

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M.Ing.

PAR
BRICE BOURGOIN

CONSTRUCTION DE CARACTÉRISTIQUES PAR PROGRAMMATION
GÉNÉTIQUE POUR UN SYSTÈME DE RECONNAISSANCE MULTICLASSE

MONTREAL, LE 5 SEPTEMBRE 2007

© Brice Bourgoïn, 2007

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

M. Jacques-André Landry, directeur de mémoire
Département du génie de la production automatisée à l'École de technologie supérieure

M. Robert Sabourin, président du jury
Département du génie de la production automatisée à l'École de technologie supérieure

M. Richard Lepage, membre du jury
Département du génie de la production automatisée à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC
LE 5 SEPTEMBRE 2007
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens en premier lieu à remercier mon directeur de mémoire Jacques-André Landry au département de génie de la production automatisée à l'École de technologie supérieure, grâce auquel cette maîtrise a été possible. Il a toute ma gratitude pour son soutien, sa patience et ses conseils qu'il n'a jamais hésité à me fournir.

Je remercie également Annie Gélinas pour ses encouragements tout au long de ce cursus ainsi que pour sa relecture attentive de mon mémoire.

L'ambiance du laboratoire d'imagerie, de vision et d'intelligence artificielle (LIVIA) a aussi été bien appréciée, une bonne ambiance de travail (et de détente parfois) est un atout indéniable.

Enfin, je souhaite exprimer ma reconnaissance envers ma famille qui non seulement m'a soutenu au cours de ma maîtrise, mais aussi durant toutes les étapes qui l'ont précédée.

CONSTRUCTION DE CARACTÉRISTIQUES PAR PROGRAMMATION GÉNÉTIQUE POUR UN SYSTÈME DE RECONNAISSANCE MULTICLASSE

BOURGOIN, Brice

RÉSUMÉ

L'objectif de la recherche présentée est d'optimiser la reconnaissance automatisée d'objets en vision ou en télédétection. Nous partons du constat que les classificateurs sont sensibles à l'espace de représentation des données et qu'un remaniement de cet espace pourrait aider certains d'entre eux. Nous avons poursuivi deux objectifs : avoir un concept de système qui nécessite le moins possible d'interventions humaines lors de sa mise en place et minimiser le taux d'erreurs absolu.

Pour cela, nous avons utilisé un algorithme de Programmation Génétique avec coévolution. Son objectif a été de construire un nouvel ensemble de caractéristiques en se basant sur son potentiel de classification selon les plus proches voisins. Ensuite ce jeu de caractéristiques a été testé sur plusieurs types de classificateurs : plus proches voisins, réseaux de neurones artificiels et machines à vecteurs de support.

Nous avons préféré restreindre la Programmation Génétique au remaniement de l'espace de représentation plutôt que d'en faire un classificateur complet afin de mieux cibler sa recherche. Nous avons ainsi espéré profiter de la force de classificateurs avancés tels que les machines à vecteurs de support pour éviter que la Programmation Génétique ne réinvente ce que nous savons déjà faire. Elle a eu pour seul objectif de se concentrer sur ce qui est parfois une faiblesse dans un système de classification : la représentation des données.

Nous avons utilisé deux bases de données totalement distinctes : la première concernant des chiffres manuscrits, la seconde la différenciation de céréales telles que l'orge, le blé ou l'avoine. La première base comporte dix classes, la seconde sept. Ce sont donc des problèmes réels de vision largement multiclassés. En plus de confirmer les résultats, l'intérêt de l'utilisation de deux bases a été de mettre en évidence le peu d'interventions humaines nécessaires pour mettre en place notre système. En effet, pour la seconde base, nous avons utilisé exactement les mêmes paramètres que ceux choisis pour la première : ces paramètres internes de notre algorithme se veulent donc assez universels.

Après de nombreuses simulations, nous avons observé de bonnes performances sur les nouveaux espaces de représentation quel que soit le classificateur final utilisé. En cela, la robustesse de notre système vis-à-vis de l'espace de représentation semble avoir été améliorée par rapport à l'utilisation d'un classificateur seul. Ainsi, nous avons conforté l'objectif du peu d'interventions nécessaires pour la mise en place du système.

Aussi, les performances absolues semblent avoir été améliorées, en particulier avec l'usage d'une machine à vecteurs de support en aval. Cette amélioration n'a cependant par toujours été énorme, mais a semblée suffisamment prometteuse pour envisager de pousser notre investigation plus loin. En effet, notre approche offre encore de nombreuses voies de perfectionnement, en grande partie grâce aux nombreuses extensions possibles des algorithmes à base de Programmation Génétiques.

FEATURE CONSTRUCTION BY GENETIC PROGRAMMING FOR A MULTICLASS RECOGNITION SYSTEM

BOURGOIN, Brice

ABSTRACT

The goal of this research is to optimize the automated recognition of objects in computer vision or remote sensing applications. The premise is that classifiers are sensitive to the data representation space and that a reorganization of this space could improve the performance of some of them. We aim at two goals: to propose a framework for a system requiring the least possible human interventions at the time of its installation and to minimize the absolute error rate.

For that, we have used a Genetic Programming algorithm with coevolution. Its objective was to build a new set of characteristics being based on its potential for classification according to its closest neighbours. Then this set of characteristics was tested on several types of classifiers: closer neighbours, artificial neurons networks and support vector machines.

In order to better target his research, we preferred to restrict the Genetic Programming algorithm at the reorganization of the representation space than to generate a complete classifier. Thus, we hoped to benefit from the force of advanced classifier such as support vector machines to prevent the Genetic Programming algorithm from reinventing what is already known. The algorithm had for only objective to concentrate on what is sometimes a weakness in a classification system: the data representation space.

We have used two completely distinct data bases: the first containing handwritten digits, the second concerned with the differentiation of cereals such as barley, corn or oats. The first base contains ten classes, the second seven. Thus, they are real problems of computer vision and strongly multiclass systems. In addition to confirming the results, the interest in using two bases was to highlight the reduced need for human interventions in the initial setup of a classification system. Indeed, for the second base, we have used exactly the same parameters as those selected for the first: these internal parameters of our algorithm claim to be rather universal.

Several simulations allowed us to observe good performances with new space representations for whatever final classifier we used. In that, the robustness of the proposed system in reorganizing the representation space seems to offer improved performance when compared to a single classifier. Thus, we demonstrated the possibility of reducing human interventions needed for the system installation.

Moreover, the absolute performances seem to be improved, in particular with the use of a support vector machine downstream. This improvement was not always huge, but seemed sufficiently promising to pursue our investigation further. Indeed, our approach still offers many ways for improvement, mainly possible thanks to the many possibilities offered by algorithms based on the Genetic Programming paradigm.

TABLE DES MATIÈRES

	Page
INTRODUCTION	1
CHAPITRE 1 LE CHOIX D'ESPACE DE REPRÉSENTATION, L'ÉTAT DE L'ART.....	3
1.1 Introduction.....	3
1.2 Les données collectées.....	4
1.3 Calcul de caractéristiques.....	7
1.3.1 Intérêt	7
1.3.2 Mises en pratique	8
1.3.3 Limites	9
1.4 La « simple » sélection de caractéristiques.....	10
1.4.1 Présentation.....	10
1.4.2 Le problème de l'interdépendance.....	10
1.4.3 L'évaluation de la pertinence.....	11
1.4.4 Les algorithmes classiques.....	12
1.5 Transformation linéaire de l'espace de représentation.....	14
1.5.1 Définition	14
1.5.2 L'analyse en composantes principales.....	14
1.5.3 Autres méthodes.....	15
1.6 Transformation non linéaire.....	16
1.6.1 Définition	16
1.6.2 Où s'arrêter?.....	16
1.6.3 Mises en pratique	16
1.6.4 La programmation génétique	17
1.7 Exemple de Krzysztof Krawiec	18
1.7.1 Idée directrice.....	18
1.7.2 Réalisation.....	20
1.7.3 Résultats et conclusion.....	21
CHAPITRE 2 LA PROGRAMMATION GÉNÉTIQUE POUR REMANIER L'ESPACE DE REPRÉSENTATION.....	24
2.1 Introduction à la programmation génétique.....	24
2.1.1 Présentation générale	24
2.1.2 Vocabulaire associé à la PG.....	25
2.1.3 Codage des individus	26
2.1.4 Opérateurs génétiques	28
2.1.5 Fonctionnement général de la PG	30
2.2 Pourquoi utiliser la PG pour remanier l'espace de représentation?	30

2.3	La coévolution.....	31
2.3.1	Présentation générale	31
2.3.2	Adaptation au remaniement de l'espace de représentation.....	31
2.3.3	Calcul de l'adéquation.....	32
2.3.4	Création des individus initiaux	34
2.3.5	Fonctionnement.....	35
2.4	Garde-fous.....	36
2.5	Les complexités.....	37
2.5.1	Présentation générale	37
CHAPITRE 3 PROTOCOLE EXPÉRIMENTAL		39
3.1	Idée directrice.....	39
3.2	Weka	39
3.2.1	Présentation.....	39
3.2.2	Pourquoi l'utiliser?.....	40
3.3	Méthodologie	41
3.3.1	Les données.....	41
3.3.2	Les caractéristiques simples.....	41
3.3.3	Points de comparaisons.....	43
3.3.4	Grammaire de la PG.....	44
3.4	L'implémentation en Java.....	45
3.5	Les résultats à chercher	47
3.6	Séparation des bases de données.....	49
3.6.1	Entre la PG et la classification finale	49
3.6.2	Au cours de la PG	50
CHAPITRE 4 RÉSULTATS EXPÉRIMENTAUX.....		51
4.1	Analyse en quartiles	51
4.1.1	Définitions.....	51
4.1.2	Les simulations effectuées	52
4.1.3	La « Boîte à moustaches »	53
4.2	Résultats sans PG	55
4.2.1	Notations des différents classificateurs.....	55
4.2.2	Sur la base des céréales	56
4.2.3	Sur la base des chiffres manuscrits	58
4.3	Évolution type de la PG	59
4.3.1	Vision d'ensemble.....	59
4.3.2	Influence du nombre d'espèces.....	64
4.3.3	Influence de la taille de la population	68
4.3.4	Autres paramètres de la PG.....	69
4.4	Performances des nouveaux jeux de caractéristiques	70
4.4.1	Sur les chiffres manuscrits	70
4.4.2	Sur les céréales.....	74
4.5	Juxtaposition aux attributs bruts	78

4.5.1	Sur les chiffres manuscrits	78
4.5.2	Sur les céréales	81
4.6	Calcul du gain de la PG sur d'autres méthodes	85
4.6.1	Sur les chiffres manuscrits	85
4.6.2	Sur les céréales	89
4.7	Analyse des résultats	91
4.7.1	Sur les chiffres manuscrits	91
4.7.2	Sur les céréales	92
4.7.3	Récapitulatif	92
CHAPITRE 5 ANALYSE STATISTIQUE AVANCÉE		94
5.1	Intervalles de confiance	94
5.1.1	Mesure de la performance d'un système	94
5.1.2	Intervalle de confiance de la performance d'une réalisation du système	95
5.1.3	Intervalle de confiance de l'espérance de la performance du système	96
5.1.4	Comparaison fine de deux systèmes	97
5.1.5	Validation croisée?	98
5.2	Application sur la base des chiffres manuscrits	99
5.2.1	Performance d'une réalisation du système	99
5.2.2	Espérance de la performance des systèmes	101
5.2.3	Comparaisons	103
5.3	Application sur la base des céréales	104
5.3.1	Performance d'une réalisation du système	104
5.3.2	Espérance de la performance des systèmes	106
5.3.3	Comparaisons	109
5.4	Discussion	110
5.4.1	Bilan des comparaisons	110
5.4.2	Critique des résultats	111
CONCLUSION		113
RECOMMANDATIONS		116
ANNEXE 1	SIMULATIONS SUR LA BASE DES CHIFFRES MANUSCRITS	117
ANNEXE 2	SIMULATIONS SUR LA BASE DES CÉRÉALES	122
ANNEXE 3	ANALYSE EN QUARTILES : BASE DES CHIFFRES MANUSCRITS	127
ANNEXE 4	ANALYSE EN QUARTILES : BASE DES CÉRÉALES	131
ANNEXE 5	INTERVALLES DE CONFIANCE : BASE DES CHIFFRES MANUSCRITS	134

ANNEXE 6	INTERVALLES DE CONFIANCE : BASE DES CÉRÉALES	137
BIBLIOGRAPHIE		140

LISTE DES TABLEAUX

	Page
Tableau 1	Notations des classificateurs55
Tableau 2	Quantile de Student.....96
Tableau 3	Taux d'erreurs du kNN sur différentes représentations de la base des chiffres manuscrits.....118
Tableau 4	Taux d'erreurs du MLP sur différentes représentations de la base des chiffres manuscrits.....119
Tableau 5	Taux d'erreurs du SMO sur différentes représentations de la base des chiffres manuscrits.....120
Tableau 6	Taux d'erreurs du LibSVM sur différentes représentations de la base des chiffres manuscrits.....121
Tableau 7	Taux d'erreurs du kNN sur différentes représentations de la base des céréales123
Tableau 8	Taux d'erreurs du MLP sur différentes représentations de la base des céréales124
Tableau 9	Taux d'erreurs du SMO sur différentes représentations de la base des céréales125
Tableau 10	Taux d'erreurs du LibSVM sur différentes représentations de la base des céréales126
Tableau 11	Analyse en quartiles sur la base des chiffres manuscrits128
Tableau 12	Analyse en quartiles du gain entre systèmes à classificateur identiques (base des chiffres manuscrits)129
Tableau 13	Analyse en quartiles du gain entre systèmes à classificateur différents (base des chiffres manuscrits).....130
Tableau 14	Analyse en quartiles sur la base des céréales.....132
Tableau 15	Analyse en quartiles du gain entre systèmes (base des céréales).....133
Tableau 16	Intervalles de confiance pour une réalisation du système (base des chiffres manuscrits).....135

Tableau 17	Intervalles de confiance de l'espérance de la performance d'un système (base des chiffres manuscrits).....	136
Tableau 18	Intervalles de confiance pour une réalisation du système (base des céréales)	138
Tableau 19	Intervalles de confiance de l'espérance de la performance d'un système (base des céréales).....	139

LISTE DES FIGURES

		Page
Figure 1	<i>Chèque numérisé.....</i>	5
Figure 2	<i>Montage de prise de photo.....</i>	6
Figure 3	<i>Chiffres manuscrits extraits de la base MNIST [3]</i>	7
Figure 4	<i>Les deux axes d'une ACP sur la photo d'un poisson.....</i>	15
Figure 5	<i>Extrait de la base de données MSTAR utilisée par Krzysztof Krawiec .</i>	19
Figure 6	<i>Taux de reconnaissance sur la base de test en fonction du nombre de classes et du nombre de sous-système.....</i>	21
Figure 7	<i>Exemples d'arbre pour la PG.....</i>	27
Figure 8	<i>La mutation</i>	28
Figure 9	<i>Le croisement</i>	29
Figure 10	<i>Illustration du calcul de l'adéquation. Le meilleur individu de chaque espèce est marqué par un « X ».</i>	33
Figure 11	<i>Exemple d'arbre par la « méthode de croissance » (à gauche) et la « méthode complète » (à droite) pour une hauteur 3.....</i>	35
Figure 12	<i>Interface de Weka</i>	40
Figure 13	<i>Exemple de « boîte à moustaches »</i>	53
Figure 14	<i>Comparaison de « boîtes à moustaches ».....</i>	54
Figure 15	<i>Taux d'erreurs des classificateurs sur les caractéristiques originales de la base de céréales</i>	57
Figure 16	<i>Taux d'erreurs des classificateurs sur les caractéristiques originales de la base des chiffres manuscrits</i>	58
Figure 17	<i>Évolution de l'adéquation maximale sur la base des chiffres manuscrits</i>	61
Figure 18	<i>Évolution des adéquations moyennes sur la base des chiffres manuscrits. L'adéquation maximale est représentée en gras</i>	62

Figure 19	<i>Évolution de la taille des élites sur la base des chiffres manuscrits.....</i>	63
Figure 20	<i>Évolution de l'adéquation maximale suivant le nombre d'espèces</i>	65
Figure 21	<i>Évolution de l'adéquation maximale suivant le nombre d'espèces à effort de calcul équivalent.....</i>	66
Figure 22	<i>Évolution de l'adéquation maximale suivant la taille des populations à effort de calcul équivalent.....</i>	69
Figure 23	<i>Taux d'erreurs des classificateurs sur les caractéristiques obtenues par PG sur la base des chiffres manuscrits</i>	71
Figure 24	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base des chiffres manuscrits et ces mêmes classificateurs sur les données brutes normalisées.....</i>	72
Figure 25	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de chiffres manuscrits et le kNN norm sur les données brutes normalisées</i>	73
Figure 26	<i>Taux d'erreurs des classificateurs sur les caractéristiques obtenues par PG sur la base de céréales</i>	75
Figure 27	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de céréales et ces mêmes classificateurs sur les données brutes normalisées.....</i>	76
Figure 28	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de céréales et le MLP sur les données brutes normalisées.....</i>	77
Figure 29	<i>Taux d'erreurs des classificateurs sur les caractéristiques obtenues par PG sur la base de chiffres manuscrits associées aux données brutes.....</i>	78
Figure 30	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques brutes et obtenues par PG sur la base des chiffres manuscrits et ces mêmes classificateurs sur les données brutes</i>	79
Figure 31	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques brutes et obtenues par PG sur la base des chiffres manuscrits et le kNN norm sur les données brutes normalisées.....</i>	80
Figure 32	<i>Taux d'erreurs des classificateurs sur les caractéristiques obtenues par PG sur la base de céréales associées aux données brutes</i>	82

Figure 33	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques brutes et obtenues par PG sur la base de céréales et ces mêmes classificateurs sur les données brutes normalisées.....</i>	83
Figure 34	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques brutes et obtenues par PG sur la base de céréales et le MLP sur les données brutes normalisées.....</i>	84
Figure 35	<i>Taux d'erreurs des classificateurs sur les trente caractéristiques conservées par AG de la base des chiffres manuscrits</i>	85
Figure 36	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base des chiffres manuscrits et ces mêmes classificateurs sur les données conservées par AG</i>	86
Figure 37	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de chiffres manuscrits et le kNN norm sur les données conservées par AG et normalisées</i>	87
Figure 38	<i>Taux d'erreurs des classificateurs sur les trente caractéristiques conservées par ACP de la base des chiffres manuscrits.....</i>	88
Figure 39	<i>Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de chiffres manuscrits et ces mêmes classificateurs sur les données conservées par ACP</i>	89
Figure 40	<i>Taux d'erreurs des classificateurs sur les caractéristiques transformées par ACP sur la base des chiffres manuscrits</i>	90
Figure 41	<i>Intervalles de confiance d'une réalisation de système. Les valeurs représentent les bornes minimum et maximum de ces intervalles</i>	100
Figure 42	<i>Intervalles de confiance de l'espérance de la performance des systèmes.....</i>	102
Figure 43	<i>Intervalles de confiance d'une réalisation de système. Les valeurs représentent les bornes minimum et maximum de ces intervalles</i>	105
Figure 44	<i>Intervalles de confiance de l'espérance de la performance des systèmes.....</i>	107
Figure 45	<i>Intervalles de confiance des meilleures espérances selon une échelle permettant de les départager plus facilement.....</i>	108

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACP	Analyse en Composantes Principales
AG	Algorithmes Génétiques
B&B	« Branch & Bound »
BFF	« Best First Search »
BS	« Beam Search »
C4.5	Algorithme utilisé pour générer un arbre de décision, développé par Ross Quinlan
GPL	« General Public Licence »
kNN	Algorithme de classification de type « k plus proches voisins »
LISP	Contraction de « list processing ». Langage de programmation de type impératif et fonctionnel.
LRS	« Plus-L Minus-R Selection »
MLP	Réseau de neurones de type perceptron multicouche (« Multi-Layer Perceptron »)
PG	Programmation génétique
RGSS	« Random Selection plus Sequential Selection »
SBS	« Sequential Backward Selection »
SFS	« Sequential Forward Selection »
SMO	Optimisation séquentiel minimale, de l'anglais « Sequential Minimal Optimization »
SVM	Machine à vecteurs de support (« Support Vector Machine »)

LISTE DES SYMBOLES ET UNITÉS DE MESURE

PARAMÈTRES DE LA PROGRAMMATION GÉNÉTIQUE

p	nombre de populations
n	taille de chaque population
g	nombre de générations

COMPLEXITÉ ALGORITHMIQUE

m	taille de la base d'apprentissage
$O(a(n))$	suite dominée par $a(n)$: $b(n) = O(a(n)) \Leftrightarrow \exists A \in \mathbb{R} \ \forall n \in \mathbb{N} \ b(n) \leq A.a(n)$

STATISTIQUES

$\bar{x}$	moyenne empirique des résultats : $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$
$\hat{\sigma}^2$	variance empirique corrigée : $\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$
Δ_x	erreur due à la dispersion, estimée par : $\Delta_x = t_{\frac{1-\alpha}{2}}^n \cdot \hat{\sigma}$
t_{γ}^k	quantile d'ordre $1-\gamma$ de la loi de <i>Student</i> à k degrés de liberté
α	niveau de confiance
C_n^p	nombre de combinaisons de p éléments parmi n : $C_n^p = \frac{n!}{p!(n-p)!}$

INTRODUCTION

En classification, le choix de l'espace de représentation est un problème récurrent. Il est bien connu que les conséquences sur les performances du système s'en ressentent fortement. Souvent, dans un but d'améliorer les performances et de diminuer la masse des données, nous n'utilisons pas directement les données brutes collectées sur les capteurs (caméra, microphone, radar, ou autres) mais nous préférons calculer des attributs. Il s'agit de procédés d'extraction de caractéristiques pertinentes pour distinguer les objets entre eux, comme des mesures statistiques, des filtres, etc. Afin de déterminer des attributs pertinents, le plus souvent nous cherchons un problème classique auquel notre tâche peut s'apparenter : reconnaissance de caractères, méthodes usuelles de traitements d'images ou autres.

Bien entendu, un tel choix d'espace de représentation n'est pas optimal car non spécifiquement adapté au problème à résoudre. Des efforts peuvent être faits en réalisant une étude plus approfondie de ce qui est utile d'extraire de l'objet à l'aide, par exemple, d'un expert du domaine. Mais une telle approche est coûteuse car elle exige beaucoup de temps de personnes qualifiées.

L'idée de notre recherche est d'automatiser la tâche du choix de l'espace de représentation à l'aide d'un algorithme de type « Programmation Génétique ». Des travaux ont déjà été effectués en ce sens que ce soit par Programmation Génétique ou avec d'autres types d'algorithmes, mais ils sont pour la plupart assez récents. Il reste donc de la place à l'investigation ou à la mise en pratique de certaines idées sur des applications plus variées.

L'objectif de ce projet de maîtrise est d'investiguer la possibilité d'améliorer un système de classification en recherchant un espace de représentation suffisamment pertinent par programmation génétique. L'accent sera mis sur les gains que nous obtiendrons par différentes variantes. Nous nous concentrerons sur un problème de classification

d'objets de types végétaux, mais nous examinerons également les performances sur d'autres types de problèmes comme par exemple la reconnaissance de chiffres isolés.

Dans ce mémoire, nous allons donc commencer par un état de l'art en ce qui concerne le choix de l'espace de représentation, puis nous présenterons l'approche que nous souhaitons investiguer et enfin ses résultats.

CHAPITRE 1

LE CHOIX D'ESPACE DE REPRÉSENTATION, L'ÉTAT DE L'ART

1.1 Introduction

Le rôle d'un classificateur est de déterminer les frontières entre les classes à distinguer dans l'espace de représentation. Pour cela, il dispose d'un certain nombre d'exemples étiquetés ou non qui constituent l'ensemble d'apprentissage. Là dessus nous ajoutons parfois des heuristiques qui sont le reflet de la connaissance du problème que peut en avoir un expert. Dans notre étude, ce deuxième type de connaissance ne sera pas pris en compte car l'objectif est justement d'automatiser la mise au point d'un système de classification.

Il faut bien comprendre que nous ne connaissons pas la fonction de répartition des classes, sinon un simple classificateur bayésien serait théoriquement imbattable. C'est une loi de mathématique statistique élémentaire. Et cela quel que soit l'espace de représentation choisi du moment que nous avons conservé toute l'information de départ. Mais dans le cas usuel où nous avons seulement une base d'apprentissage, le tracé de la frontière par le classificateur dépend plus ou moins du choix de l'espace de représentation. Avec une bonne représentation, la frontière de décision optimale aura un tracé plus simple, et sera donc plus facile à déterminer [1]. En facilitant la recherche de la frontière optimale, nous avons alors plus de chance de nous en rapprocher davantage, et par conséquent d'augmenter les performances du classificateur. De plus, le fait d'avoir un espace de données plus concis peut réduire l'effort de calcul nécessaire à la classification.

Il est donc bien important de comprendre que nous n'avons pas la prétention de créer de l'information nouvelle en remodelant l'espace de représentation. Ceci serait de toute

façon impossible. C'est le constat qu'un classificateur n'est en général pas optimal lorsque l'espace de représentation n'est pas suffisamment explicite qui nous fait penser qu'il y a un gain de performance à réaliser en amont du classificateur. Le nouvel espace ne contient donc pas d'informations qui n'auraient pas été présentes avant. Par contre nous le construirons de façon à ce qu'il soit plus facile pour le classificateur de trouver une frontière de décision intéressante, que nous espérons donc plus proche de l'optimale.

1.2 Les données collectées

Avant d'aller plus loin, introduisons la provenance et la nature des données. De manière générale, le problème de classification se pose face à une scène réelle. Cela signifie qu'il existe physiquement un ou plusieurs objets avec certaines propriétés. Nous connaissons a priori une partie de celles-ci et nous tentons d'avoir davantage d'informations grâce à des capteurs. Prenons un exemple pour clarifier :

- Supposons que l'objet soit un chèque dont nous cherchons à identifier le montant. Dans la pratique les chèques sont numérisés à l'aide d'un numériseur, mais le principe est similaire à une prise de photographie conventionnelle. Par la suite nous ne ferons pas cette distinction, le chèque n'a été choisi qu'à titre d'illustration.
- Les principales propriétés physiques de ce chèque sont : sa composition (en papier), sa taille, sa position, son orientation, l'encre utilisée, le montant indiqué.
- A priori, nous savons qu'un chèque est en papier, nous pouvons connaître approximativement sa position et son orientation mais nous ne savons pas quel est le montant indiqué.
- Étant données les différences de réflectance de la lumière sur du papier recouvert d'encre ou vierge, nous pouvons utiliser un capteur de type caméra afin de prendre des photos. Ces photos sont donc des informations

complémentaires que nous avons choisies de récolter afin de nous donner une chance pour identifier le montant inscrit sur ce chèque.

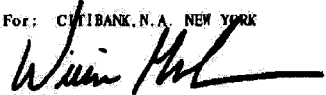
SWEDBANK BRUNKEBERGSTORG 8 STOCKHOLM S-10534 SWEDEN		FOR INFORMATION CONCERNING THIS INSTRUMENT CONTACT CITIBANK GLOBAL CASH MANAGEMENT USA: 1-800-888-8488 OR 1-212-524-7220 LONDON: 020-7500 8961 OR 44-20-7500 8961		CHECK NUMBER: 0140 CLIENT ID:
Pay to the Order of:		FRANCE	CHECK DATE: FEBRUARY 20, 2001 USD \$ 123	
THE SUM OF ONE HUNDRED TWENTY THREE AND 00/100 U.S. DOLLARS*****				
Payable Through:	CITIBANK (NEW YORK STATE) FOR CITIBANK, N.A. ABA 0220 00868			
		For: CITIBANK, N.A. NEW YORK  AUTHORIZED SIGNATURE		
⑈0 3400⑈ ⑆02200⑆		99⑈8		

Figure 1 Chèque numérisé

Source : <http://membres.lycos.fr/topbestcasino/images/cheque.jpg>

Concentrons-nous sur la nature de la photo de la Figure 1. Elle a été obtenue en projetant un certain éclairage sur le chèque. Une partie de cette lumière émise par une source est dirigée vers le chèque, puis a été partiellement réfléchiée avec des propriétés modifiées vers les cellules photosensibles de la caméra. Ces cellules retranscrivent un signal électrique dépendant de la nature et de la puissance du spectre électromagnétique reçu selon une certaine fonction de transfert. Ces signaux électriques sont alors échantillonnés puis stockés dans une matrice appelée image. Voyons ce qu'il en résulte :

- L'information collectée dépend de beaucoup de facteurs indépendants du montant du chèque, par exemple : la nature de l'éclairage, les positions, les caractéristiques de la caméra.
- Les données ont été recueillies et rangées en matrice selon des contraintes physiques et non en fonction de l'information recherchée.
- L'information fournie par la caméra n'est pas la description complète de la surface du chèque mais un simple échantillonnage de certaines des propriétés de réflectance de la lumière.

- Les points d'échantillonnage sont quantifiés selon une dynamique bornée (risque de saturation du capteur) et discrète (approximation des niveaux réels).
- Vu le processus d'acquisition, le bruit est complexe et potentiellement important.
- Alors que nous cherchons peu d'information (un montant de chèque étant un nombre de taille modérée, il pourrait être transcrit sur 3 ou 4 octets), nous recueillons pourtant un grand nombre de données (de l'ordre de plusieurs millions d'octets).

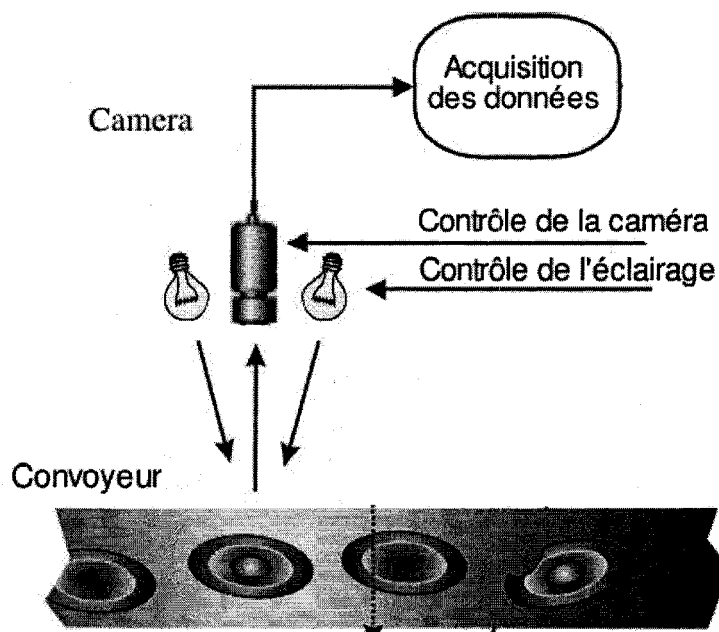


Figure 2 Montage de prise de photo

Source : [http://bruce.cs.cf.ac.uk/bruce/LVM/Methods 060-069/LA Method 062/LA_062\(a\).jpg](http://bruce.cs.cf.ac.uk/bruce/LVM/Methods%20060-069/LA%20Method%20062/LA_062(a).jpg)

Au bilan, nous avons énormément de données par rapport à l'information recherchée. Pourtant elles ne sont qu'une infime partie des propriétés physiques de l'objet, nous sommes très loin d'une description exhaustive de la scène à étudier. Les données

recueillies sont « brutes », au sens où leur nature découle de contraintes essentiellement physiques, budgétaires et de décisions partiellement arbitraires (choix de la caméra, de l'éclairage). La détermination du montant du chèque directement à partir de ces données sera donc ardue pour un classificateur.

1.3 Calcul de caractéristiques

1.3.1 Intérêt

Nous avons mentionné auparavant le concept de connaissance a priori. Reprenons l'exemple du chèque. Nous savons qu'il s'agit d'écriture manuscrite. Nous pouvons donc penser qu'elle sera mieux décrite par la trajectoire du trait de crayon plutôt que par une matrice de pixels. Nous pourrions alors calculer sa courbure, ses recouvrements, le nombre de boucles, ou autres.

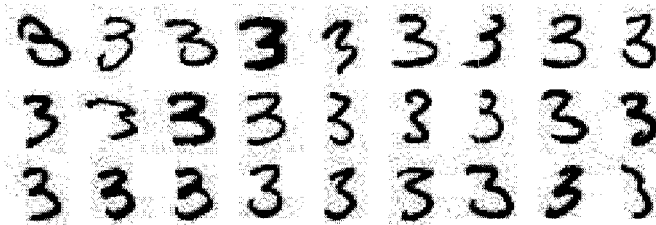


Figure 3 *Chiffres manuscrits extraits de la base MNIST [3]*

Il faut garder à l'esprit qu'un classificateur tente de prendre une décision à partir des données que nous lui fournissons. Plus ces données seront « parlantes », plus il lui sera facile de prendre cette décision et ses performances seront potentiellement accrues. Il y a plusieurs raisons à cela :

- La définition du calcul des caractéristiques est issue de connaissances a priori d'un expert sur le problème. Que le classificateur utilise ces données revient

indirectement à injecter une partie des connaissances a priori de l'expert, et qui dit connaissances supplémentaires dit décision potentiellement meilleure.

- Nous ne connaissons pas la fonction de répartition des classes, le classificateur n'a donc pas l'information nécessaire pour tracer la frontière optimale entre ces classes. Il tente de la tracer à partir des données qu'il a en y ajoutant une heuristique : la frontière doit être d'une complexité modérée pour ne pas tomber dans le sur-apprentissage. Cette complexité est implicitement liée à la topologie de l'espace de représentation utilisé par les données. Or, avoir un espace de représentation choisi en fonction de ce que nous cherchons (à l'aide de calcul de caractéristiques) aura probablement une frontière théorique entre les classes moins complexe que dans le cas de l'espace de représentation des données brutes. Par conséquent, l'heuristique de complexité modérée de la frontière sera davantage justifiée. Nous pourrions donc nous attendre à de meilleurs résultats.

D'autre part, une fois un bon ensemble de caractéristiques choisi, nous éliminerons généralement les données brutes de l'entrée du classificateur. Ce qui implique que la dimension de l'entrée sera nettement inférieure à celle de l'ensemble initial. Cela aide bien entendu le classificateur à faire son travail plus simplement et plus rapidement.

1.3.2 Mises en pratique

Le plus souvent, si nous ne souhaitons pas utiliser les données brutes directement à l'entrée du classificateur, le jeu de caractéristiques est choisi par quelqu'un (que nous appellerons expert) qui tentera de déterminer les informations intéressantes à extraire. Pour cela, il s'inspire de ce qui a déjà pu être fait pour des problèmes similaires, qu'il adaptera éventuellement aux spécificités du système à concevoir. Par exemple, pour une analyse de photos de grain de blé, il pourrait décider de calculer la rondeur, la surface, et les concavités.

1.3.3 Limites

Bien que la détermination d'un ensemble de caractéristiques par un expert soit justifiée dans un certain nombre de cas, il faut toutefois être conscient des limites de cette approche. Ses choix seront probablement pertinents, mais partiellement arbitraires : un autre expert aurait pu avoir une appréciation du problème ou une expérience différente. De ce fait, le choix de l'espace de représentation n'aurait probablement pas été le même. Il serait possible de réduire cet aléa en choisissant plusieurs experts pour créer ce que nous appelons un comité d'expert. Mais dans ce cas, c'est le coût financier de l'opération qui monte en flèche.

En effet, mandater un expert est en général onéreux. En fonction du budget disponible pour réaliser le système de classification, cela pourrait être rédhibitoire, ou du moins prendre des ressources qui auraient pu être utilisées autrement au cours du projet, peut-être avantageusement.

Enfin, si un expert peut apporter des connaissances intéressantes, ce n'est pas le cas d'un calculateur. L'expert apportera certainement des atouts indéniables de par son expérience et son intuition, mais l'énorme travail de calcul numérique d'optimisation d'une base sera probablement plus adapté à un algorithme. Nous pouvons d'ailleurs deviner que la construction de caractéristiques faite par un homme ou une machine sera différente, et sans doute complémentaire.

1.4 La « simple » sélection de caractéristiques

1.4.1 Présentation

Voyons maintenant une première façon de changer un espace de représentation : en éliminant purement et simplement certaines caractéristiques pour ne conserver que celles qui seront jugées les plus pertinentes. En réduisant la dimension du problème, la charge de calcul du classificateur sera allégée, ce qui laisse envisager des gains de vitesse d'exécution.

De plus, des caractéristiques « parasites » (c'est-à-dire qui fluctuent sans véritable corrélation avec l'information recherchée) peuvent gêner la phase d'apprentissage du système et ainsi entacher le taux de bonnes classifications. Cela est d'ailleurs largement vérifié empiriquement [4]. Ce phénomène montre bien qu'un classificateur ne trouve pas dans la pratique la frontière de décision optimale et qu'il est possible de faire des gains en changeant l'espace de représentation.

1.4.2 Le problème de l'interdépendance

Nous avons mentionné la notion de caractéristiques jugées les plus pertinentes. Cependant cette appréciation est plus ardue qu'elle n'y paraît car ces caractéristiques ne doivent pas être évaluées individuellement. En effet, une information ne révèle parfois son importance que si elle est couplée à une autre. Prenons un exemple simple :

- les objets seraient les nombres réels compris entre 0,5 et 9,5;
- les caractéristiques seraient leurs $n^{\text{ième}}$ chiffre en écriture décimale;
- l'information recherchée serait le nombre entier le plus proche de notre objet.

Ainsi, prise individuellement, la deuxième caractéristique (donc la première décimale du nombre) ne donne statistiquement aucune information sur ce qui est recherché (le

nombre entier le plus proche). Une évaluation individuelle des caractéristiques conclurait que seul le premier chiffre en écriture décimale est pertinent. Pourtant, dans cet exemple, il est évident que la deuxième caractéristique revêt son importance quand elle est couplée à la première. De même pour les suivantes.

Nous voyons alors bien qu'il est très utile d'évaluer la pertinence des caractéristiques jumelées. Le fait de ne pas les évaluer risque de nous faire passer à côté d'informations essentielles pour la bonne prise de décision du classificateur. Certains algorithmes de sélection, dans un souci de rapidité ou de simplicité, utilisent tout de même une évaluation individuelle, mais cela doit être fait en connaissance de cause.

Nous pouvons également noter qu'il est illusoire de croire que décorréler les caractéristiques avant la sélection permettent de s'affranchir de leur interdépendance. Le simple exemple donné ci-dessus permettra de s'en convaincre car il n'y a dans ce cas aucune corrélation entre les caractéristiques.

1.4.3 L'évaluation de la pertinence

Revenons à l'évaluation proprement dite d'un jeu de caractéristiques. En effet un algorithme de sélection aura besoin de quantification pour faire ses choix. Idéalement, nous devrions utiliser le taux de bonne classification du classificateur qui sera utilisé. De par l'effort de calcul nécessaire pour refaire l'apprentissage sur chacun de ses sous-ensembles, cette solution est parfois difficilement réalisable. Dans certains cas il est possible de faire des aménagements pour approximer les résultats escomptés du classificateur. Ainsi, dans le cas d'un réseau de neurones de type perceptron, plutôt que de refaire systématiquement l'apprentissage, nous réutilisons le classificateur entraîné sur toutes les caractéristiques et nous inhibons par la suite celles qui ne sont pas dans le jeu à évaluer (en les remplaçant par des valeurs moyennes).

Une autre approche serait une étude statistique, qui a l'énorme avantage d'être beaucoup plus rapide. Nous pouvons par exemple citer le critère de Fisher [2]. Pour simplifier, nous pouvons dire qu'il sera d'autant plus fort que les classes seront espacées dans l'espace de représentation des caractéristiques, et que les objets d'une même classe seront regroupés. Il s'agit donc de maximiser le rapport entre la variance interclasse et la variance intraclasse.

1.4.4 Les algorithmes classiques

Maintenant, voyons où se situe la difficulté de déterminer ce sous-ensemble. Il se trouve que c'est la complexité combinatoire qui est problématique. En effet, si nous avons n caractéristiques possibles, le nombre de sous-ensembles que nous pouvons en extraire est alors 2^n . Pour se donner une idée, avec un jeu de 30 caractéristiques, cela fait plus d'un milliard de combinaisons possibles. Il existe plusieurs familles d'algorithmes de sélection de caractéristiques [5][6]. Elles se différencient par la nature de leur recherche :

- Les complètes : « Branch & Bound » (B&B), « Best First Search » (BFF), ou « Beam Search » (BS). Elles recherchent systématiquement plusieurs combinaisons, parfois toutes et en déduisent quel jeu retenir. Voici ce qui les caractérise :
 - o Très forte complexité
 - o Optimale
- Celles faisant appel à des heuristiques : « Sequential Forward Selection » (SFS), « Sequential Backward Selection » (SBS), ou « Plus-L Minus-R Selection » (LRS). Ici, afin de limiter les calculs, il faut émettre des hypothèses pour décrire comment varient les performances quand nous passons d'un jeu de caractéristiques à un autre. En conséquence :

- La complexité est moindre (en général de l'ordre n^2)
 - La solution est toutefois sous-optimale car les hypothèses formulées ne sont souvent pas rigoureusement respectées par le problème.
- Les méthodes stochastiques : « Random Selection plus Sequential Selection » (RGSS), Recuit Simulé (« Simulated Annealing »), ou Algorithmes Génétiques (AG). Ici, nous regroupons toutes les méthodes qui font appel à un tirage de nombre aléatoire. Ainsi, deux exécutions successives du même algorithme avec les mêmes paramètres ne donneront pas forcément la même solution. Cependant, après un certains temps, ces solutions auront normalement tendance à se rejoindre. Nous pouvons noter au sujet de ces algorithmes qu'ils sont :
- Lourds, mais de complexité acceptable. Ainsi, nous voyons tout de suite qu'ils seront préférablement proposés pour des problèmes où nous ne disposons pas de beaucoup de caractéristiques.
 - Optimaux si nous leur laissons suffisamment de temps pour converger convenablement. Dans la pratique, pour un problème complexe, nous ne pouvons jamais être certain que la convergence est atteinte, il se peut aussi que cette convergence ne soit pas atteignable dans un temps raisonnable. Toutefois, théoriquement le caractère partiellement aléatoire de ces méthodes assure que la convergence se fait sur la meilleure solution possible.

