *Journal of Heuristics*, 4(2):107–122, 1998.
- [Baldi et Brunak, 2001] cité p. 6
P. Baldi and S. Brunak. *Bioinformatics : The machine learning approach (second edition)*. A Bradford Book. The MIT Press, 2001.

- [Barichard, 2003] cité p. 85
V. Barichard. *Approches hybrides pour les problèmes multiobjectifs*. PhD thesis, Université d'Angers, 2003.
- [Batitti et Techioiii, 1994] cité p. 45
R. Batitti and G. Techioiii. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, 1994.
- [Baum, 1986] cité p. 45
E.B. Baum. Iterated descent: A better algorithm for local search in combinatorial optimization problems. In *Neural Information Processing Systems*, 1986.
- [Baxter, 1981] cité p. 45
J. Baxter. Local optima avoidance in depot location. *Journal of the Operational Research Society*, 32:815–819, 1981.
- [Bäck et Hoffmeister, 1991] cité p. 65
T. Bäck and F. Hoffmeister. Extended selection mechanisms in genetic algorithms. In *International conference on genetic algorithms and their applications*, pages 92–99. University of California, Morgan Kaufmann, 1991.
- [Bertoni et al., 2005] cité p. 79, 79, 79
A. Bertoni, R. Folgieri, and G. Valentini. Bio-molecular cancer prediction with random subspace ensembles of support vector machines. *Neurocomputing-Elsevier*, 63:535–539, 2005.
- [Besten et al., 2001] cité p. 46
M.L. Den Besten, T. Stützle, and M. Dorigo. Design of iterated local search algorithms : An example application to the single machine total weighted tardiness problem. *EvoStim2001 - LNCS*, 2037:441–452, 2001.
- [Bishop, 2006] cité p. 17
C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.
- [Blum et Langley, 1997] cité p. 22, 22, 25
A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97:245–271, 1997.
- [Boser et al., 1992] cité p. 15, 37, 39
B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992.
- [Breiman, 1996] cité p. 18
L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
- [Breimann et al., 1984] cité p. 18, 32
L. Breimann, J. Friedman, R. Olsen, and C. Stone. *Classification and regression trees*. California: Wadworth International, 1984.
- [Burges, 1998] cité p. 16, 37
C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.
- [Chambers, 2001] cité p. 64
I. L. Chambers. *The practical handbook of genetic algorithms, applications*. Chapman & Hall/CRC, 2001.
- [Chen et al., 1998] cité p. 85
H. Chen, N.S. Flann, and D.W. Watson. Parallel genetic simulated annealing: A massively parallel simd algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 9(2):126–136, 1998.

Références bibliographiques

- [Chiarandini et Stützle, 2002] cité p. 46
M. Chiarandini and T. Stützle. An application of iterated local search to graph coloring. In *The Computational Symposium on Graph Coloring and its Generalizations*, pages 112–125, 2002.
- [Chu *et al.*, 2005] cité p. 31
W. Chu, Z. Ghahramani, F. Falciani, and D. Wild. Biomarker discovery in microarray gene expression data with gaussian processes. *Bioinformatics*, 21(16):3385–3393, 2005.
- [Corne *et al.*, 1999] cité p. 87
D. Corne, F. Glover, and M. Dorigo, editors. *New ideas in optimization*. McGraw-Hill, 1999.
- [Cornuéjols et Miclet, 2002] cité p. 13, 16, 37
A. Cornuéjols and L. Miclet. *Apprentissage artificielle: Concepts et algorithmes*. Eyrolles, 2002.
- [Cortes et Vapnik, 1995] cité p. 15, 37, 39
C. Cortes and V. Vapnik. Support vector network. *Learning machine*, 20:1–25, 1995.
- [Cotta et Moscato, 2003] cité p. 26
C. Cotta and P. Moscato. The k-feature set problem is w[2]-complete. *Journal of computer and system sciences*, 67:686–690, 2003.
- [Cotta et Troya, 1998] cité p. 85
C. Cotta and J.M. Troya. A hybrid genetic algorithm for the 0-1 multiple knapsack problem. *Artificial Neural Nets and Genetic Algorithms*, 3:250–254, 1998.
- [Cristianini et Shawe-Taylor, 2000] cité p. 13, 16, 37
N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines*. Cambridge University Press, Cambridge, UK, 2000.
- [Dai *et al.*, 2006] cité p. 22, 22
J. J. Dai, L. Lieu, and D. Rocke. Dimension reduction for classification with gene expression microarray data. *Statistical Applications in Genetics and Molecular Biology*, 5(1):1–19, 2006.
- [Dash et Liu, 1997] cité p. 22, 22, 27
M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1:131–156, 1997.
- [Dash et Liu, 2006] cité p. 25
M. Dash and H. Liu. Hybrid search of feature subsets. In *PRICAI98*. Springer, 2006.
- [Davies et Russell, 1994] cité p. 26
S. Davies and S. Russell. Np-completeness of searches for smallest possible feature sets. In *Proceedings of the 1994 AAAI Fall Symposium on Relevance*, 1994.
- [Dawid, 1999] cité p. 64
H. Dawid. *Adaptive learning by genetic algorithms: Analytical results and applications to economic models*. Springer, 1999.
- [de Souza et de Carvalho, 2004] cité p. 77, 77, 77
B. Feres de Souza and A.C.Ponce Leon Ferreira de Carvalho. Gene selection using genetic algorithms. In *ISBMDA*, pages 479–490, 2004.
- [Deb et Raji, 2003] cité p. 77, 79
K. Deb and A. Raji. Reliable classification of two-class cancer data using evolutionary algorithms. *BioSystems*, 72:111–129, 2003.
- [Deng *et al.*, 2004] cité p. 31
L. Deng, J. Pei, J. Ma, and D.L. Lee. A rank sum test method for informative gene discovery. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouche, editors, *KDD*, 2004.