Ce sont des approches différentes, et le choix de l'une d'entre elle se fera suivant le problème que nous avons à résoudre, en particulier selon la dimension de l'espace de recherche.

1.5 Transformation linéaire de l'espace de représentation

1.5.1 Définition

Pour aller plus loin que la sélection simple de caractéristiques, nous pouvons envisager une transformation linéaire de l'espace. En réalité, quand nous profitons de cette transformation pour réduire la dimensionnalité, il s'agit d'une projection. Concrètement, d'un ensemble de départ de 100 caractéristiques, nous pourrions en déduire 20 dont chacune serait une combinaison linéaire des 100 originales. Cela laisse donc beaucoup de liberté quant au nouvel espace de représentation. Mais inévitablement, de par la complexité de cette recherche encore supérieure à la sélection, il faudra se limiter à certaines approches à fortes heuristiques.

1.5.2 L'analyse en composantes principales

L'analyse en composantes principales (ACP) [7] est une méthode mathématique d'analyse de données qui consiste à rechercher les directions qui représentent le mieux la dispersion des points disponibles. Nous retiendrons alors les principaux axes en termes de variances et nous éliminerons les autres. De ce fait, nous aurons réduit la dimension en gardant une grande partie de la variabilité des données d'origine. Comme l'ACP est une projection de matrice, cette transformation est donc linéaire. Enfin, de par leur construction, les données ainsi projetées sont complètement décorrélées.

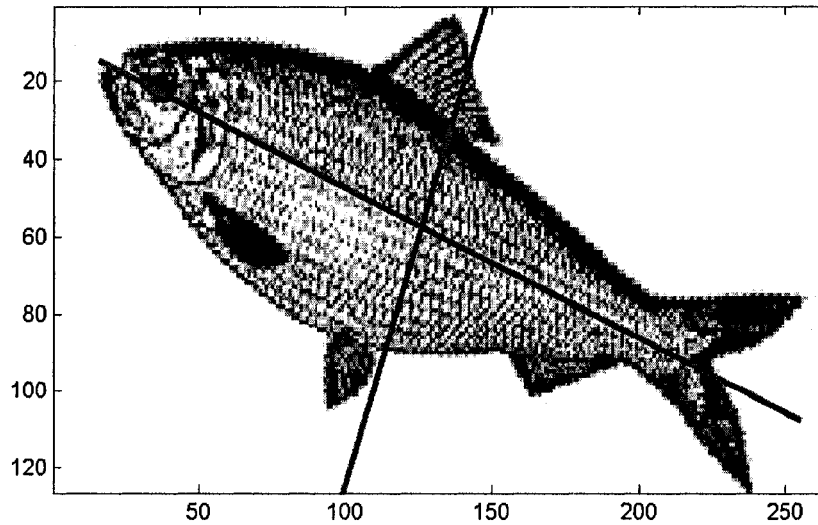


Figure 4 *Les deux axes d'une ACP sur la photo d'un poisson*

Source : Wikipedia [7].

1.5.3 Autres méthodes

Nous pouvons aisément concevoir d'autres algorithmes permettant de construire une transformation linéaire de l'espace. Par exemple, un perceptron multicouche avec une fonction d'activation sigmoïde pourrait très bien accomplir cette tâche. Le nouvel espace serait alors une des couches cachées du perceptron. C'est un bel exemple de détournement en transformateur d'espace d'un algorithme habituellement utilisé comme classificateur.

1.6 Transformation non linéaire

1.6.1 Définition

Enfin, rien n'empêche d'envisager des transformations non linéaires de l'espace de représentation. Dans ce cas nous avons toutes les libertés, ce qui veut également dire que l'espace de recherche est vraiment très vaste.

1.6.2 Où s'arrêter?

Comme nous le mentionnions, dans le cas d'une transformation non linéaire, tout est permis, l'objectif étant de faciliter le travail du classificateur. Or, nous pouvons très bien voir un classificateur comme un cas particulier de transformation non linéaire. Dès lors, il serait en théorie possible d'effectuer tout le travail lors du prétraitement, ce qui est pourtant à proscrire car nous ne ferions que déplacer le problème. En effet, notre objectif est de tirer le meilleur parti de classificateurs éprouvés en optimisant leur entrée, pas de réinventer la roue en concevant un classificateur à part entière en guise de prétraitement. Où se fixer une limite? Comme la frontière est floue, tout ce que nous pouvons faire est de veiller à ce que notre prétraitement n'empiète pas trop sur les domaines où le classificateur est performant mais qu'au contraire il se concentre sur ses lacunes.

1.6.3 Mises en pratique

Le calcul de caractéristiques est une forme de transformation non linéaire des données brutes. Une façon de réaliser ce type de transformation est donc de faire appel à un expert ou de se référer à ce qui a pu être utilisé pour des problèmes de classification similaires.

Cependant nous poursuivons ici l'objectif d'automatiser le processus de choix du prétraitement. Nous allons donc chercher un algorithme pouvant se charger de ce travail. Comme nous allons le montrer plus loin dans ce mémoire, la programmation génétique est un candidat possible (mais il y en a d'autres [8][9]).

1.6.4 La programmation génétique

Dans cette section, nous n'allons pas exposer la programmation génétique (PG) en détail (c'est l'objet du CHAPITRE 2), mais plutôt introduire l'idée de ce choix pour la transformation de l'espace de représentation. La PG aurait également pu être utilisée pour réaliser un classificateur complet, il faut justifier notre choix.

L'un des principaux avantages de la PG est la liberté d'évoluer dans de très grands espaces de recherche. Généralement, assez peu de contraintes sont imposées au mécanisme d'évolution afin de permettre l'obtention d'une solution novatrice, qui vraisemblablement n'aurait pas pu être « pensée » ou « déduite ». Nous pouvons alors nous permettre d'émettre l'hypothèse que nous pourrions évoluer vers des programmes spécifiquement adaptés à un problème sans pour autant avoir eu à apporter les connaissances d'un expert. Néanmoins, il faut être réaliste, il ne s'agit pas de dire que la PG est une intelligence artificielle supérieure à l'être humain. Elle a une grande liberté d'action avec peu de préjugés qui la détournerait de voies qui pourraient être pertinentes. Toutefois, si nous voulons de bons résultats il faut avant tout lui fixer des objectifs clairs et aussi simples que possible. « Aussi simples que possible » ne signifie bien sûr pas trivial, sinon il n'y aurait aucun intérêt à faire usage d'un tel dispositif. Ce que nous disons, c'est qu'il faut simplifier la tâche de la PG :

- Ne la faire travailler que sur ce qui pose vraiment problème. Les tâches que nous savons déjà résoudre, mieux vaut les résoudre manuellement comme à l'habitude. Inutile de faire réinventer à la PG ce que nous connaissons déjà.

Cela risquerait de lui compliquer la tâche et donc au final qu'elle fournisse un moins bon résultat sur ce qui nous intéressait.

- Éventuellement séparer les problèmes à résoudre s'ils sont aisément définissables et à peu près indépendants.

Plus le problème sera simple, moins la PG aura le risque de « se perdre » dans l'espace de recherche. En effet, même si en théorie (si une longueur d'arbre maximum est fixée) la PG devrait converger vers un optimum global, il faut bien se rendre compte que nous nous trouvons dans une telle complexité combinatoire que cette convergence peut très bien ne pas apparaître dans des délais raisonnables [10].

C'est pour le premier point, i.e. concentrer la recherche sur les tâches les plus problématiques, que nous avons choisi de séparer la tâche de remaniement des données de celle de classification. En effet, pour la classification, de très bons algorithmes ont déjà été proposés, par exemple les réseaux de neurones artificiels ou les machines à vecteurs de support (SVM). En réalité tant que les données sont bien représentées, un SVM par exemple donnera d'excellents résultats. C'est pourquoi nous focaliserons la PG sur la construction de caractéristiques, ensuite nous utiliserons par exemple un SVM sur ces résultats pour avoir une idée des performances d'un système complet basé sur ce type d'apprentissage.

1.7 Exemple de Krzysztof Krawiec

1.7.1 Idée directrice

L'idée de construire de nouvelles caractéristiques par transformation non linéaire avec la PG a déjà été proposée dans la littérature [11][12][13], bien que la plupart de ces travaux soient relativement récents. Le polonais Krzysztof Krawiec [14] est un de ceux qui se sont attachés à explorer une des avenues les plus proches de celle que nous proposons.

En effet, il a lui aussi utilisé un algorithme à base de PG, et il a travaillé sur la classification d'image.

Son idée est que les caractéristiques intéressantes peuvent être obtenues à partir de filtres sur l'image suivie d'une fonction de traitement d'image pour en extraire un nombre réel. Il a donc conçu un ensemble de filtres pouvant être appliqués successivement sur une image, comme par exemple :

- Laplacien,
- Prewitt,
- valeur absolue de la différence de deux images.

Il décide également de prendre une base d'images radar (base de données publique MSTAR [15]) pour ses expériences. Ces images représentent des cibles militaires (char d'assaut, véhicule léger, avion), prises sous différents angles ou orientations. Cette base est d'après lui un problème de reconnaissance ardue à cause de leur faible résolution (30 cm par pixel), de la visibilité réduite, des variations importantes lors de rotations (à cause des ombres principalement) et d'un fort niveau de bruit.

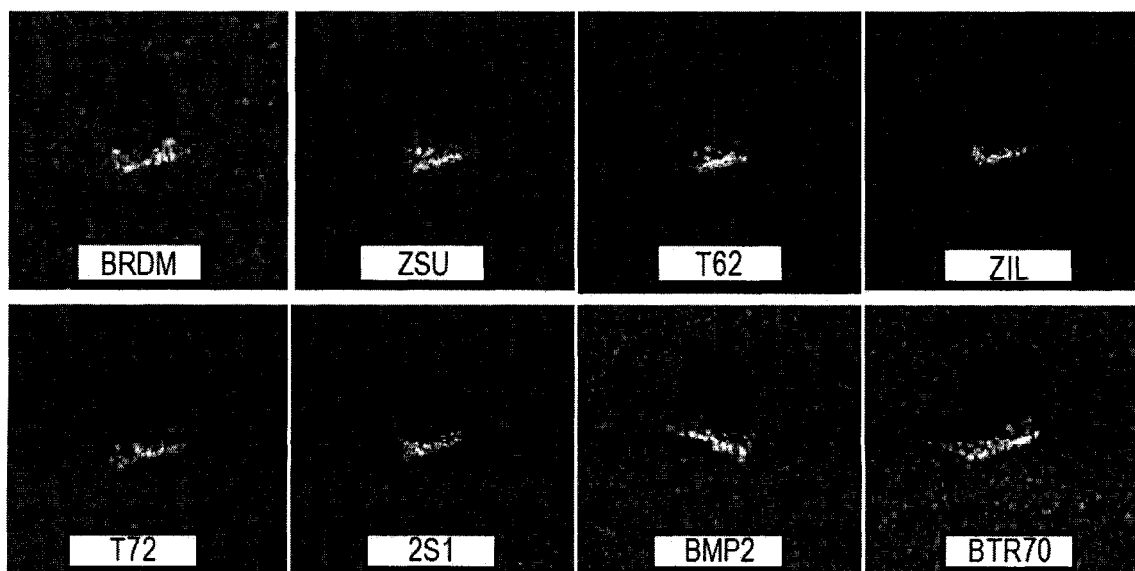


Figure 5 *Extrait de la base de données MSTAR utilisée par Krzysztof Krawiec [14]*

1.7.2 Réalisation

Dans la pratique, K. Krawiec copie plusieurs fois l'image à classer dans des registres images. Ce sera sur ces registres images que les filtres seront appliqués. Les opérateurs utilisés sont ceux provenant de « Intel Image Processing » [16] et de « OpenCV » [17], pour un total de 70 fonctions implémentées dans son système. À côté de cela, il crée également un registre numérique qui servira à stocker le résultat de la caractéristique calculée.

L'assemblage des opérateurs afin d'obtenir des modifications sur les registres est fait sous la forme d'une séquence d'octets de longueur fixe, représentant un programme séquentiel. Nous parlons donc d'une approche hybride entre les Algorithmes Génétique (AG) et la PG, généralement appelée « Programmation génétique linéaire » [10]. En effet, l'évolution se fera de manière similaire aux AG mais les individus codent un programme. Dans le détail, chaque groupe d'octets code pour une opération : le premier représente le choix de la fonction, alors que les trois autres codent pour des paramètres de cette fonction ou pour le choix des registres sur lesquels l'appliquer.

Comme l'objectif n'est pas de construire une seule caractéristique pour la classification, K. Krawiec utilise la coévolution (*voir* CHAPITRE 2) : plusieurs populations d'individus évoluent en parallèle, chacune d'entre elles cherchant une des caractéristiques du jeu à construire. L'évaluation de la pertinence d'un jeu de caractéristiques se fait à l'aide d'un classificateur de type arbre de décision à induction C4.5 [18][19] (il utilise l'implémentation de Weka [20]). Comme l'évaluation des caractéristiques ne se fait pas de façon indépendante, l'évolution des différentes populations de la coévolution est conjointe également.

Au terme de la coévolution, il retient alors le meilleur individu de chaque population, ce qui lui permet donc d'avoir son nouveau jeu de caractéristiques, de dimension égale au

nombre de populations. À partir de ce jeu, il utilise alors une machine à vecteurs de support (SVM) pour la classification. En effet, ces résultats sont alors meilleurs qu'avec le C4.5, et comme la contrainte de temps de calcul n'est plus aussi forte que pendant la coévolution, ce choix se comprend bien.

1.7.3 Résultats et conclusion

Nous avons vu que le système de K. Krawiec commence par construire un jeu de caractéristiques par programmation génétique linéaire, et procède ensuite à une classification avec un SVM. Cependant, au moment de présenter ces résultats, l'auteur avance l'importance d'effectuer plusieurs simulations, chacune d'entre elles classant les objets. De plus, il assemble tous ces résultats sous forme de comité d'experts. La décision finale de classification est celle du vote majoritaire de tous les sous-systèmes de classification réalisés. Les résultats obtenus sont présentés à la *Figure 6*.

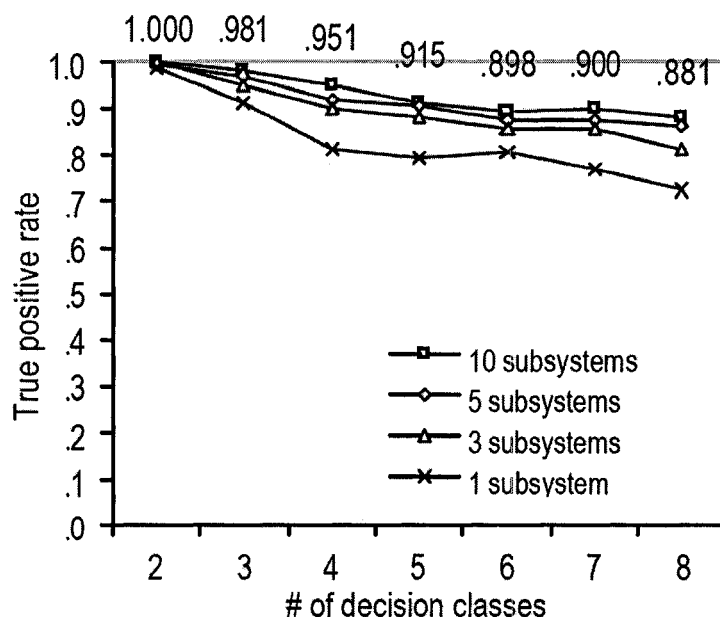


Figure 6 *Taux de reconnaissance sur la base de test en fonction du nombre de classes et du nombre de sous-système [14]*

Ces résultats sont d'après lui d'un bon niveau, et même comparable à ceux que nous pourrions obtenir à l'aide de caractéristiques qui auraient été conçues par un expert ou un modèle approprié. Comparable ne signifie évidemment pas meilleur mais, dans le cas du système qu'il propose, peu d'interventions humaines ont été nécessaires. C'est surtout sur ce point qu'il juge son approche intéressante. D'autre part, il remarque que les solutions codées par les individus de ses populations sont souvent très différentes et innovantes par rapport à ce qu'aurait pu proposer un expert pour la construction de caractéristiques. Selon lui, la solution apportée par son système est davantage objective car libre des préjugés que pourraient avoir un humain.

K. Krawiec travaille toujours aujourd'hui sur le concept de construction de caractéristiques par coévolution de la PG (« coevolutionary feature programming », [21]). Ses derniers travaux ont été dévoilés lors d'une conférence IEEE de 2006 [22], tentant d'acquérir des concepts visuels simples (segments et triangles) d'une scène à l'aide de construction de caractéristiques par PG. Il devrait également être présent cet été à la conférence GECCO 2007 à Londres en Angleterre [21][23].

Les résultats obtenus par K. Krawiec sont donc encourageants. Cependant l'approche que nous proposons est légèrement différente car :

- elle s'appuie sur de la PG à base d'individus-arbres;
- elle utilise des opérateurs arithmétiques et logiques sur un jeu de caractéristiques originales (brutes ou évoluées, mais toujours suffisamment standards et peu spécifiques au problème) plutôt que des filtres sur des images;
- les résultats intermédiaires et finaux sont intrinsèques aux arbres, donc pas de registres;
- l'évaluation de la pertinence des jeux de caractéristiques durant la coévolution se fait à l'aide d'un kNN (classificateur selon les plus proches voisins).

Notre approche est donc plus générique, dans le sens où elle peut s'appliquer à tous systèmes de classification et non aux seuls problèmes de vision comme K. Krawiec. De plus, nous évitons le choix arbitraire des filtres s'appliquant sur les images : les opérateurs arithmétiques sont davantage universels. Enfin, nous espérons que l'utilisation du kNN plutôt que C4.5 permettra une évaluation plus précise de la pertinence des jeux de caractéristiques durant la coévolution de la PG.

CHAPITRE 2

LA PROGRAMMATION GÉNÉTIQUE POUR REMANIER L'ESPACE DE REPRÉSENTATION

2.1 Introduction à la programmation génétique

2.1.1 Présentation générale

La programmation génétique (PG) introduite par John Koza en 1992 [24] est un algorithme de recherche évolutionnaire s'inspirant des mécanismes de la sélection naturelle. Il est important de distinguer la PG des algorithmes génétiques (AG) généralement plus connus.

Comme les AG, la PG utilise une population d'individus qui évoluent au fil des générations par un processus de reproduction. Chaque individu code une solution au problème que nous cherchons à résoudre. Les individus les plus pertinents pour le problème ont davantage de chances d'être sélectionnés pour la reproduction, et nous espérons ainsi que la population sera de plus en plus performante.

Mais à la différence des AG aux génotypes ayant une structure fixe et linéaire, le codage des individus de la PG est plus flexible. Nous pouvons considérer les individus de la PG comme des programmes permettant de résoudre le problème. Typiquement, ces programmes sont représentés sous forme d'arbres (au sens informatique du terme), mais ce n'est pas exclusif : de nombreuses autres méthodes de codage ont été proposées dans la littérature comme des graphes [25], la syntaxe LISP [26], ou encore le code machine [27]. Dans ce travail nous utiliserons uniquement des arbres, nous détaillerons donc ce cas.

2.1.2 Vocabulaire associé à la PG

Avant d'aller plus loin, commençons par définir les termes couramment utilisés pour la PG. Beaucoup de ce vocabulaire est également utilisé pour les AG, avec des significations similaires.

Individu : Entité codant une solution potentielle du problème à résoudre. Ici il s'agira donc d'un arbre, son codage sera détaillé dans la section suivante.

Population : Ensemble d'individus à un moment donné.

Reproduction : Nous parlerons de reproduction au sens de la genèse d'enfants à partir de parents.

Génération : Dans le cas de la formation d'une population complète de remplacement à partir d'une population, nous pouvons distinguer la population des parents et des enfants. Nous parlons alors de génération.

Adéquation : Mesure la pertinence d'un individu à résoudre le problème posé. Sa pertinence est généralement sa performance (taux de bonne reconnaissance pour un classificateur par exemple), mais rien n'empêche d'y inclure d'autres facteurs comme sa concision ou le temps de calcul qu'il requiert. L'objectif est donc de maximiser cette adéquation. Bien entendu, pour ne pas biaiser les résultats finaux, cette mesure ne doit pas faire intervenir la base de test du problème. En anglais, l'adéquation est connue sous le nom « *fitness* ».

Sélection : Il s'agit du choix de parents à reproduire afin de constituer la génération suivante. Les mêmes méthodes que pour les AG peuvent être employées pour la PG.

Sélection par la roulette : Cas où la probabilité de sélection d'un individu parmi une population est proportionnelle à son adéquation.

Sélection par tournoi : Nous choisissons un certain nombre d'individus au hasard (distribution uniforme), puis nous ne retenons que les deux meilleurs individus de ce groupe. Cette méthode est donc insensible à toute transformation strictement monotone de la fonction adéquation, contrairement à la sélection par la roulette.

Élitisme : Technique qui consiste à assurer la conservation par réplique des meilleurs individus de la population. Cela assure de ne pas perdre ces bonnes solutions mais trop d'élitisme peut nuire à la diversité génétique et donc à l'évolution.

Génotype : Nous utiliserons ce terme pour préciser que nous parlons du code en lui-même d'un individu et non de l'action de son programme.

2.1.3 Codage des individus

Tel que mentionné précédemment, un individu code un programme, c'est-à-dire une suite d'instructions afin de résoudre le problème. Voyons le cas où ce code se présente sous forme d'arbre.

Un arbre est une structure issue d'une grammaire récursive, cas particulier d'un graphe ayant une unique source et aucun cycle. Dans un arbre nous pouvons distinguer deux catégories d'éléments :

- les feuilles, éléments ne possédant pas de fils dans l'arbre, ce sont donc des éléments terminaux;
- les nœuds, éléments possédant des fils (dits branches ou sous-arbres).

Nous appellerons racine le nœud n'ayant pas de parents. Par abus de langage nous parlerons parfois de fils pour la racine d'un sous-arbre fils. Nous dirons également que les feuilles sont un cas particulier de nœud n'ayant aucun fils. Dans le cas de la PG, les nœuds sont généralement des instructions ayant pour arguments ses fils.

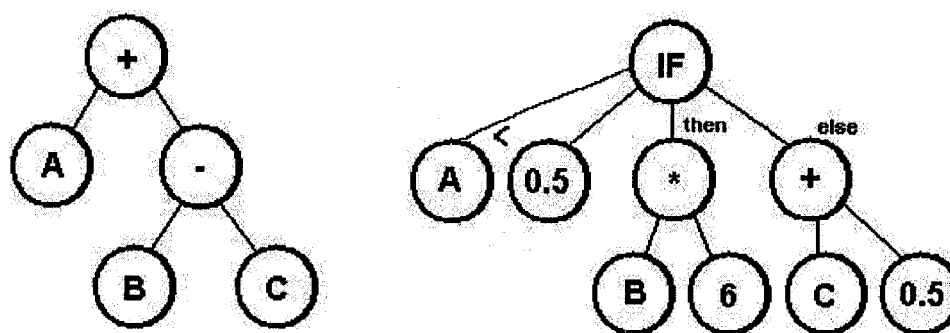


Figure 7 Exemples d'arbre pour la PG

Source : Cette figure, ainsi que les autres montrant des arbres, a été tiré du rapport du projet de synthèse de Yan Levasseur, « Conception et implémentation d'un module de programmation génétique ». Ce rapport a été écrit dans le cadre du cours GPA-790 en génie de la production automatisée à l'École de technologie supérieure (Montréal, Qc, session d'hiver 2005).

Enfin, nous désignerons par « hauteur d'un arbre » son nombre de niveaux ou étages. Plus formellement, sa définition est récursive : la hauteur d'un arbre est égale au maximum des hauteurs de ses sous-arbres augmenté d'une unité. La hauteur d'un arbre ne comportant que sa racine étant un.

2.1.4 Opérateurs génétiques

Un opérateur génétique est une fonction qui à partir d'un certain nombre de parents (en général un ou deux) va produire un certain nombre d'enfants. En voici une liste non exhaustive :

- La **réplication** : Simple copie d'un parent pour en faire un enfant (un clonage en quelque sorte). L'enfant et le parent sont donc identiques.
- La **mutation** : Elle consiste à modifier aléatoirement une partie de l'individu. En pratique, nous choisissons (probabilité uniforme) un nœud, nous détruisons ce qui se trouve en dessous et nous le remplaçons par un nouveau sous-arbre généré aléatoirement. Cela permet d'introduire des types de nœuds qui ne sont peut-être pas présents dans la population. La mutation a donc pour rôle d'accroître la diversité génétique.

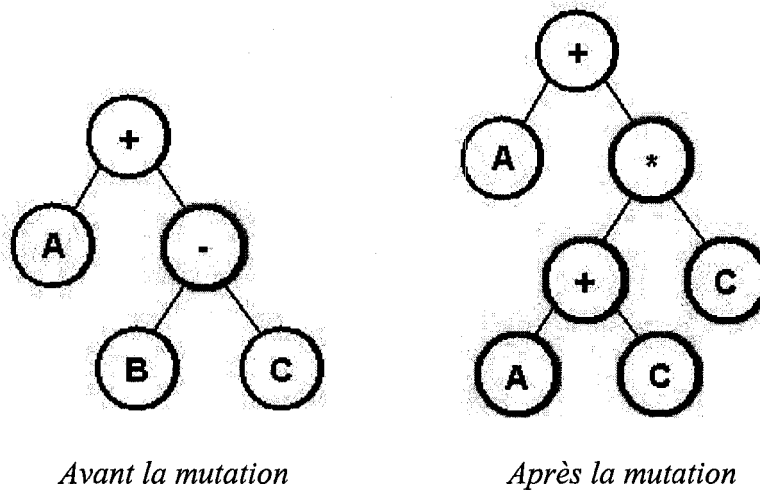
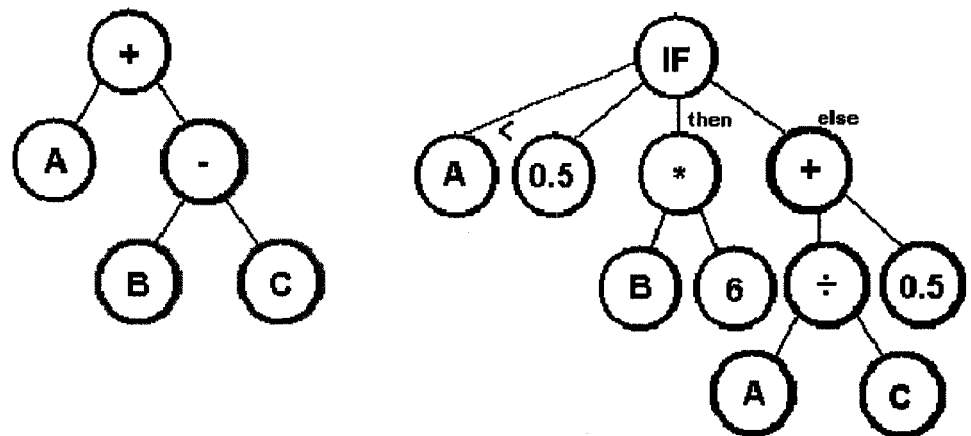
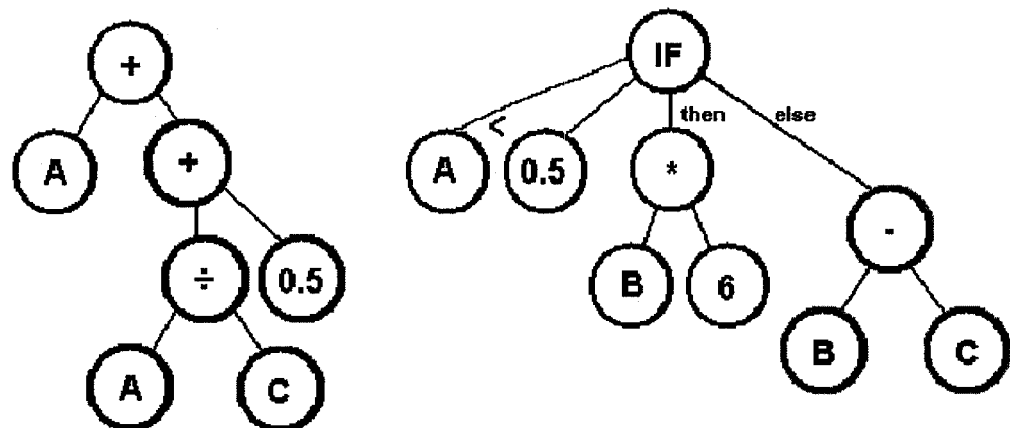


Figure 8 La mutation

- Le **croisement** : Nous le retrouvons dans la littérature anglophone sous le nom de « *crossover* ». Il s'agit de créer deux enfants à partir de deux parents en mélangeant leurs génotypes. Concrètement, nous choisissons un nœud dans chacun des parents, ces nœuds (ainsi que ce qui se trouve en dessous) seront alors permutés, ce qui formera les deux enfants.



Avant le croisement



Après le croisement

Figure 9 *Le croisement*

2.1.5 Fonctionnement général de la PG

La PG est un algorithme qui se déroule comme suit :

1. La première étape est de générer aléatoirement une population initiale.
2. Ensuite nous évaluons l'adéquation des individus.
3. Puis, création de la nouvelle génération : nous choisissons des parents selon une méthode de sélection, lesquels seront utilisés pour donner naissance à leurs enfants grâce aux opérateurs génétiques. Nous créons ces enfants jusqu'à ce que la nouvelle génération ait autant d'individus que la précédente. Éventuellement une partie de la nouvelle génération sera réservée à l'élitisme.
4. Tant que le critère d'arrêt n'est pas respecté (par exemple nombre maximal de génération atteint ou adéquation stable), retour à l'étape 2.

2.2 Pourquoi utiliser la PG pour remanier l'espace de représentation?

Remanier l'espace de représentation est un problème d'optimisation : disposer au mieux l'information de façon à ce que le classificateur puisse l'assimiler facilement lors de sa phase d'apprentissage. Nous avons vu que la PG est un algorithme de recherche, elle peut donc être utilisée pour changer l'espace de représentation.

L'intérêt principal de la PG est, à l'inverse de la plupart de ses concurrents, qu'elle permet d'explorer un espace de recherche très vaste, en particulier les transformations non linéaires. Cependant, telle que décrite précédemment, la PG ne créerait qu'un seul nouvel attribut, elle est donc trop limitative pour ce type particulier de recherche. Nous pourrions envisager que chaque individu code pour la construction d'un n-uplet d'attributs, mais l'évolution serait alors particulièrement délicate de par la complexité de la recherche. Il existe par ailleurs une autre façon de palier à cette difficulté : la coévolution. C'est ce que nous présentons dans la prochaine section.

2.3 La coévolution

2.3.1 Présentation générale

L'idée de la coévolution est d'avoir plusieurs populations qui évolueront en parallèle, chacune ayant sa propre évolution. Si nous devons rejoindre l'inspiration du monde vivant et de sa sélection naturelle, la coévolution modéliserait les évolutions conjointes de diverses espèces (animales ou végétales) dans un même monde. L'environnement d'une espèce inclut alors également l'état des autres espèces. L'interaction entre les espèces se fait donc par le biais de la modification de l'environnement (c'est-à-dire au niveau du calcul de l'adéquation), il n'y a pas de croisements ou d'échanges d'individus entre les espèces. Ce dernier point correspond d'ailleurs à la définition biologique d'une espèce, d'où l'utilisation de ce terme. Certains travaux ont toutefois tenté de faire intervenir des échanges entre populations [28], mais dans ce cas ce n'est plus à proprement parlé de la coévolution. Il s'agit alors davantage d'éviter les impasses évolutionnaires que de poursuivre un objectif multiple comme nous le souhaitons.

La coévolution se situe donc un niveau au dessus de la PG conventionnelle. Afin d'adopter un vocabulaire spécifique, nous appellerons les différentes populations *espèces*, et l'ensemble des espèces sera nommé *monde*. Le choix de ces termes est fait en relation à la littérature anglophone qui utilise généralement « *species* » et « *world* ».

2.3.2 Adaptation au remaniement de l'espace de représentation

Pour notre problème, l'intérêt de la coévolution est d'introduire nativement un ensemble de nouveaux attributs, une espèce correspondant à chacun d'eux. La pression de sélection d'une espèce devra alors mettre l'accent sur ce qu'elle apporte de plus par rapport aux autres espèces. Ainsi nous évoluerons vers un jeu d'attributs complémentaires pour la classification ultérieure. L'objectif est que l'évolution

conjointe des espèces mène à un ensemble cohérent de nouveaux attributs pour le classificateur.

2.3.3 Calcul de l'adéquation

Nous avons mentionné précédemment que le calcul de l'adéquation d'un individu appartenant à une espèce dépendait également de l'état des autres espèces. En effet notre fonction d'adéquation évalue en réalité la pertinence d'un jeu d'attribut pour un classificateur. Une solution serait de faire la moyenne des évaluations des pertinences de tous les jeux d'attributs possibles constitués de l'individu à évaluer et d'un individu de chacune des autres espèces. Cependant nous pouvons deviner que cela serait très lourd en effort de calcul : pour 20 espèces avec chacune 500 individus, il faudrait évaluer $(500)^{20-1}$ jeux d'attributs! Heureusement il y a une autre façon de procéder : ne considérer que le meilleur individu de chacune des autres espèces. Nous calculerons donc la pertinence du jeu d'attributs constitué de l'individu à évaluer accompagné du meilleur individu de chacune des autres espèces. Voici alors la procédure pour évaluer l'adéquation du monde (illustrée par la *Figure 10*) :

Pour chaque espèce E, faire :

 Pour chaque individu I de l'espèce E faire :

 Donner comme adéquation à I la pertinence du jeu d'attributs constitué de I ainsi que du meilleur individu (au sens de l'adéquation) de chacune des espèces autres que E.

 Fin de la boucle faire

Fin de la boucle faire

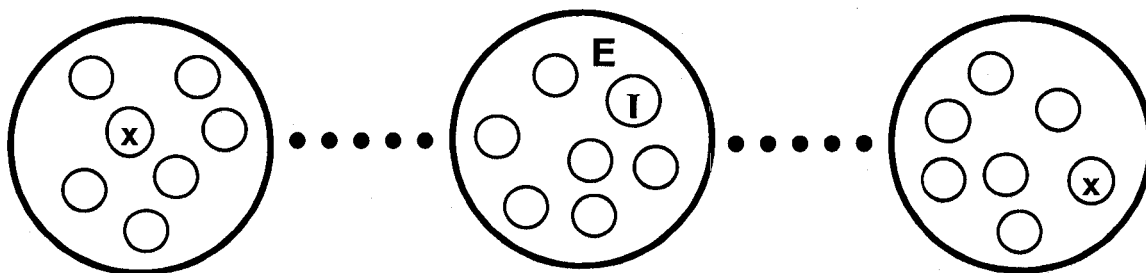


Figure 10 *Illustration du calcul de l'adéquation. Le meilleur individu de chaque espèce est marqué par un « X ».*

Comme nous pouvons le remarquer, pour l'évaluation d'une espèce cette procédure nécessite de connaître les adéquations des autres espèces. En général cela ne pose pas problème, sauf bien sûr pour la première génération. Pour cette phase d'initialisation, nous donnerons des valeurs aléatoires aux adéquations.

Ensuite, il s'agit donc d'évaluer la pertinence d'un jeu d'attribut. Plusieurs méthodes auraient été possibles :

- Statistique : critère de Fisher [2], qui consiste à faire le rapport de la variance intraclasse sur la variance interclasse. Cette méthode a l'avantage d'être assez rapide, mais ne reflète pas forcément bien les performances qu'aura un classificateur avec une telle représentation.
- Classification : utiliser un classificateur réel afin de déterminer son taux de bonne reconnaissance en validation croisée sur la base qui sert d'apprentissage pour la PG. L'indication de pertinence du jeu d'attribut sera donc réellement la performance empirique d'une classification, ce que nous souhaitons *in fine*.

Comme la représentation de la base est différente à chaque évaluation d'un jeu d'attribut, une phase d'apprentissage aurait été lourdement pénalisante au niveau du temps de calcul. C'est pourquoi nous avons choisi l'utilisation d'un kNN. De plus, de

par cette absence de phase d'apprentissage, nous pourrions utiliser une forme particulière de validation croisée, dite « omettre un élément » (« leave one out »). Autrement dit, pour classer un objet nous nous servons de tous les autres éléments de la base pour prendre une décision, afin d'avoir un résultat aussi précis que possible. Enfin, le kNN possède un autre atout important qui lui permettra d'être assez rapide dans le cas de son utilisation au sein de la PG. En effet, le fonctionnement d'un kNN est fondé sur le calcul de distances. Or, lors de l'évaluation de l'adéquation d'une espèce, seul un attribut de l'espace de représentation change, les autres restent identiques au sein d'une génération. En décomposant le calcul d'une distance euclidienne selon les termes de sa somme, nous pouvons conserver en mémoire tous ces termes sauf un. L'évaluation de l'adéquation d'un jeu d'attributs devient alors pratiquement indépendante du nombre d'attributs.

2.3.4 Création des individus initiaux

Pour commencer l'évolution, il faut bien sûr un point de départ, c'est-à-dire construire une population initiale. Nous ne présumerons pas de ce que pourrait être de bonnes solutions, la seule chose que nous désirons avoir pour favoriser une meilleure recherche est la diversité. Nous entendons par là diversité des nœuds utilisés, diversité de taille des arbres et de leurs formes. Nous utiliserons pour cela la méthode « étagée moitié-moitié » (« ramped half-half ») proposée par Banzhaf dans son livre [10].

Selon cette méthode, nous prenons en compte toutes les hauteurs d'arbres comprises entre deux et la hauteur maximale permise. Chacune de ces hauteurs aura la même part de la population initiale. Pour une hauteur maximale de 6, nous aurons donc :

- 20 % des arbres initiaux qui auront une hauteur de deux;
- 20 % une hauteur de trois;
- 20 % une hauteur de quatre;
- 20 % une hauteur de cinq;
- et 20 % une hauteur de six.

Pour chacune de ces hauteurs, la moitié est générée selon la « méthode de croissance » (« grow method »), l'autre selon la « méthode complète » (« full method »). La « méthode de croissance » consiste à générer des arbres en choisissant aléatoirement si un nœud est terminal ou non quel que soit sa hauteur dans l'arbre, sauf pour la hauteur maximale autorisée où il est alors obligatoirement terminal. En conséquence, les branches de l'arbre ainsi générées n'atteignent pas forcément toutes la hauteur maximale, ce qui permet d'avoir une certaine diversité de forme. À l'inverse, la « méthode complète » n'introduit d'éléments terminaux qu'à la hauteur maximale. L'arbre est alors dit complet. Ces méthodes sont illustrées à la *Figure 11*.

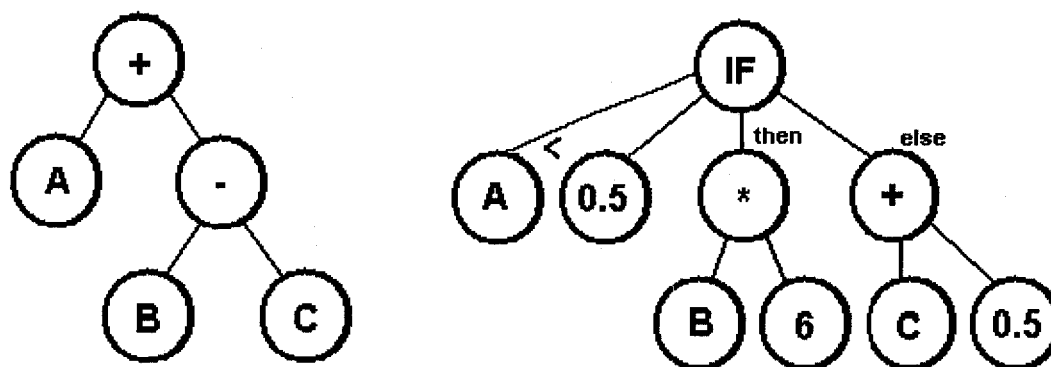


Figure 11 *Exemple d'arbre par la « méthode de croissance » (à gauche) et la « méthode complète » (à droite) pour une hauteur 3*

2.3.5 Fonctionnement

À l'initialisation, nous devons créer autant de populations que nous voulons d'espèces, nombre qui correspond dans notre cas à la dimension souhaitée du nouvel espace de représentation des attributs. Ensuite le fonctionnement est identique à la PG conventionnelle, en alternant les espèces.