- [Digalakis et Margaritis, 2000] cité p. 85
 J.G. Digalakis and K. G. Margaritis. A performance comparison of parallel genetic and memetic algorithms using mpi. Technical report, Parallel Distributed Processing Laboratory, University of Macedonia, 2000.
- [Dubitzky *et al.*, 2003] cité p. 6, 6, 8
 W. Dubitzky, M. Granzow, S. Downes, and D. Berrar. *A practical approach to microarray data analysis*, chapter Introduction to microarray data analysis, pages 1–46. Kluwer Academic Publishers, 2003.
- [Duch, 2006] cité p. 22
 W. Duch. *Feature extraction, foundations and applications*, chapter Filter Methods, pages 89–118. Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, 2006.
- [Duda *et al.*, 2000] cité p. 17, 31, 32
 R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, Inc., New York, 2nd edition, 2000.
- [Dudoit *et al.*, 2002] cité p. 6, 7, 8, 10, 30, 31
 S. Dudoit, J. Fridlyand, and T. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97:77–87, 2002.
- [Eiben et Smith, 2007] cité p. 68, 87
 A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2007.
- [Falkenauer, 1996] cité p. 87
 E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.
- [Fleurent et Ferland, 1996] cité p. 85
 C. Fleurent and J.A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63:437–463, 1996.
- [Fogarty, 1989] cité p. 67
 T.C. Fogarty. Varying the probability of mutation in the genetic algorithm. In Morgan Kaufmann Publishers Inc, editor, *the 3rd International Conference on Genetic Algorithms*, pages 104–109, 1989.
- [Freisleben et Merz, 1996a] cité p. 87
 B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 616–621, 1996.
- [Freisleben et Merz, 1996b] cité p. 87
 B. Freisleben and P. Merz. New genetic local search operators for the traveling salesman problem. *PPSN IV - Lecture Notes in Computer Science*, 1141:890–899, 1996.
- [Fukunaga, 1990] cité p. 22
 K. Fukunaga. *Statistical pattern recognition : Second edition*. Morgan Kaufmann, 1990.
- [Galinier et Hao, 1999] cité p. 85, 87
 P. Galinier and J.K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.
- [Gen et Cheng, 1997] cité p. 68, 87
 M. Gen and R. Cheng. *Genetic algorithms and engineering design*. John Wiley & Sons, 1997.
- [Gendreau, 2003] cité p. 45
 M. Gendreau. *Handbook of metaheuristics*, chapter An introduction to tabu search, pages 37–54. Springer, 2003.

Références bibliographiques

- [Glover *et al.*, 1993] cité p. 45
F. Glover, E. Taillard, M. Laguna, and D. De Werra. A user's guide to tabu search. *Annals of operations research*, 41:3–28, 1993.
- [Glover et Laguna, 1993] cité p. 45
F. Glover and M. Laguna. *Modern heuristic techniques for combinatorial problems*, chapter Tabu search, pages 70–150. John Wiley and Sons, Inc., 1993.
- [Glover et Laguna, 1998] cité p. 43, 45
F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1998.
- [Glover, 1977] cité p. 43
F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166, 1977.
- [Glover, 1986] cité p. 43
F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986.
- [Glover, 1989] cité p. 43
F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.
- [Glover, 1990a] cité p. 43
F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.
- [Glover, 1990b] cité p. 45
F. Glover. Tabu search: A tutorial. *Interface*, 20(1):74–94, 1990.
- [Godoy, 2001] cité p. 6
W. W. Godoy. *DNA Arrays: methods and protocols*, chapter Ethical ramifications of genetic analysis using DNA arrays, pages 53–69. Humana Press, 2001.
- [Goëffon, 2006] cité p. 85, 87
A. Goëffon. *Nouvelles heuristiques de voisinage et mémétiques pour le problème maximum de parcimonie*. PhD thesis, Université d'Angers, 2006.
- [Goldberg et Deb, 1991] cité p. 64
D.E. Goldberg and K. Deb. *Foundations of genetic algorithms*, chapter A comparative analysis of selection schemes used in genetic algorithms, pages 69–93. Morgan Kaufmann, 1991.
- [Goldberg, 1989] cité p. 63, 65
D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*, volume 3. Reading, MA: Addison-Wesley., 1989.
- [Goldberg, 1991] cité p. 63
G.E. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, 5:139–167, 1991.
- [Golub *et al.*, 1999] cité p. 9, 10, 30, 31
T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
- [Gordon *et al.*, 2002] cité p. 11
G.J. Gordon, R.V. Jensen, L.L. Hsiao, S.R. Gullans, J.E. Blumenstock, S. Ramaswamy, W.G. Richards, D.J. Sugarbaker, and R. Bueno. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research*, 17(62):4963–4967, 2002.

- [Grefenstette, 1987] cité p. 76
 J.J. Grefenstette. *Genetic Algorithms and Simulated Annealing*, chapter Incorporating Problem Specific Knowledge into Genetic Algorithms, pages 42–60. Morgan Kaufmann Pub, 1987.
- [Grody, 2001] cité p. 6
 W. W. Grody. *DNA Arrays : methods and protocols*, chapter Ethical ramifications of genetic analysis using DNA arrays, pages 53–69. Humana Press, 2001.
- [Guyon *et al.*, 2002] cité p. 22, 29, 32, 32, 36, 37, 37, 41, 56
 I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.
- [Guyon et Elisseeff, 2003] cité p. 22, 22, 25
 I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [Hancock, 1994] cité p. 64
 P.J.B. Hancock. An empirical comparison of selection methods in evolutionary algorithms. *AISB-Springer-verlag*, pages 80–94, 1994.
- [Hansen, 1986] cité p. 43
 P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Numerical Methods in Combinatorial Optimization*, 1986.
- [Hart *et al.*, 2005] cité p. 87
 W.E. Hart, N. Krasnogor, and J.E. Smith, editors. *Recent advances in memetic algorithms*. Springer, 2005.
- [Haupt et Haupt, 2004] cité p. 68
 R.L Haupt and S.E. Haupt. *Practical genetic algorithms*. John Wiley & Sons, Inc. Hoboken, New Jersey, second edition edition, 2004.
- [Herbrich, 2002] cité p. 37
 R. Herbrich. *Learning kernel classifiers. Theory and algorithms*. MIT press, 2002.
- [Hernandez-Hernandez *et al.*, 2007a] cité p. 62
 J. C. Hernandez-Hernandez, B. Duval, and J.-K. Hao. A genetic embedded approach for gene selection and classification of microarray data. In *EvoBIO*, pages 90–101, 2007.
- [Hernandez-Hernandez *et al.*, 2007b] cité p. 62
 J.C. Hernandez-Hernandez, B. Duval, and J.-K. Hao. A study of crossover operators for gene selection of microarray data. *EA2007-LNCS*, 4926:243–254, 2007.
- [Hernandez-Hernandez *et al.*, 2008] cité p. 36
 J.C. Hernandez-Hernandez, B. Duval, and J.-K. Hao. Svm-based local search for gene selection and classification of microarray data. *BIRD2008-Communications in Computer and Information Science*, 13:499–508, 2008.
- [Hertz et Werra, 2003] cité p. 45
 A. Hertz and D. De Werra. *Local search in combinatorial optimization*, chapter Tabu search, pages 121–136. Princeton University Press, 2003.
- [Holland, 1962] cité p. 63
 J.H. Holland. Outline for a logical theory of adaptative systems. *Journal of the Association on Computing Machinery*, 9(3):297–314, 1962.
- [Holland, 1975] cité p. 67
 J.H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.