1. Générer aléatoirement les populations initiales.
2. Pour chacune des espèces, faire :
 - a. Évaluer l'adéquation des individus (pour la première génération, donner des adéquations aléatoires).
 - b. Créer la nouvelle génération pour cette espèce : Nous choisissons des parents selon une méthode de sélection, lesquels seront utilisés pour donner naissance à leurs enfants grâce aux opérateurs génétiques. Nous répétons cette étape jusqu'à ce que la nouvelle génération ait autant d'individus que la précédente. Éventuellement une partie de la nouvelle génération sera réservée à l'élitisme.
3. Tant que le critère d'arrêt n'est pas respecté (par exemple nombre maximal de génération atteint ou adéquation stable), retour à l'étape 2.

2.4 Garde-fous

Nous avons présenté les opérateurs de mutation et de croisement, mais leurs applications répétées pourraient très bien faire exploser la taille des individus (nombre de nœuds des arbres). Cependant au vu des restrictions dues à la réalisation matérielle des simulations (mémoire des ordinateurs limitées), il faut impérativement fixer une taille maximum des individus. Cette taille sera arbitrairement fixée à 200 nœuds. Certaines publications [24] préfèrent limiter la hauteur des arbres, mais nous avons préféré se référer au nombre de nœuds car cela reflète plus fidèlement l'espace mémoire ainsi que l'effort de calcul nécessaire.

Dans la pratique cette limite se met en place de manière fort simple : si un opérateur génétique crée un arbre trop grand, ce dernier est détruit et un nouvel arbre est construit de la même façon qu'à l'initialisation de la première génération. Nous aurions également pu tronquer ou élaguer ces arbres trop grands. Cependant, cette méthode aurait fait intervenir des choix arbitraires comme les points de coupe et nous aurions aussi dû

porter une attention particulière au respect de la grammaire. L'arbre serait alors probablement fortement dénaturé, c'est pourquoi nous avons préféré la méthode de la destruction des arbres problématiques. La création d'arbres totalement nouveaux permettra également de maintenir une bonne diversité dans la population.

2.5 Les complexités

2.5.1 Présentation générale

La complexité combinatoire d'un algorithme est le moyen de savoir de quelle façon l'effort de calcul augmente lorsque nous lui donnons un problème plus grand. Cette complexité est souvent un indice primordial pour savoir si un programme pourra se terminer dans un temps raisonnable. Prenons les notations suivantes :

- p : nombre de populations
- n : taille de chaque population
- g : nombre de générations
- m : taille de la base d'apprentissage.

Dans le cas de notre système de construction de caractéristiques et de classification, il y a plusieurs éléments (algorithmes ou sous-algorithmes) qui s'enchaînent où encore se font appels l'un dans l'autre. Pour connaître la complexité de notre système, nous avons donc besoin de le décomposer :

- Le kNN qui sert à l'évaluation de la pertinence des individus a une complexité en $O(m^2)$ pour chaque individu, donc en $O(n.m^2)$ pour l'évaluation d'une population au complet et enfin, en $O(g.p.n.m^2)$ pour une simulation au complet. À noter que si nous n'avions pas utilisé l'astuce présentée en 2.3.3, la complexité aurait été en $O(g.p^2.n.m^2)$.

- L'évaluation des arbres est quand à elle en $O(1)$ pour chaque individu car il a une taille maximale. Au niveau d'une simulation complète, cela donne donc $O(g.p.n.m)$
- L'évaluation des performances finales du système à l'aide de différents classificateurs a quant à elle un coût en $O(p.m^2)$. En effet, tous les classificateurs utilisés ont des complexités égales ou inférieures à cela.

Notre système complet a donc une complexité globale en $O(g.p.n.m^2) + O(g.p.n.m) + O(p.m^2) = O(g.p.n.m^2)$. Cette complexité est donc dictée par celle du kNN utilisée en interne pour l'évaluation de l'adéquation des jeux de caractéristiques. Quoique importante, elle demeure quadratique, ce qui laisse bon espoir quant à la réalisation des simulations. Dans la pratique, nous avons en effet des temps de calculs importants, mais qui demeurent dans le domaine du raisonnable : environ 8 heures sur un Athlon XP 2500+ sur la base des céréales et environ 5 jours pour la base des chiffres manuscrits. Quant à la consommation mémoire, elle est de l'ordre de 300 méga octets pour les bases que nous avons utilisées.

CHAPITRE 3

PROTOCOLE EXPÉRIMENTAL

3.1 Idée directrice

Dans un premier temps, nous allons travailler à partir d'un niveau de représentation intermédiaire des données. En cela, il est entendu que nous disposerons de caractéristiques assez simples. Celles-ci sont tout de même plus en rapport avec notre compréhension d'une image qu'avec les seules valeurs de pixels. L'objectif sera de remanier automatiquement ces caractéristiques afin d'en construire de nouvelles qui soient plus adaptées au problème spécifique de classification que nous voulons résoudre. Nous appellerons ce nouvel ensemble « représentation de troisième niveau » par opposition aux caractéristiques élémentaires dites de deuxième niveau et les données brutes (pixels) de premier niveau.

3.2 Weka

3.2.1 Présentation

Weka [20] est un programme d'exploration de données proposé par l'Université de Waikato en Nouvelle-Zélande. Sa licence est de type GPL (« *General Public License* » [29]), donc le code est ouvert et disponible. Weka est le rassemblement dans une même application Java de nombreux algorithmes d'apprentissage. Ainsi nous y retrouvons des outils de prétraitement, de classification et de régression. Il propose également des visualisations graphiques des bases de données. Un livre détaillant son fonctionnement est également disponible [30].

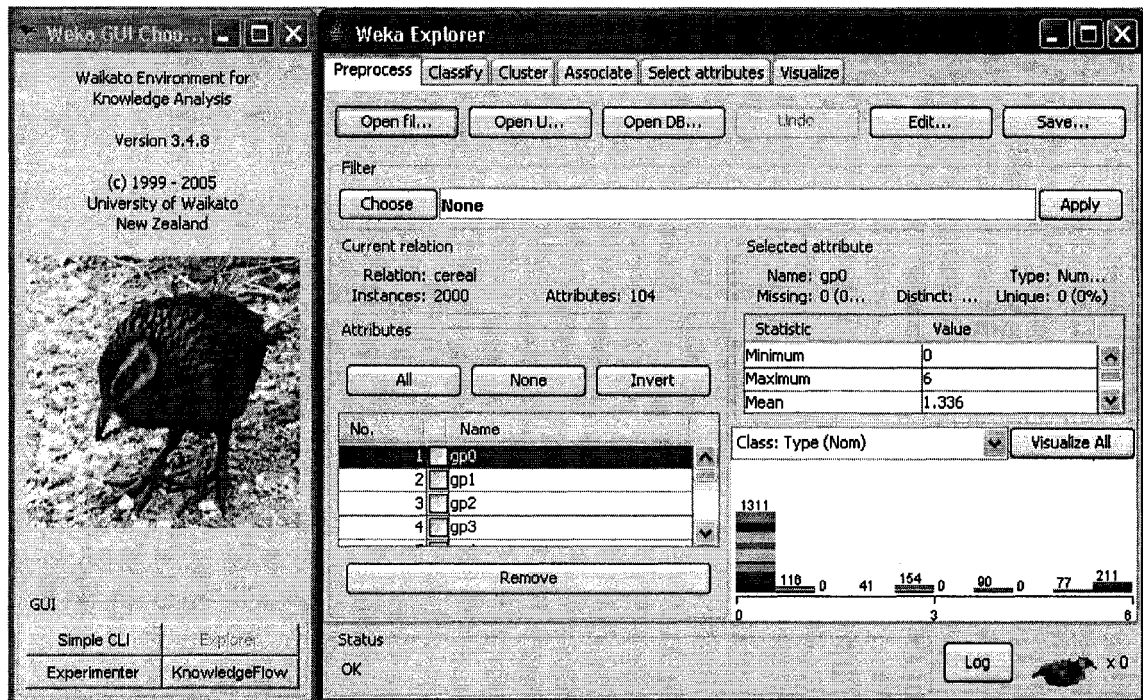


Figure 12 *Interface de Weka*

3.2.2 Pourquoi l'utiliser?

Weka nous permettra de manipuler nos bases de données sans avoir nécessairement à recoder les algorithmes classiques de sélection et de classification. Pour cela, il nous fera gagner du temps tout en nous permettant d'avoir des résultats plus complets. La visualisation rapide des données de manière graphique nous a également permis au début d'identifier les bogues grossiers de nos algorithmes ou encore de mieux interpréter les résultats. Enfin, comme Weka est librement utilisable par tout le monde, c'est un moyen d'assurer la reproductibilité de nos travaux. Ainsi nos résultats pourront être confirmés par une tierce personne ou bien repris pour aller plus loin.

3.3 Méthodologie

3.3.1 Les données

Nous avons choisi de travailler sur des photos de grains de céréales, afin par exemple de pouvoir déterminer quelle est le pourcentage de grains étrangers à celui que nous souhaitons dans un lot. N'ayant pas à notre disposition une base de données assez vaste, nous en avons développée une nous-même en prenant nos propres photos. Nous avons numérisé 1150 exemplaires au total afin de pouvoir avoir une bonne généralisation. Nous avons quatre classes : blé mou rond, orge perlé, orge mondé et avoine. Comme l'objet de notre problématique n'est pas la segmentation, nous avons pris soin d'isoler chaque grain sur la tablette, ainsi un simple seuillage nous permet d'obtenir les grains individuellement.

Enfin, nous envisagerons un autre type d'image : des chiffres manuscrits. Cette base pourrait facilement être reprise pour fin de comparaison car elle est gracieusement mise à disposition du public sur Internet par Robert P.W. Duin, Delft University of Technology [31]. Il s'agit d'une base de 2000 chiffres manuscrits, 200 dans chaque classe.

3.3.2 Les caractéristiques simples

Dans cette approche, nous voulons travailler à partir de caractéristiques assez simples dans le sens où elles doivent nécessiter un faible effort de calcul mais surtout être « générales ». En effet, ici le but de la PG est de s'adapter automatiquement à un problème spécifique, donc ça n'aurait aucun sens de dire que nous devons choisir au préalable un jeu de caractéristiques déjà adapté. Mais il faut faire attention à une chose toutefois : quelques soient les manipulations que nous pourrons faire pour créer de nouvelles caractéristiques, ça restera la même information que celle de départ mais

mieux organisée pour le classificateur. Donc si le jeu de caractéristiques initiales vient à éliminer certaines informations utiles pour la classification, elles ne pourront être recréées. Par exemple, si nous ne tenons pas compte de la couleur au départ, il est évident qu'aucune des nouvelles caractéristiques ne pourra y faire référence. Au bilan, il faut se trouver un jeu de caractéristiques simples mais sans être trop restrictif.

Dans le cas de nos deux bases de données, nous allons utiliser des jeux de caractéristiques différents. Pour les céréales, nous allons utiliser dix attributs élémentaires fournis par MATLAB [32] (bibliothèque « regionprops » de « image processing toolbox ») :

- la surface de l'objet;
- la longueur de l'axe majeur;
- la longueur de l'axe mineur;
- l'excentricité;
- l'orientation : angle entre l'axe des abscisses et l'axe majeur de l'objet;
- la surface convexe;
- la surface une fois les trous bouchés;
- le nombre d'Euler;
- la solidité : rapport entre la surface et la surface convexe;
- l'ampleur : rapport entre la surface de l'objet et l'aire du rectangle parallèle aux axes et délimitant l'objet.

En ce qui concerne les chiffres manuscrits, nous prendrons des données directement liées aux valeurs des pixels des imagerie, plus précisément les 240 moyennes de pixels réalisées sur des fenêtres de deux par trois. Ces données sont celles du fichier « *mfeat-pixel.arff* » présent dans la base disponible au téléchargement [31]. Afin de réduire la masse de données, nous avons éliminé certaines des caractéristiques par simple sélection pour ne retenir que les 103 qui ont la capacité individuelle de prédiction de la classe (Weka : `weka.attributeSelection.CfsSubsetEval` et `weka.attributeSelection.BestFirst`).

Cette dimension de 103 provient du seuil par défaut de la sélection de caractéristiques de Weka.

3.3.3 Points de comparaisons

Notre PG doit construire de nouveaux jeux d'attributs, lesquels sont sensés améliorer les performances de classificateurs en aval. Il faut donc que nous sachions à quoi comparer les résultats obtenus. Comme points de référence, nous utiliserons donc les taux de bonnes classifications de ces différents classificateurs :

- kNN (classement selon les k plus proches voisins) : il s'agit d'un classificateur simple connu de tous ceux qui ont approché le domaine de classification. De plus c'est le classificateur utilisé en interne par notre PG pour évaluer la pertinence des nouvelles caractéristiques. Enfin, il n'est pas stochastique, ces résultats sont tout à fait reproductibles. Pour ces raisons, le kNN était incontournable. Nous utiliserons la distance euclidienne.
- MLP (réseau de neurones de type perceptron multicouche) : il s'agit là aussi d'un classificateur bien connu souvent utilisé comme référence. Les résultats obtenus sont toutefois sujets à changements suivant les aléas de sa phase d'apprentissage. La phase de test est relativement rapide mais la phase d'apprentissage est tout de même assez lourde. Cette lourdeur est toutefois bien en deçà de celle de notre PG.
- Deux implémentations de SVM (SMO [33] et LibSVM [34]) seront également utilisées. Les machines à vecteurs de support sont connues pour être performantes et rapides. Nous utiliserons les deux implémentations disponibles pour Weka car elles donnent souvent des résultats assez différents et elles nous sont facilement accessibles.

Les éventuels paramètres de ces classificateurs seront fixés à des valeurs aussi génériques que possible. Pour cela ils seront laissés aux valeurs par défaut de Weka. En

effet, un des objectifs de notre recherche est de minimiser les interventions humaines pour la mise en place d'un système de classification. Pour chacun de ces classificateurs, nous prendrons la base de données avec toutes les caractéristiques brutes dans un premier temps. Ensuite, nous ferons une comparaison à dimension égale, donc nous réduirons les caractéristiques à l'aide d'un algorithme de sélection pour rejoindre la même dimensionnalité que le jeu produit par la PG.

3.3.4 Grammaire de la PG

Nous avons vu au chapitre précédent que le codage des individus à l'intérieur de la PG était en fait une combinaison d'opérateurs qui s'appliquent sur les données brutes pour produire un nombre réel (la nouvelle caractéristique). Nous devons donc définir un certain nombre d'opérateurs :

- Les opérateurs arithmétiques : l'addition, la soustraction, la multiplication et la division protégée. Pour chacun de ces opérateurs nous avons besoin de deux entrées réelles, celui-ci calcule alors une sortie réelle. La division est protégée au sens que si nous demandons une division par zéro, elle retournera zéro pour fonctionner dans tous les cas de figure. C'est une définition couramment utilisée en programmation.
- Un test conditionnel : si $A < B$ alors C sinon D. Cet opérateur a besoin de quatre fils réels (nommés A, B, C et D) et donne un nombre réel en sortie. Son mode d'opération est que si A est inférieur à B alors il retourne C sinon il retourne D.
- Maximum : il s'agit d'un opérateur binaire qui pour deux réels A et B retournera le plus grand des deux. Nous pouvons remarquer que cet opérateur aurait facilement pu se construire à partir du test conditionnel de cette façon : si $A < B$ alors B sinon A. Il peut donc sembler superflu d'inclure cet opérateur de maximum. Cependant cet opérateur est souvent utilisé dans la construction

de caractéristiques habituelles et nous avons observé une meilleure convergence grâce à son ajout. Il s'agit d'un petit coup de pouce pour la PG!

Les individus sont alors des arbres dont les nœuds internes sont des opérateurs tels que décrits ci-dessus et dont les feuilles (nœuds terminaux) sont soit une référence à une caractéristique de l'ancien espace de représentation, soit une constante réelle (choisie selon la loi normale lors de la création de la feuille).

3.4 L'implémentation en Java

Nous en venons maintenant à l'implémentation. Nous avons choisi le langage Java pour plusieurs raisons : très usuel, portabilité, rapide (lorsque compilé). Comme toujours, en programmation, mieux vaut ne pas se donner d'objectifs trop ambitieux pour le premier jet. Voilà pourquoi nous ne nous sommes pas tout de suite attaqué au problème de la construction de caractéristiques. Au contraire, nous nous sommes fixés des objectifs intermédiaires à atteindre afin de glisser progressivement vers l'algorithme que nous souhaitons :

- Classificateur complet : chaque individu est un arbre qui à partir des caractéristiques simples indique la classe de l'objet. La grammaire est la même que celle décrite dans la section précédente (3.3.4). Cet objectif est plus simple que la construction de caractéristiques car il n'y a aucun besoin d'implanter un classificateur annexe pour l'évaluation de l'adéquation. En effet, il suffit de prendre le taux de bonne classification.
- « Pairwise » : afin de simplifier le problème de classification, nous pouvons séparer un problème multiclasse (n classes) en $\frac{n(n-1)}{2}$ paires de classes.

Dans ce cas le problème est plus simple et nous pouvons donc espérer obtenir une meilleure performance. La décision de la classe à partir de toutes les

réponses des classificateurs biclasses s'obtient soit par vote, soit à l'aide d'un algorithme proposé dans la littérature du « pairwise » [35]. En plus de tenter d'améliorer les performances, cet objectif introduit une PG avec plusieurs populations, ce que nous souhaitons pour la construction de caractéristiques. Dans un premier temps, l'adéquation des classificateurs biclasses sera leurs taux d'erreurs sur les deux classes concernées.

- « Pairwise » avec coévolution : dans le point précédent, il y avait bien présence de plusieurs populations, mais pas de véritable coévolution car chacune des évolutions se faisaient indépendamment. Comme en réalité la reconstruction de la décision multiclassée à partir des réponses biclasses est inconsistante, il pourrait être intéressant de prendre en compte l'algorithme de reconstruction dans l'évaluation des classificateurs biclasses. Nous cherchons donc à évaluer les performances d'un ensemble de classificateurs biclasses sur la véritable réponse multiclassée qui sera obtenue. Pour cela, nous pouvons utiliser la technique de coévolution. Nous entendons par là que l'adéquation de chaque individu d'une population est sa performance multiclassée quand il est pris avec les autres classificateurs biclasses (sur toutes les autres paires de classes). Nous choisirons pour ces autres classificateurs les meilleurs de chaque population de la génération précédente. Pour la première génération, habituellement nous choisissons arbitrairement un individu par population comme le « meilleur » de sa population. Cet objectif nous amènera à l'implémentation de la coévolution que nous souhaitons, mais sur un problème un peu plus simple où il sera plus facile d'examiner le comportement de ce type d'algorithme. Pour ce problème spécifique, à la première génération nous pourrions évaluer facilement les performances de chaque classificateur indépendamment, mais ce ne sera pas transposable pour la construction de caractéristiques.

- Construction de caractéristiques : Nous arrivons enfin au véritable objectif. L'algorithme ressemblera à celui de « pairwise » avec coévolution, excepté que cette fois les programmes ne fourniront pas de décision de classe mais seulement un nombre réel. Pour l'évaluation d'un ensemble de nouvelles caractéristiques, nous utiliserons le taux d'erreurs obtenu en fournissant ces nouvelles données à un classificateur de type kNN. La performance du meilleur individu au terme de l'évolution sera ensuite réévaluée avec divers types de classificateurs dont des SVM pour de meilleures performances.

Les résultats obtenus lors de ces différentes étapes de programmation ne sont pas rapportés dans ce mémoire car ils ne correspondent pas au schéma que nous voulons étudier. Nous avons seulement vérifié grossièrement les performances pour corriger les erreurs dans le code le plus facilement et le plus tôt possible. Le travail qui aurait été nécessaire pour présenter les résultats pertinents avec ces méthodes intermédiaires n'était pas justifié compte tenu de notre objectif.

3.5 Les résultats à chercher

Pour ce qui est des résultats intermédiaires sur le schéma du « pairwise » par exemple, nous pouvons déjà nous attendre à des performances honorables, probablement comparables à d'autres classificateurs. Il est même possible d'avoir des résultats intéressants, mais c'est tout de même au prix d'une lourdeur assez importante tant dans la structure que dans le mécanisme pour l'obtenir. Nous ne détaillerons pas de toute façon les résultats de cette partie car cela sortirait du cadre de notre projet, l'objectif ici n'était que de vérifier que l'implémentation préliminaire de notre PG était correcte.

Avec l'algorithme de construction de caractéristiques, le but est d'obtenir de meilleures performances qu'avec un bon classificateur, grâce à un espace de représentation plus adéquat. Nous prendrons comme « bon classificateur » un SVM ou un MLP. Plus

précisément nous utiliserons les implémentations de Weka de deux SVM (*SMO* et *LibSVM*) ainsi que celle nommée *MultilayerPerceptron*. Les paramètres de ces classificateurs seront laissés aux valeurs par défaut de Weka excepté, parfois, pour la normalisation préliminaire des attributs. Puisque c'est lui qui a été utilisé au cours de l'évolution de la PG pour l'évaluation des jeux d'attributs, nous étudierons aussi les résultats finaux du kNN.

En fait, nous pourrions explorer deux voies pour les performances de la construction de caractéristiques : chercher à tout prix le plus bas taux d'erreurs quel que soit la complexité (au final, nous pourrions par exemple ajouter les nouvelles caractéristiques aux anciennes puis faire une sélection par AG), ou bien chercher un espace concis qui, de plus, prend appui sur un nombre raisonnable de caractéristiques simples et donne des performances intéressantes. Pour ce faire, nous allons comparer les taux de bonnes classifications de différentes approches :

- avec uniquement les données brutes;
- avec uniquement les caractéristiques produites par la PG;
- en prenant à la fois les données brutes et celles de la PG;
- en faisant une simple sélection de caractéristiques sur les données brutes afin de ramener leur nombre à celles données par la PG. Nous utiliserons pour cela un algorithme de sélection de type AG, et en évaluant la pertinence des jeux d'attributs à l'aide d'un classificateur C4.5 à base d'arbre. Dans Weka, cela correspond à (paramètres laissés par défaut) :
 - `weka.filters.supervised.attribute.AttributeSelection`
 - `weka.attributeSelection.ClassifierSubsetEval`
 - `weka.classifiers.trees.J48`
 - `weka.attributeSelection.GeneticSearch`

3.6 Séparation des bases de données

3.6.1 Entre la PG et la classification finale

Comme la PG demande de longs temps de calcul et que les bases de données sont suffisamment grandes, nous opterons pour une séparation généreuse pour la partie test au détriment de la partie apprentissage. Ainsi, les résultats de chacun des essais de la PG seront plus précis (espace de test plus grand) et donc probablement moins variables. Nous espérons donc avoir à faire moins de répétitions de l'apprentissage afin de confirmer le caractère significatif d'un gain ou d'une détérioration des performances. Nous découperons donc nos bases en deux parties égales, une pour l'apprentissage de la PG et une autre pour l'indication de performance du nouveau jeu de caractéristiques avec un classificateur. Si ce dernier nécessite lui aussi une phase d'apprentissage (cas du MLP et du SVM), celle-ci sera effectuée par validation croisée sur la base qui a servi à l'apprentissage de la PG afin de ne pas introduire de biais sur la base de test.

Par ailleurs, ce découpage en deux moitiés sera aléatoire mais stratifié, c'est-à-dire que la proportion de chacune des classes sera respectée dans ces divisions. Là encore, il s'agit d'avoir les résultats les plus stables possibles. Cette stratification est légitime dans le cas où la répartition des classes est connue *a priori* (et non issue de l'analyse de la base au complet), ce que nous considérerons être le cas pour les chiffres manuscrits ainsi que les céréales.

Dans la pratique nous utiliserons Weka pour faire ce découpage. La fonction se nomme « *weka.filters.supervised.instance.StratifiedRemoveFolds* », et il suffit de changer le paramètre « *seed* » pour avoir des découpages différents. Le paramètre « *numFolds* » sera fixé à deux afin d'avoir une séparation en deux moitiés égales.

3.6.2 Au cours de la PG

Nous avons dit que la PG travaillait sur une base d'apprentissage. Seulement, en interne, l'évaluation de la pertinence des jeux d'attributs nécessite de faire une classification par kNN. Donc, là encore, nous aurions besoin de découper à nouveau cette base. Nous avions dans un premier temps séparé selon une proportion 2/3-1/3, mais nous avons remarqué que le calcul de l'adéquation n'était pas assez précis pour une évolution correcte de la PG (mauvais pouvoir de généralisation). Nous avons donc affiné la mesure de l'adéquation par une validation croisée de type particulier : « leave one out » [36]. C'est-à-dire que le kNN tentera de classer chacun des éléments de la base à l'aide de tous les autres. Ce genre de validation croisée est en général peu utilisé par la plupart des classificateurs car il nécessite de faire autant d'apprentissage qu'il y a d'éléments dans la base. Mais comme le kNN n'a pas de phase d'apprentissage, ce souci ne sera pas gênant pour nous.

CHAPITRE 4

RÉSULTATS EXPÉRIMENTAUX

4.1 Analyse en quartiles

4.1.1 Définitions

Dans ce chapitre, nous allons présenter des résultats expérimentaux, principalement sous la forme de taux d'erreurs de système de classification. Bien entendu, au vu du nombre de mesures de performances effectuées, les tableaux reprenant ces valeurs sont difficilement lisibles (*voir* ANNEXE 1 et ANNEXE 2). C'est pourquoi nous présenterons dans ce chapitre les résultats d'une manière plus synthétique, sous forme de « boîtes à moustaches ». Les tableaux exhaustifs des valeurs associées seront quant à eux consultables en ANNEXE 3 (base des chiffres manuscrits) et ANNEXE 4 (base des céréales). Enfin, le CHAPITRE 5 sera consacré à des analyses statistiques plus avancées.

Avant de présenter la « boîte à moustaches », il est nécessaire de définir quelques termes statistiques :

- **La médiane** : c'est la valeur qui sépare en deux parties égales les valeurs mesurées. La première moitié des mesures est donc sous la médiane, l'autre moitié au dessus. S'il existe plusieurs valeurs qui séparent les mesures en deux (cas d'un nombre de mesures pair), c'est le point milieu de ces valeurs qui sera choisi comme valeur médiane, évitant ainsi toute ambiguïté de la définition. En général, nous pouvons dire que la médiane est la valeur typique des mesures, car comparativement à la moyenne la médiane est moins sensible aux valeurs extrêmes.
- **Les quartiles** : ce sont les valeurs qui séparent les mesures en quatre parties égales. Nous définissons alors les quartiles suivant : le premier, le deuxième

et le troisième. Le premier est celui qui sépare les 25 % des mesures les plus basses des autres, le deuxième est la médiane, le troisième est celui qui sépare les 75 % les plus basses des autres. Il découle de ces définitions que la moitié des mesures se situe entre le premier et le troisième quartile. Cela donne donc un bon aperçu des valeurs auxquelles nous pouvons nous attendre.

- **Écart interquartile** : c'est tout simplement la différence entre le troisième et le premier quartile. Si les mesures sont plutôt dispersées, cet écart sera donc grand. Au contraire si elles sont regroupées il sera petit.
- **Les déciles** : à la manière des quartiles, les déciles sont les valeurs qui séparent les mesures en dix groupes. Nous nous intéresserons surtout au premier décile qui sépare les 10 % des mesures les plus basses et au neuvième décile pour les 10 % les plus hautes. Entre ces deux valeurs se trouvent donc 80 % des mesures.
- **Les valeurs extrêmes** : il s'agit de la plus petite mesure pour le minimum et de la plus grande pour le maximum.

4.1.2 Les simulations effectuées

Nous avons besoin pour estimer la performance d'un système de classification de faire des simulations, c'est-à-dire l'appliquer sur une base réelle et limitée. Cependant, des simulations d'un même système ne donnent pas toujours exactement la même mesure de performance. En effet, elles dépendent de tirages aléatoires dans leurs algorithmes et surtout de la base utilisée avec son découpage en apprentissage et test. Pour avoir une meilleure idée de la performance réelle d'un système, il est donc important de faire plusieurs simulations. Afin de s'affranchir le plus possible de l'influence du découpage de la base, il sera différent pour chaque simulation d'un même système. Par contre, pour pouvoir comparer efficacement les différents systèmes, le même ensemble de découpages sera réutilisé à chaque série de simulations. Par ailleurs nous avons choisi de

faire quarante simulations pour chaque système afin d'avoir un nombre suffisant de mesures.

4.1.3 La « Boîte à moustaches »

La « boîte à moustaches » est la traduction la plus courante au Québec de « *box and whiskers plot* », mais nous trouvons également l'appellation « boîte à pattes », cette dernière étant davantage utilisée en France [37]. Il s'agit d'un diagramme introduit en 1977 par J.W. Tukey [38] qui présente graphiquement des valeurs statistiques simples tels les quartiles. De nombreuses variantes ont été utilisées depuis (plus de soixante d'après [37]), différant principalement par la manière de traiter les valeurs extrêmes. Plutôt que de reprendre exactement la description de Tukey, nous allons utiliser la plus courante aujourd'hui dans les explorateurs de données et les logiciels de statistiques [37], laquelle est représentée à la *Figure 13*.

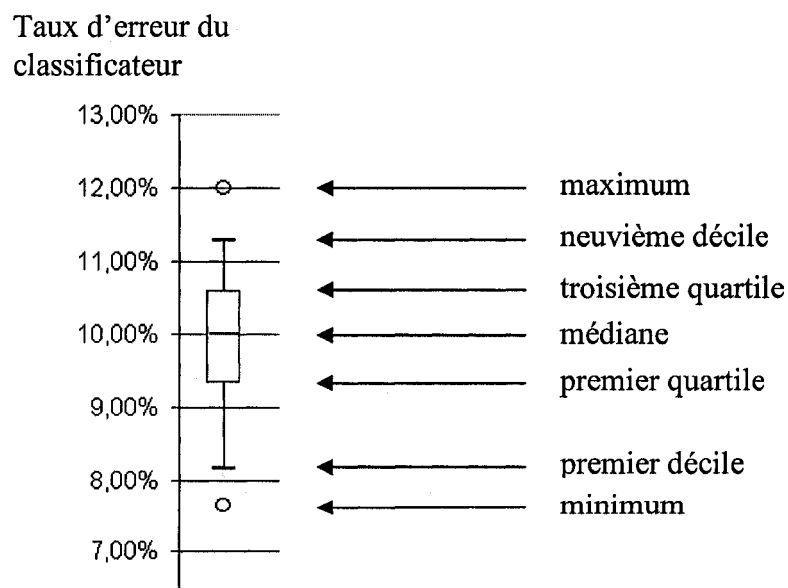


Figure 13 Exemple de « boîte à moustaches »

Nous pouvons constater que ce diagramme permet de visualiser rapidement où se situe la moitié (donc pour ainsi dire la majorité) des mesures grâce au rectangle ainsi que la médiane qui coupe ce rectangle en deux. Le trait vertical permet quant à lui d'avoir facilement un aperçu de la dispersion d'un plus grand nombre de mesures car il en représente 80 %. Pour comparer les performances de plusieurs classificateurs, la première chose à faire est de confronter les médianes. Ensuite, il sera intéressant de voir si les rectangles des « boîtes à moustaches » se recouvrent (cas A de la *Figure 14*), ou du moins si la médiane est à l'extérieur du rectangle de l'autre (cas C). Ainsi cela signifiera que la valeur typique de l'un est supérieure au trois quarts des mesures de la performance de l'autre. Dans le cas B, la distinction des boîtes est alors évidente. Finalement, afin de faciliter davantage la lecture de ces graphiques, nous ne présenterons pas les valeurs extrêmes, ce qui nous épargnera la présence des petits cercles.

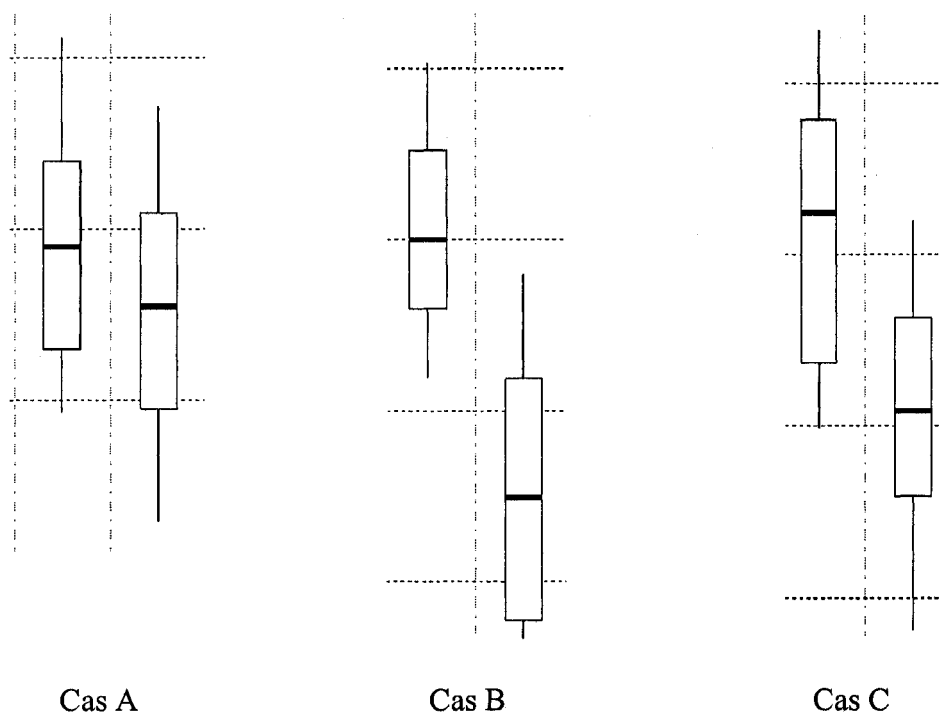


Figure 14 *Comparaison de « boîtes à moustaches »*

4.2 Résultats sans PG

4.2.1 Notations des différents classificateurs

Nous allons étudier l'impact qu'a l'espace de représentation des données sur différents classificateurs. Pour éviter toute confusion le *Tableau 1* recense ceux qui seront utilisés ainsi que les notations qui leurs seront attribuées par la suite.

Tableau 1
Notations des classificateurs

Notation	Description
kNN	Classification selon les plus proches voisins. Le classificateur détermine le meilleur k entre 1 et 10 à partir de son ensemble d'apprentissage (par validation croisée), puis utilise cette valeur pour classer l'ensemble de test. Le classificateur détermine quels sont les plus proches voisins à l'aide de la distance euclidienne.
kNN norm	Identique au kNN sauf le fait que les données sont préalablement normalisées, c'est-à-dire qu'il fait un prétraitement linéaire de sorte que la dynamique de chaque caractéristique soit [0, 1].
MLP norm	Réseau de neurones (de l'anglais « Multi-Layer Perceptron ») à rétro-propagation selon les paramètres par défaut de Weka. Il aura donc une couche cachée ayant une taille de la somme du nombre de classes avec la dimension de l'entrée. Nous n'utiliserons le MLP qu'avec normalisation préalable des données car nous nous sommes vite aperçus que c'était une condition nécessaire pour que son apprentissage se déroule correctement.

Tableau 1
Notations des classificateurs (suite)

Notation	Description
SMO	Il s'agit d'une implémentation des machines à vecteurs de support (SVM) proposée par Weka. Comme pour les autres classificateurs, nous conserverons les paramètres par défaut de Weka.
SMO norm	Identique au SMO mais en normalisant les données avant de les traiter.
LibSVM	LibSVM est une implémentation alternative des SVM proposée par Yasser El-Manzalawy [34].
LibSVM norm	LibSVM, mais en normalisant d'abord.

4.2.2 Sur la base des céréales

Les « boîtes à moustaches » des classificateurs LibSVM et SMO ne figurent pas sur la *Figure 15* car elles sont hors échelle : ils présentent des valeurs médianes de plus de 65 % d'erreurs. Nous avons préféré garder une échelle qui permette d'apprécier les différences des autres classificateurs plutôt que d'écraser les boîtes sur la gauche. Nous pouvons toutefois en retenir que LibSVM et SMO ne peuvent faire correctement leurs apprentissages que si les données sont de même ordre de grandeur.

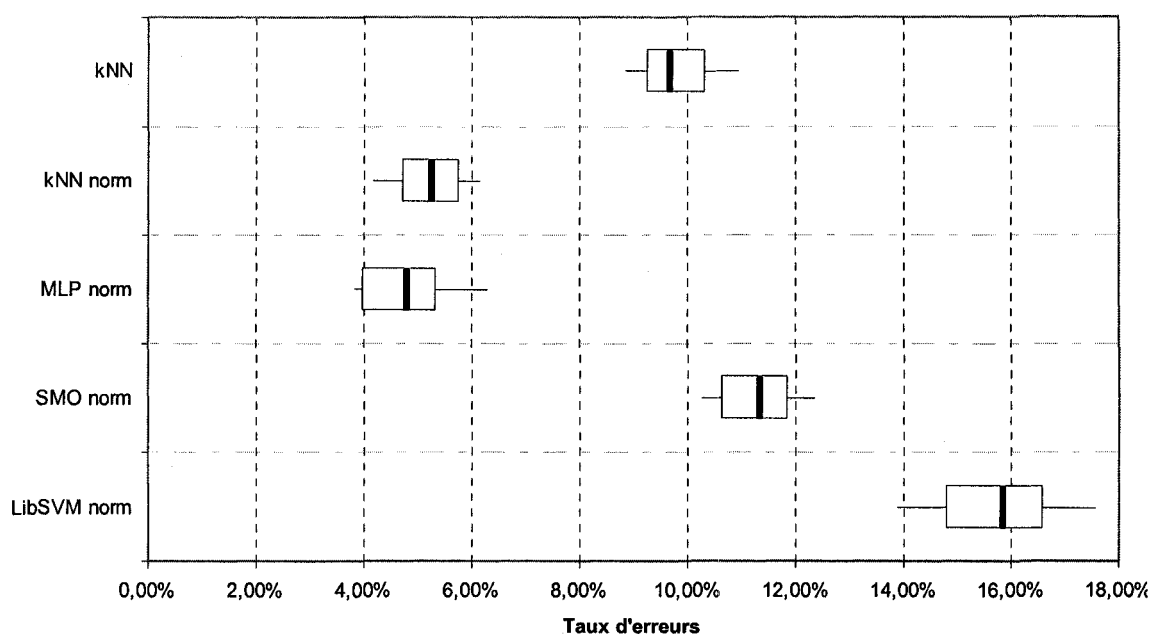


Figure 15 *Taux d'erreurs des classificateurs sur les caractéristiques originales de la base de céréales*

Ensuite, la principale observation à faire est que le taux d'erreurs obtenu est très différent d'un classificateur à l'autre. Même en exceptant LibSVM et SMO, les médianes vont de 4,8 % pour le MLP à 15,8 % pour le LibSVM normalisé. Enfin, nous pouvons retenir que sur les caractéristiques originales de la base des céréales, il semble que le meilleur classificateur soit le MLP, suivi de proche par le kNN, à condition que les données soient préalablement normalisées. LibSVM, même normalisé ne semble pas avoir une bonne performance, tandis que l'autre SVM, le SMO s'en tire mieux s'il est normalisé, sans pour autant exceller.

4.2.3 Sur la base des chiffres manuscrits

Sur la base des chiffres manuscrits (*Figure 16*), nous observons encore des différences de performances d'un classificateur à l'autre, bien que moins marquées. Par exemple, le SMO produit 1 % de plus d'erreurs que le kNN (4,25 % au lieu de 3,2 % en médiane). Cependant, contrairement à la base des céréales, l'influence de la normalisation des données n'est pas largement déterminante. Cela s'explique par le fait que les données représentent des nombres de pixels allumés dans des zones de surfaces égales, donc les ordres de grandeurs des caractéristiques sont semblables. Nous notons toutefois que LibSVM a tout de même bien tiré partie de la normalisation.

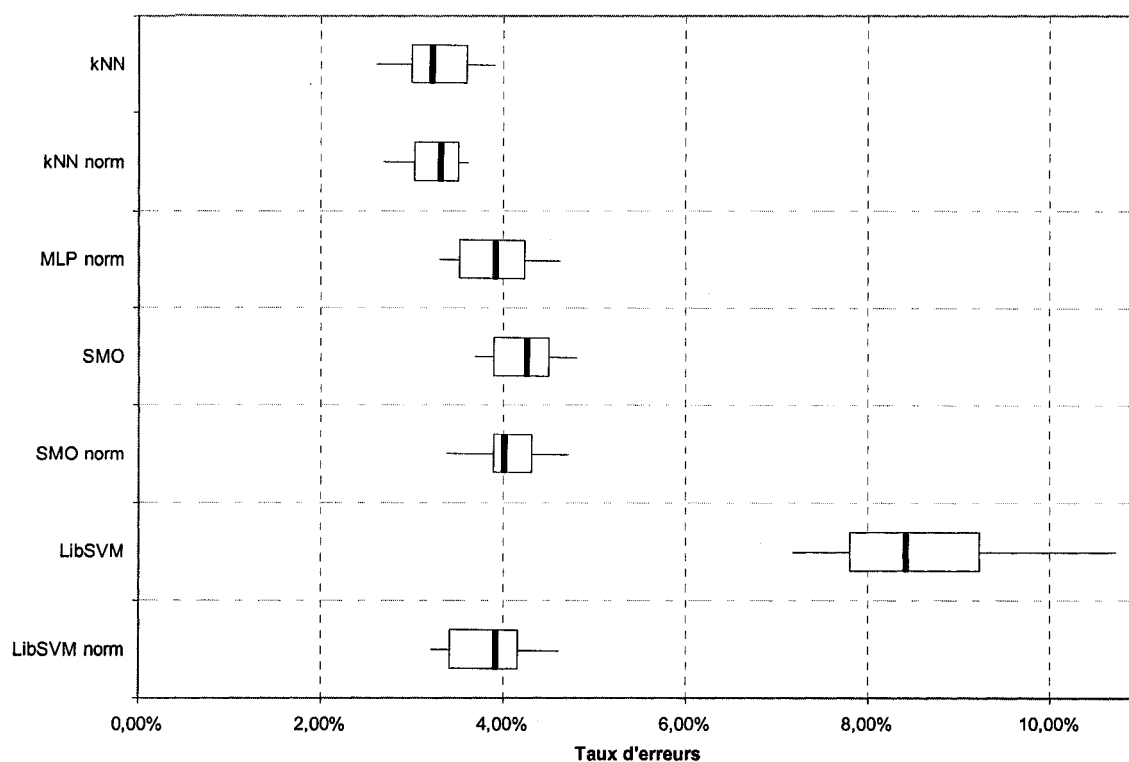


Figure 16 *Taux d'erreurs des classificateurs sur les caractéristiques originales de la base des chiffres manuscrits*

Nous pouvons distinguer principalement trois groupes dans les résultats :

- kNN en version normalisée ou pas obtiennent les meilleures performances avec respectivement des médianes à 3,3 % et 3,2 %.
- le MLP norm et les SVM (SMO, SMO norm, LibSVM norm) excepté le LibSVM ont des résultats légèrement moins bons mais tout de même honorables avec des médianes entre 3,9 % et 4,3 % et des quartiles qui se recoupent largement. Ces quartiles se distinguent tout de même de ceux de kNN et kNN norm, ce qui fait penser que l'algorithme du plus proche voisin semble significativement plus performant sur cette base.
- LibSVM accuse quant à lui un large retard avec une médiane de 8,4 %. De plus, son écart interquartile est assez grand. Il semble donc avoir des problèmes d'apprentissage ou de généralisation.

Les deux premiers groupes ont des écarts interquartiles relativement modestes, ce qui signifie qu'ils obtenaient des résultats assez stables. Cela devrait aider à faire les comparaisons qui s'imposeront avec les résultats après PG.

4.3 Évolution type de la PG

4.3.1 Vision d'ensemble

Il s'agit de visualiser l'évolution des populations par la PG. Tout d'abord, il faut bien garder à l'esprit qu'il s'agit d'une coévolution, il y a donc plusieurs espèces à observer. Voyons plusieurs indices sur lesquels il sera intéressant de se pencher :

- L'adéquation maximale : il s'agit de tracer pour chaque génération l'adéquation maximale calculée. Rappelons que l'adéquation correspond au taux de bonnes classifications d'un kNN sur la base d'apprentissage de la PG. Grâce à l'élitisme, l'adéquation maximale sera toujours croissante selon les espèces. De plus, par la nature du calcul de l'adéquation (pour évaluer une

espèce, nous nous servons du meilleur individu de chacune des autres espèces), une espèce aura toujours une adéquation maximale supérieure à l'espèce précédemment évaluée. Ainsi, pour plus de lisibilité, il sera possible de ne tracer qu'une seule courbe regroupant toutes les espèces plutôt qu'une courbe par espèce.

- L'adéquation moyenne de chaque population : dans ce cas, il est nécessaire de tracer autant de courbe qu'il y a d'espèces. Ce qui est intéressant au sujet de la moyenne des adéquations, c'est de la comparer à l'adéquation maximale à la même génération. Si cette différence est petite, cela peut signifier deux choses : que la population est peu diversifiée ou bien qu'aucun de ses individus n'arrive à apporter d'informations pertinentes de classification qui soient nouvelles par rapport aux autres espèces. Comme la PG n'a pas autant le défaut des AG d'avoir des populations homogènes, nous pouvons dire assez paradoxalement que les espèces qui ont une adéquation moyenne plus élevée sont moins bonnes pour la classification. Il est bien important de comprendre ce point et de bien prendre conscience que ça ne remet pas en cause que les meilleurs individus au sein d'une espèce sont ceux qui ont l'adéquation la plus haute. Pour évaluer ce qu'apporte une espèce par rapport aux autres, il aurait également été possible de mesurer la performance du kNN sans l'espèce dont il est question. En faisant la différence avec l'adéquation maximale de cette espèce, nous aurions alors une bonne mesure de son apport au tout. Cependant cette approche aurait nécessité une charge de calcul relativement importante. La remarque faite sur la différence entre l'adéquation maximale et moyenne, bien que certainement moins précise, nous permet de ne pas ralentir les simulations.
- La taille moyenne des élites : il s'agit du nombre de nœuds du meilleur individu de chaque espèce. Cette mesure permet de visualiser deux choses : l'évolution de la complexité de la meilleure solution du moment ainsi que de repérer quand l'élite est remplacée dans une espèce.

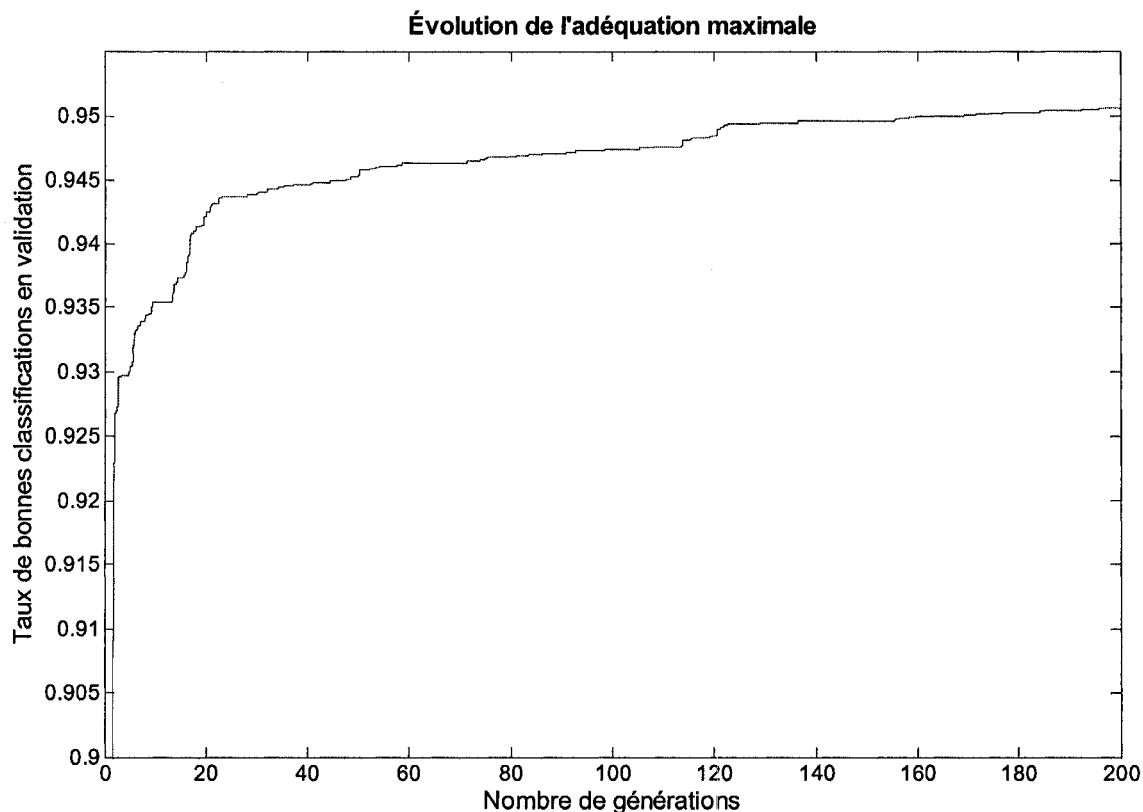


Figure 17 *Évolution de l'adéquation maximale sur la base des chiffres manuscrits*

Première chose rassurante, la courbe de l'adéquation maximale de la *Figure 17* augmente effectivement bien. Il y a donc un apprentissage qui se fait. Ensuite nous remarquons que l'adéquation croît rapidement lors des toutes premières générations (moins de trente), puis monte plus lentement. Nous remarquons aussi quelque chose qui est caractéristique d'un algorithme évolutionnaire : il y a de nombreux petits paliers suivis de décrochages marqués. Même loin dans l'évolution (ici à la génération 120), nous observons encore ce genre de saut dans l'évolution. Nous avons décidé d'arrêter l'évolution après 200 générations, principalement pour des contraintes techniques car le temps de calcul nécessaire est déjà important (environ 5 jours sur un Athlon 2500+ sur la

base des chiffres manuscrits). Aussi, l'évolution semblait être toujours assez lente à ce stade.

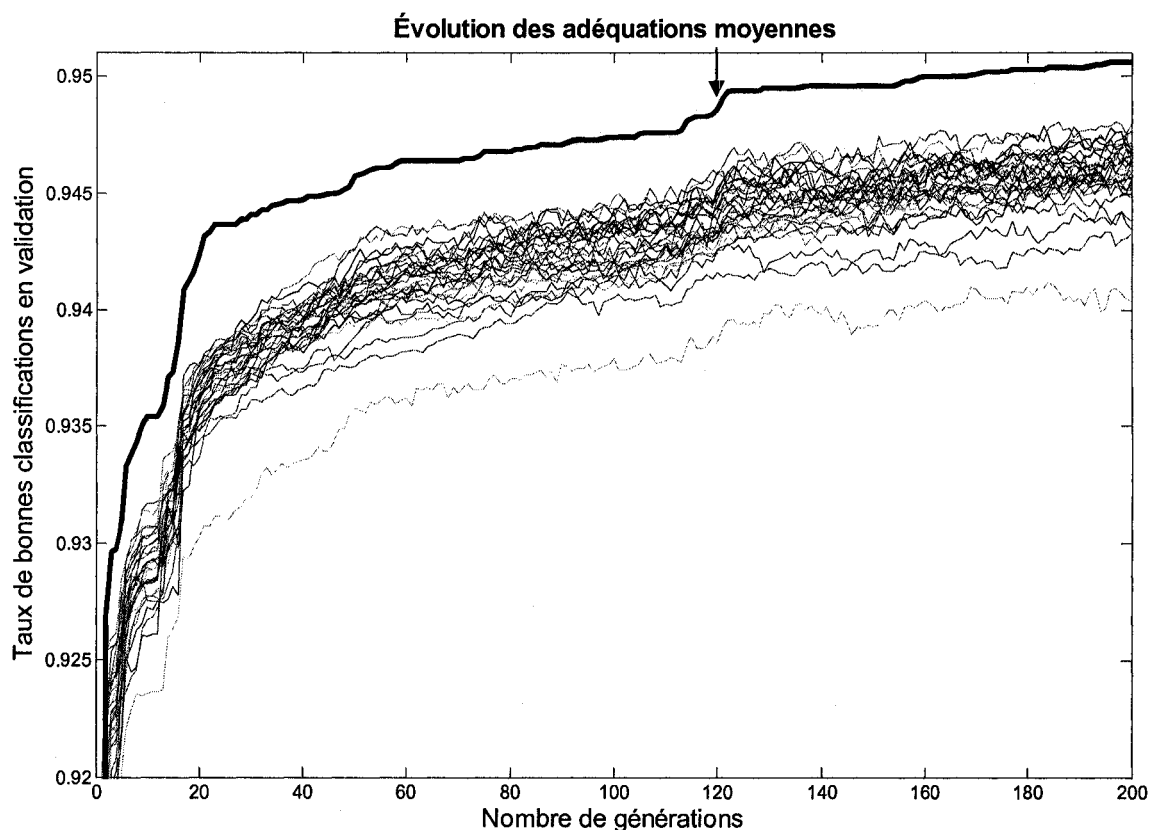


Figure 18 *Évolution des adéquations moyennes sur la base des chiffres manuscrits.*
L'adéquation maximale est représentée en gras

Étudions maintenant les courbes de l'évolution de l'adéquation moyenne de chacune des espèces. Sur la *Figure 18*, nous avons également tracé d'un trait plus épais et noir l'adéquation maximale à des fins de comparaisons. Comme toutes les courbes sont largement sous l'adéquation maximale, nous en déduisons que chacune des espèces a réussi à trouver par son élite des informations pertinentes pour la classification qui n'était pas déjà présentes dans l'élite des autres espèces. Cependant cette différence va

facilement du simple au triple, certaines espèces ont donc trouvé des niches plus importantes que les autres. Sur l'exemple de la *Figure 18*, l'élite d'une espèce arrive à elle seule à faire passer le taux d'erreurs de 6 % à 5 % par rapport à la moyenne des autres individus de son espèce. Pour comparaison, les élites des autres espèces de cet exemple parviennent en moyenne à un gain de 0,5 %.

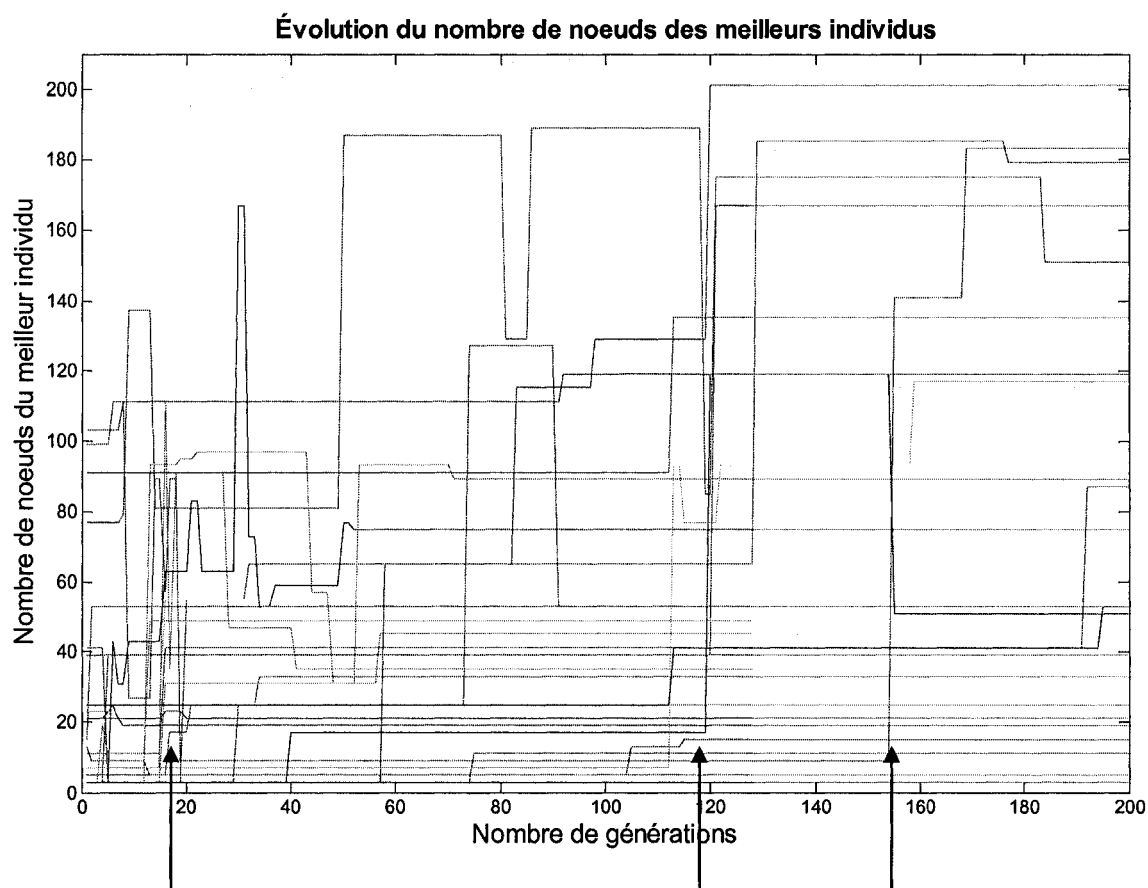


Figure 19 *Évolution de la taille des élites sur la base des chiffres manuscrits*

La *Figure 19* montre l'évolution du nombre de nœuds des élites de chacune des espèces. Sans grande surprise, au cours des premières générations, les élites ont des tailles plutôt modestes et changent souvent. Cette taille réduite peut s'expliquer par le fait que tant que l'évolution n'a pas une certaine maturité, les calculs les plus simples ont plus de

chances de refléter les données, valeurs certes conservatrices mais sûres pour la classification. Ensuite, nous constatons qu'après quelques dizaines de générations les courbes ont de longs paliers, reflet que les élites ne changent pas très souvent (il est peu probable que deux élites consécutives aient exactement le même nombre de nœuds). Cette observation est d'ailleurs conforme avec la présence de paliers sur la courbe de l'évolution de l'adéquation maximale (*Figure 17*). Il ne faut pas y voir une défaillance de la recherche de la PG, c'est une observation habituelle pour les algorithmes évolutionnaires [24]. En y regardant de plus près, nous pouvons remarquer que le changement de l'élite d'une espèce (saut sur une courbe) est souvent suivi d'autres changements d'élite dans les autres espèces. Cela est particulièrement visible à la génération 120, par ailleurs nous constatons que cela a entraîné une forte augmentation de l'adéquation totale. Ce phénomène s'explique par l'interaction indirecte entre les espèces : si une espèce change d'élite, cela induit une modification du calcul de l'adéquation des autres espèces. À un niveau supérieur, le changement d'une caractéristique du jeu modifie l'intérêt des autres pour la classification, d'où la possibilité d'un profond remaniement.

Au final, l'étude de ces trois figures (*Figure 17*, *Figure 18* et *Figure 19*) révèle que l'évolution de la PG semble bien se comporter, tant par rapport aux comportements généraux des algorithmes évolutionnaires (nombreux changements au début, présence de paliers ensuite, complexité des individus croissante) que par rapport à ce que nous pouvions attendre de la recherche d'un nouvel espace de représentation (interaction indirecte entre les caractéristiques).

4.3.2 Influence du nombre d'espèces

Afin de mettre en place la PG, il faut fixer ses paramètres tels le nombre d'espèces (donc la dimension du nouvel espace de représentation), la taille des populations, le nombre de générations ainsi que les paramètres liés à l'évolution elle-même (taux de croisement, de

mutation). Notre objectif est de prendre des valeurs raisonnables pour avoir une évolution respectable et non de les ajuster finement au mieux de ce qu'il serait possible. En effet, nous voulons avant tout un système qui demandera peu d'interventions humaines, nous essayerons donc de déterminer des paramètres génériques qui pourraient s'appliquer à toutes bases de données. Ainsi, nous déterminerons nos paramètres sur la base des chiffres manuscrits uniquement, puis nous prendrons autant que possible les mêmes pour la base des céréales.

Dans un premier temps, intéressons-nous au nombre d'espèces. Comme il s'agit seulement de changer la représentation des données, nous aurions pu conserver la même dimensionnalité, c'est-à-dire 100 dans le cas des chiffres manuscrits. Cependant, cela impliquerait une masse de calcul trop importante par rapport au matériel informatique dont nous disposions. Nous avons donc cherché à réduire ce nombre, et pour cela nous avons observé l'évolution en fonction du nombre d'espèce (*Figure 20*).

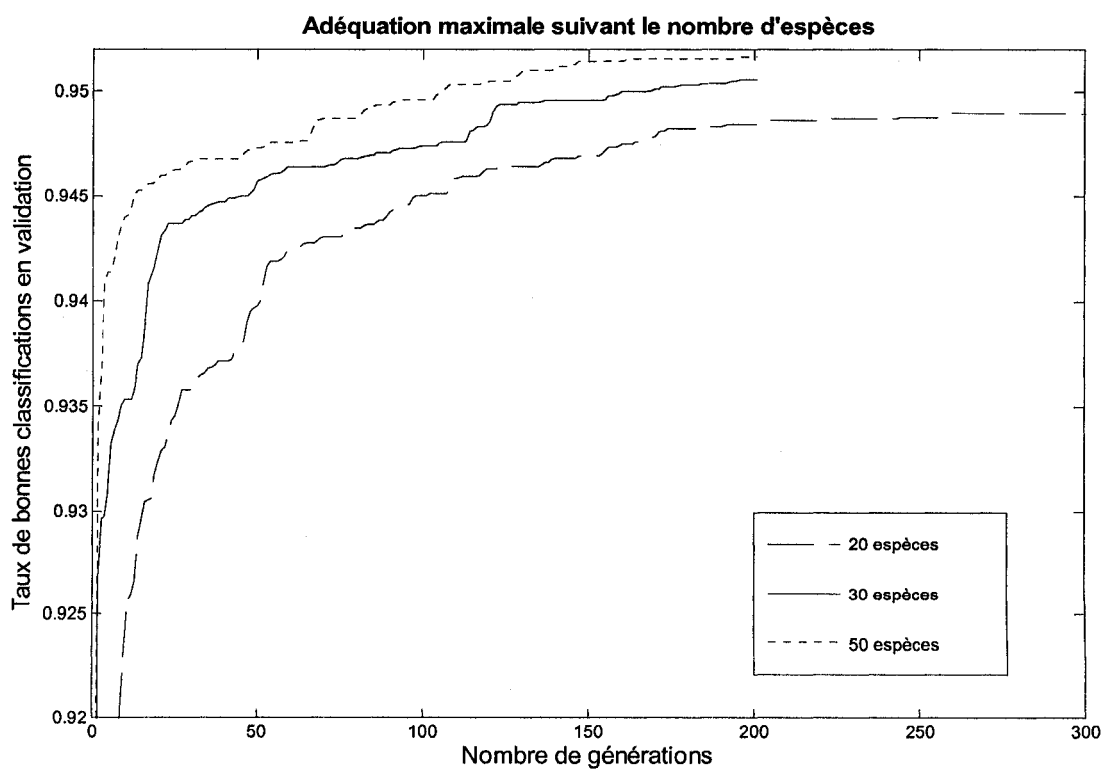


Figure 20 *Évolution de l'adéquation maximale suivant le nombre d'espèces*

Assez naturellement, à nombre égal de générations, plus d'espèces implique des taux de bonnes classifications supérieurs. Par contre, comme la masse de calcul nécessaire à l'évolution augmente quasi-linéairement selon le nombre d'espèce, il serait sans doute plus intéressant de considérer les taux de bonnes classifications à effort de calcul équivalent. Pour cela ramenons le nombre réel de générations à son équivalent en termes de temps de calcul pour 30 espèces.

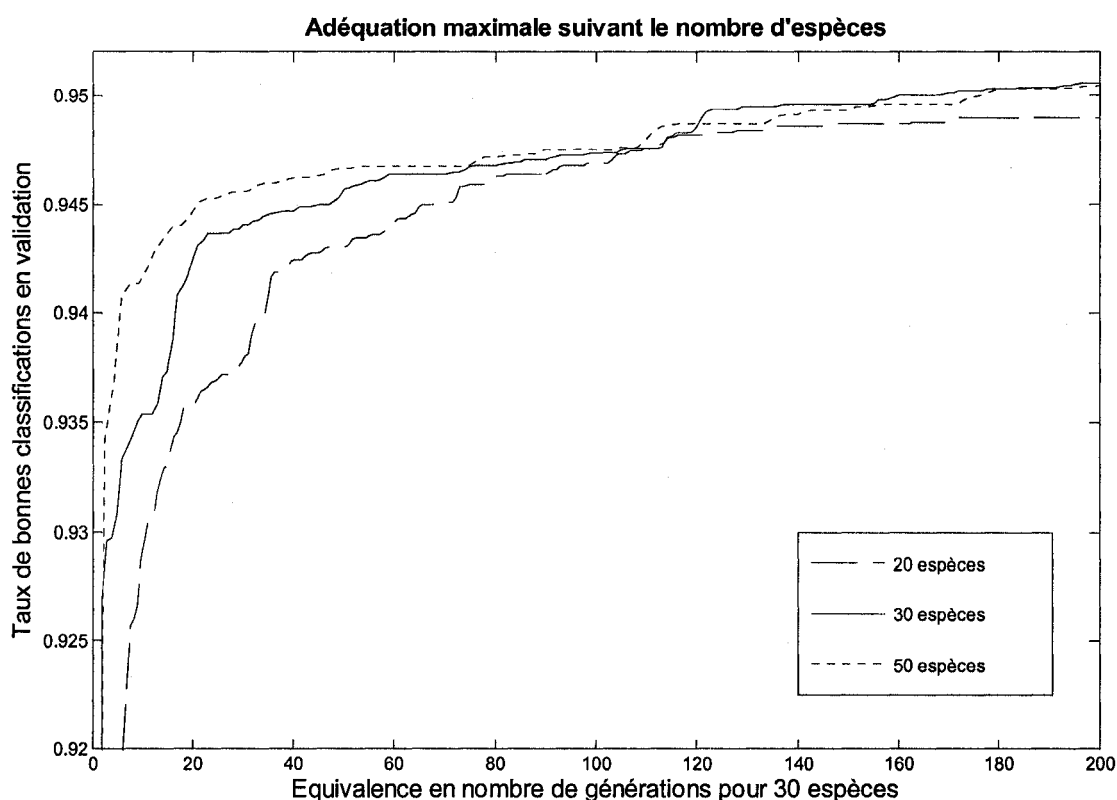


Figure 21 *Évolution de l'adéquation maximale suivant le nombre d'espèces à effort de calcul équivalent*

Au début de l'évolution, un nombre plus élevé d'espèces donne de meilleures classifications. Cependant, assez vite (à la génération 110 dans cet exemple) les courbes semblent se croiser. À partir de là, l'influence du nombre d'espèce semble nettement

moins importante. Enfin, au bout de l'effort de calcul jugé raisonnable compte tenu de notre matériel, le choix de 30 espèces semble le plus judicieux, tout en remarquant que la différence est relativement faible. Nous pouvons avancer une hypothèse pour expliquer l'allure de ces courbes :

- Aux premières générations, l'évolution n'a pas encore pu faire son œuvre. Statistiquement, avec un plus grand nombre d'espèces et donc en retenant davantage de caractéristiques, nous avons plus de chance d'avoir conservé d'informations pertinentes pour la classification. Toutefois, il y a probablement plus de redondances et d'informations inutiles.
- Ensuite, l'écart entre les courbes se resserre. Nous pouvons penser que c'est la lourdeur du grand nombre d'espèces qui ralentit l'évolution à effort de calcul équivalent. La croissance avec moins d'espèces est donc plus rapide.
- Enfin, le risque d'un nombre d'espèces trop peu élevé peut provoquer une stagnation du taux de bonnes classifications. En effet, il devient difficile d'ajouter des informations pertinentes pour la classification sans en retirer d'autres si toutes les espèces ont déjà réussi à se spécialiser sur des éléments intéressants.
- Nous pouvons imaginer qu'à terme ce soit les mondes avec les plus grands nombres d'espèces qui reprendraient le dessus, car ce sont elles qui laissent le plus la place à une solution étoffée.

Bien entendu, l'expérience que nous avons menée, si elle va dans le sens de cette hypothèse, ne suffit pas pour la valider de manière satisfaisante. Pour cela, il faudrait une investigation plus poussée, laquelle déborderait nettement de notre projet étant donné que nous avons précisé que nous ne voulions faire qu'un ajustement grossier des paramètres d'évolution.

4.3.3 Influence de la taille de la population

Un autre paramètre important des algorithmes évolutionnaires est la taille de chacune des populations. De manière générale, une population plus grande demande un effort de calcul plus important à génération égale (complexité linéaire dans notre cas). À l'inverse, une population trop petite risque de produire une population trop homogène, c'est-à-dire ayant un manque de diversité critique. Dans ce cas, l'évolution risque d'avoir du mal à innover et de se contenter de peaufiner la première solution satisfaisante qu'elle rencontre (maximum local). Il ne faut donc pas se laisser séduire par une évolution qui semble rapide en termes de générations mais qui comporterait de grosses populations. Nous préférons rechercher un compromis qui, pour un effort de calcul auquel nous puissions nous permettre, donnera les meilleures solutions à notre problème de classification [39]. Pour cela, à la manière de l'étude sur le nombre d'espèces, nous allons observer l'évolution non pas à nombre de génération égal mais à effort de calcul égal. Comme la complexité est linéaire, la conversion en équivalent de génération de populations de taille moyenne (500 individus par exemple) va de soi.

La *Figure 22* illustre qu'avec des populations de plus petite taille, l'évolution précoce est plus rapide. Par la suite, il est plus hasardeux de tirer des conclusions de cette expérience. De toute évidence, à effort de calcul équivalent, les performances sont très similaires au moment où nous arrêtons l'expérience. Il semble qu'une plus grande taille de populations permet une progression par génération plus importante, ce qui par ailleurs se conçoit très bien. Par contre, comme l'évolution des différentes espèces se fait à tour de rôle à chaque génération, de plus grandes populations impliquent que les espèces « se passent la main » moins souvent en termes de temps de calcul. Cela induit qu'une espèce réagit de manière plus différée à l'évolution des autres, ce qui pourrait ralentir l'évolution globale au point de compenser l'apport de la diversité accrue. Au bilan, il semble que des tailles des populations de 200, 500 ou 1000 n'influent que très peu sur la

performance globale à effort de calcul égal. De ce fait, nous utiliserons une taille de 500, par ailleurs souvent utilisée dans la littérature [39].

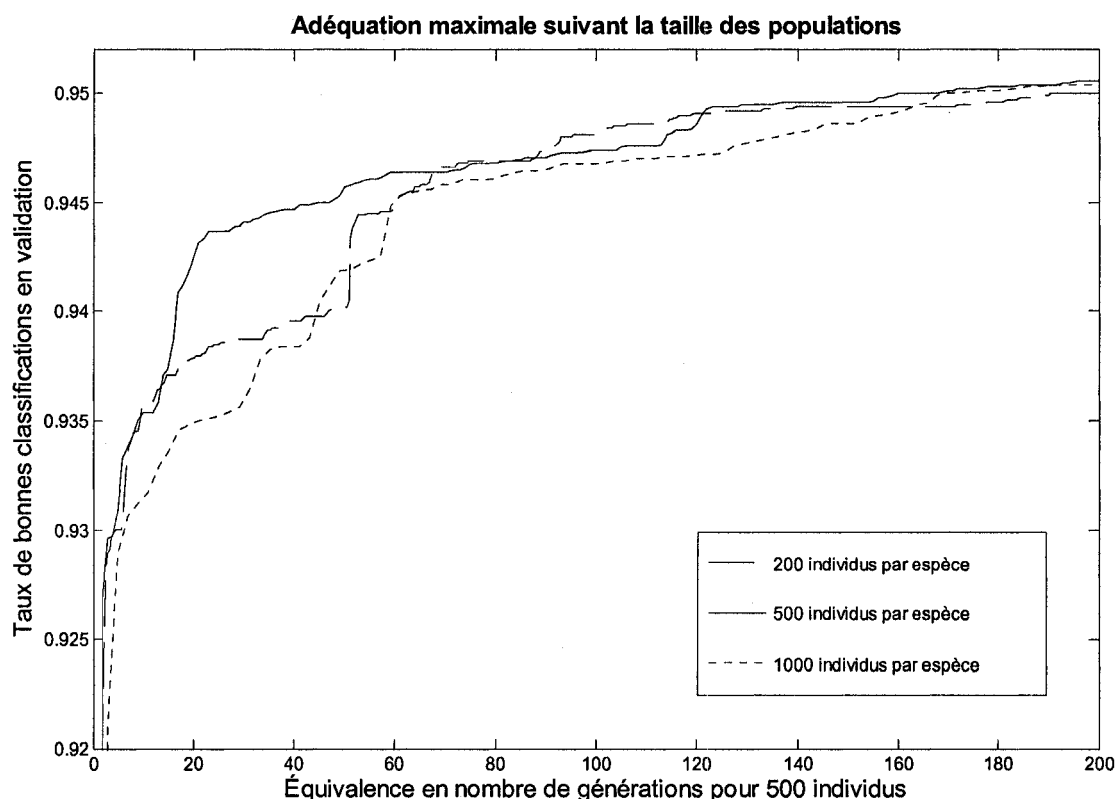


Figure 22 *Évolution de l'adéquation maximale suivant la taille des populations à effort de calcul équivalent*

4.3.4 Autres paramètres de la PG

Il reste encore à définir les autres paramètres de la PG tels que le taux de mutation, le taux de croisement, la taille maximale des arbres, et le taux de création d'individus complètement nouveaux. Afin d'être suffisamment générique, nous utiliserons les valeurs les plus souvent utilisées dans la littérature pour ce type d'algorithme [40]. En l'occurrence, voici nos choix :

- taux de mutation : 10 %
- taux de croisement : 70 %
- taux de création d'individus totalement nouveaux : 15 %
- nombre de nœuds maximal des arbres : 200.

Du fait que ces valeurs aient été choisies selon la littérature et non d'après le comportement sur les bases étudiées permet également de s'affranchir d'un biais optimiste en cas de nouvelles bases.

4.4 Performances des nouveaux jeux de caractéristiques

4.4.1 Sur les chiffres manuscrits

Observons maintenant les performances des classificateurs après remaniement de l'espace de représentation par PG (30 espèces, 500 individus par espèce, 200 générations). La dimension du nouvel espace de représentation est donc 30. La *Figure 23* montre que les différents classificateurs ont tous eu des performances beaucoup plus rapprochées avec ce nouvel espace qu'avec les données originales. En effet, les médianes sont ici comprises entre 4,5 % (pour LibSVM) et 7,0 % (SMO) contre 3,2 % à 8,4 % pour les données originales. Nous remarquons également que les meilleurs classificateurs ne sont plus les mêmes : après la PG c'est LibSVM qui obtient les meilleurs résultats avec une médiane à 4,5 % de bonnes classifications. De plus les performances de celui-ci semblent assez stables avec un écart interquartile de 0,65 %.

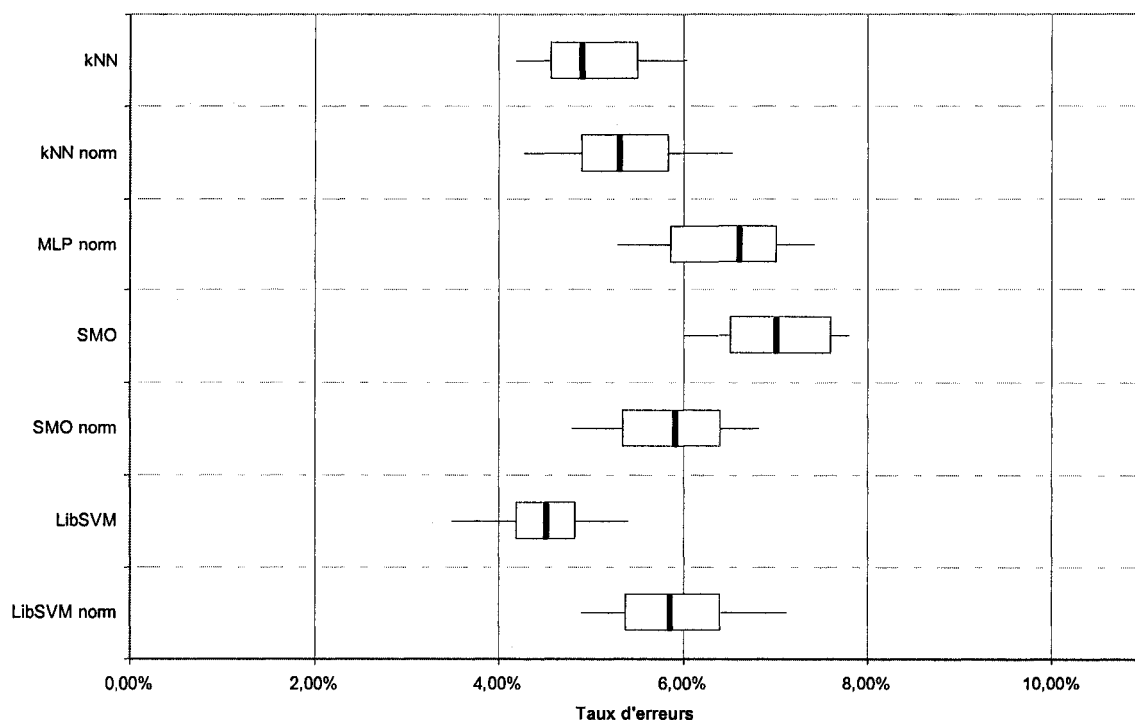


Figure 23 *Taux d'erreurs des classificateurs sur les caractéristiques obtenues par PG sur la base des chiffres manuscrits*

Après ces observations générales, nous allons nous intéresser à la comparaison directe de ces résultats avec ceux obtenus sans PG. Toutefois, comme il a été remarqué que les données originales nécessitaient une normalisation préalable pour un apprentissage correct, nous ferons pour chaque classificateur les deux comparaisons suivantes :

- version normalisée après PG versus normalisée sans PG;
- version non normalisée après PG versus normalisée sans PG.

Rappelons que pour chacun des découpages de la base nous avons mesuré la performance des classificateurs avant et après PG. Il est donc légitime de faire la différence entre ces deux résultats afin de déterminer le gain. Nous cherchons donc à évaluer le gain apporté par la PG à découpage apprentissage/test identique mais non fixé. Ces résultats sont reportés sous forme de quartiles dans la *Figure 24*.

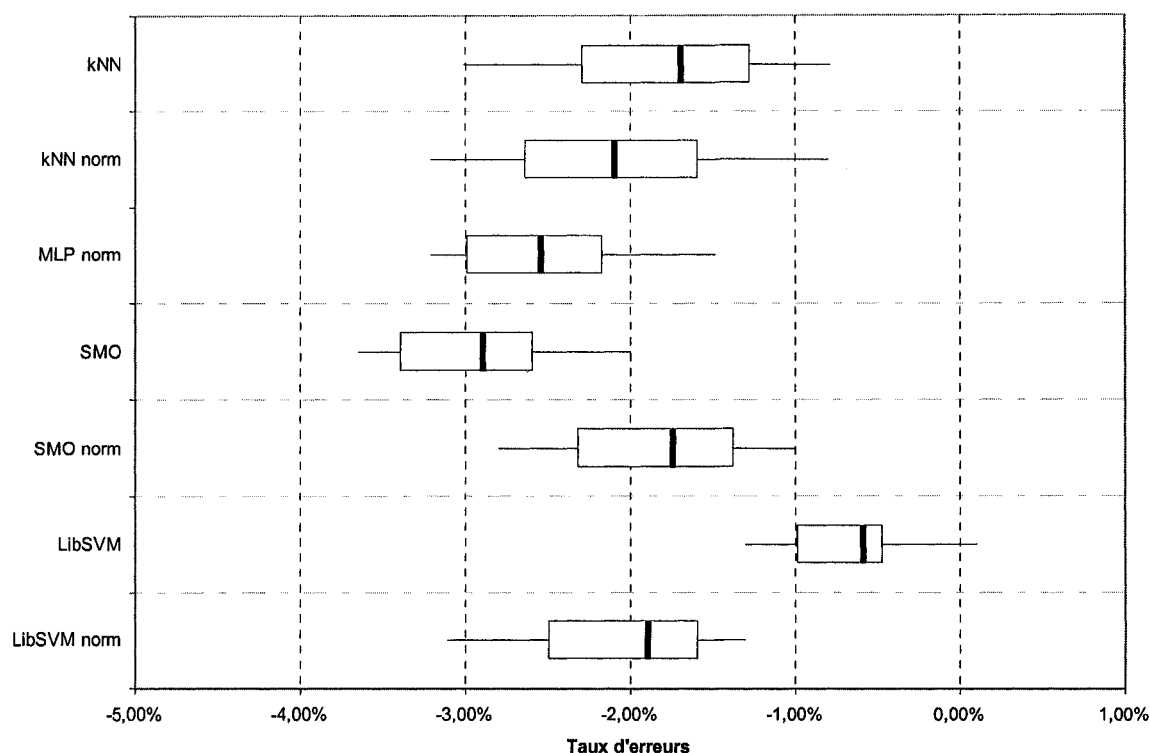


Figure 24 *Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base des chiffres manuscrits et ces mêmes classificateurs sur les données brutes normalisées*

Première constatation, tous les gains médians sont sensiblement négatifs. Les 30 caractéristiques générées par la PG ont donc été moins discriminantes pour les classificateurs que les 100 originales. Pour la plupart des classificateurs, la perte médiane se situe autour de 2 %. Dans le cas de LibSVM, la perte est toutefois assez limitée avec une médiane de 0,6 % alors que nous pourrions espérer un gain de vitesse d'exécution grâce à la réduction de la dimensionnalité. Ce gain de vitesse est toutefois hypothétique car la construction des nouvelles caractéristiques nécessitent également des calculs (bien que très simples grâce à la grammaire choisie pour les arbres de la PG).