Références bibliographiques

- [Hoos et Stützle, 2004] cité p. 43, 46, 85
H.H. Hoos and T. Stützle. *Stochastic local search: Foundations and applications*. Morgan Kaufmann, 2004.
- [Houck *et al.*, 1996] cité p. 85
C.R. Houck, J. A. Joines, and M.G. Kay. Utilizing lamarckian evolution and the baldwin effect in hybrid genetic algorithms. Technical Report 96-01, Dept. of Industrial Engineering North Carolina State University, 1996.
- [Huang et Liao, 2004] cité p. 79, 79, 79
C.J. Huang and W.C. Liao. Application of probabilistic neural networks to the class prediction of leukemia and embryonal tumor of center nervous system. *Neural processing letters*, 19:211–226, 2004.
- [Jaeger *et al.*, 2003] cité p. 25
J. Jaeger, R. Sengupta, and W. Ruzzo. Improved gene selection for classification of microarrays. In *Pacific Symposium on Biocomputing*, pages 53–64, 2003.
- [Jain et Zongker, 1997] cité p. 22, 22
A. Jain and D. Zongker. Feature selection : Evaluation, application, and small sample performance. *IEEE Trans Pattern Anal. Mach. Intell.*, 19(2):153–157, 1997.
- [Jiang *et al.*, 2008] cité p. 8
N. Jiang, L. J Leach, X. Hu, E. Potokina, T. Jia, A. Druka, R. Waugh, M. J Kearsey, and Z. W Luo. Methods for evaluating gene expression from affymetrix microarray datasets. *BMC Bioinformatics*, 9(284):1–10, 2008.
- [John *et al.*, 1994] cité p. 25
G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proc. ICML*, pages 121–129, 1994.
- [Jong, 1975] cité p. 67
K.A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.
- [Kecman, 2001] cité p. 37
V. Kecman. *Learning and soft computing. Support vector machines, neural networks and fuzzy logic models*. The MIT press, 2001.
- [Knudsen, 2004] cité p. 6
S. Knudsen. *Guide to analysis of DNA microarray data (second edition)*. Wiley-Liss, 2004.
- [Kohavi et John, 1997] cité p. 22, 22, 23, 31
R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.
- [Kolen et Pesch, 1994] cité p. 85
A.W. J. Kolen and E. Pesch. Genetic local search in combinatorial optimization. *Discrete applied mathematics and combinatorial operations research and computer science*, 48(3):273–284, 1994.
- [Koller et Sahami, 1996] cité p. 24, 24, 24, 28
D. Koller and M. Sahami. Toward optimal feature selection. *13th International conference on machines learning*, pages 1–15, 1996.
- [Küçükural *et al.*, 2007] cité p. 92, 92
A. Küçükural, R. Yeniterzi, S. Yuniterzi, and O.U. Sezerman. Evolutionary selection of minimum number of features for classification of gene expression data using genetic algorithms. *GECCO-2007-ACM*, pages 401–406, 2007.

- [Lander, 1999] cité p. 6, 6
E. Lander. Array of hope. *Nature Genetics*, 21:3–4, 1999.
- [Lardeux, 2005] cité p. 85, 87
F. Lardeux. *Approches hybrides pour les problèmes de satisfiabilité (SAT et MAX-SAT)*. PhD thesis, Université d'Angers, 2005.
- [Lavrac *et al.*, 1996] cité p. 85
N. Lavrac, D. Gamberger, and P. Turney. Cost-sensitive feature reduction applied to a hybrid genetic algorithm. In *Proceedings of the Seventh International Workshop on Algorithmic Learning Theory*, pages 127–134, 1996.
- [Li *et al.*, 2001] cité p. 58
L. Li, C. Weinberg, T. Darden, and L. Pedersen. Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method. *Bioinformatics*, 17(12):1131–1142, 2001.
- [Li *et al.*, 2008] cité p. 92
S. Li, X. Wu, and M. Tan. Gene selection using hybrid particle swarm optimization and genetic algorithm. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 12:1039–1048, 2008.
- [Liu *et al.*, 2002] cité p. 79, 79, 79
H. Liu, J. Li, and L. Wong. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, 13:51–60, 2002.
- [Liu *et al.*, 2004] cité p. 92, 92
B. Liu, Q. Cui, T. Jiang, and S. Ma. A combinational feature selection and ensemble neural network method for classification of gene expression data. *BMC Bioinformatics*, 5(138):1–12, 2004.
- [Liu et Iba, 2002] cité p. 79, 79, 79
J. Liu and H. Iba. Selecting informative genes using a multiobjective evolutionary algorithm. In IEEE Press, editor, *Congress on Evolutionary Computation*, volume 1, pages 297–302, 2002.
- [Liu et Motoda, 2008a] cité p. 22
H. Liu and H. Motoda. *Computational methods of feature selection*. Taylor & Francis group, 2008.
- [Liu et Motoda, 2008b] cité p. 25
H. Liu and H. Motoda, editors. *Feature selection for genomic data analysis*, chapter 17, pages 337–353. Chapman & Hall/CRC, 2008.
- [Liu et Yu, 2005] cité p. 23, 23, 28
H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowledge and Data Engineering*, 17(4):491–502, 2005.
- [Lourenço *et al.*, 2002a] cité p. 46
H.R. Lourenço, O. Martin, and T. Stützle. *Handbook of metaheuristics*, chapter Iterated local search, pages 321–353. Springer, 2002.
- [Lourenço *et al.*, 2002b] cité p. 45
H.R. Lourenço, O. Martin, and T. Stützle. Iterated local search. Technical report, Technical university of Darmstadt, 2002.
- [Mao et Tang, 2007] cité p. 25
K. Mao and W. Tang. Correlation-based relevancy and redundancy measures for efficient gene selection. In *Pattern Recognition in Bioinformatics*, volume 4774 of *Lecture Notes in Computer Science*, pages 230–241. Springer, 2007.