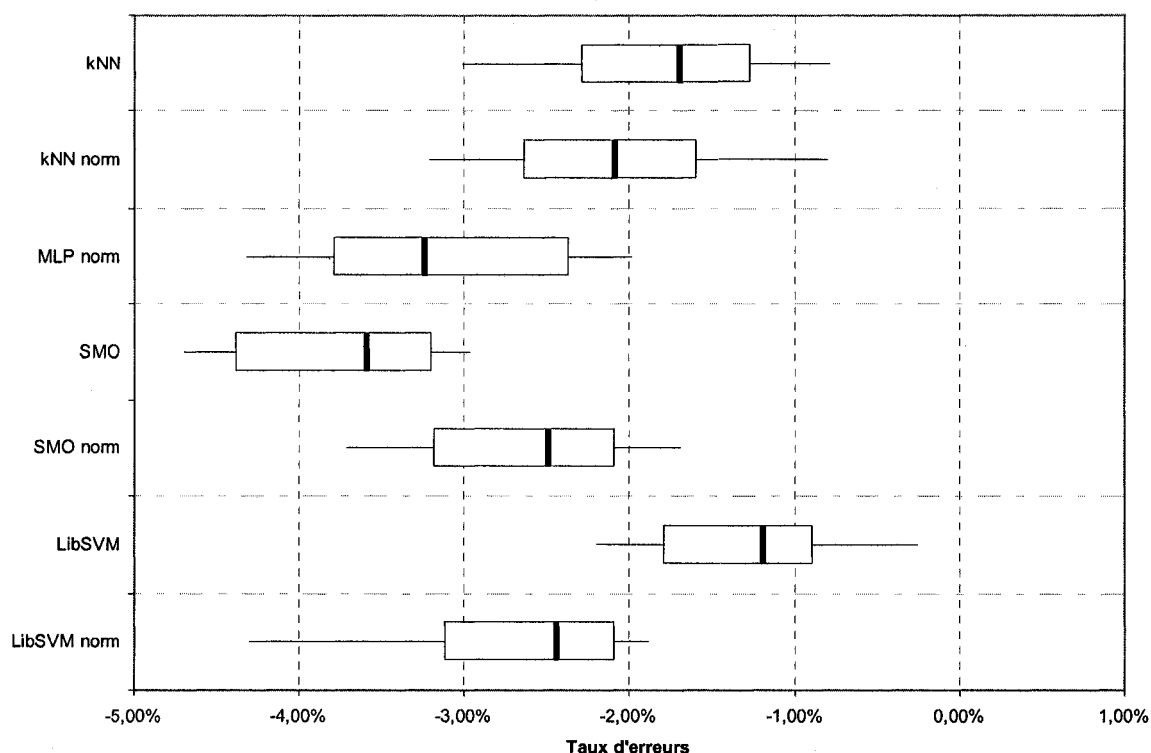


Figure 25 *Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de chiffres manuscrits et le kNN norm sur les données brutes normalisées*

Cependant, les comparaisons à classificateur égal sont un peu fictives. En effet dans la pratique, si nous n'avions pas eu à notre disposition la PG, nous aurions choisi le classificateur le plus performant sur ces données. Il serait donc plus honnête de comparer systématiquement les classificateurs après PG au meilleur classificateur avant PG. Bien entendu, les gains (ou pertes) seront moins avantageux. En fait, la mesure de gain sera même cette fois pessimiste par rapport à la réalité car il aurait fallu choisir le classificateur de référence (sans PG) sur une base de validation et non celle de test. Toutefois la taille des bases à notre disposition ne nous permettait que difficilement un tel découpage supplémentaire, et cela aurait causé une réduction de la taille de la base de test et donc un intervalle de confiance moins intéressant (*voir* CHAPITRE 5).

Pour en revenir aux résultats expérimentaux, le meilleur classificateur avant PG était le kNN norm. La *Figure 25* reporte donc les différences de performances des différents classificateurs après PG au kNN norm avant PG. Comme attendu, les pertes sont encore plus importantes que dans la comparaison précédente (*Figure 24*) : elles sont comprises entre 1 % et 3 %. En l'état, la PG n'est donc pas bénéfique en termes de taux de bonnes classifications.

4.4.2 Sur les céréales

Avant d'aller plus loin, observons les résultats sur la base des céréales. Cette fois, la dimensionnalité du problème a été conservée : nous construisons 10 nouvelles caractéristiques qui remplacent celles d'origine. Rappelons également que mise à part ce point, les paramètres de la PG sont les mêmes que pour les chiffres manuscrits. Les résultats expérimentaux après les 40 simulations sont reportés sur la *Figure 26*.

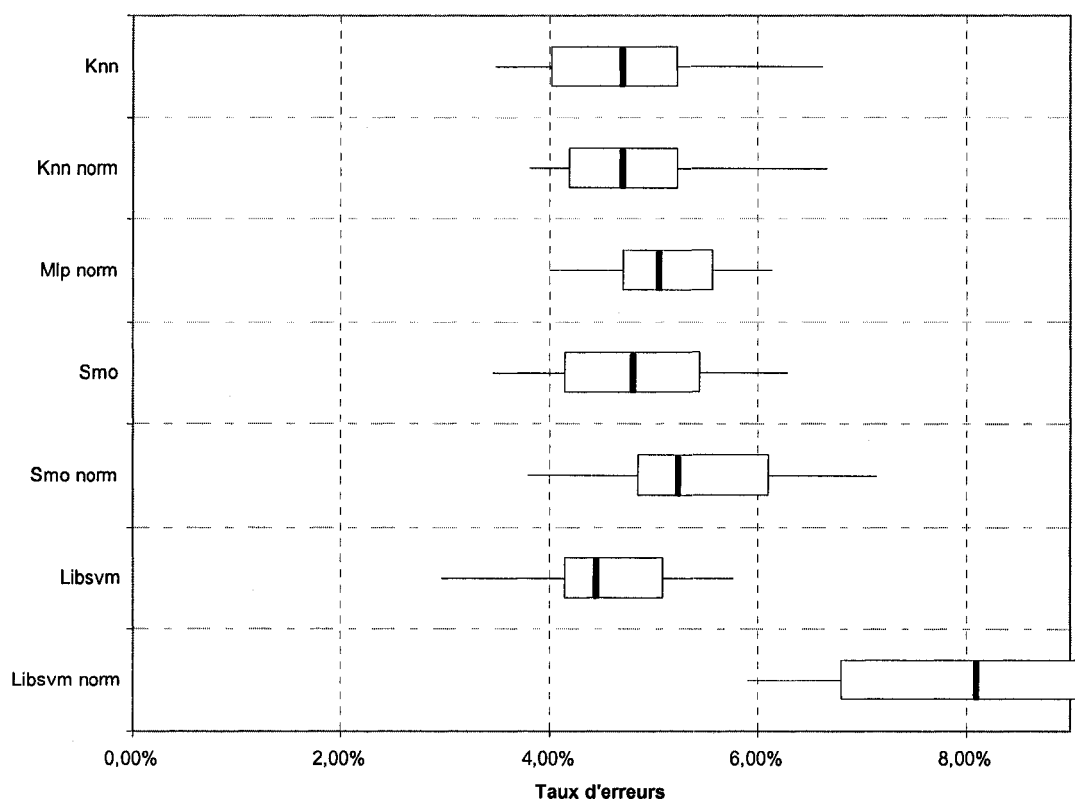


Figure 26 *Taux d'erreurs des classificateurs sur les caractéristiques obtenues par PG sur la base de céréales*

Tout d'abord, nous pouvons constater qu'après la PG les performances sont très similaires d'un classificateur à l'autre après des taux d'erreurs généralement compris entre 4 % et 5 %, excepté pour LibSVM norm. Ces performances sont également de l'ordre du meilleur classificateur sur les données non remaniées (voir 4.2.2). Dans le cas du problème de classification des céréales nous avons donc là un premier atout de la PG : le choix du classificateur final n'est plus critique. L'objectif de réduire les interventions humaines pour la conception d'un système de classification semble donc atteint.

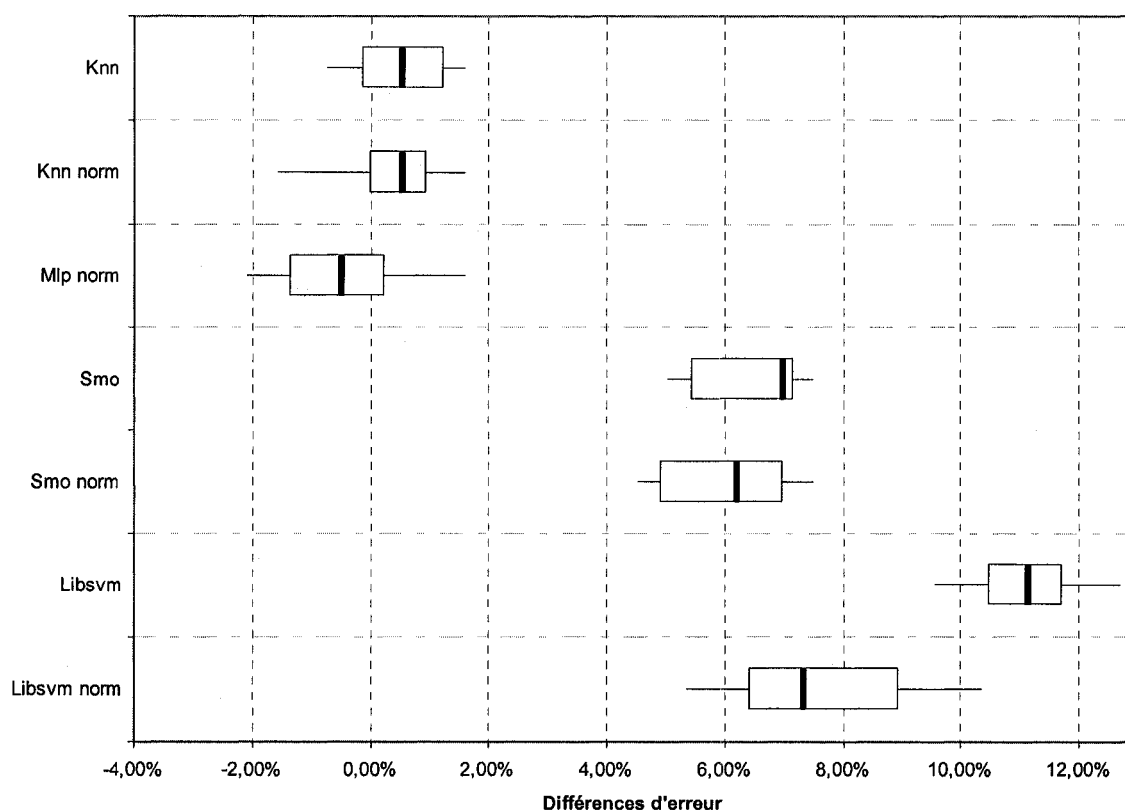


Figure 27 *Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de céréales et ces mêmes classificateurs sur les données brutes normalisées*

De la même manière que pour les chiffres manuscrits, nous allons analyser le gain apporté par la PG à classificateur égal à l'aide de la différence d'erreurs appariées (Figure 27). MLP norm mis à part, dans au moins les trois quarts des simulations (ou presque pour kNN) les performances ont été meilleures sur les caractéristiques calculées par la PG. Pour les classificateurs à base de SVM, le gain est même très important, au delà de 5 %. Pour le MLP norm, qui était le meilleur classificateur sur les données non remaniées, nous enregistrons une faible perte (0,52 %).

Quant à la comparaison des classificateurs après PG au MLP avant PG, la *Figure 28* nous confirme que la plupart des classificateurs ont effectivement rejoints celle du meilleur système avant PG. Plus précisément, excepté les SVM normalisés, le gain nul se situe toujours dans le troisième quartile, la présente analyse ne permettant donc pas de conclure en un écart statistiquement significatif.

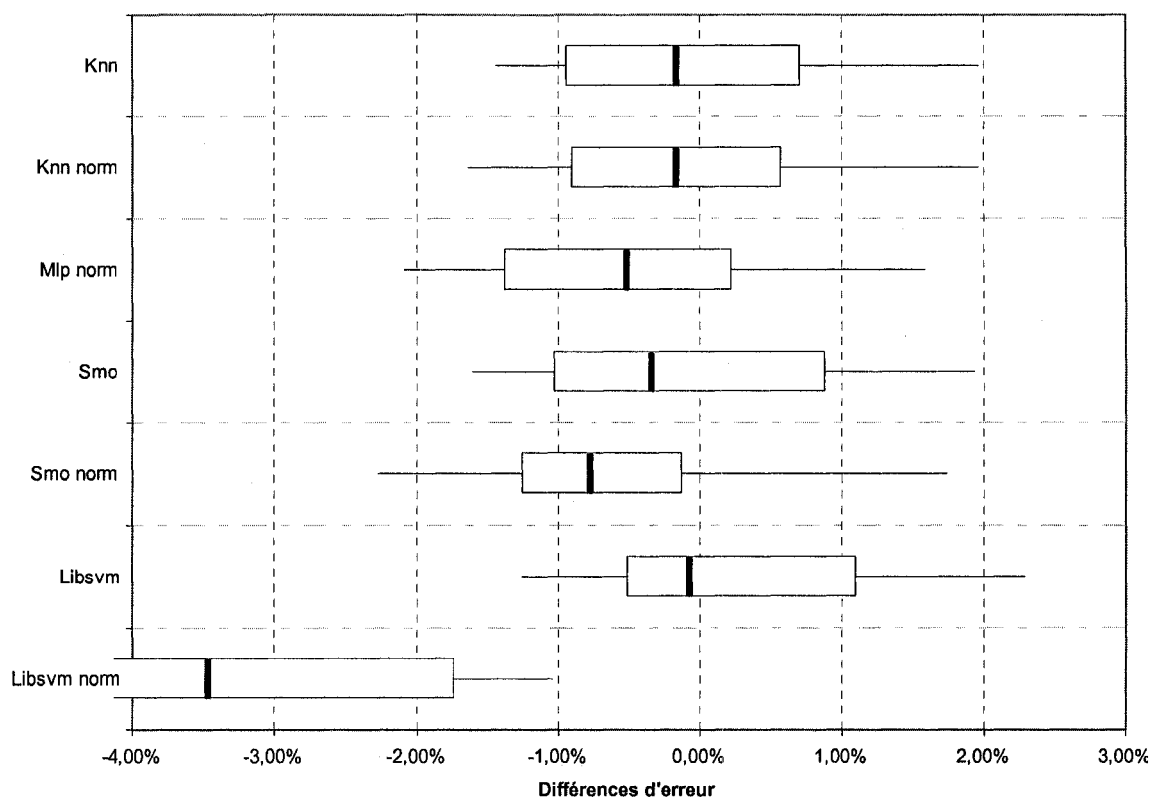


Figure 28 *Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de céréales et le MLP sur les données brutes normalisées*

4.5 Juxtaposition aux attributs bruts

4.5.1 Sur les chiffres manuscrits

Sur la base des chiffres manuscrits, les résultats des classificateurs sur les nouvelles caractéristiques n'ont pas été à la hauteur de ce qu'ils étaient sur les données d'origine. Nous avons avancé l'hypothèse de la réduction trop drastique de la dimensionnalité (voir 4.4.1), laquelle était inévitable compte tenu de la puissance de calcul à notre disposition. Cependant, si ces nouvelles caractéristiques facilitent réellement l'apprentissage des classificateurs, il pourrait être intéressant de combler leur manque d'exhaustivité par un autre moyen. Nous proposons donc de juxtaposer les deux jeux de caractéristiques : avant et après la PG. Nous obtenons donc une dimension totale de 130.

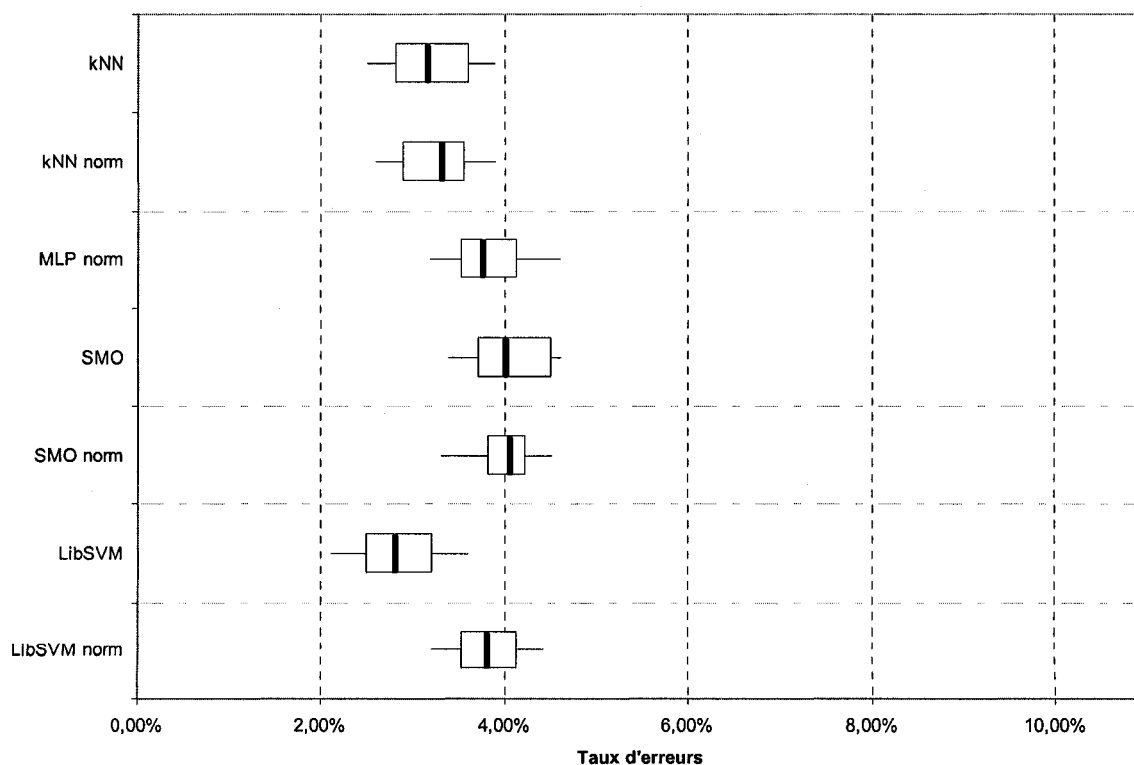


Figure 29 *Taux d'erreurs des classificateurs sur les caractéristiques obtenues par PG sur la base de chiffres manuscrits associées aux données brutes*

À première vue, la *Figure 29* montre des résultats semblables à ce qui avait été obtenu sur les caractéristiques brutes uniquement : assez regroupées et de l'ordre de 3 %. Pour comparer de manière plus fine, le gain apporté par la PG pour chaque type de classificateur est rapporté sur la *Figure 30*. Celle-ci nous confirme qu'excepté pour LibSVM, effectivement les performances avant et après PG sont très similaires avec des médianes presque nulles et des écarts interquartiles inférieurs à 0,4 %. Au moins, ces résultats montrent que la PG utilisée comme nous l'avons fait ne nuit pas à la classification. De plus, pour le LibSVM nous observons un gain significatif de l'ordre de 1 %.

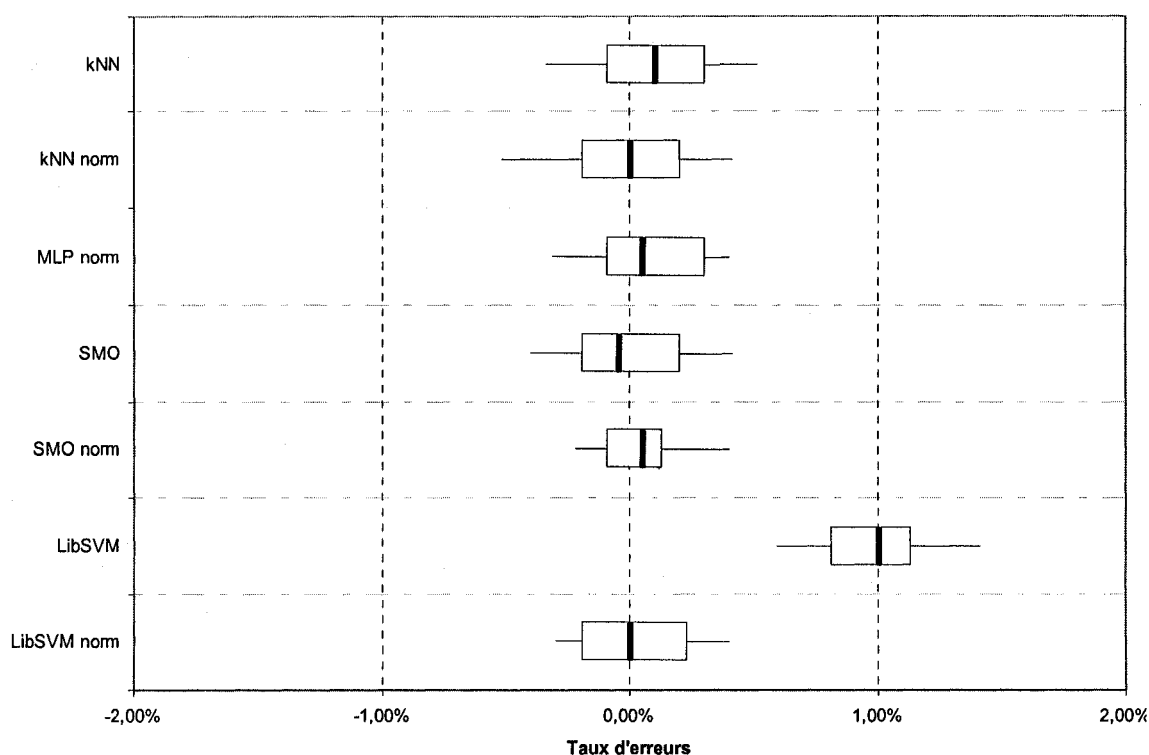


Figure 30 *Différences d'erreurs appariées des classificateurs avec les caractéristiques brutes et obtenues par PG sur la base des chiffres manuscrits et ces mêmes classificateurs sur les données brutes*

Comme le LibSVM n'était pas le plus performant avant la PG, il est intéressant de savoir si ce gain est un rattrapage des mauvaises performances préliminaires de ce classificateur ou bien s'il s'agit d'un gain absolu. Pour cela la *Figure 31* montre le gain des classificateurs sur le jeu hybride de 130 caractéristiques par rapport au meilleur système avant PG (i.e. le kNN). Nous constatons alors que plus des trois quarts des simulations ont attribuées de meilleures performances au couple PG/LibSVM qu'au kNN sur les données brutes, la médiane du gain si situant à 0,4 %. Grâce à la PG nous avons donc été en mesure de concevoir un meilleur système de classification sur notre base de chiffres manuscrits.

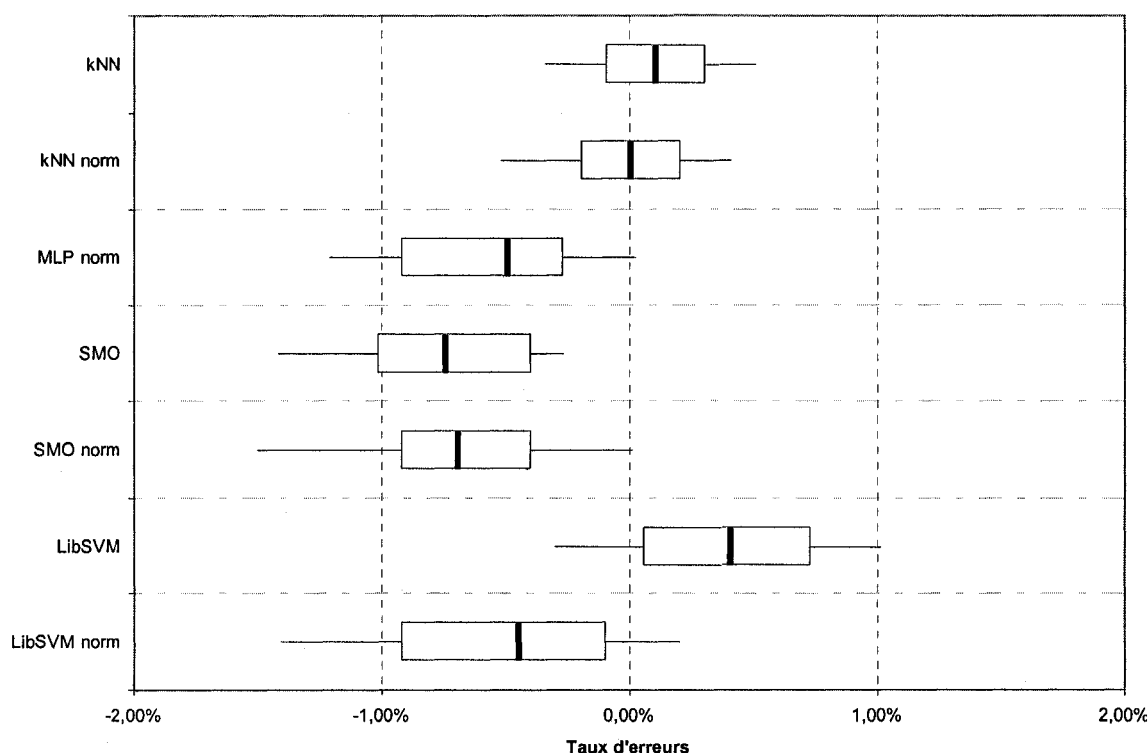


Figure 31 *Différences d'erreurs appariées des classificateurs avec les caractéristiques brutes et obtenues par PG sur la base des chiffres manuscrits et le kNN norm sur les données brutes normalisées*

4.5.2 Sur les céréales

Pour la base de céréales, l'analyse en quartiles des résultats des classificateurs sur les caractéristiques calculées par la PG nous a montré que notre système permettait d'être moins sensible au choix du classificateur, qui pourtant est habituellement critique. Cependant, les performances finales n'étaient pas tout à fait à la hauteur de ce que nous obtenions avec un bon choix de classificateur sur les données d'origine. Nous pouvons donc penser que les nouvelles caractéristiques construites par la PG sont effectivement plus simples à assimiler lors de l'apprentissage des classificateurs, mais qu'elles sont légèrement incomplètes. Autrement dit, l'information pertinente est mieux disposée mais une petite partie a été perdue.

Afin d'allier la robustesse des nouvelles caractéristiques face aux différents mécanismes d'apprentissage à l'exhaustivité des données originales, nous avons donc envisagé de juxtaposer ces deux jeux. Bien entendu, la dimension du problème s'en trouve augmentée. Pour ne pas faire face à « la malédiction de la dimension » [41][42], il faut donc bien s'assurer que nos bases disposent de suffisamment d'objets, ce qui est le cas pour les bases que nous avons choisies.

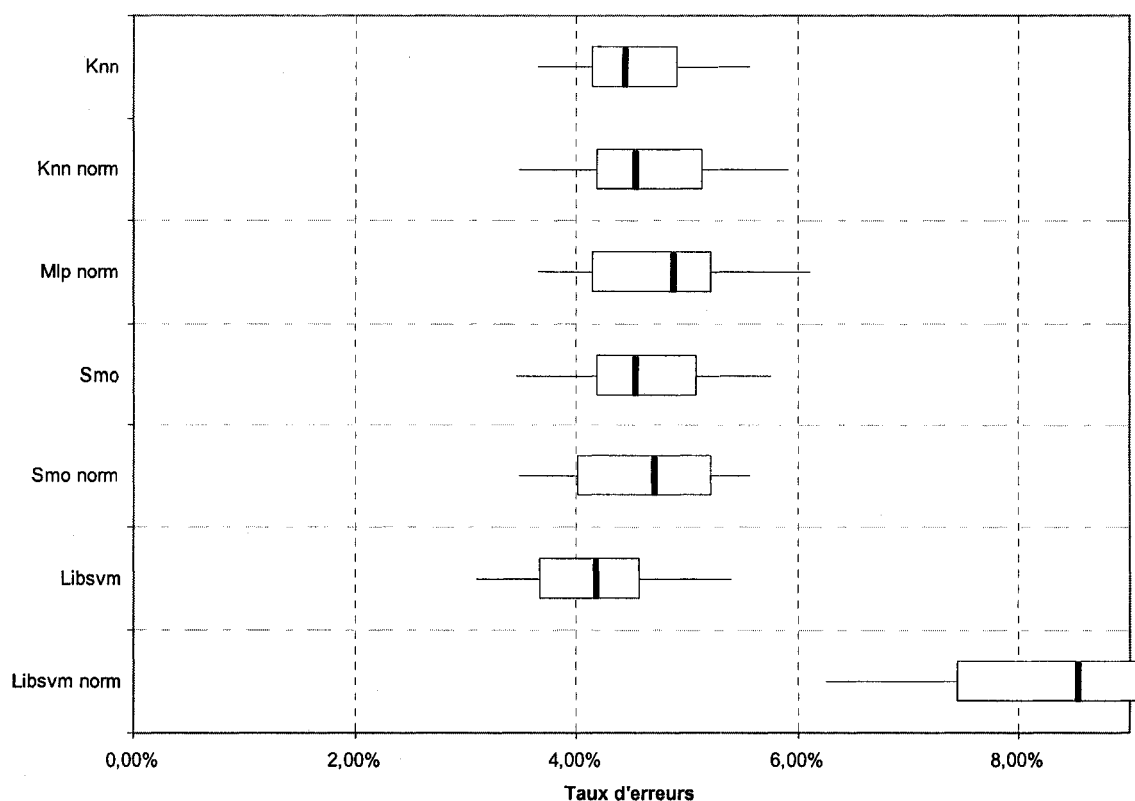


Figure 32 *Taux d'erreurs des classificateurs sur les caractéristiques obtenues par PG sur la base de céréales associées aux données brutes*

La Figure 32 nous rapporte les performances des classificateurs sur la base des céréales, avec à la fois les caractéristiques calculées par la PG et celles d'origine. Nous remarquons que tous les classificateurs ont des performances très proches, de l'ordre de 4 à 5 %, à la manière de la Figure 26 obtenue avec les caractéristiques de la PG uniquement. Cela nous conforte donc encore plus dans la robustesse du système vis-à-vis du choix du classificateur. Le LibSVM norm semble pourtant faire exception avec un taux d'erreurs encore très élevé.

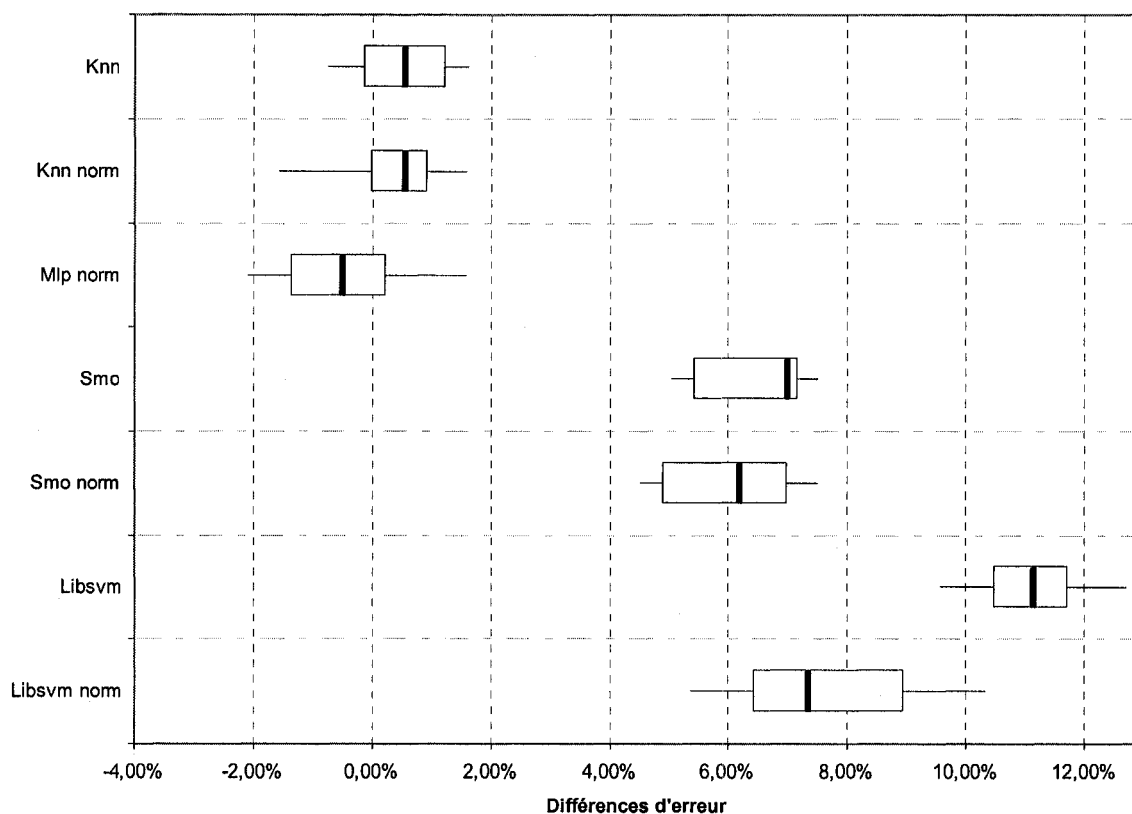


Figure 33 *Différences d'erreurs appariées des classificateurs avec les caractéristiques brutes et obtenues par PG sur la base de céréales et ces mêmes classificateurs sur les données brutes normalisées*

Si nous observons le gain apporté par la PG pour les différents classificateurs (Figure 33) nous remarquons qu'ils sont sensiblement positifs (supérieur à 5 %) pour tous les SVM. Pour les autres, l'analyse en quartiles ne permet pas de trancher. Quant à la Figure 34, elle nous permet de comparer les classificateurs après PG avec le meilleur classificateur avant PG (MLP norm). Celle-ci montre que hormis le LibSVM norm, tous les classificateurs ont rejoint les performances du MLP avant PG. La PG a donc bien apporté une bonne robustesse face au choix du classificateur. Nous pouvons par ailleurs nous attarder sur le cas de LibSVM : en excluant les cas d'égalité (10 % des cas), la performance de LibSVM après PG jumelée aux données brutes a été supérieure à la

performance de MLP norm deux fois plus souvent qu'elle n'a été inférieure. La médiane du gain se situe à 0,52 %, soit une diminution relative du nombre d'erreurs de 11 % par rapport au MLP norm. Ce dernier résultat, bien qu'encourageant, aurait avantage à être confirmé par une analyse statistique plus poussée.

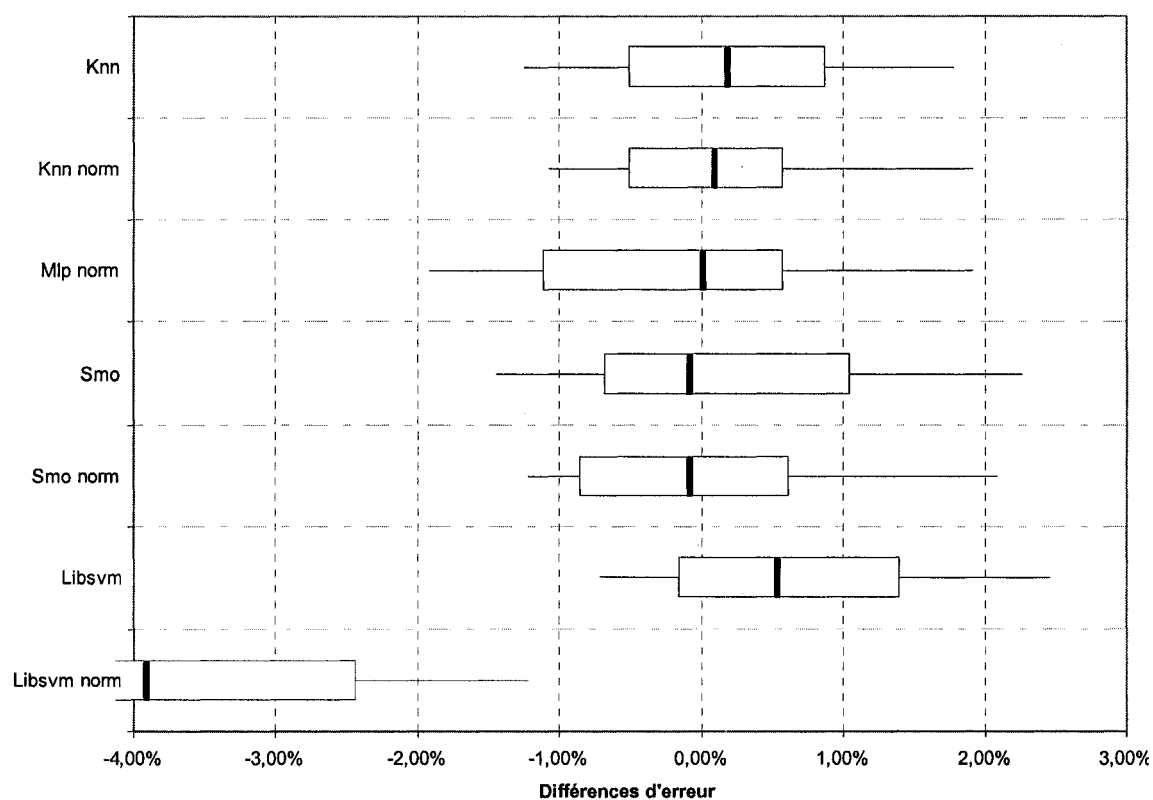


Figure 34 *Différences d'erreurs appariées des classificateurs avec les caractéristiques brutes et obtenues par PG sur la base de céréales et le MLP sur les données brutes normalisées*

4.6 Calcul du gain de la PG sur d'autres méthodes

4.6.1 Sur les chiffres manuscrits

Pour la base des chiffres manuscrits, la PG avait construit 30 caractéristiques. Nous allons donc comparer la PG à une autre méthode de réduction de la dimension. Parmi les différents algorithmes de sélection de caractéristiques, nous avons choisi un algorithme génétique (AG). Il a l'avantage de donner de bons résultats même si les caractéristiques d'origine sont fortement interdépendantes. Dans la pratique, nous avons utilisé l'AG de Weka avec ses paramètres par défaut pour la sélection de 30 caractéristiques. La *Figure 35* rapporte les performances des classificateurs sur ce sous-ensemble. Nous notons que le kNN, le kNN norm, le SMO norm et le LibSVM norm ont tous les quatre de faibles taux d'erreurs aux environs de 6 %. Les autres classificateurs ont des performances significativement moins bonnes.

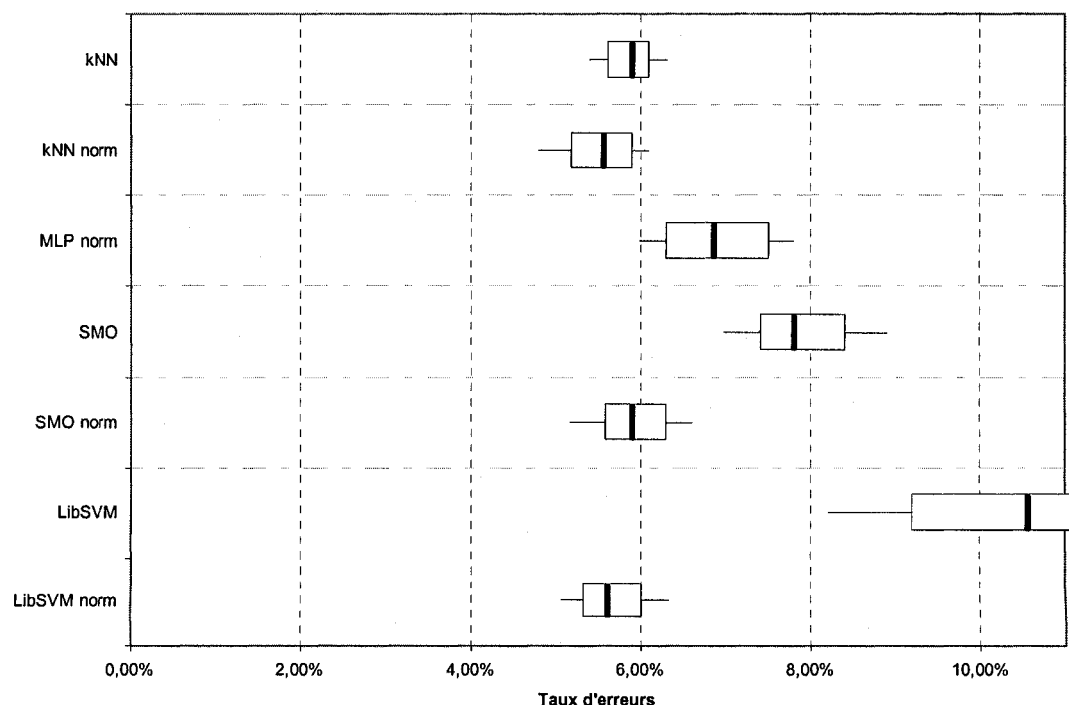


Figure 35 *Taux d'erreurs des classificateurs sur les trente caractéristiques conservées par AG de la base des chiffres manuscrits*

Si nous comparons l'AG à la PG (Figure 36), nous observons que le gain est parfois en faveur de l'AG (SMO), parfois en faveur de la PG (LibSVM), mais dans la plupart des cas il est faible. Toutefois, avec la Figure 37, en comparant la PG au meilleur classificateur après AG (kNN norm), nous remarquons qu'encore une fois LibSVM se démarque avec les trois quart de ses simulations affichant un gain supérieur à 0,47 %. Le gain médian se situe quant à lui à 0,80 %, ce qui est beaucoup puisque nous parlons de taux d'erreurs de 5 à 6 %. La PG permet donc d'obtenir de meilleurs résultats avec un bon choix de classificateur, et si nous considérons l'ensemble des classificateurs étudiés, les résultats sont également plus homogènes avec la PG qu'avec une sélection par AG.

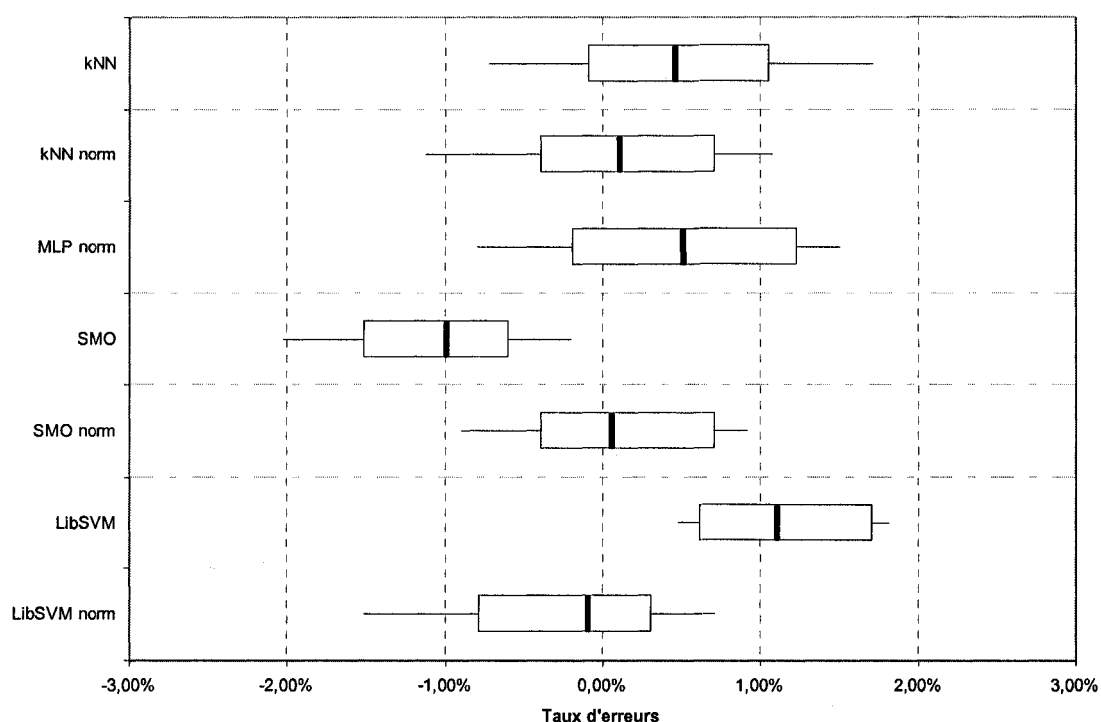


Figure 36 *Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base des chiffres manuscrits et ces mêmes classificateurs sur les données conservées par AG*

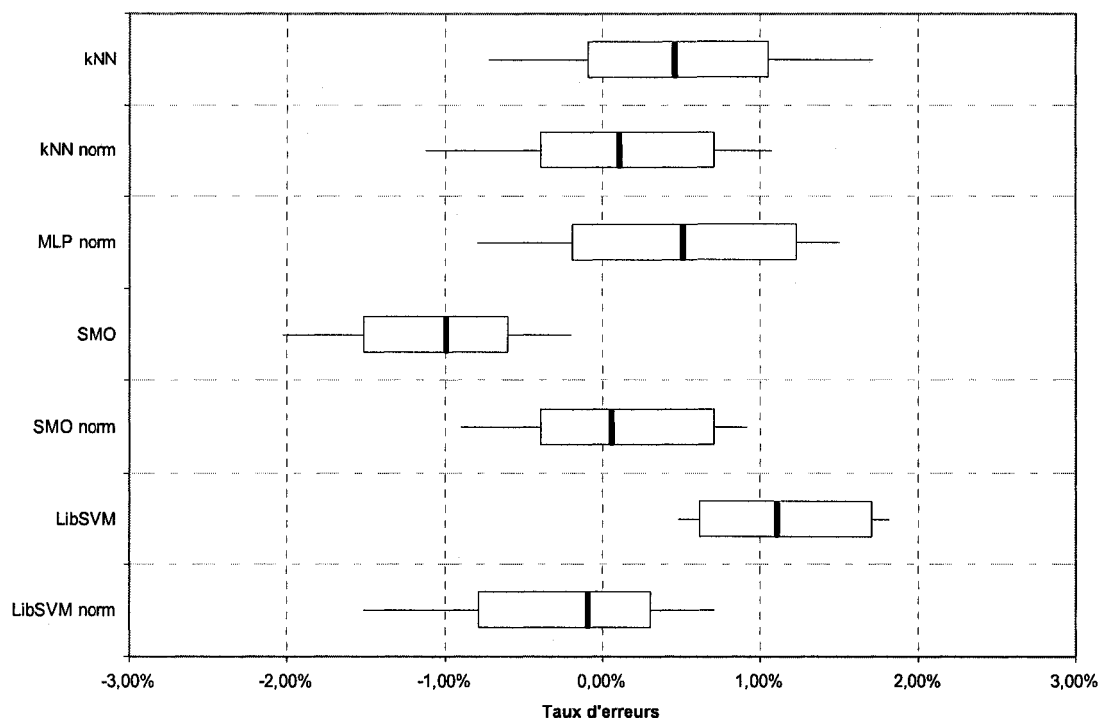


Figure 37 *Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de chiffres manuscrits et le kNN norm sur les données conservées par AG et normalisées*

Nous avons également voulu comparer notre système à un autre moyen de réduction de la dimension : l'analyse en composantes principales (ACP). Pour rester consistant, nous avons conservé les 30 premières caractéristiques par importance données par l'ACP. Nous les avons ensuite passées aux classificateurs, ce qui a donné les résultats de la *Figure 38*. Sur le problème des chiffres manuscrits, l'ACP semble donc donner de très bons résultats avec des erreurs médianes comprises entre 3,0 % et 5,4 %. L'excellence de ces résultats est confirmée par l'étude de la différence d'erreurs appariées avec les 30 caractéristiques de la PG (*Figure 39*). Cependant, l'erreur médiane du meilleur classificateur après ACP (SMO norm avec 3,0 % d'erreurs) reste plus élevée que celle obtenue avec les 30 caractéristiques de la PG jumelées aux données brutes (LibSVM avec 2,8 % d'erreurs).

Nous avons également commencé l'étude de la PG menée sur les caractéristiques fournies par l'ACP mais les résultats n'ont malheureusement pas été encourageant (le gain ne semblait pas significatif). Compte tenu que l'étude complète de ce système aurait doublé le nombre de simulations nécessaires, nous avons décidé de ne pas pousser plus loin cette voie.

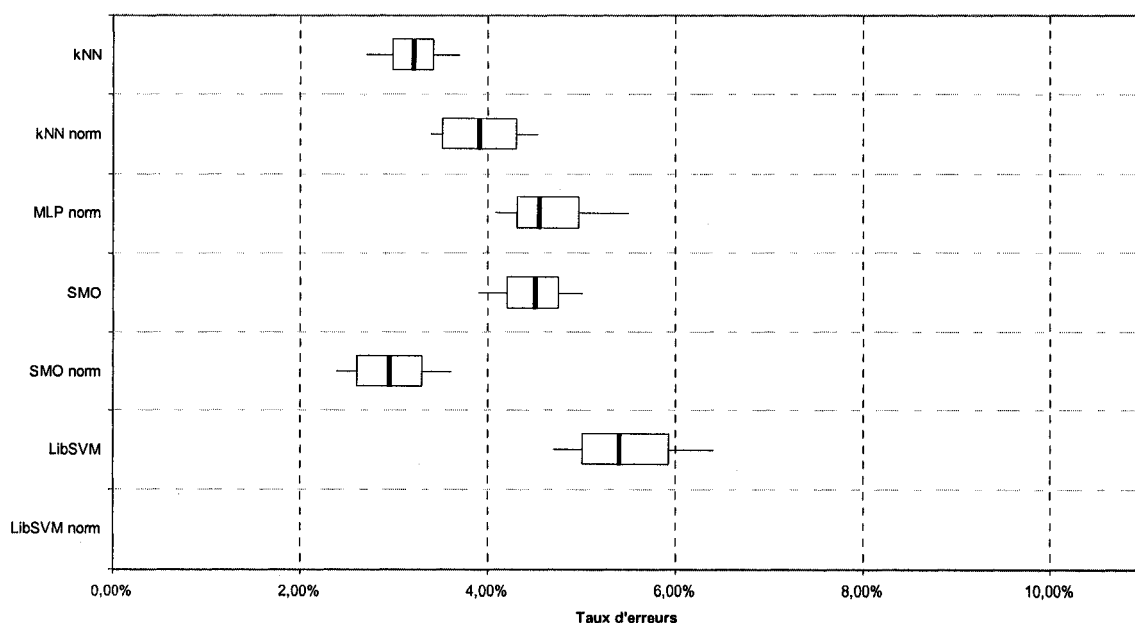


Figure 38 *Taux d'erreurs des classificateurs sur les trente caractéristiques conservées par ACP de la base des chiffres manuscrits*

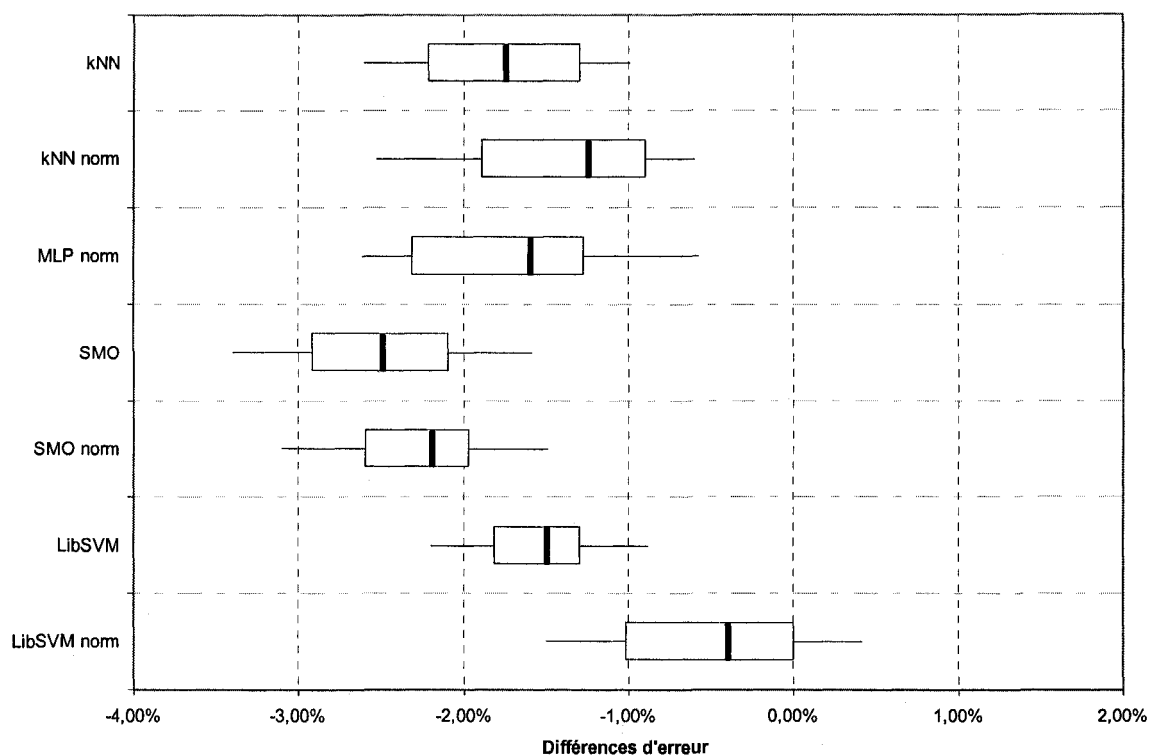


Figure 39 *Différences d'erreurs appariées des classificateurs avec les caractéristiques obtenues par PG sur la base de chiffres manuscrits et ces mêmes classificateurs sur les données conservées par ACP*

4.6.2 Sur les céréales

En ce qui concerne la base des céréales, la dimension est la même avant et après PG. Il n'y a donc pas lieu de faire une sélection par AG comme pour la base des chiffres manuscrits. Par contre, il est tout à fait possible de faire une ACP pour décorréler les caractéristiques. La *Figure 40* rapporte ces résultats.

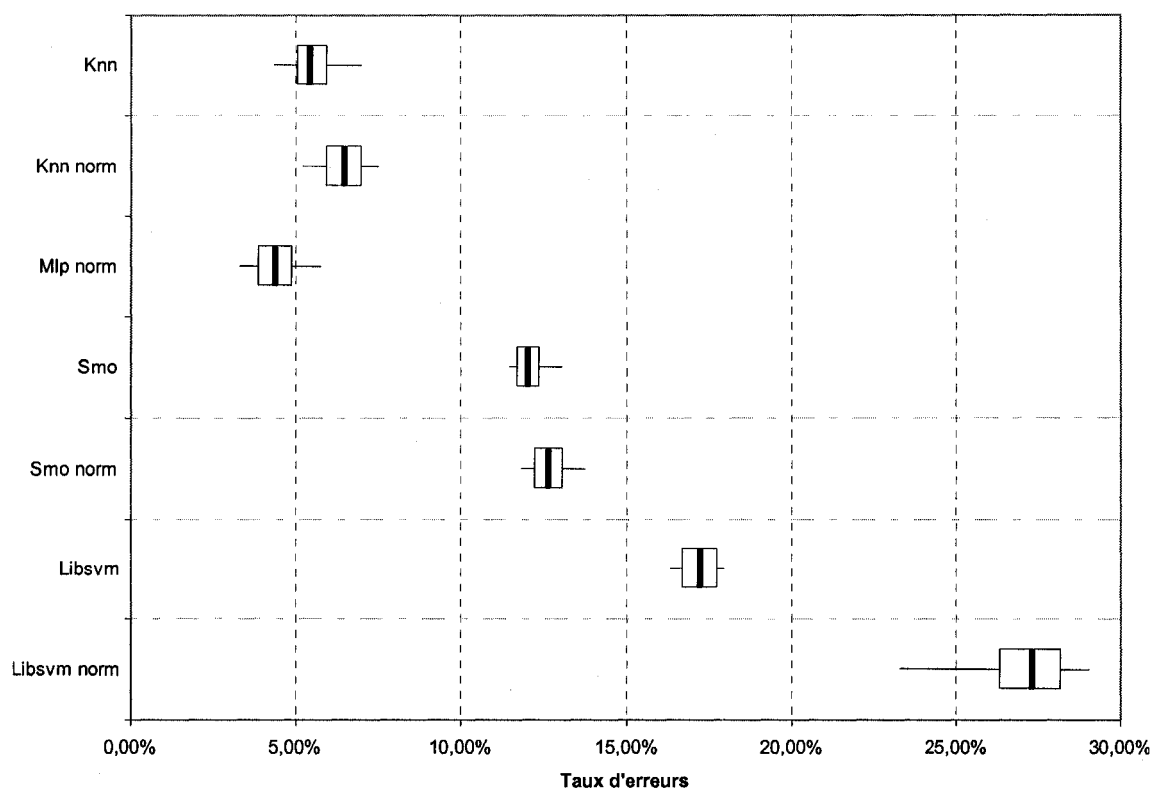


Figure 40 *Taux d'erreurs des classificateurs sur les caractéristiques transformées par ACP sur la base des céréales*

Comme nous pouvons le constater, les taux d'erreurs obtenus sont plusieurs fois très élevées, supérieures à 12 % avec les SVM. Pour le kNN, les résultats sont compris entre 5 et 7 %, ce qui est du même ordre de grandeur que sur les données brutes, et donc un peu moins bon qu'en utilisant la PG à la place de l'ACP. Enfin le MLP norm semble bien tirer parti de l'ACP avec une erreur médiane de 4,35 %. C'est mieux que tout ce qui avait été obtenu sur les données brutes, à la hauteur de PG/LibSVM (4,43 %) et un peu moins bon que ce qui avait été obtenu en associant les caractéristiques de la PG avec les brutes sur LibSVM (4,17 %).

4.7 Analyse des résultats

4.7.1 Sur les chiffres manuscrits

L'analyse en quartiles a montré que le passage à 30 caractéristiques par PG apportait une perte de performance par rapport aux 100 caractéristiques brutes, et cela quel que soit le classificateur utilisé. Toutefois, si nous comparons à dimension égale, la PG semble avoir des résultats similaires à la sélection de caractéristiques par AG. De plus, si nous nous concentrons sur le couple PG/LibSVM (LibSVM semble être de manière générale le meilleur complément à la PG comme nous allons le voir également avec la base des céréales), il semble que nous surpassions l'AG, quel que soit le classificateur utilisé pour ce dernier. Par contre, comparativement à l'ACP, à une dimension de 30, la PG est significativement surpassée.

Nous avons également mesuré les performances des classificateurs en juxtaposant les caractéristiques de la PG avec celles d'origine. En considérant LibSVM par la suite, c'est pour ce système que les meilleures performances ont été trouvées, avec une erreur médiane de 2,8 %. Cependant, l'analyse en quartiles ne montre pas de manière très significative ce gain, une analyse statistique plus poussée est préférable pour pouvoir prendre position (*voir* CHAPITRE 5).

Concernant la robustesse apportée par la PG en ce qui concerne le choix du classificateur, notre base des chiffres manuscrits ne permet pas de tirer de conclusion. En effet, bien que les différents classificateurs aient tous eu des performances comparables après PG, c'était déjà le cas sur les données brutes. Nous remarquons toutefois qu'après PG LibSVM semble avoir un avantage systématique sur ses concurrents.

4.7.2 Sur les céréales

En ce qui concerne la base des céréales, ce qui a été le plus flagrant est que la PG a permis de se rendre beaucoup moins sensible au choix du classificateur. En effet, sur cette base, les résultats obtenus en utilisant directement les classificateurs sur les données brutes étaient très disparates, alors qu'après passage par la PG ils étaient bien mieux regroupés.

Quant aux performances absolues, comme pour la base des chiffres manuscrits, c'est le couple PG+données brutes/LibSVM qui sort vainqueur avec une erreur médiane de 4,17 %. Cependant d'autres méthodes comme l'ACP/MLP produisent des résultats honorables eux aussi (*voir* 4.6.2). L'analyse en quartiles ne permet pas d'être catégorique sur le gain réalisé. Les résultats absolus de la PG sont tout de même encourageants, et méritent donc de poursuivre le travail de confirmation de ce possible gain.

4.7.3 Récapitulatif

Nos résultats sur nos deux bases ont montré que les performances à l'aide de la PG sont comparables aux meilleurs résultats des autres méthodes, surtout quand elle est associée à LibSVM. Nous observons même un léger gain lorsque nous réunissons les caractéristiques obtenues par la PG avec les données brutes, lequel est toutefois à confirmer par une analyse statistique plus poussée.

Cependant, l'apport le plus clair que la PG nous a donné est certainement la robustesse face au choix du classificateur. En effet, bien que LibSVM semble avoir un avantage, les autres types de classificateurs ont également donné des performances respectables, ce qui n'était pas toujours le cas avec les données d'origine. Au-delà du gain absolu qui

reste à confirmer, la PG semble donc être un bon moyen de limiter les interventions humaines lors de la conception d'un système de classification.

CHAPITRE 5

ANALYSE STATISTIQUE AVANCÉE

5.1 Intervalles de confiance

5.1.1 Mesure de la performance d'un système

Une exécution d'un classificateur nous donne par son taux de bonnes classifications une mesure de sa performance. Seulement, c'est en général une information insuffisante car nous ne savons pas à quel point cette mesure est proche de sa performance réelle. C'est pourquoi nous devons répéter plusieurs fois l'expérience, avec des découpages différents de la base de données. C'est ce que nous avons fait dans le chapitre précédent, mais en nous contentant d'une approche principalement visuelle (analyse en quartiles). Cependant, les chiffres obtenus peuvent être exploités de façon plus rigoureuse à l'aide d'une analyse statistique. En effet, les simulations que nous avons effectuées sont des expériences de Bernoulli, qui pourront nous donner un intervalle de confiance pour la performance réelle. Autrement dit, nous pourrions affirmer avec une probabilité fixée α (niveau de confiance) que la performance réelle du classificateur est dans cet intervalle.

En résumé, l'intervalle de confiance de la performance réelle (nous supposons que les mesures suivent une loi normale de variance inconnue) peut être déterminé à partir des valeurs de n observations indépendantes $x_1, \dots, x_n$ suivant toutes cette même loi. En réalité, nous nous intéresserons à deux intervalles de confiance : celui concernant la performance d'une réalisation du système de classification ainsi que celui de l'espérance de cette performance. Ces notions seront détaillées dans les paragraphes suivants.

5.1.2 Intervalle de confiance de la performance d'une réalisation du système

Ici, nous chercherons à déterminer avec un niveau de confiance α dans quel intervalle devrait se situer la performance d'une réalisation de notre système de classification. Nous entendons par réalisation un système qui a fait son apprentissage sur une base donnée, et sa performance est le taux de bonnes classifications sur la base de test, elle aussi fixée. Plus simplement, nous tentons donc de prévoir la performance à laquelle nous devons nous attendre avec ce type d'algorithme sur ce genre et cette taille de base (d'apprentissage et de test). Cette estimation a donc un intérêt pratique pour ceux qui voudront utiliser notre système sur les bases que nous avons choisies. Cependant, nous verrons plus loin que ce n'est pas forcément la seule bonne méthode pour comparer les performances de différents types de systèmes.

Plus mathématiquement, il s'agit de la dispersion statistique de la mesure de la performance du système, le découpage de la base en apprentissage et test changeant à chaque mesure. Pour un grand nombre de mesures, nous supposons que la loi suivie est gaussienne. Définissons alors quelques termes :

- La moyenne empirique des résultats : $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$
- La variance empirique corrigée : $\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$

L'erreur due à la dispersion est alors estimée par :

$$\Delta_x = t_{\frac{1-\alpha}{2}}^n \cdot \hat{\sigma} \quad (1)$$

t_{γ}^k étant le quantile d'ordre $1-\gamma$ de la loi de *Student* à k degrés de liberté, qui dépend donc uniquement du nombre de mesures et du niveau de confiance souhaité. La preuve

de ce résultat est disponible sur [43][44]. Donc pour un niveau de confiance $\alpha = 80 \%$, $\gamma = \frac{1-\alpha}{2} = 0,1$. Voici alors le tableau rassemblant les valeurs qui nous seront utiles :

Tableau 2

Quantile de Student

k	1	2	3	4	5	10	20	30	39	40
$t_{0.1}^k$	3,078	1,886	1,638	1,533	1,476	1,372	1,325	1,310	1,304	1,303

L'intervalle de confiance est alors $[\bar{x} - \Delta_x, \bar{x} + \Delta_x]$. Cela signifie que quelqu'un qui découperait la même base de données en apprentissage et test selon le même ratio aura une probabilité de α de mesurer une performance comprise entre $\bar{x} - \Delta_x$ et $\bar{x} + \Delta_x$. Nous pouvons remarquer que cette formule respecte bien notre intuition : l'intervalle de confiance est d'autant plus fin que nous répétons l'expérience et que les observations sont stables.

5.1.3 Intervalle de confiance de l'espérance de la performance du système

Nous tenterons ici d'avoir un résultat un peu plus fin. Il s'agit de déterminer avec un certain niveau de confiance l'espérance de la performance du système. En d'autres mots, si nous faisons une infinité de découpages apprentissage/test de la base et qu'à chaque fois nous mesurons la performance, il y aurait une probabilité α que la moyenne soit située dans l'intervalle de confiance calculé. Entendons-nous bien : la moyenne de cette infinité de mesures n'est pas une variable aléatoire, elle n'est ni plus ni moins égale à l'espérance recherchée. Cependant cette espérance n'est pas connue, c'est pour ça que nous essayons de l'encadrer.

Plus mathématiquement, il s'agit de la détermination rigoureuse de l'intervalle de confiance associé à l'espérance d'une variable aléatoire de loi normale de variance inconnue. Cet intervalle est le suivant [46] :

$$\left[\bar{x} - t_{\frac{1-\alpha}{2}}^{n-1} \frac{\hat{\sigma}}{\sqrt{n}}, \bar{x} + t_{\frac{1-\alpha}{2}}^{n-1} \frac{\hat{\sigma}}{\sqrt{n}} \right] \quad (2)$$

avec :

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (3)$$

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (4)$$

et

t_{γ}^k le quantile d'ordre $1-\gamma$ de la loi de *Student* à k degrés de liberté, déjà défini dans la section précédente.

Nous pouvons remarquer que cet intervalle est nécessairement strictement inclus dans celui de la section précédente, ce qui permettra de départager plus de systèmes.

5.1.4 Comparaison fine de deux systèmes

Les paragraphes précédents définissent des intervalles de confiance sur la performance réelle d'un système. Pour comparer deux systèmes, nous pouvons donc tout simplement voir si un intervalle est au dessus de l'autre. Cependant, si les performances sont

proches, il arrive que ces intervalles se chevauchent, dans ce cas il n'est pas possible de dire qu'il y a une différence significative de cette façon.

Pourtant, il est envisageable de faire une comparaison plus fine sans pour autant multiplier le nombre de simulations. En effet, une bonne partie des fluctuations des performances observées proviennent du fait que la base de test est difficile ou non et que la base d'apprentissage est bien représentative. Ces fluctuations vont alors nuire à la comparaison des systèmes si nous ne nous intéressons qu'à leurs intervalles de confiance. En revanche, il est possible de calculer directement un intervalle de confiance sur la différence de performances des deux systèmes sur une même base [30][45]. Il suffit de répéter l'expérience autant de fois que nous le voulons en changeant le découpage de la base et en notant la différence de performance pour chacun de ces découpages. L'intervalle de confiance se calcule alors des mêmes façons que dans les sections précédentes. Dès lors, si l'intervalle ne chevauche pas zéro, nous pourrions conclure sur la supériorité d'un système par rapport à l'autre.

5.1.5 Validation croisée?

Nous n'avons pas parlé jusqu'ici de validation croisée pour la mesure de la performance des systèmes de classification (seulement en interne à la PG pour calculer l'adéquation). La raison est que nous n'en ferons pas. Avant d'expliquer pourquoi, nous allons rappeler l'intérêt de la validation croisée.

En temps normal, nous cherchons à résoudre un problème de classification. Nous avons donc une base qui nous a été fournie et nous devons construire un système ainsi que prévoir sa performance sur les futures données (non fournies) qu'il aura à classer. Donc ultimement, l'apprentissage se fait sur la base fournie au complet. Cependant, pour déterminer une estimation de la performance, il faut avant ça découper la base afin d'en garder une partie pour le test. C'est ici qu'intervient la validation croisée. L'estimation

est d'ailleurs en général légèrement pessimiste car la base d'apprentissage finale est un peu plus grosse.

Dans le cadre de notre travail, l'objectif est sensiblement différent. En effet, nous cherchons à prouver l'utilité d'un concept, celui d'utiliser la PG en amont pour améliorer les performances d'un classificateur. En autant que les bases de données sont suffisamment grandes, il n'est pas nécessaire de procéder par validation croisée. Il est tout à fait correct de procéder à des découpages en deux moitiés égales pour l'apprentissage et le test. À chaque fois que nous voudrions reprendre une mesure de la performance du système, nous utiliserons un découpage différent mais ayant le même ratio. Nous pouvons même aller plus loin, le temps de calcul est ainsi mieux utilisé car chaque itération d'une validation croisée demande un lourd apprentissage pour une mesure de performance peu fiable car la base de test ne contient en général que 10 % des données.

5.2 Application sur la base des chiffres manuscrits

5.2.1 Performance d'une réalisation du système

Comme expliqué plus haut, il s'agit d'une prévision de la performance à laquelle nous devrions nous attendre lors d'une réalisation du système. La *Figure 41* fait référence à tous les systèmes qui ont été expérimentés au chapitre précédant, mais en exprimant les résultats sous la forme d'intervalles de confiance d'une réalisation. Les résultats numériques sont consultables dans le Tableau 16 de l'ANNEXE 5.

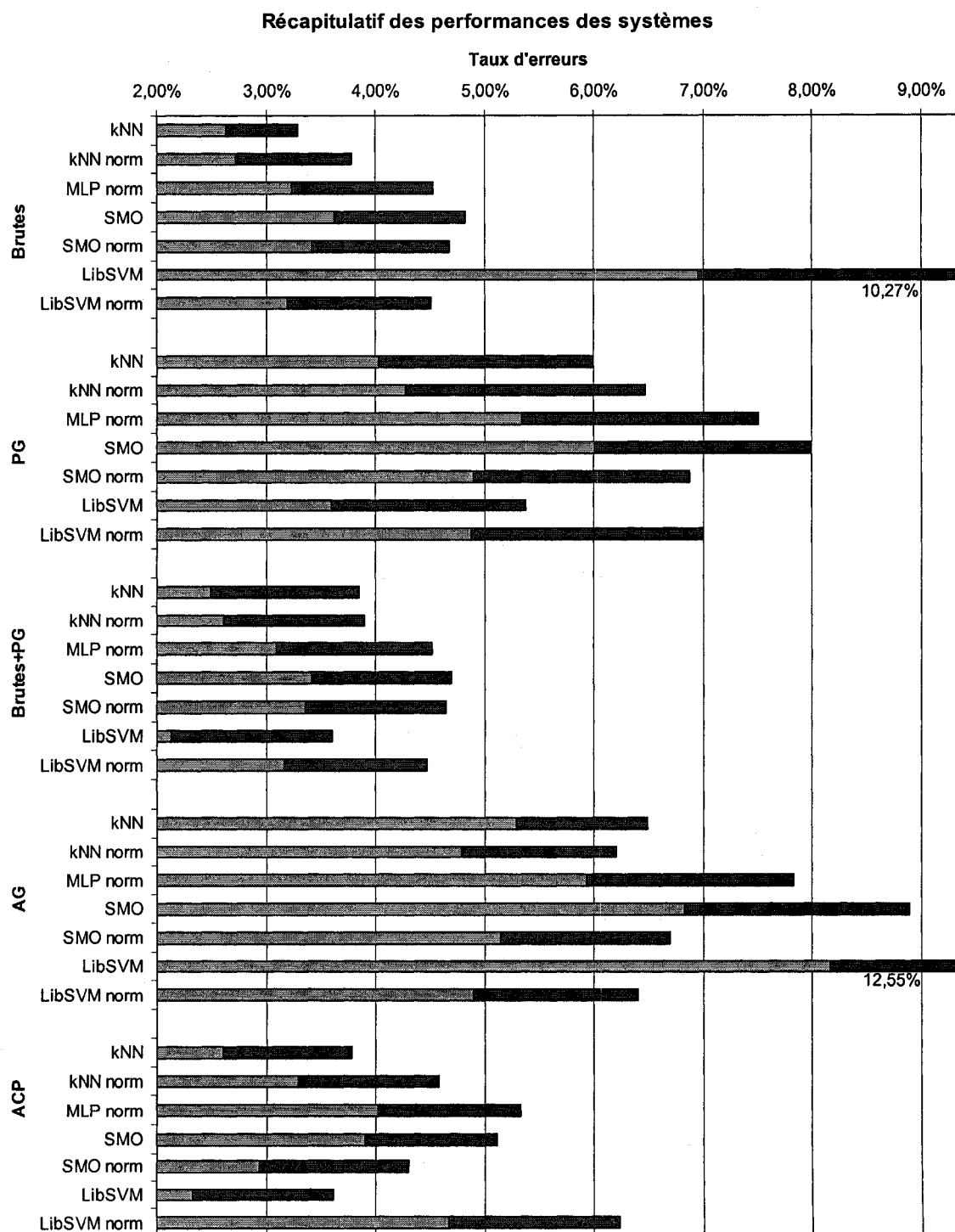


Figure 41 *Intervalle de confiance d'une réalisation de système. Les valeurs représentent les bornes minimum et maximum de ces intervalles*

Nous pouvons remarquer que bien souvent ces intervalles de confiance se chevauchent, ce qui ne laisse que peu de conclusions précises à tirer. La tendance qui veut que ce soit LibSVM le mieux adapté dans les systèmes à base de PG est bien là. Toutefois, ces seuls résultats ne permettent pas de dire à coup sûr (du moins avec une confiance suffisante) qu'une réalisation avec PG/LibSVM sera meilleure qu'une autre réalisation de système PG/autre classificateur. Par contre dans de nombreux cas (en fait tous hormis le MLP dont les intervalles se chevauchent), nous pouvons affirmer qu'adjoindre les données brutes à celles obtenues par PG améliore sensiblement les performances par rapport aux données de la PG seules.

En fait, ces résultats sont principalement là pour nous donner une idée des résultats à attendre lors d'une réalisation d'un système de classification sur cette base. Pour des fins de comparaisons, il est inutile de croiser ces intervalles davantage car il est plus précis de déterminer les intervalles de confiance des espérances de performances ou encore des erreurs appariées.

5.2.2 Espérance de la performance des systèmes

La *Figure 42* donne les intervalles de confiance des espérances de performance, lesquels sont nécessairement plus étroits que ceux concernant les réalisations ponctuelles. Les résultats numériques sont consultables dans le Tableau 17 de l'ANNEXE 5. Rappelons que ces intervalles ne donnent en aucune manière une prévision des performances auxquelles s'attendre lors d'une simulation. Ils fournissent seulement une indication de l'aptitude du système à résoudre le problème de classification.

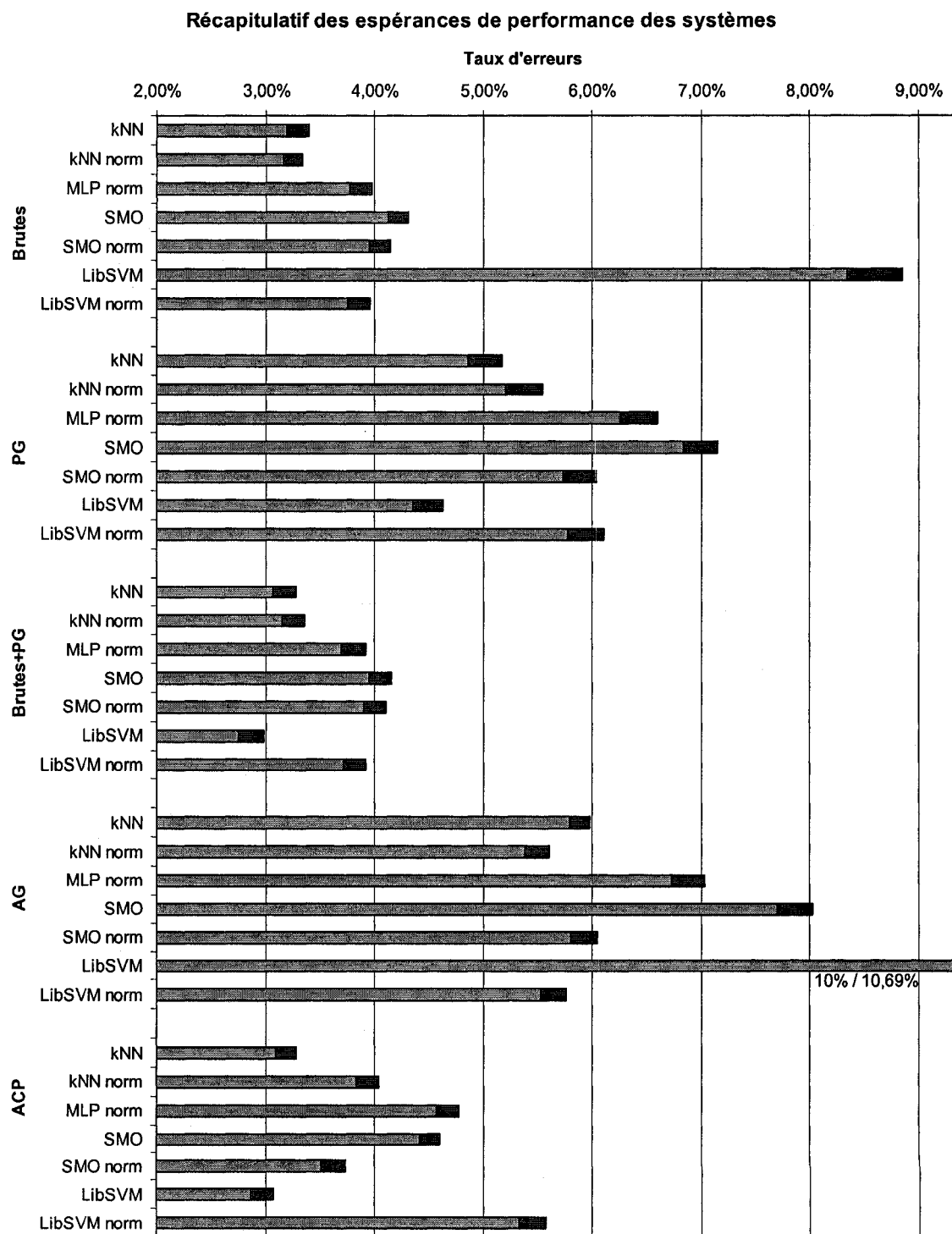


Figure 42 Intervalles de confiance de l'espérance de la performance des systèmes

Ces intervalles confirment en grandes parties les observations qui avaient été faites à l'aide de l'analyse en quartiles :

- Les performances avec PG sont meilleures qu'avec AG mais moins bonnes qu'avec les données brutes (de dimension plus grande) ou les données de l'ACP.
- Dans le cas des données fournies par la PG, LibSVM est plus avantageux que tous les autres classificateurs, avec un intervalle qui se détache des autres. Même constat dans le cas du cumul des données brutes et de celles de la PG.
- En absolu, deux systèmes sortent du lot. Le premier est le couple PG+Brutes/LibSVM avec un intervalle de confiance [2,75 %; 2,98 %]. Le second est le couple ACP/LibSVM dont l'espérance se situe dans [2,87 %; 3,07 %]. Rappelons que sur les données brutes c'est le kNN norm qui était le plus performant avec [3,17 %; 3,33 %], soit un gain relatif de l'ordre de 10 %.

5.2.3 Comparaisons

Nous avons vu qu'il était possible de calculer un intervalle de confiance directement sur la différence de performance à découpage de base identique. Au total, nous pourrions donc faire cette étude pour chaque couple de système de classification. Puisque nous avons 35 systèmes, cela ferait $C^2_{35} = 595$ comparaisons possibles. Nous ne ferons donc cette étude que dans les cas intéressants.

Nous avons vu dans la section précédente qu'en tant que performance absolue, deux systèmes semblaient sortir du lot : PG+Brutes/LibSVM et ACP/LibSVM. Le calcul de l'intervalle de confiance de l'espérance du gain entre ces deux systèmes donne [0,05 %; 0,15 %]. Celui-ci étant strictement situé au dessus de zéro, nous pouvons affirmer que le couple PG+Brutes/LibSVM est le meilleur système de tous ceux que nous avons expérimentés sur cette base de chiffres manuscrits. Par rapport au meilleur

classificateur sur les données brutes (kNN), l'espérance de gain est dans [0,27 %; 0,50 %]. Ce qui signifie, compte tenu des mesures que nous avons faites, qu'il y a une probabilité de 90 % que l'espérance du gain apporté par la PG soit supérieure à 0,27 %. Comme nous parlons de taux d'erreurs se situant autour de 3 %, ce gain est tout de même important, avec une dilution relative du nombre d'erreurs de classification de l'ordre de 10 % grâce à la PG. En pratique, l'intervalle de confiance de ce gain sur une réalisation des deux systèmes est [-0,36 %; 1,13 %].

Si nous ne considérons que les caractéristiques de la PG avec LibSVM sans y adjoindre les données brutes, l'espérance du gain est négative [-1,40 %; -1,08 %] par rapport au kNN sur les données brutes (il s'agit donc plus d'une perte que d'un gain). Cette perte est toutefois accompagnée d'une forte réduction de la dimension. À dimension réduite, la PG a réussi à surpasser l'AG mais pas l'ACP. La PG, même si elle s'en est sortie honorairement, n'est donc pas la meilleure méthode de réduction de dimension pour cette base.