Références bibliographiques

- [Martin *et al.*, 1991] cité p. 45
O. Martin, S.W. Otto, and E.W. Felten. Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.
- [Merz et Freisleben, 1997] cité p. 87
P. Merz and B. Freisleben. A genetic local search approach to the quadratic assignment problem. *ICGA'97 - Morgan Kaufmann*, pages 465–472, 1997.
- [Merz et Freisleben, 1999] cité p. 87
P. Merz and B. Freisleben. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, 3:2063–2070, 1999.
- [Michalewicz, 1995] cité p. 63
Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In MIT Press, editor, *the 4th Annual Conference on Evolutionary Programming*, pages 135–155, 1995.
- [Michalewicz, 1998] cité p. 63
Z. Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer, 1998.
- [Mitchell, 1997] cité p. 16, 63
T.M. Mitchell. *Machine Learning*. Mc-Graw Hill, 1997.
- [Mitchell, 1999] cité p. 68
M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1999.
- [Molina *et al.*, 2002a] cité p. 22, 26
L. C. Molina, L. Belanche, and A. Nebot. Evaluating feature selection algorithms. *CCIA-LNCS*, 2504:216–227, 2002.
- [Molina *et al.*, 2002b] cité p. 22
L. C. Molina, L. Belanche, and A. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*, pages 306–313, 2002.
- [Moscato, 1989] cité p. 85
P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report 05.20 02.50 87.10, California Institute of technologie, Pasadena, CA 91125, U.S.A., 1989.
- [Moscato, 1999] cité p. 85
P. Moscato. *New Ideas in Optimization*, chapter Memetic Algorithms: A Short Introduction. McGraw-Hill, 1999.
- [Mundra et Rajapaks, 2007] cité p. 25
P.A. Mundra and J.C. Rajapaks. SVM-RFE with relevancy and redundancy criteria for gene selection. In *Pattern Recognition in Bioinformatics*, volume 4774 of *Lecture Notes in Computer Science*, pages 242–252. Springer, 2007.
- [Navin *et al.*, 2006] cité p. 32
T. Navin, O. Chapelle, J. Weston, and A. Elisseeff. *Feature extraction, foundations and applications*, chapter Embedded Methods, pages 139–167. Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, 2006.
- [Nguyen *et al.*, 2005] cité p. 77, 77
H.N. Nguyen, S.Y. Ohn, J. Park, and K.S. Park. Combined kernel function approach in svm for diagnosis of cancer. *ICNC2005-LNCS*, 3610:1017–1026, 2005.

- [Nguyen *et al.*, 2006] cité p. 79, 79, 79
H.N. Nguyen, S.Y. Ohn, S.O. Chae, D.H. Song, and I. Lee. Optimizing weighted kernel function for support vector machine by genetic algorithm. *MICAI2006-LNAI*, 4293:583–592, 2006.
- [Nguyen et Rocke, 2002] cité p. 30
D. Nguyen and D. Rocke. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, 18(1):39–50, 2002.
- [Ni et Liu, 2004] cité p. 77, 77
B. Ni and J. Liu. A novel method of searching the microarray data for the best gene subsets by using a genetic algorithm. *PPSN2004-LNCS*, 3242:1153–1162, 2004.
- [Nijijima et Kuhara, 2006] cité p. 79, 79, 79
S. Nijijima and S. Kuhara. Recursive gene selection based on maximum margin criterion: A comparison with svm-rfe. *BMC-Bioinformatics*, 7(543), 2006.
- [Orsenigo, 2008] cité p. 58
C. Orsenigo. Gene selection and cancer microarray data classification via mixed-integer optimization. *EvoBio2008-LNCS*, 4973:141–152, 2008.
- [Pang *et al.*, 2007] cité p. 92, 92
S. Pang, I. Havukkala, Y. Hu, and N. Kasabov. Classification consistency analysis for bootstrapping gene selection. *Neural Computing and Applications*, 16:527,539, 2007.
- [Paul et Iba, 2004] cité p. 79, 79, 79
T.K. Paul and H. Iba. Selection of the most useful subset of genes for gene expression-based classification. In IEEE, editor, *Evolutionary Computation.*, volume 2, pages 2076 – 2083, 2004.
- [Peng *et al.*, 2003] cité p. 77, 79, 79
S. Peng, Q. Xu, X.B. Ling, X. Peng, W. Du, and L.Chen. Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. *FEBS Letters*, 555(2):358–362, 2003.
- [Petricoin *et al.*, 2002] cité p. 12
E. F. Petricoin, A. M. Ardekani, B. A. Hitt, P. J. Levine, V. A. Fusaro, S. M. Steinberg, G. B. Mills, C. Simone, D. A. Fishman, E. C. Kohn, and L. A. Liotta. Use of proteomic patterns in serum to identify ovarian cancer. *Lancet*, 359:572–577, 2002.
- [Pochet *et al.*, 2004] cité p. 58
N. Pochet, F. De Smet, J.A.K. Suykens, and B.L.R. De Moor. Systematic benchmarking of microarray data classification: assessing the role of nonlinearity and dimensionality reduction. *Bioinformatics*, 20(17):3185–3195, 2004.
- [Pomeroy *et al.*, 2002] cité p. 11
S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L.M. Sturia, M. Angelo, M.E. McLaughlin, J.Y.H. Kim, L.C. Goumnerova, P.M. Black, C. Lau, J.C. Allen, D. Zagzag, M.M. Olson, T. Curran, C. Wetmore, J.A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D.N. Louis, J.P.E.S. Mesirov, Lander, and T.R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415:436–442, 2002.
- [Quinlan, 1993] cité p. 18, 32
J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc, 1993.
- [Rakotomamonjy, 2003] cité p. 11, 32, 37, 58, 96
A. Rakotomamonjy. Variable selection using svm-based criteria. *Machine Learning Research*, 3:1357–1370, 2003.
- [Rechenberg, 1971] cité p. 63
I. Rechenberg. *Evolutionsstrategie: Optimierung technischer systeme und prinzipien der biologischen evolution*. PhD thesis, Frommann-Holzboog, Stuttgart, 1971.