5.3 Application sur la base des céréales

5.3.1 Performance d'une réalisation du système

De la même façon que pour la base des chiffres manuscrits, la *Figure 43* représente les performances des simulations faites sur la base des céréales. Les résultats numériques sont consultables dans le Tableau 18 de l'ANNEXE 6. Pour cette base également ces résultats confirment les observations faites à partir des quartiles. Tout d'abord, nous voyons que les performances des classificateurs après PG sont regroupées, ce qui confirme la robustesse vis-à-vis de ce choix. Hormis LibSVM norm, les intervalles de confiance de tous les classificateurs après PG sont toujours inclus dans [3,02 %; 7,17 %].

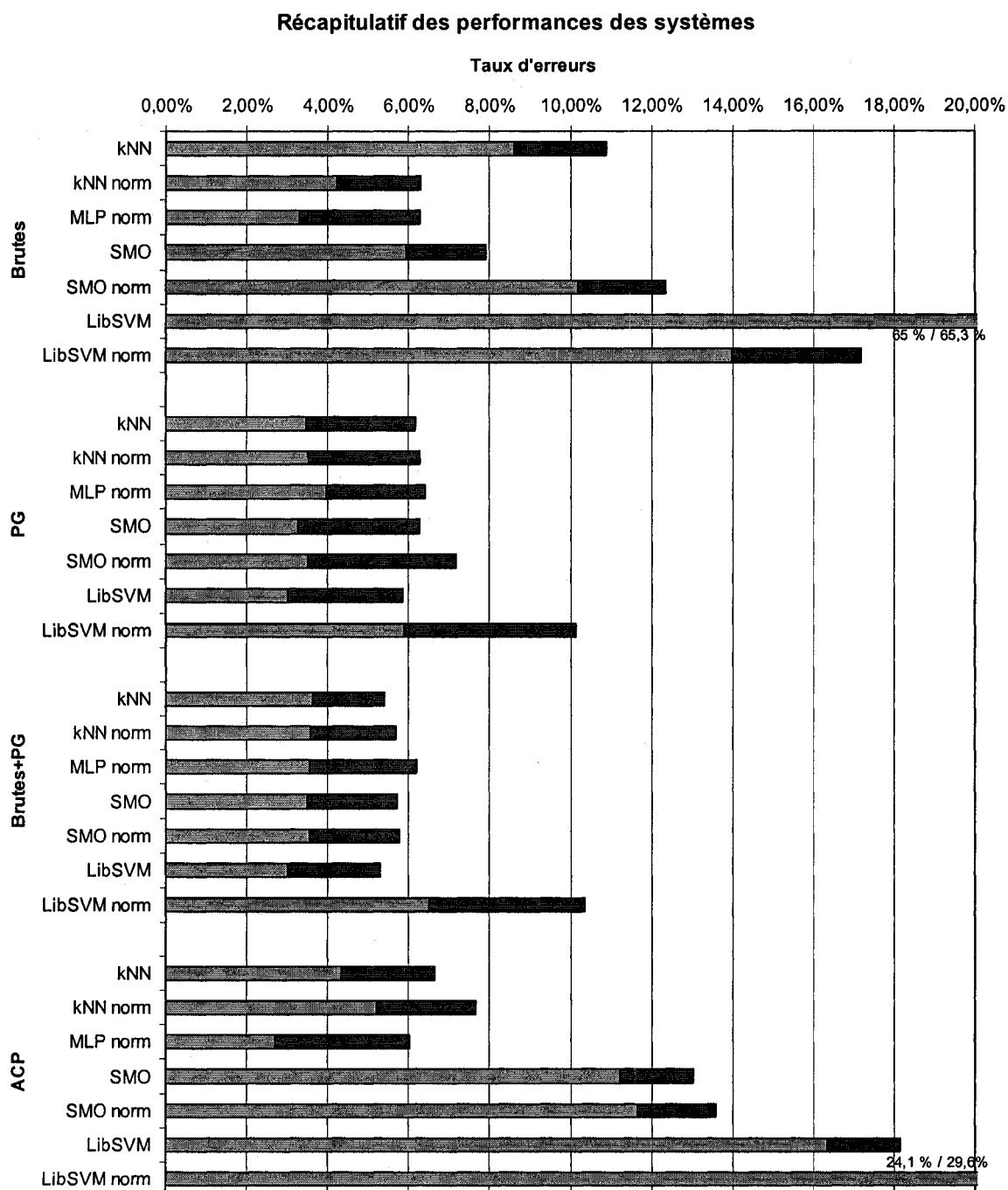


Figure 43 *Intervalle de confiance d'une réalisation de système. Les valeurs représentent les bornes minimum et maximum de ces intervalles*

Nous pouvons faire la même constatation pour la mise en commun des caractéristiques d'origine avec celle de la PG. Les intervalles semblent toutefois légèrement plus resserrés dans ce cas. Quant à l'ACP, les résultats sont très disparates, bien que dans certains cas (MLP norm en particulier) la performance est d'un très bon niveau : [2,70 %; 6,02 %].

5.3.2 Espérance de la performance des systèmes

Pour représenter les intervalles de confiance des espérances de performance des systèmes, nous avons choisi de faire deux graphiques : un qui donne une vue d'ensemble (*Figure 44*) et un autre qui met particulièrement en lumière les différences plus fines parmi les meilleurs systèmes (*Figure 45*). Les résultats numériques sont consultables dans le *Tableau 19* de l'ANNEXE 6.

Sur la vue d'ensemble, nous distinguons très clairement qu'à partir des caractéristiques d'origine ou de celles de l'ACP les performances dépendent grandement du choix du classificateur. En effet, nous retrouvons des taux d'erreurs pouvant aller de 5 % à 65 %, en passant par 10 % ou 15 %. Autant dire qu'il est indispensable de faire les expérimentations adéquates pour faire un bon choix avant de mettre en place un système de classification sur de telles données.

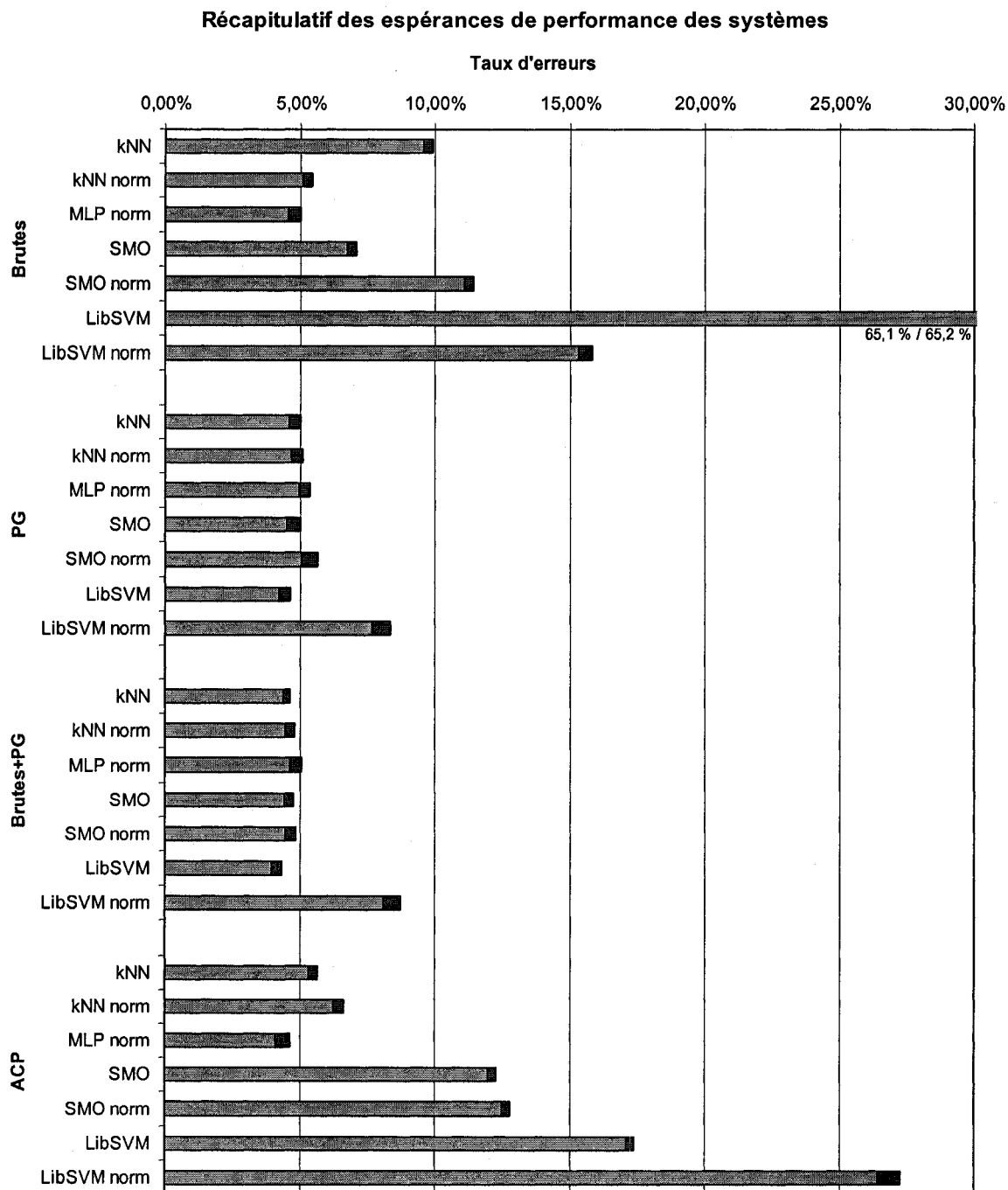


Figure 44 *Intervalles de confiance de l'espérance de la performance des systèmes*

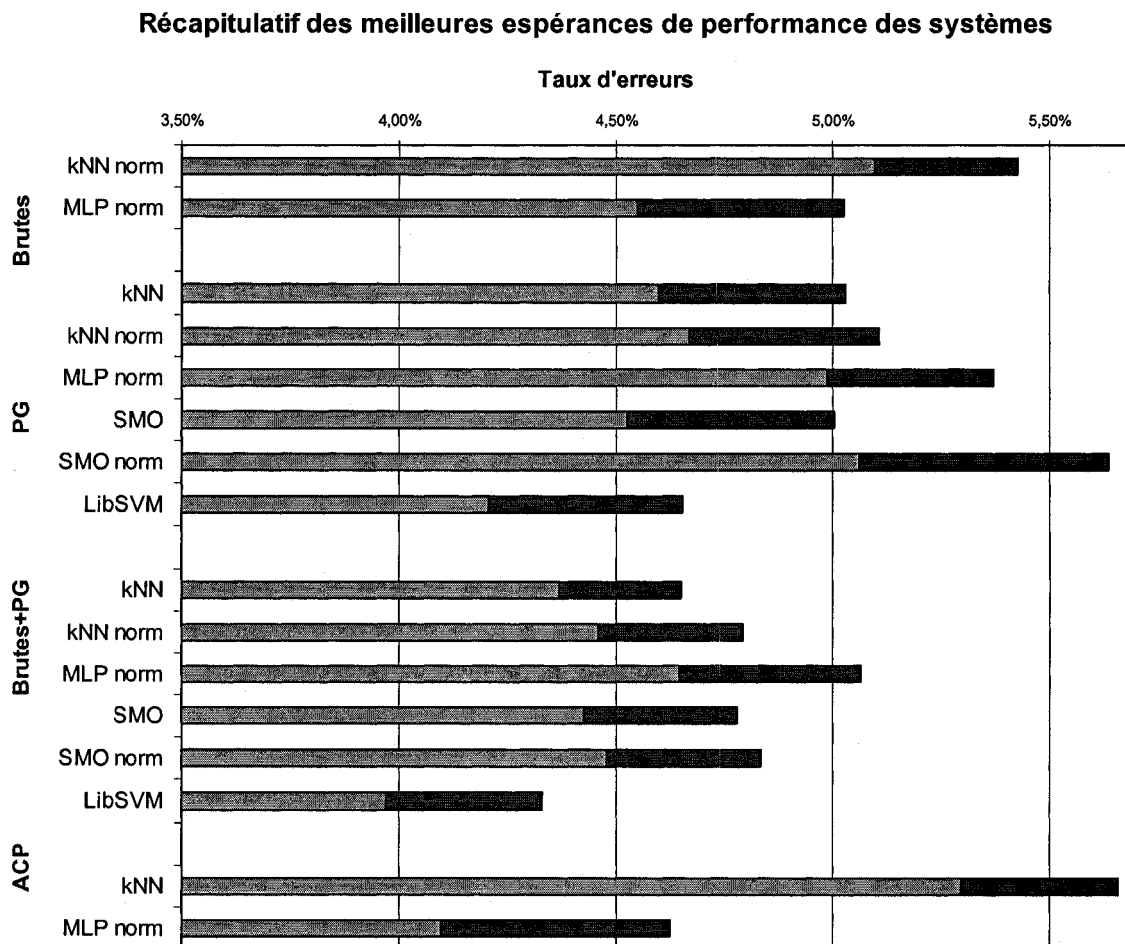


Figure 45 *Intervalles de confiance des meilleures espérances selon une échelle permettant de les départager plus facilement*

Sur la *Figure 45*, nous pouvons plus aisément distinguer les différences entre les classificateurs qui semblent tirer leur épingle du jeu. Nous remarquons qu'avec les données issues de la PG (adjointes ou non aux caractéristiques d'origine), LibSVM semble avoir une avance sur ses concurrents. Toutefois, pour les caractéristiques de la PG seules, son intervalle de confiance recoupe de peu celui de SMO et kNN. Nous essayerons donc de les départager à la section suivante.

Le plus prometteur de tous ces systèmes est le couple PG+caractéristiques d'origine/LibSVM avec un intervalle de confiance [3,97 %; 4,33 %] pour son espérance d'erreurs. Cet intervalle recoupe toutefois ceux de ACP/MLP norm ([4,10 %; 4,62 %]) et PG/LibSVM ([4,21 %; 4,65 %]). Nous devons donc également comparer ces systèmes dans la section suivante.

5.3.3 Comparaisons

Nous voulions déterminer si LibSVM était bien le classificateur le plus performant sur les caractéristiques fournies par la PG. Comparativement à SMO, l'espérance du gain est [0,12 %; 0,54 %] en faveur de LibSVM, et par rapport à kNN, l'intervalle est [0,23 %; 0,54 %], toujours en faveur de LibSVM (*voir* ANNEXE 6). Ces intervalles étant strictement positifs, nous pouvons donc affirmer que c'est bien LibSVM qui est le plus adapté en sortie de la PG.

Enfin, en tant que performance absolue, l'espérance du gain apporté au LibSVM par la juxtaposition des caractéristiques d'origine avec celles de la PG par rapport à celles de la PG seules est sensible : [0,10 %; 0,45 %]. En revanche, nos expérimentations ne permettent pas de départager le couple ACP/MLP norm de libSVM sur l'ensemble PG+caractéristiques d'origine : l'espérance du gain est dans [-0,07 %; 0,49 %]. LibSVM semble avoir un avantage, mais il faudrait davantage de simulations pour trancher.

Enfin, si nous imposons une dimension identique à celle d'origine, ce sont les couples ACP/MLP norm et PG/LibSVM qui mènent. Ces deux systèmes ne pouvant pas être départagés car leurs performances se recoupent largement.

5.4 Discussion

5.4.1 Bilan des comparaisons

L'analyse à l'aide des intervalles de confiance a largement confirmée les observations faites à partir des quartiles. D'une manière générale, nous avons montré que les caractéristiques fournies par la PG étaient peu sensibles au choix du classificateur, ce qui lui confère une certaine robustesse. Cette constatation va bien dans le sens de l'objectif que nous nous étions fixé au début de ce travail : avoir un système de classification qui nécessite le moins possible d'interventions humaines ou d'ajustements lors de son élaboration.

Bien que nous ayons remarqué que le choix du classificateur en aval de la PG ne soit pas primordial, nous avons tout de même montré une supériorité de LibSVM. Cette domination s'est retrouvée à la fois sur la base des chiffres manuscrits et celles des céréales. Il aurait fallu expérimenter sur d'autres bases pour confirmation, mais il serait appréciable que LibSVM soit le choix à faire quel que soit le problème que nous traitons. Ainsi, ce serait un pas de plus vers l'universalité du concept que nous avons exploré.

En ce qui concerne les performances absolues, notre système à base de PG s'en tire assez bien. Pour la base des chiffres manuscrits, le couple PG+Brutes/LibSVM est au coude à coude avec ACP/LibSVM, tous deux en tête des résultats. Tandis que pour la base des céréales, c'est avec ACP/MLP norm que la PG se bat pour la première place, avec semble-t-il un léger avantage en faveur de cette dernière. Ces performances sont donc très encourageantes, et il semble effectivement que la PG aide bien les classificateurs à tracer une frontière de décision plus proche de l'optimale.

Cependant nous avons constaté sur ces deux bases que les classificateurs avaient de plus bas taux d'erreurs quand nous utilisions à la fois les caractéristiques créées par la PG et celles d'origine. Cela traduit peut-être le fait que nous ayons trop limité le nombre d'espèces dans la PG.

5.4.2 Critique des résultats

Comme le soulignait Steven L. Salzberg [47], le domaine de la reconnaissance de forme prend certaines libertés face aux lois de la statistique. Bien que les approches que nous avons adoptées aient été utilisées et acceptées dans la littérature scientifique du domaine [48], cela n'empêche pas de noter les écarts de conduite. Dans un souci d'honnêteté, nous allons donc exposer les points litigieux de notre analyse.

La construction de l'intervalle de confiance tel que nous l'avons faite a nécessité plusieurs hypothèses. Cependant, ces hypothèses n'étant pas toutes pleinement vérifiées, il est important de garder une certaine distance face aux conclusions qui ont été tirées. Une première hypothèse était que la variable aléatoire de la performance d'un système de classification suit une loi normale (*voir* 5.1.2). Une indication que cette hypothèse est vérifiée dans une mesure raisonnable est souvent faite par le test de Fisher, mais ce n'est pas cette hypothèse qui devrait soulever les principales inquiétudes. En effet, le plus problématique est que nous avons également fait l'hypothèse que nos simulations étaient indépendantes. Bien entendu, puisque nous utilisons la même base de données en changeant seulement son découpage, cette hypothèse n'est pas vérifiée. Cependant il aurait été difficile de faire autrement car nos bases avaient une taille limitée, nous n'avions donc pas le choix de la réutiliser pour chaque simulation. Certaines publications tentent alors d'introduire des facteurs pour corriger ce biais [49][50][51].

Un autre écueil auquel notre travail pourrait avoir à faire face est la conséquence de la multiplication des systèmes de classification que nous comparons. En effet, même si des

systemes ont la même espérance de taux d'erreurs, le fait de répéter les comparaisons de ces systemes pourrait aboutir à ce qu'une de ces comparaisons concluent en un écart significatif. Cela est dû à la dispersion statistique des résultats. Par ailleurs, S. Salzberg [47] avertit que cette multiplication peut ne pas être interne à une publication mais la conséquence que plusieurs scientifiques utilisent les mêmes bases de données publiques et ne publient que lorsque qu'ils ont des résultats intéressants. Dans notre cas, bien que nous ayons comparé de nombreux systemes et utilisé une base publique, le risque que nos résultats soient faussement significatifs est fortement réduit par le fait que nous les ayons confirmés sur des bases distinctes.

CONCLUSION

Dans ce mémoire, nous avons étudié l'intérêt de changer l'espace de représentation de façon non linéaire afin de minimiser le taux d'erreurs de classification et de réduire les interventions humaines nécessaires lors de la conception de ce système. Pour cela, nous avons exposé une manière d'exploiter une coévolution sur fond de Programmation Génétique. Nous avons bien constaté que ce type d'algorithme est correctement adapté à ce genre de problème car la nature de la coévolution permet de ne pas imposer une structure et des objectifs rigides aux nouvelles caractéristiques. L'ajustement se fait ainsi « naturellement ».

Parmi les résultats, ce qui semble le plus incontestable c'est que le nouvel espace de représentation est beaucoup moins sensible au choix d'un classificateur, contrairement à ce qui a pu être observé sur d'autres représentations (*voir 5.3.2*). Ce premier point est crucial car notre principal objectif était de concevoir un système qui minimise les interventions humaines lors de sa conception. Une bonne robustesse face à de tels choix évitera alors de longues vérifications ou comparaisons de classificateurs. Toutefois ce fait était davantage évident sur la base des céréales que sur la base des chiffres manuscrits. En effet, sur cette dernière, tous les classificateurs offraient des performances comparables sur les données non transformées, il n'y avait donc pas de grande marge de progression possible pour la PG. En ce qui concerne la base des céréales, le progrès de robustesse vis-à-vis du choix du classificateur est nettement plus visible. Les données brutes de cette base étaient très hétérogènes au départ, il semble donc que la PG soit capable d'en tirer un ensemble plus cohérent de caractéristiques, qui ne soit pas un enfer pour certains classificateurs.

Nous avons mis un point d'honneur à utiliser exactement les mêmes réglages de notre algorithme pour les deux bases de données. Ces paramètres avaient alors été obtenus par une étude sommaire de leur influence avec la base des chiffres manuscrits. Le fait que la

PG ait bien réagi sur la base des céréales va donc bien dans le sens du peu d'interventions humaines nécessaires. Ce point renforce donc l'idée que notre prétraitement à l'aide de PG rend le système de classification plus autonome vis-à-vis de l'homme.

Nous avons également constaté que la PG permettait des taux d'erreurs plus bas que sur les données brutes. Cela confirme l'hypothèse que ce changement d'espace de représentation induit une frontière de classification optimale moins complexe, et donc il est plus facile pour le classificateur de s'en approcher. Par conséquent, les performances des classificateurs peuvent s'améliorer (*voir* 5.2.3 et 5.3.3). Cette différence a été constatée sur les deux bases que nous avons utilisées. Le gain n'était pas énorme dans l'absolu, mais compte tenu de l'ordre de grandeur assez faible des taux d'erreurs de départ, le gain relatif est tout de même intéressant.

Dans tous les cas de figure, il a été remarqué que le meilleur classificateur à prendre après remaniement de l'espace de représentation est LibSVM. Nous retrouvons donc le choix qu'avait fait K. Krawiec sur le même genre d'algorithme.

Néanmoins, comme nous l'avons vu en 5.4.2, la confiance que nous avons placée dans la précision de nos résultats est sans doute un peu trop optimiste. En effet, les simulations que nous avons faites ne sont pas totalement indépendantes. Bien que les hypothèses de calcul que nous ayons faites soient courantes dans le domaine de la classification, il faut garder à l'esprit que certains résultats sont peut-être contestables : une contre étude ne serait donc pas superflue.

En définitive, la principale application de la PG utilisée à des fins de construction de caractéristiques est assurément les problèmes de classification très spécifiques ou isolés, qui ne permettent pas l'adaptation manuelle fine du système. Cependant, les résultats encourageants en tant que performance laissent penser que cette approche pourrait

avantageusement être utilisée comme complément de constructions manuelles de caractéristiques. L'apport humain, quand il est possible, aura toutefois son importance pour les systèmes sur lesquels le moindre ajustement est essentiel.

Au-delà des expériences que nous avons menées, la PG est également intéressante de par le potentiel des améliorations possibles. Par exemple, la PG produit par nature de nombreux candidats par caractéristique que nous avons voulue construire. Dans notre travail, nous n'avons retenu que celui qui semblait la plus prometteur, mais il serait possible d'assembler plusieurs de ses candidats pour former un espace de représentation encore plus performant. Aussi, la multiplicité des candidats se prête bien à la formation d'un « comité d'experts » [52]. Cela pourrait alors notablement pousser vers le haut les taux de bonnes classifications.

Enfin, si les temps de calcul importants auxquels nous avons dû faire face peuvent sembler décourageants pour de plus grosses bases de données, il existe des moyens de contourner cette difficulté. En effet, des techniques de réduction de la masse de calcul pour l'obtention de l'adéquation sont disponibles, comme la « sélection dynamique d'un sous-ensemble d'apprentissage » qui a été testée avec succès pour la PG [53]. Notre système de construction de caractéristiques à l'aide de coévolution de PG est donc envisageable pour des problèmes de tailles nettement plus grandes que ceux que nous avons traités.

RECOMMANDATIONS

Comme nous l'avons souligné dans la conclusion, le traitement du sujet de la transformation de l'espace de représentation par la PG pourrait être continué. Suite à l'expérience que nous avons acquise, nous pouvons conseiller ces quelques voies de recherche, non exhaustives bien entendues :

- introduire la notion de comité d'experts afin de tirer un maximum du travail de la PG;
- envisager l'utilisation de « sélection dynamique d'un sous-ensemble d'apprentissage » (de l'expression anglaise « Dynamic Training Subset Selection »);
- diversifier les genres de bases utilisées afin de vérifier la robustesse du système;
- mesurer de façon plus précise et rigoureuse les intervalles de confiance des gains apportés par le système. Nous pourrions envisager l'utilisation de la validation croisée de type « 5x2 cv F test » d'Ethem Alpaydin [54].

ANNEXE 1

SIMULATIONS SUR LA BASE DES CHIFFRES MANUSCRITS

Tableau 3

Taux d'erreurs du kNN sur différentes représentations de la base des chiffres manuscrits

kNN										
Données	Brutes		ACP		AG		PG		Brutes+PG	
Norm.	Non	Oui	Non	Oui	Non	Oui	Non	Oui	Non	Oui
n°1	3,7%	3,7%	3,0%	3,7%	5,6%	5,8%	4,8%	5,3%	3,9%	3,7%
n°2	2,6%	2,9%	2,7%	3,7%	5,9%	5,1%	5,2%	5,3%	3,0%	2,6%
n°3	2,7%	2,5%	2,8%	3,3%	6,1%	5,3%	5,4%	5,8%	2,8%	2,8%
n°4	3,1%	3,6%	3,6%	4,2%	5,4%	5,2%	6,7%	6,7%	3,3%	3,5%
n°5	3,2%	3,5%	3,4%	3,4%	5,9%	5,2%	5,1%	5,3%	3,4%	3,4%
n°6	2,6%	2,7%	2,6%	4,0%	5,3%	5,4%	4,6%	5,5%	3,7%	2,9%
n°7	3,2%	2,5%	2,3%	3,3%	6,1%	6,4%	4,2%	4,5%	2,5%	2,5%
n°8	3,5%	3,1%	3,5%	4,3%	5,7%	5,5%	5,5%	5,9%	3,9%	3,9%
n°9	3,9%	3,3%	3,4%	4,5%	6,4%	5,9%	4,7%	5,5%	3,1%	3,3%
n°10	3,2%	3,1%	3,0%	4,4%	5,8%	5,9%	4,6%	5,2%	2,7%	2,7%
n°11	3,4%	3,6%	3,3%	4,0%	5,7%	5,9%	4,9%	5,5%	3,3%	4,1%
n°12	3,4%	3,5%	3,1%	3,7%	6,1%	6,0%	4,2%	4,3%	3,0%	3,2%
n°13	3,9%	3,8%	3,7%	4,9%	6,0%	5,5%	4,6%	5,4%	3,6%	3,8%
n°14	3,8%	3,9%	3,6%	4,1%	6,0%	5,6%	5,2%	4,7%	3,6%	3,8%
n°15	2,9%	3,2%	3,1%	4,3%	5,7%	4,7%	5,1%	5,0%	3,1%	3,0%
n°16	3,0%	3,1%	3,3%	3,4%	6,1%	5,2%	4,8%	5,3%	3,2%	3,3%
n°17	3,7%	3,6%	3,7%	3,8%	6,1%	5,8%	5,1%	4,9%	3,9%	3,8%
n°18	3,2%	3,4%	3,3%	4,4%	6,2%	5,8%	4,9%	5,1%	3,1%	3,5%
n°19	3,1%	3,5%	2,9%	4,0%	5,4%	4,9%	5,5%	6,0%	2,4%	3,4%
n°20	4,4%	3,0%	4,0%	4,9%	7,1%	6,1%	6,0%	6,5%	3,7%	4,1%
n°21	3,3%	3,5%	2,9%	3,6%	5,8%	5,7%	6,6%	6,7%	3,0%	3,5%
n°22	2,7%	3,1%	3,0%	3,5%	4,7%	4,2%	4,0%	4,1%	3,2%	2,7%
n°23	3,1%	2,9%	3,2%	3,8%	5,6%	4,5%	5,5%	4,8%	2,8%	2,7%
n°24	3,8%	3,5%	3,4%	4,8%	6,0%	5,5%	5,5%	5,9%	3,7%	3,4%
n°25	3,2%	3,4%	3,7%	3,5%	6,3%	6,0%	5,9%	6,4%	3,3%	3,4%
n°26	2,4%	2,5%	1,7%	3,5%	5,7%	5,4%	4,2%	4,9%	2,5%	2,5%
n°27	3,5%	3,4%	3,2%	3,2%	5,5%	5,8%	4,0%	4,1%	2,7%	2,9%
n°28	3,6%	3,3%	3,1%	3,6%	5,5%	5,0%	3,6%	4,0%	2,9%	2,6%
n°29	3,4%	3,3%	3,3%	3,3%	6,2%	6,7%	5,3%	5,7%	3,4%	3,0%
n°30	2,6%	2,5%	2,9%	4,0%	5,8%	5,9%	4,2%	5,1%	2,6%	3,2%
n°31	4,5%	4,4%	3,9%	4,3%	6,5%	5,6%	4,8%	5,3%	4,1%	3,9%
n°32	2,6%	2,8%	2,7%	3,5%	6,0%	6,1%	3,9%	4,4%	2,6%	3,0%
n°33	3,4%	2,9%	3,0%	3,4%	5,4%	4,8%	4,9%	4,7%	2,5%	2,8%
n°34	3,3%	3,5%	3,1%	4,3%	6,2%	5,7%	4,8%	5,6%	2,9%	3,3%
n°35	2,8%	3,2%	3,2%	3,7%	5,2%	4,8%	5,5%	6,1%	3,3%	3,2%
n°36	3,6%	3,5%	3,7%	4,5%	6,2%	6,0%	5,0%	5,6%	3,6%	3,9%
n°37	2,9%	3,2%	3,1%	4,3%	5,7%	4,7%	4,4%	4,0%	2,9%	3,5%
n°38	3,0%	3,1%	3,3%	3,4%	6,1%	5,2%	6,6%	7,0%	3,3%	3,4%
n°39	3,1%	3,5%	2,9%	4,0%	5,4%	4,9%	4,4%	5,1%	2,0%	2,1%
n°40	4,4%	3,0%	4,0%	4,9%	7,1%	6,1%	6,3%	7,7%	4,4%	3,9%
Moyenne	3,29%	3,25%	3,19%	3,94%	5,89%	5,50%	5,01%	5,37%	3,17%	3,26%
Écart-type	0,51%	0,41%	0,45%	0,49%	0,46%	0,55%	0,76%	0,84%	0,52%	0,50%

Tableau 4

Taux d'erreurs du MLP sur différentes représentations de la base des chiffres manuscrits

MLP					
Données	Brutes	ACP	AG	PG	Brutes+PG
<i>Norm.</i>	<i>Oui</i>	<i>Oui</i>	<i>Oui</i>	<i>Oui</i>	<i>Oui</i>
n°1	3,7%	4,6%	6,9%	6,0%	3,7%
n°2	3,5%	4,3%	7,1%	6,7%	3,7%
n°3	3,3%	3,8%	6,0%	7,6%	3,3%
n°4	4,8%	5,4%	7,7%	7,0%	4,7%
n°5	3,9%	5,5%	7,0%	6,7%	3,5%
n°6	3,9%	4,6%	6,0%	5,0%	4,0%
n°7	3,1%	4,3%	7,5%	5,5%	2,8%
n°8	4,6%	5,6%	7,2%	7,0%	4,6%
n°9	3,4%	4,5%	6,2%	4,7%	3,4%
n°10	3,4%	3,9%	6,7%	5,3%	2,6%
n°11	4,1%	5,6%	8,0%	6,6%	4,6%
n°12	4,0%	4,5%	6,7%	7,1%	3,7%
n°13	4,2%	5,6%	6,8%	7,0%	4,3%
n°14	3,5%	4,2%	7,8%	6,5%	3,2%
n°15	3,9%	4,5%	7,3%	6,1%	3,0%
n°16	4,4%	4,5%	5,5%	6,6%	4,2%
n°17	4,3%	4,6%	7,2%	6,9%	4,0%
n°18	3,5%	4,2%	6,2%	6,4%	3,8%
n°19	4,0%	5,2%	6,5%	5,4%	4,4%
n°20	4,7%	5,2%	7,8%	7,3%	5,0%
n°21	4,4%	4,5%	7,2%	6,9%	4,1%
n°22	3,6%	4,3%	6,3%	4,9%	3,7%
n°23	3,4%	4,0%	6,0%	5,9%	3,2%
n°24	4,2%	5,6%	6,5%	6,9%	4,0%
n°25	4,2%	4,6%	6,7%	6,8%	4,4%
n°26	3,0%	4,5%	6,4%	5,9%	2,9%
n°27	3,5%	4,3%	6,2%	5,2%	3,6%
n°28	3,9%	4,6%	6,8%	7,2%	3,6%
n°29	3,3%	4,1%	7,5%	5,7%	3,5%
n°30	3,3%	4,9%	7,8%	6,3%	3,7%
n°31	4,7%	4,8%	7,8%	6,6%	4,1%
n°32	3,5%	3,9%	5,8%	6,6%	3,3%
n°33	3,5%	4,8%	5,9%	6,7%	3,5%
n°34	4,5%	4,8%	7,9%	7,4%	4,2%
n°35	3,2%	4,5%	7,3%	7,7%	3,9%
n°36	3,7%	4,7%	7,9%	5,7%	3,8%
n°37	3,9%	4,5%	7,3%	6,0%	3,5%
n°38	4,4%	4,5%	5,5%	7,9%	4,1%
n°39	4,0%	5,2%	6,5%	5,5%	4,1%
n°40	4,7%	5,2%	7,8%	7,8%	4,6%
<i>Moyenne</i>	<i>3,88%</i>	<i>4,67%</i>	<i>6,88%</i>	<i>6,43%</i>	<i>3,81%</i>
<i>Écart-type</i>	<i>0,50%</i>	<i>0,50%</i>	<i>0,73%</i>	<i>0,83%</i>	<i>0,55%</i>

Tableau 5

Taux d'erreurs du SMO sur différentes représentations de la base des chiffres manuscrits

SMO										
Données	Brutes		ACP		AG		PG		Brutes+PG	
Norm.	Non	Oui	Non	Oui	Non	Oui	Non	Oui	Non	Oui
n°1	4,1%	4,1%	4,9%	3,6%	9,0%	5,4%	7,1%	5,8%	4,4%	4,0%
n°2	3,6%	3,6%	4,5%	4,0%	6,8%	6,2%	6,0%	6,4%	3,8%	3,8%
n°3	3,2%	3,0%	3,9%	2,7%	5,9%	5,3%	7,2%	5,8%	3,2%	3,0%
n°4	4,7%	4,7%	5,7%	4,1%	8,9%	7,1%	7,8%	6,4%	4,6%	4,5%
n°5	3,8%	4,0%	4,3%	3,0%	7,5%	5,9%	6,9%	6,0%	4,5%	4,2%
n°6	4,2%	4,1%	4,2%	4,4%	7,7%	6,6%	7,4%	5,9%	4,0%	4,2%
n°7	3,4%	3,2%	3,6%	3,0%	7,4%	5,5%	7,3%	4,7%	3,4%	3,4%
n°8	5,0%	4,6%	5,0%	4,6%	6,0%	4,7%	7,6%	5,6%	4,7%	4,6%
n°9	4,1%	4,3%	4,9%	3,3%	8,0%	6,2%	6,5%	5,4%	3,7%	3,9%
n°10	4,0%	3,8%	4,6%	2,9%	7,9%	5,9%	5,5%	5,1%	3,1%	3,0%
n°11	4,3%	4,3%	5,0%	4,1%	7,3%	6,2%	7,0%	6,1%	4,6%	4,5%
n°12	4,3%	4,0%	4,2%	3,4%	7,1%	6,1%	6,9%	5,0%	4,2%	4,1%
n°13	4,4%	4,3%	4,4%	4,3%	7,3%	5,8%	7,8%	6,7%	4,5%	4,2%
n°14	3,8%	3,9%	4,3%	4,1%	8,4%	5,9%	7,3%	6,2%	3,9%	3,8%
n°15	4,5%	3,9%	4,6%	2,7%	8,4%	6,1%	6,8%	5,0%	4,3%	4,0%
n°16	3,7%	3,8%	4,2%	3,8%	7,5%	5,8%	7,2%	6,2%	4,5%	3,7%
n°17	4,7%	4,9%	5,4%	4,0%	8,9%	6,6%	7,7%	5,7%	4,5%	4,3%
n°18	4,0%	3,4%	4,3%	3,3%	7,7%	5,8%	6,8%	4,8%	3,3%	3,3%
n°19	4,5%	3,9%	4,5%	3,1%	8,9%	6,3%	6,5%	6,7%	3,8%	4,3%
n°20	4,8%	4,9%	4,6%	3,9%	7,6%	6,9%	7,6%	7,0%	5,1%	5,1%
n°21	4,5%	4,1%	5,0%	3,6%	7,8%	6,4%	8,7%	5,9%	4,2%	4,5%
n°22	3,7%	3,2%	3,9%	3,6%	7,8%	4,7%	5,0%	4,6%	3,5%	3,3%
n°23	4,1%	4,2%	4,2%	3,6%	8,0%	6,1%	7,6%	5,8%	4,6%	4,1%
n°24	5,0%	4,8%	4,7%	4,1%	8,3%	6,5%	7,6%	7,4%	3,9%	4,1%
n°25	4,6%	4,2%	5,1%	3,7%	8,5%	5,4%	7,8%	7,5%	3,7%	3,8%
n°26	3,5%	3,0%	4,0%	3,5%	7,4%	5,2%	6,4%	4,6%	3,1%	2,9%
n°27	4,9%	4,6%	5,4%	3,7%	8,3%	5,8%	6,0%	5,1%	4,4%	4,2%
n°28	4,1%	3,9%	4,5%	3,5%	6,5%	5,3%	7,0%	6,1%	4,0%	4,0%
n°29	3,9%	3,7%	3,7%	3,4%	8,2%	5,6%	6,6%	6,0%	3,7%	3,6%
n°30	3,8%	3,9%	4,5%	3,8%	7,8%	5,9%	6,9%	6,4%	3,6%	3,9%
n°31	4,8%	4,3%	4,6%	3,8%	7,0%	6,4%	7,1%	6,0%	4,0%	4,2%
n°32	4,5%	4,0%	3,9%	2,8%	8,4%	4,7%	6,0%	4,7%	3,8%	3,8%
n°33	4,5%	4,5%	4,4%	3,6%	9,1%	4,9%	6,5%	5,8%	4,2%	4,6%
n°34	4,5%	4,3%	5,0%	5,0%	8,2%	6,6%	7,7%	7,0%	4,5%	4,2%
n°35	3,9%	4,2%	4,2%	3,8%	9,1%	6,0%	6,4%	6,0%	4,0%	4,0%
n°36	4,0%	3,9%	4,2%	3,5%	7,4%	5,9%	6,5%	5,6%	3,9%	3,5%
n°37	4,5%	3,9%	4,6%	2,7%	8,4%	6,1%	6,2%	5,1%	3,6%	3,9%
n°38	3,7%	3,8%	4,2%	3,8%	7,5%	5,8%	8,6%	6,8%	4,5%	4,2%
n°39	4,5%	3,9%	4,5%	3,1%	8,9%	6,3%	6,8%	5,9%	4,1%	4,4%
n°40	4,8%	4,9%	4,6%	3,9%	7,6%	6,9%	7,5%	6,6%	4,8%	5,0%
Moyenne	4,22%	4,05%	4,51%	3,62%	7,86%	5,92%	7,00%	5,89%	4,06%	4,00%
Écart-type	0,46%	0,48%	0,46%	0,53%	0,79%	0,60%	0,76%	0,76%	0,49%	0,50%

Tableau 6

Taux d'erreurs du LibSVM sur différentes représentations de la base des chiffres manuscrits