Références bibliographiques

- [Reeves, 1993] cité p. 45
C.R. Reeves. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, 1993.
- [Reeves, 1995] cité p. 67
C.R. Reeves. A genetic algorithm for flowshop sequencing. *Operations Research*, 22:5–13, 1995.
- [Reeves, 1996] cité p. 85
C.R. Reeves. Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research*, 63:371–396, 1996.
- [Rodriguez-Tello, 2007] cité p. 85, 87
E.A. Rodriguez-Tello. *Nouvelles fonctions d'évaluation pour les problèmes d'étiquetage de graphes BMP et MINLA*. PhD thesis, Université d'Angers, 2007.
- [Saeys et al., 2007] cité p. 32
Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.
- [Schapire, 1990] cité p. 19
R.E. Schapire. The strength of weak learning. *Machine Learning*, 5(2):197–227, 1990.
- [Schoemaker et Lin, 2005] cité p. 6
J. S. Schoemaker and S. M. Lin, editors. *Methods of microarray data analysis IV*. Springer, 2005.
- [Scholkopf et Smola, 2001] cité p. 16, 37
B. Scholkopf and A. Smola. *Learning with kernels*. MIT Press, 2001.
- [Shah et Kusiak, 2007] cité p. 58
S. Shah and A. Kusiak. Cancer gene search with data-mining and genetic algorithms. *Computers in Biology and Medicine*, 37:251–261, 2007.
- [Singh et al., 2002a] cité p. 12
D. Singh, P. Febbo, K. Ross, D. Jackson, J. Manola, C. Ladd, P. Tamayo, A. Renshaw, A. D'Amico, and J. Richie. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:203–209, 2002.
- [Singh et al., 2002b] cité p. 22
S. Singh, M. Singh, and M. Markou. Feature selection for face recognition based on data partitioning. *ICPR*, 1:680–683, 2002.
- [Soularue et Gidrol, 2002] cité p. 6, 7
P. Soularue and X. Gidrol. Puces à adn. Recherche, 06 2002.
- [Southern, 2001a] cité p. 6, 7, 8
E. M. Southern. *DNA Arrays methods and protocols*, chapter DNA Microarrays, pages 1–15. Humana Press, 2001.
- [Southern, 2001b] cité p. 7
E. M. Southern. *DNA Arrays: Methods and Protocols*, chapter DNA Microarrays, pages 1–15. Humana Press, 2001.
- [Speed, 2000] cité p. 6
T. Speed, editor. *Statistical analyse of gene expression microarray data*. Chapman & Hall/CRS, 2000.
- [Stafford, 2008] cité p. 8
P. Stafford, editor. *Methods in microarray normalization*. CRC Press, 2008.
- [Stekel, 2003] cité p. 6, 7, 8
D. Stekel. *Microarray bioinformatics*. Cambridge University Press, 2003.

- [Stutzle, 1999] cité p. 45
T. Stutzle. Iterated local search for the quadratic assignment problem. Technical report, Technical university of Darmstadt, 1999.
- [Taillard, 1991] cité p. 45
E. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel computing-elsevier*, 17(4-5):443–455, 1991.
- [Tan et Gilbert, 2003] cité p. 58
A. C. Tan and D. Gilbert. Ensemble machine learning on gene expression data for cancer classification. *Applied Bioinformatics*, 2(2):75–83, 2003.
- [Troyanskaya *et al.*, 2001] cité p. 11
O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.
- [Ulder *et al.*, 1991] cité p. 85
N.L.J. Ulder, E.H.L. Aarts, H.J. Bandelt, P. J. M. Van Laarhoven, and E. Pesch. Genetic local search algorithms for the traveling salesman problem. *Parallel Problem Solving from Nature*, pages 109–116, 1991.
- [Vapnik, 1998] cité p. 15, 37
V. Vapnik. *Statistical learning theory*. John Wiley, 1998.
- [Veer *et al.*, 2002] cité p. 12
L.J. Van't Veer, H. Dai, M.J. Van de Vijver, Y.D. He, A.A.M. Hart, M. Mao, H.L. Peterse, K. Van der Kooy, M.J. Marton, A.T. Witteveen, G.J. Schreiber, R.M. Kerkhoven, C. Roberts, P.S. Linsley, R. Bernards, and S.H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–536, 2002.
- [Wang *et al.*, 2005a] cité p. 37
Y. Wang, F. Makedon, J. C. Ford, and J. D. Pearlman. Hykgene: a hybrid approach for selecting marker genes for phenotype classification using microarray gene expression data. *Bioinformatics*, 21(8):1530–1537, 2005.
- [Wang *et al.*, 2005b] cité p. 77, 79, 79
Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes. Gene selection from microarray data for cancer classification : A machine learning approach. *Computational biology and Chemistry*, 29:37–46, 2005.
- [Wang *et al.*, 2007] cité p. 77, 79, 79
S. Wang, H. Chen, S. Li, and D. Zhang. Feature extraction from tumor gene expression profiles using dct and dft. In *EPIA Workshops*, pages 485–496, 2007.
- [Wang, 2006] cité p. 92, 92
C.W. Wang. New ensemble machine learning method for classification and prediction on gene expression data. In *IEEE EMBS Annual International Conference. Institute of Electrical and Electronics Engineers*, pages 3478–3481, 2006.
- [Weston *et al.*, 2002] cité p. 11, 32, 58
J. Weston, A. Elisseeff, B. Scholkopf, and M. Tipping. The use of zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461, 2002.
- [Whitley, 1989] cité p. 67
D. Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *The third international conference on genetic algorithms*, number 116-121, 1989.

Références bibliographiques

- [Whitley, 1995] cité p. 68, 85
Darrell Whitley. *Genetic Algorithms in Engineering and Computer Science*, chapter Modeling hybrid genetic algorithms, pages 191–201. John Wiley, 1995.
- [Wright, 1991] cité p. 63
A.H. Wright. Genetic algorithms for real parameter optimization. In *the foundation of genetic algorithms*, pages 205–218, 1991.
- [Wu *et al.*, 2008] cité p. 16, 17
X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z-H Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 1:1–37, 2008.
- [Xiong *et al.*, 2006] cité p. 77, 77
W. Xiong, C. Zhang, C. Zhou, and Y. Liang. Selection for feature gene subset in microarray expression profiles based on a hybrid algorithm using svm and ga. *ISPA 2006 - LNCS*, 4331:637–647, 2006.
- [Xiong *et al.*, 2007] cité p. 58
H. Xiong, Y. Zhang, and X.W. Chen. Data-dependent kernel machines for microarray data classification. *Transactions on computational biology and bioinformatics*, 4:583–595, 2007.
- [Yang *et al.*, 2002] cité p. 8
Y.H. Yang, S. Dudoit, P. Luu, D. M. Lin, V. Peng, J. Ngai, and T.P. Speed. Normalization for cdna microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res*, 30:1–12, 2002.
- [Yang *et al.*, 2006] cité p. 92, 92
W-H. Yang, D-Q. Dai, and H. Yan. Generalized discriminant analysis for tumor classification with gene expression data. *Machine Learning and Cybernetics.*, 1:4322–4327, 2006.
- [Yang et Honovar, 1998] cité p. 22
J. Yang and V. Honovar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13:44–49, 1998.
- [Yu et Liu, 2004a] cité p. 22, 24, 24, 25, 25
L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, 2004.
- [Yu et Liu, 2004b] cité p. 22, 25
L. Yu and H. Liu. Redundancy based feature selection for microarray data. In *KDD*, pages 737–742, 2004.
- [Zhang *et al.*, 2007] cité p. 77, 79, 79
L. Zhang, Z. Li, and H. Chen. An effective gene selection method based on relevance analysis and discernibility matrix. In *PAKDD*, pages 1088–1095, 2007.
- [Zhu *et al.*, 2007] cité p. 91, 92
Z. Zhu, Y.S. Ong, and M. Dash. Markov blanket-embedded genetic algorithm for selection. *Pattern Recognition*, 40:3236–3248, 2007.
- [Zhu et Hastie, 2004] cité p. 32
J. Zhu and T. Hastie. Classification of gene microarrays by penalized logistic regression. *Bio-statistics*, 5(3):427–443, 2004.

Liste des publications personnelles

Conférences internationales avec comité de sélection

1. J.C. Hernandez-Hernandez, B. Duval, and J.-K. Hao. A genetic embedded approach for gene selection and classification of microarray data. *Proceeding of the 5th European Conference on Evolutionary Computation Machine Learning and Data Mining in Bioinformatics (EvoBIO'07)*, volume 4447 of *Lecture Notes in Computer Science*, pages 90–101, Valencia, Spain, April 2007. Springer.
2. J.C. Hernandez-Hernandez, B. Duval, and J.-K. Hao. A study of crossover operators for gene selection of microarray data. *Proceedings of the 8th International Conference on Artificial Evolution (AE'07)*, volume 4926 of *Lecture Notes in Computer Science*, pages 243–254, Tours, France, October 2007. Springer.
3. J.C. Hernandez-Hernandez, B. Duval, and J.-K. Hao. SVM-based local search for gene selection and classification of microarray data. *Proceedings of the 2nd International Conference on Bioinformatics Research and Development (BIRD'08)*, volume 13 of *Communications in Computer and Information Science*, pages 499–508, Vienna, Austria, July 2008. Springer.

Conférences francophones avec actes de résumés étendus

1. J.C. Hernandez-Hernandez, B. Duval, and J.-K. Hao. Sélection de gènes par algorithme génétique pour la classification des données de puces à ADN. *Actes du Huitième Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'07)*, Grenoble, France, Février 2007.
2. J.C. Hernandez-Hernandez, B. Duval, and J.-K. Hao. Recherche locale pour la sélection de gènes des données de puces à ADN. *Actes du Neuvième Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'08)*, Clermont-Ferrand, France, Février 2008.

ALGORITHMES MÉTAHEURISTIQUES HYBRIDES POUR LA SÉLECTION DE GÈNES ET LA CLASSIFICATION DE DONNÉES DE BIOPUCES

Résumé

Les biopuces permettent de mesurer simultanément l'activité d'un grand nombre de gènes au sein d'échantillons biologiques et de réaliser un diagnostic (reconnaissance tissu sain/tissu cancéreux ou distinction entre différents types de cancer) à partir de ces données. Pour cette tâche de classification, on dispose d'un faible nombre d'échantillons alors que chaque échantillon est décrit par un très grand nombre de gènes.

Dans cette thèse, nous nous intéressons à la sélection de gènes qui permet de proposer un sous-ensemble de gènes pertinents afin de construire un classifieur prédisant le type de tumeur qui caractérise un échantillon cellulaire. Le problème de la sélection de gènes est un problème très difficile et les algorithmes métaheuristiques à base de voisinage (méthodes de recherche locale) et à base de populations (algorithmes génétiques et algorithmes mémétiques) semblent bien appropriés pour traiter ce problème.

Dans cette thèse, nous proposons plusieurs méthodes de sélection dites intégrées, combinant des algorithmes métaheuristiques avec un séparateur à vaste marge linéaire (SVM). Dans ces algorithmes, la qualité d'un sous-ensemble de gènes sélectionnés est évaluée grâce au classifieur SVM. De plus, nos algorithmes exploitent l'information de pertinence fournie par le classifieur SVM sur les différents gènes pour guider les mécanismes de recherche locale ou pour proposer des opérateurs génétiques spécialisés.

Des expérimentations ont été réalisées sur les différents jeux de données disponibles dans la littérature et nos méthodes se révèlent très compétitives par rapport aux travaux existants.

Mots-clés : sélection de gènes, classification, algorithmes métaheuristiques, optimisation combinatoire, données de biopuces.

HYBRID METAHEURISTICS ALGORITHMS FOR GENE SELECTION AND CLASSIFICATION OF MICROARRAY DATA

Abstract

DNA microarray technologies permit to measure simultaneously gene expressions for thousands of genes in a sample and enable to consider molecular cancer diagnosis based on gene expression. Data that are currently available in this field concern a very large number of variables (thousands of gene expressions) relative to a small number of observations (typically under one hundred samples).

This thesis deals with the problem of gene selection, which aims to propose a subset of relevant genes in order to build efficient classifiers to recognize different types of tumor. The problem of gene selection is a very hard problem, for which metaheuristics algorithms based on neighbourhood (local search methods) and population (genetic algorithms and memetic algorithms) seem appropriate.

In this thesis, we propose several embedded gene selection methods, that combine metaheuristics algorithms with a support vector machine. In these algorithms, the quality of a selected gene subset is evaluated by a linear SVM classifier trained on this subset. Moreover, these algorithms use the relevance measure, given by the linear SVM about each gene, to inform the search process or to build very specialized genetic operators.

Experimentations performed on available data sets show very competitive results when compared to the state-of-the-art works.

Keywords: gene selection, classification, metaheuristic algorithms, combinatorial optimisation, microarray data.