LibSVM										
Données	Brutes		ACP		AG		PG		Brutes+PG	
Norm.	Non	Oui	Non	Oui	Non	Oui	Non	Oui	Non	Oui
n°1	6,9%	3,9%	3,0%	6,0%	8,0%	5,3%	3,5%	5,6%	2,7%	3,9%
n°2	7,8%	3,9%	3,2%	4,6%	10,4%	6,2%	4,5%	6,1%	3,2%	4,1%
n°3	8,7%	3,3%	2,0%	5,5%	10,6%	5,4%	4,8%	7,1%	2,3%	3,9%
n°4	8,4%	4,9%	4,1%	5,5%	11,7%	6,9%	5,5%	6,6%	3,9%	4,4%
n°5	8,4%	3,6%	2,6%	4,5%	10,8%	5,5%	4,8%	6,3%	2,8%	3,6%
n°6	7,9%	4,4%	3,3%	6,0%	9,8%	5,4%	4,8%	5,7%	3,0%	4,2%
n°7	7,2%	3,2%	2,4%	5,3%	11,5%	5,5%	3,7%	4,8%	2,2%	3,4%
n°8	9,2%	4,1%	3,3%	5,9%	9,2%	5,6%	4,7%	6,4%	3,2%	4,1%
n°9	9,3%	4,1%	3,5%	5,1%	12,2%	5,4%	4,9%	5,4%	3,6%	3,9%
n°10	8,1%	4,0%	2,7%	5,1%	7,3%	5,4%	3,0%	5,2%	2,4%	3,8%
n°11	10,0%	4,6%	2,9%	6,6%	10,7%	5,1%	4,5%	6,3%	2,9%	4,4%
n°12	8,7%	3,7%	2,7%	4,8%	9,2%	4,7%	4,4%	5,1%	2,7%	3,3%
n°13	7,0%	3,9%	2,9%	6,5%	8,8%	5,1%	4,2%	5,9%	2,8%	3,5%
n°14	10,7%	3,7%	2,8%	4,6%	11,6%	6,2%	4,2%	5,9%	2,9%	3,7%
n°15	8,9%	3,0%	2,2%	4,7%	11,0%	5,2%	4,1%	5,1%	2,1%	2,7%
n°16	7,8%	4,3%	3,1%	6,4%	8,9%	5,9%	4,2%	5,8%	3,2%	4,3%
n°17	6,2%	4,3%	3,2%	4,9%	7,4%	6,2%	4,5%	5,8%	3,3%	3,9%
n°18	11,1%	3,5%	2,7%	5,4%	12,5%	5,3%	4,2%	5,4%	2,8%	3,7%
n°19	11,0%	3,4%	2,5%	5,5%	14,5%	4,7%	4,4%	6,1%	2,2%	3,7%
n°20	8,3%	4,6%	3,8%	6,0%	8,2%	6,8%	5,9%	7,3%	4,2%	4,7%
n°21	9,9%	3,2%	3,0%	5,4%	11,7%	5,6%	5,3%	5,7%	2,6%	3,5%
n°22	7,3%	3,4%	2,7%	4,7%	9,2%	5,3%	3,4%	4,9%	2,7%	3,2%
n°23	7,6%	3,1%	2,8%	5,1%	11,2%	5,8%	4,1%	6,0%	2,6%	3,3%
n°24	9,7%	4,0%	3,2%	6,3%	8,4%	5,6%	4,7%	7,8%	2,5%	4,4%
n°25	8,4%	4,1%	3,3%	5,7%	10,8%	6,0%	5,4%	7,9%	3,1%	4,2%
n°26	6,8%	3,7%	2,6%	5,3%	10,5%	5,7%	4,3%	4,9%	2,0%	3,2%
n°27	8,4%	3,8%	3,4%	5,0%	9,6%	6,3%	3,3%	5,6%	2,9%	3,8%
n°28	9,1%	3,8%	3,1%	5,7%	11,0%	6,0%	4,5%	5,6%	2,8%	3,8%
n°29	8,9%	4,0%	3,4%	5,3%	10,0%	5,9%	4,5%	5,8%	3,2%	3,4%
n°30	7,7%	3,5%	2,6%	5,0%	10,6%	5,3%	4,7%	5,1%	2,4%	3,6%
n°31	10,9%	4,3%	3,8%	5,4%	11,2%	6,0%	5,3%	6,5%	4,0%	3,9%
n°32	7,4%	3,2%	2,3%	4,7%	9,4%	5,7%	4,0%	4,9%	2,1%	3,1%
n°33	8,0%	4,0%	2,9%	5,0%	10,2%	5,1%	4,5%	6,1%	2,8%	3,6%
n°34	8,4%	4,8%	3,6%	6,4%	11,5%	6,5%	5,4%	6,4%	3,6%	4,9%
n°35	10,2%	3,4%	2,4%	5,8%	12,4%	4,6%	4,9%	6,6%	2,6%	3,5%
n°36	7,6%	4,1%	3,2%	5,7%	9,1%	6,1%	4,7%	4,9%	3,1%	3,8%
n°37	8,9%	3,0%	2,2%	4,7%	11,0%	5,2%	3,5%	4,9%	2,0%	3,1%
n°38	7,8%	4,3%	3,1%	6,4%	8,9%	5,9%	5,3%	7,4%	3,6%	4,6%
n°39	11,0%	3,4%	2,5%	5,5%	14,5%	4,7%	3,4%	5,9%	2,1%	3,6%
n°40	8,3%	4,6%	3,8%	6,0%	8,2%	6,8%	5,6%	6,9%	3,6%	4,9%
Moyenne	8,60%	3,85%	2,97%	5,45%	10,3%	5,65%	4,49%	5,94%	2,87%	3,82%
Écart-type	1,26%	0,50%	0,49%	0,60%	1,67%	0,58%	0,68%	0,82%	0,56%	0,50%

ANNEXE 2

SIMULATIONS SUR LA BASE DES CÉRÉALES

Tableau 7

Taux d'erreurs du kNN sur différentes représentations de la base des céréales

kNN								
Données	Brutes		ACP		PG		Brutes+PG	
Norm.	Non	Oui	Non	Oui	Non	Oui	Non	Oui
n°1	7,8%	4,0%	5,7%	5,2%	3,5%	3,7%	4,2%	3,8%
n°2	9,2%	7,1%	7,3%	8,9%	6,3%	5,9%	5,6%	6,1%
n°3	9,2%	5,9%	7,1%	7,1%	3,5%	3,8%	4,3%	4,5%
n°4	8,9%	4,5%	4,2%	5,7%	4,0%	3,8%	4,5%	4,2%
n°5	9,2%	5,7%	5,9%	7,5%	4,3%	4,9%	3,1%	4,2%
n°6	10,1%	5,2%	5,4%	6,6%	5,2%	4,9%	4,3%	4,3%
n°7	10,3%	5,4%	4,9%	6,8%	6,6%	7,1%	5,4%	5,9%
n°8	9,2%	4,9%	5,2%	7,0%	5,0%	5,0%	4,3%	3,8%
n°9	11,5%	6,8%	6,4%	6,6%	5,2%	5,2%	5,0%	4,9%
n°10	11,0%	5,0%	5,0%	7,0%	5,7%	7,1%	5,6%	5,4%
n°11	9,9%	5,2%	5,7%	5,9%	5,4%	5,2%	5,2%	5,4%
n°12	10,1%	5,2%	4,9%	5,7%	4,7%	4,7%	4,5%	4,7%
n°13	9,7%	5,4%	5,4%	7,3%	4,7%	4,9%	4,9%	5,0%
n°14	9,4%	4,2%	4,3%	4,9%	3,7%	4,0%	3,7%	4,5%
n°15	10,6%	5,9%	6,1%	6,3%	4,0%	4,2%	3,3%	3,1%
n°16	10,6%	4,5%	5,6%	6,1%	4,2%	4,0%	3,7%	3,8%
n°17	9,7%	5,4%	5,6%	6,3%	3,5%	3,7%	4,2%	3,5%
n°18	9,6%	4,9%	7,0%	6,3%	4,2%	4,2%	4,5%	4,3%
n°19	9,9%	4,5%	4,5%	5,4%	5,2%	4,7%	4,7%	4,7%
n°20	9,0%	4,7%	5,0%	4,7%	5,0%	4,7%	4,3%	4,3%
n°21	10,4%	5,6%	5,7%	7,0%	3,5%	3,5%	3,8%	3,5%
n°22	9,6%	5,0%	5,0%	6,1%	6,8%	6,6%	4,9%	5,9%
n°23	8,3%	4,7%	5,2%	5,9%	4,0%	4,2%	4,0%	4,2%
n°24	10,6%	6,1%	6,1%	5,9%	4,9%	4,9%	4,5%	5,4%
n°25	10,8%	5,2%	5,2%	5,9%	4,7%	4,7%	4,3%	4,5%
n°26	9,2%	4,2%	4,3%	5,0%	4,2%	4,2%	5,0%	4,7%
n°27	11,0%	5,2%	4,2%	5,4%	4,5%	4,3%	4,3%	4,7%
n°28	9,4%	5,7%	5,0%	7,5%	4,5%	4,7%	4,5%	4,5%
n°29	9,4%	4,2%	4,2%	6,1%	4,7%	4,5%	4,7%	4,9%
n°30	9,7%	5,4%	5,6%	6,6%	7,1%	7,1%	5,6%	6,3%
n°31	7,8%	4,0%	5,7%	5,2%	3,3%	3,3%	3,7%	3,8%
n°32	9,2%	7,1%	7,3%	8,9%	5,9%	5,9%	4,2%	4,7%
n°33	9,2%	5,9%	7,1%	7,1%	5,0%	5,2%	5,0%	5,6%
n°34	8,9%	4,5%	4,2%	5,7%	5,0%	5,0%	4,3%	4,3%
n°35	9,2%	5,7%	5,9%	7,5%	4,5%	4,5%	4,5%	4,5%
n°36	10,1%	5,2%	5,4%	6,6%	4,7%	4,5%	4,0%	4,3%
n°37	10,3%	5,4%	4,9%	6,8%	4,0%	4,9%	3,7%	3,5%
n°38	9,2%	4,9%	5,2%	7,0%	3,7%	4,0%	4,0%	3,5%
n°39	11,5%	6,8%	6,4%	6,6%	7,0%	7,1%	6,3%	5,9%
n°40	11,0%	5,0%	5,0%	7,0%	6,6%	6,6%	5,6%	5,7%
Moyenne	9,74%	5,26%	5,48%	6,42%	4,81%	4,89%	4,51%	4,63%
Écart-type	0,87%	0,79%	0,88%	0,94%	1,04%	1,06%	0,68%	0,80%

Tableau 8

Taux d'erreurs du MLP sur différentes représentations de la base des céréales

MLP				
Données	Brutes	ACP	PG	Brutes+PG
<i>Norm.</i>	<i>Oui</i>	<i>Oui</i>	<i>Oui</i>	<i>Oui</i>
n°1	5,2%	4,3%	3,7%	3,1%
n°2	5,0%	7,3%	5,9%	5,2%
n°3	6,4%	4,0%	4,7%	3,8%
n°4	4,2%	4,9%	4,9%	6,1%
n°5	4,0%	3,8%	4,2%	3,7%
n°6	3,8%	3,8%	5,2%	4,7%
n°7	3,8%	4,3%	7,5%	5,7%
n°8	2,8%	3,8%	4,9%	5,2%
n°9	5,6%	3,3%	5,6%	5,2%
n°10	3,8%	5,7%	6,6%	7,3%
n°11	5,2%	4,7%	5,4%	4,7%
n°12	5,0%	4,3%	5,0%	4,3%
n°13	5,2%	5,2%	6,1%	5,0%
n°14	3,1%	3,7%	4,0%	4,5%
n°15	5,0%	4,9%	5,0%	4,9%
n°16	4,2%	4,5%	4,7%	4,0%
n°17	6,3%	6,3%	4,3%	4,5%
n°18	4,7%	3,7%	5,4%	4,7%
n°19	8,3%	0,0%	5,0%	4,9%
n°20	3,8%	2,8%	5,6%	5,2%
n°21	4,0%	3,5%	4,2%	4,2%
n°22	5,6%	5,7%	5,9%	6,3%
n°23	4,7%	3,8%	4,5%	3,7%
n°24	4,3%	4,3%	4,0%	4,9%
n°25	7,1%	5,6%	5,6%	5,2%
n°26	4,9%	3,0%	3,8%	4,7%
n°27	4,0%	3,1%	4,7%	5,4%
n°28	5,7%	4,2%	4,9%	4,0%
n°29	5,0%	5,9%	5,0%	4,5%
n°30	5,7%	4,5%	7,1%	7,1%
n°31	5,2%	4,3%	3,5%	3,3%
n°32	5,0%	7,3%	5,7%	5,6%
n°33	6,4%	4,0%	5,6%	5,2%
n°34	4,2%	4,9%	4,7%	4,9%
n°35	4,0%	3,8%	4,7%	4,0%
n°36	3,8%	3,8%	5,4%	4,9%
n°37	3,8%	4,3%	5,4%	3,8%
n°38	2,8%	3,8%	5,4%	3,0%
n°39	5,6%	3,3%	7,5%	7,0%
n°40	3,8%	5,7%	5,9%	5,9%
Moyenne	4,79%	4,36%	5,18%	4,86%
Écart-type	1,15%	1,28%	0,94%	1,01%

Tableau 9

Taux d'erreurs du SMO sur différentes représentations de la base des céréales

SMO								
Données	Brutes		ACP		PG		Brutes+PG	
Norm.	Non	Oui	Non	Oui	Non	Oui	Non	Oui
n°1	5,6%	10,3%	12,3%	12,2%	3,3%	4,0%	3,1%	3,1%
n°2	7,8%	11,7%	12,0%	13,0%	5,4%	5,4%	5,4%	5,4%
n°3	7,5%	11,7%	13,6%	12,3%	4,3%	7,1%	4,2%	5,0%
n°4	6,4%	11,8%	12,0%	13,0%	4,5%	4,9%	4,9%	4,0%
n°5	5,7%	10,6%	11,7%	11,8%	3,5%	4,9%	4,0%	3,5%
n°6	7,1%	12,3%	12,5%	13,7%	5,2%	5,0%	5,2%	5,0%
n°7	7,8%	11,3%	11,7%	12,0%	5,9%	6,1%	5,7%	5,2%
n°8	6,1%	11,1%	12,0%	12,7%	4,9%	5,2%	4,7%	3,7%
n°9	7,0%	11,5%	11,5%	12,3%	6,4%	6,1%	4,5%	5,4%
n°10	7,0%	11,0%	12,2%	12,2%	5,7%	5,6%	5,7%	5,6%
n°11	7,0%	10,8%	12,2%	13,7%	6,3%	5,9%	4,5%	4,7%
n°12	7,1%	11,3%	12,0%	11,8%	4,3%	4,0%	4,5%	4,2%
n°13	8,2%	12,5%	12,7%	13,6%	5,6%	6,1%	6,1%	5,6%
n°14	6,4%	9,9%	11,7%	12,7%	2,8%	2,8%	3,0%	3,0%
n°15	7,0%	12,3%	12,3%	12,9%	5,4%	5,9%	4,5%	5,0%
n°16	7,7%	11,1%	12,3%	12,9%	1,6%	2,4%	4,9%	4,7%
n°17	7,1%	11,8%	13,0%	12,9%	4,3%	7,5%	3,8%	4,2%
n°18	7,5%	10,4%	11,3%	13,9%	5,2%	7,8%	4,9%	5,2%
n°19	6,6%	12,9%	13,0%	14,3%	4,5%	4,9%	4,5%	4,9%
n°20	5,7%	9,4%	10,6%	11,7%	4,2%	4,9%	3,8%	4,0%
n°21	7,7%	11,7%	11,7%	12,5%	4,0%	4,2%	4,5%	3,7%
n°22	6,8%	11,1%	12,3%	12,7%	5,6%	6,3%	5,2%	6,1%
n°23	7,7%	10,3%	11,5%	11,3%	3,1%	3,8%	3,7%	3,5%
n°24	6,6%	12,2%	12,3%	12,9%	3,7%	5,7%	3,5%	4,3%
n°25	8,5%	11,8%	13,9%	13,9%	4,7%	4,9%	4,9%	4,7%
n°26	6,3%	10,6%	10,8%	11,0%	3,5%	9,2%	3,3%	5,0%
n°27	6,8%	9,7%	11,5%	12,3%	4,9%	5,2%	5,0%	4,5%
n°28	6,8%	12,3%	12,7%	12,5%	5,2%	7,1%	4,3%	4,0%
n°29	7,8%	10,1%	12,3%	12,0%	4,3%	5,2%	4,9%	4,7%
n°30	5,9%	11,8%	11,8%	12,7%	6,8%	6,4%	6,3%	5,4%
n°31	5,6%	10,3%	12,3%	12,2%	3,5%	2,6%	3,0%	3,1%
n°32	7,8%	11,7%	12,0%	13,0%	6,3%	5,9%	5,4%	6,3%
n°33	7,5%	11,7%	13,6%	12,3%	4,9%	4,7%	4,5%	4,7%
n°34	6,4%	11,8%	12,0%	13,0%	4,7%	5,2%	4,7%	5,0%
n°35	5,7%	10,6%	11,7%	11,8%	4,5%	4,0%	4,2%	4,2%
n°36	7,1%	12,3%	12,5%	13,7%	5,2%	5,4%	4,2%	5,0%
n°37	7,8%	11,3%	11,7%	12,0%	5,4%	5,2%	4,3%	4,9%
n°38	6,1%	11,1%	12,0%	12,7%	3,8%	3,5%	4,2%	3,8%
n°39	7,0%	11,5%	11,5%	12,3%	6,8%	6,6%	6,3%	6,4%
n°40	7,0%	11,0%	12,2%	12,2%	6,4%	6,3%	5,9%	5,6%
Moyenne	6,93%	11,27%	12,12%	12,62%	4,77%	5,35%	4,60%	4,66%
Écart-type	0,76%	0,83%	0,69%	0,74%	1,15%	1,40%	0,85%	0,86%

Tableau 10

Taux d'erreurs du LibSVM sur différentes représentations de la base des céréales

LibSVM								
Données	Brutes		ACP		PG		Brutes+PG	
<i>Norm.</i>	<i>Non</i>	<i>Oui</i>	<i>Non</i>	<i>Oui</i>	<i>Non</i>	<i>Oui</i>	<i>Non</i>	<i>Oui</i>
n°1	65,0%	13,9%	17,2%	24,3%	2,6%	8,0%	2,8%	7,7%
n°2	65,2%	17,2%	17,6%	28,2%	6,1%	5,7%	4,5%	6,8%
n°3	65,2%	14,8%	17,6%	23,3%	4,5%	9,2%	3,1%	9,6%
n°4	65,0%	17,6%	17,7%	27,7%	3,7%	5,9%	4,2%	7,0%
n°5	65,0%	15,8%	17,4%	26,8%	4,5%	7,5%	3,5%	9,0%
n°6	65,2%	15,5%	17,7%	29,2%	4,9%	7,3%	4,5%	7,8%
n°7	65,0%	15,8%	17,7%	27,3%	5,6%	8,3%	4,7%	8,9%
n°8	65,2%	16,7%	16,3%	27,0%	4,3%	7,7%	3,7%	9,2%
n°9	65,2%	16,9%	17,2%	29,0%	5,7%	9,0%	5,6%	9,4%
n°10	65,0%	13,2%	16,5%	21,9%	5,7%	7,8%	5,7%	8,2%
n°11	65,2%	16,5%	17,4%	30,1%	5,0%	9,4%	4,7%	9,2%
n°12	65,0%	13,7%	16,5%	26,4%	4,3%	6,8%	4,2%	6,1%
n°13	65,0%	15,8%	17,9%	27,1%	0,9%	9,4%	5,0%	8,7%
n°14	65,2%	15,3%	16,9%	28,2%	2,6%	8,7%	2,3%	8,7%
n°15	65,0%	16,2%	16,3%	27,0%	4,2%	6,6%	3,7%	6,8%
n°16	65,2%	15,3%	18,8%	25,6%	4,2%	7,8%	4,0%	8,2%
n°17	65,2%	14,3%	16,3%	28,2%	4,0%	11,8%	4,0%	11,5%
n°18	65,0%	16,0%	17,2%	27,8%	4,5%	10,1%	3,8%	10,3%
n°19	65,2%	16,0%	18,4%	27,5%	4,2%	5,0%	4,5%	6,3%
n°20	65,0%	14,8%	16,0%	27,7%	4,3%	7,7%	3,8%	8,5%
n°21	65,0%	14,4%	16,7%	26,4%	3,0%	8,2%	2,4%	8,2%
n°22	65,2%	17,2%	18,3%	28,5%	6,1%	6,1%	5,4%	5,7%
n°23	65,2%	13,4%	17,2%	23,3%	3,5%	5,7%	3,1%	5,9%
n°24	65,0%	17,0%	17,0%	28,9%	4,5%	7,5%	3,8%	7,3%
n°25	65,0%	15,1%	18,6%	25,7%	4,3%	8,2%	4,0%	8,3%
n°26	65,2%	15,8%	15,8%	27,7%	3,8%	11,7%	4,2%	11,7%
n°27	65,2%	15,7%	16,7%	28,0%	4,5%	9,4%	4,2%	9,6%
n°28	65,0%	15,3%	17,4%	27,3%	4,5%	10,3%	4,3%	10,1%
n°29	65,0%	14,1%	17,2%	23,3%	4,5%	6,8%	4,2%	10,4%
n°30	65,2%	16,5%	16,5%	29,0%	6,1%	9,2%	5,2%	9,0%
n°31	65,2%	13,9%	17,2%	24,3%	3,0%	6,3%	2,6%	7,5%
n°32	65,2%	17,2%	17,6%	28,2%	5,6%	8,3%	5,0%	8,3%
n°33	65,2%	14,8%	17,6%	23,3%	5,2%	8,2%	4,5%	8,9%
n°34	65,0%	17,6%	17,7%	27,7%	4,5%	11,0%	4,5%	11,0%
n°35	65,0%	15,8%	17,4%	26,8%	4,2%	5,6%	4,2%	7,7%
n°36	65,2%	15,5%	17,7%	29,2%	4,2%	8,3%	3,5%	8,7%
n°37	65,0%	15,8%	17,7%	27,3%	4,2%	8,5%	4,2%	9,0%
n°38	65,2%	16,7%	16,3%	27,0%	4,0%	6,4%	3,3%	6,3%
n°39	65,2%	16,9%	17,2%	29,0%	5,7%	6,8%	5,7%	7,1%
n°40	65,0%	13,2%	16,5%	21,9%	5,9%	8,2%	5,4%	8,5%
Moyenne	65,13%	15,58%	17,23%	26,83%	4,43%	8,01%	4,15%	8,42%
Écart-type	0,09%	1,23%	0,69%	2,09%	1,08%	1,63%	0,87%	1,47%

ANNEXE 3

ANALYSE EN QUARTILES : BASE DES CHIFFRES MANUSCRITS

Tableau 11

Analyse en quartiles sur la base des chiffres manuscrits

		minimum	1er décile	1er quartile	mediane	3ème quartile	9ème décile	maximum
Brutes	<i>kNN</i>	2,40%	2,60%	2,98%	3,20%	3,60%	3,90%	4,50%
	<i>kNN norm</i>	2,50%	2,68%	3,00%	3,30%	3,50%	3,61%	4,40%
	<i>MLP norm</i>	3,00%	3,30%	3,50%	3,90%	4,23%	4,61%	4,80%
	<i>SMO</i>	3,20%	3,69%	3,88%	4,25%	4,50%	4,80%	5,00%
	<i>SMO norm</i>	3,00%	3,38%	3,88%	4,00%	4,30%	4,71%	4,90%
	<i>LibSVM</i>	6,20%	7,18%	7,78%	8,40%	9,22%	10,72%	11,10%
	<i>LibSVM norm</i>	3,00%	3,20%	3,40%	3,90%	4,15%	4,60%	4,90%
ACP	<i>kNN</i>	1,70%	2,70%	2,98%	3,20%	3,43%	3,70%	4,00%
	<i>kNN norm</i>	3,20%	3,39%	3,50%	3,90%	4,30%	4,53%	4,90%
	<i>MLP norm</i>	3,80%	4,09%	4,30%	4,55%	4,98%	5,51%	5,60%
	<i>SMO</i>	3,60%	3,90%	4,20%	4,50%	4,75%	5,01%	5,70%
	<i>SMO norm</i>	2,70%	2,89%	3,30%	3,60%	3,93%	4,12%	5,00%
	<i>LibSVM</i>	2,00%	2,39%	2,60%	2,95%	3,30%	3,62%	4,10%
	<i>LibSVM norm</i>	4,50%	4,70%	5,00%	5,40%	5,93%	6,40%	6,60%
AG	<i>kNN</i>	4,70%	5,40%	5,60%	5,90%	6,10%	6,31%	7,10%
	<i>kNN norm</i>	4,20%	4,79%	5,18%	5,55%	5,90%	6,10%	6,70%
	<i>MLP norm</i>	5,50%	5,99%	6,28%	6,85%	7,50%	7,80%	8,00%
	<i>SMO</i>	5,90%	6,98%	7,40%	7,80%	8,40%	8,90%	9,10%
	<i>SMO norm</i>	4,70%	5,17%	5,58%	5,90%	6,30%	6,60%	7,10%
	<i>LibSVM</i>	7,30%	8,20%	9,18%	10,55%	11,28%	12,22%	14,50%
	<i>LibSVM norm</i>	4,60%	5,06%	5,30%	5,60%	6,00%	6,32%	6,90%
PG	<i>kNN</i>	3,60%	4,18%	4,55%	4,90%	5,50%	6,03%	6,70%
	<i>kNN norm</i>	4,00%	4,28%	4,88%	5,30%	5,82%	6,52%	7,70%
	<i>MLP norm</i>	4,70%	5,29%	5,85%	6,60%	7,00%	7,42%	7,90%
	<i>SMO</i>	5,00%	6,00%	6,50%	7,00%	7,60%	7,80%	8,70%
	<i>SMO norm</i>	4,60%	4,79%	5,33%	5,90%	6,40%	6,82%	7,50%
	<i>LibSVM</i>	3,00%	3,49%	4,18%	4,50%	4,83%	5,40%	5,90%
	<i>LibSVM norm</i>	4,80%	4,90%	5,35%	5,85%	6,40%	7,12%	7,90%
Brutes+PG	<i>kNN</i>	2,00%	2,50%	2,80%	3,15%	3,60%	3,90%	4,40%
	<i>kNN norm</i>	2,10%	2,60%	2,87%	3,30%	3,55%	3,90%	4,10%
	<i>MLP norm</i>	2,60%	3,18%	3,50%	3,75%	4,13%	4,60%	5,00%
	<i>SMO</i>	3,10%	3,39%	3,70%	4,00%	4,50%	4,60%	5,10%
	<i>SMO norm</i>	2,90%	3,30%	3,80%	4,05%	4,23%	4,51%	5,10%
	<i>LibSVM</i>	2,00%	2,10%	2,48%	2,80%	3,20%	3,60%	4,20%
	<i>LibSVM norm</i>	2,70%	3,20%	3,50%	3,80%	4,13%	4,42%	4,90%

Tableau 12

Analyse en quartiles du gain entre systèmes à classificateur identiques
(base des chiffres manuscrits)

		minimum	1er décile	1er quartile	mediane	3ème quartile	9ème décile	maximum
PG vs Brutes (Figure 24)	<i>kNN</i>	-3,50%	-3,01%	-2,30%	-1,70%	-1,27%	-0,79%	-0,30%
	<i>kNN norm</i>	-4,70%	-3,21%	-2,65%	-2,10%	-1,60%	-0,80%	-0,70%
	<i>MLP norm</i>	-4,50%	-3,21%	-3,00%	-2,55%	-2,17%	-1,49%	-1,10%
	<i>SMO</i>	-4,80%	-3,65%	-3,40%	-2,90%	-2,60%	-2,00%	-1,40%
	<i>SMO norm</i>	-3,30%	-2,80%	-2,33%	-1,75%	-1,37%	-1,00%	-0,50%
	<i>LibSVM</i>	-2,10%	-1,30%	-1,00%	-0,60%	-0,48%	0,10%	1,00%
	<i>LibSVM norm</i>	-3,80%	-3,11%	-2,50%	-1,90%	-1,60%	-1,30%	-0,80%
PG+Brutes vs Brutes (Figure 30)	<i>kNN</i>	-1,40%	-0,34%	-0,10%	0,10%	0,30%	0,51%	1,50%
	<i>kNN norm</i>	-1,10%	-0,52%	-0,20%	0,00%	0,20%	0,41%	1,40%
	<i>MLP norm</i>	-0,70%	-0,31%	-0,10%	0,05%	0,30%	0,40%	0,90%
	<i>SMO</i>	-0,70%	-0,40%	-0,20%	-0,05%	0,20%	0,41%	0,90%
	<i>SMO norm</i>	-0,50%	-0,22%	-0,10%	0,05%	0,13%	0,40%	0,80%
	<i>LibSVM</i>	0,30%	0,59%	0,80%	1,00%	1,13%	1,41%	1,70%
	<i>LibSVM norm</i>	-0,60%	-0,30%	-0,20%	0,00%	0,23%	0,40%	0,60%
PG vs AG (Figure 36)	<i>kNN</i>	-1,50%	-0,72%	-0,10%	0,45%	1,05%	1,71%	2,20%
	<i>kNN norm</i>	-1,80%	-1,12%	-0,40%	0,10%	0,70%	1,07%	1,90%
	<i>MLP norm</i>	-2,40%	-0,80%	-0,20%	0,50%	1,22%	1,50%	2,20%
	<i>SMO</i>	-2,90%	-2,03%	-1,53%	-1,00%	-0,60%	-0,20%	0,40%
	<i>SMO norm</i>	-2,10%	-0,90%	-0,40%	0,05%	0,70%	0,91%	1,10%
	<i>LibSVM</i>	-0,30%	0,48%	0,60%	1,10%	1,70%	1,81%	3,00%
	<i>LibSVM norm</i>	-2,20%	-1,52%	-0,80%	-0,10%	0,30%	0,70%	1,20%
PG vs ACP (Figure 39)	<i>kNN</i>	-3,70%	-2,60%	-2,23%	-1,75%	-1,30%	-0,99%	-0,50%
	<i>kNN norm</i>	-3,60%	-2,53%	-1,90%	-1,25%	-0,90%	-0,60%	0,30%
	<i>MLP norm</i>	-3,80%	-2,61%	-2,33%	-1,60%	-1,28%	-0,58%	-0,20%
	<i>SMO</i>	-4,40%	-3,40%	-2,93%	-2,50%	-2,10%	-1,59%	-0,60%
	<i>SMO norm</i>	-3,80%	-3,10%	-2,60%	-2,20%	-1,97%	-1,49%	-1,00%
	<i>LibSVM</i>	-2,80%	-2,20%	-1,83%	-1,50%	-1,30%	-0,88%	0,10%
	<i>LibSVM norm</i>	-2,20%	-1,50%	-1,03%	-0,40%	0,00%	0,41%	0,80%

Tableau 13

Analyse en quartiles du gain entre systèmes à classificateur différents
(base des chiffres manuscrits)

		minimum	1er décile	1er quartile	mediane	3ème quartile	9ème décile	maximum
PG vs kNN norm sur Brutes (Figure 25)	<i>kNN</i>	-3,50%	-3,01%	-2,30%	-1,70%	-1,27%	-0,79%	-0,30%
	<i>kNN norm</i>	-4,70%	-3,21%	-2,65%	-2,10%	-1,60%	-0,80%	-0,70%
	<i>MLP norm</i>	-5,10%	-4,32%	-3,80%	-3,25%	-2,38%	-1,99%	-1,40%
	<i>SMO</i>	-5,50%	-4,70%	-4,40%	-3,60%	-3,20%	-2,97%	-1,90%
	<i>SMO norm</i>	-4,10%	-3,72%	-3,20%	-2,50%	-2,10%	-1,69%	-1,40%
	<i>LibSVM</i>	-2,90%	-2,20%	-1,80%	-1,20%	-0,90%	-0,26%	0,20%
	<i>LibSVM norm</i>	-4,60%	-4,30%	-3,13%	-2,45%	-2,10%	-1,89%	-1,40%
PG+Brutes vs kNN norm Brutes (Fig. 31)	<i>kNN</i>	-1,40%	-0,34%	-0,10%	0,10%	0,30%	0,51%	1,50%
	<i>kNN norm</i>	-1,10%	-0,52%	-0,20%	0,00%	0,20%	0,41%	1,40%
	<i>MLP norm</i>	-2,00%	-1,21%	-0,92%	-0,50%	-0,27%	0,02%	0,70%
	<i>SMO</i>	-2,10%	-1,42%	-1,03%	-0,75%	-0,40%	-0,27%	0,40%
	<i>SMO norm</i>	-2,10%	-1,50%	-0,92%	-0,70%	-0,40%	0,01%	0,20%
	<i>LibSVM</i>	-1,20%	-0,30%	0,05%	0,40%	0,73%	1,01%	1,40%
	<i>LibSVM norm</i>	-1,90%	-1,41%	-0,93%	-0,45%	-0,10%	0,20%	0,50%
PG vs kNN norm AG (Figure 37)	<i>kNN</i>	-1,50%	-0,72%	-0,10%	0,45%	1,05%	1,71%	2,20%
	<i>kNN norm</i>	-1,80%	-1,12%	-0,40%	0,10%	0,70%	1,07%	1,90%
	<i>MLP norm</i>	-2,90%	-1,93%	-1,50%	-1,10%	-0,50%	0,60%	1,20%
	<i>SMO</i>	-3,40%	-2,33%	-2,00%	-1,60%	-0,97%	-0,47%	0,40%
	<i>SMO norm</i>	-1,90%	-1,32%	-1,12%	-0,50%	0,17%	0,82%	1,70%
	<i>LibSVM</i>	-0,30%	0,29%	0,47%	0,80%	1,40%	2,21%	2,90%
	<i>LibSVM norm</i>	-2,30%	-1,80%	-1,12%	-0,50%	0,25%	0,90%	1,60%

ANNEXE 4

ANALYSE EN QUARTILES : BASE DES CÉRÉALES

Tableau 14

Analyse en quartiles sur la base des céréales

		minimum	1er décile	1er quartile	mediane	3ème quartile	9ème décile	maximum
Brutes	<i>kNN</i>	7,83%	8,87%	9,22%	9,65%	10,30%	10,96%	11,48%
	<i>kNN norm</i>	4,00%	4,17%	4,70%	5,22%	5,74%	6,16%	7,13%
	<i>MLP norm</i>	2,78%	3,83%	3,96%	4,78%	5,30%	6,28%	8,35%
	<i>SMO</i>							
	<i>SMO norm</i>	9,39%	10,24%	10,61%	11,30%	11,83%	12,35%	12,87%
	<i>LibSVM</i>	65,04%	65,04%	65,04%	65,22%	65,22%	65,22%	65,22%
	<i>LibSVM norm</i>	13,22%	13,90%	14,78%	15,83%	16,57%	17,22%	17,57%
ACP	<i>kNN</i>	4,17%	4,33%	5,00%	5,39%	5,91%	6,97%	7,30%
	<i>kNN norm</i>	4,70%	5,22%	5,87%	6,43%	6,96%	7,48%	8,87%
	<i>MLP norm</i>	0,00%	3,29%	3,83%	4,35%	4,87%	5,76%	7,30%
	<i>SMO</i>	10,61%	11,48%	11,65%	12,00%	12,35%	13,04%	13,91%
	<i>SMO norm</i>	10,96%	11,83%	12,17%	12,61%	13,04%	13,74%	14,26%
	<i>LibSVM</i>	15,83%	16,35%	16,65%	17,22%	17,74%	17,95%	18,78%
	<i>LibSVM norm</i>	21,91%	23,30%	26,26%	27,30%	28,17%	29,04%	30,09%
PG	<i>kNN</i>	3,30%	3,48%	4,00%	4,70%	5,22%	6,61%	7,13%
	<i>kNN norm</i>	3,30%	3,81%	4,17%	4,70%	5,22%	6,66%	7,13%
	<i>MLP norm</i>	3,48%	4,00%	4,70%	5,04%	5,57%	6,14%	7,48%
	<i>SMO</i>	1,57%	3,46%	4,13%	4,78%	5,43%	6,28%	6,78%
	<i>SMO norm</i>	2,43%	3,79%	4,83%	5,22%	6,09%	7,13%	9,22%
	<i>LibSVM</i>	0,87%	2,96%	4,13%	4,43%	5,09%	5,76%	6,09%
	<i>LibSVM norm</i>	5,04%	5,90%	6,78%	8,09%	9,09%	10,10%	11,83%
Brutes+PG	<i>kNN</i>	3,13%	3,65%	4,13%	4,43%	4,91%	5,57%	6,26%
	<i>kNN norm</i>	3,13%	3,48%	4,17%	4,52%	5,13%	5,91%	6,26%
	<i>MLP norm</i>	2,96%	3,65%	4,13%	4,87%	5,22%	6,10%	7,30%
	<i>SMO</i>	2,96%	3,46%	4,17%	4,52%	5,09%	5,76%	6,26%
	<i>SMO norm</i>	2,96%	3,48%	4,00%	4,70%	5,22%	5,57%	6,43%
	<i>LibSVM</i>	2,26%	3,10%	3,65%	4,17%	4,57%	5,39%	5,74%
	<i>LibSVM norm</i>	5,74%	6,26%	7,43%	8,52%	9,22%	10,28%	11,65%

Tableau 15

Analyse en quartiles du gain entre systèmes (base des céréales)

		minimum	1er décile	1er quartile	mediane	3ème quartile	9ème décile	maximum
PG vs Brutes norm (Figure 27)	<i>kNN</i>	-1,74%	-0,75%	-0,17%	0,52%	1,22%	1,60%	2,43%
	<i>kNN norm</i>	-2,09%	-1,57%	-0,04%	0,52%	0,91%	1,58%	2,09%
	<i>MLP norm</i>	-3,65%	-2,09%	-1,39%	-0,52%	0,22%	1,58%	3,30%
	<i>SMO</i>	4,52%	5,03%	5,39%	6,96%	7,13%	7,50%	9,57%
	<i>SMO norm</i>	1,39%	4,52%	4,87%	6,17%	6,96%	7,50%	8,70%
	<i>LibSVM</i>	7,30%	9,57%	10,43%	11,13%	11,70%	12,70%	14,96%
	<i>LibSVM norm</i>	2,43%	5,36%	6,39%	7,30%	8,91%	10,33%	11,65%
PG+Brutes vs Brutes norm (Figure 33)	<i>kNN</i>	-0,87%	-0,21%	0,13%	0,70%	1,22%	1,74%	2,96%
	<i>kNN norm</i>	-0,87%	-0,54%	0,09%	0,52%	1,22%	1,91%	2,78%
	<i>MLP norm</i>	-3,48%	-1,91%	-1,13%	0,00%	0,57%	1,91%	3,48%
	<i>SMO</i>	4,70%	5,22%	6,17%	6,87%	7,13%	8,00%	8,70%
	<i>SMO norm</i>	5,04%	5,37%	5,96%	6,78%	7,30%	7,83%	8,35%
	<i>LibSVM</i>	7,48%	10,23%	10,96%	11,48%	12,17%	13,04%	13,39%
	<i>LibSVM norm</i>	2,78%	5,01%	6,22%	6,87%	7,78%	9,81%	11,48%
PG vs MLP norm sur Brutes (Figure 28)	<i>kNN</i>	-2,78%	-1,44%	-0,96%	-0,17%	0,70%	1,97%	3,13%
	<i>kNN norm</i>	-3,30%	-1,63%	-0,91%	-0,17%	0,57%	1,97%	3,65%
	<i>MLP norm</i>	-3,65%	-2,09%	-1,39%	-0,52%	0,22%	1,58%	3,30%
	<i>SMO</i>	-2,61%	-1,60%	-1,04%	-0,35%	0,87%	1,93%	3,83%
	<i>SMO norm</i>	-4,35%	-2,28%	-1,26%	-0,78%	-0,13%	1,74%	3,48%
	<i>LibSVM</i>	-2,09%	-1,25%	-0,52%	-0,09%	1,09%	2,30%	4,35%
	<i>LibSVM norm</i>	-6,78%	-5,41%	-4,52%	-3,48%	-1,74%	-1,04%	3,30%
PG+Brutes vs MLP norm Brutes (Fig. 34)	<i>kNN</i>	-1,74%	-1,25%	-0,52%	0,17%	0,87%	1,77%	3,65%
	<i>kNN norm</i>	-2,09%	-1,08%	-0,52%	0,09%	0,57%	1,91%	3,65%
	<i>MLP norm</i>	-3,48%	-1,91%	-1,13%	0,00%	0,57%	1,91%	3,48%
	<i>SMO</i>	-2,09%	-1,44%	-0,70%	-0,09%	1,04%	2,26%	3,83%
	<i>SMO norm</i>	-1,74%	-1,22%	-0,87%	-0,09%	0,61%	2,09%	3,48%
	<i>LibSVM</i>	-1,91%	-0,71%	-0,17%	0,52%	1,39%	2,45%	3,83%
	<i>LibSVM norm</i>	-6,78%	-5,57%	-5,04%	-3,91%	-2,43%	-1,22%	2,09%

ANNEXE 5

INTERVALLES DE CONFIANCE : BASE DES CHIFFRES MANUSCRITS

Tableau 16

Intervalles de confiance pour une réalisation du système (base des chiffres manuscrits)

		Borne inférieure de l'intervalle de confiance	Valeur moyenne	Borne supérieure de l'intervalle de confiance
Brutes	<i>kNN</i>	2,63%	3,29%	3,96%
	<i>kNN norm</i>	2,72%	3,25%	3,78%
	<i>MLP norm</i>	3,23%	3,88%	4,53%
	<i>SMO</i>	3,62%	4,22%	4,82%
	<i>SMO norm</i>	3,42%	4,05%	4,68%
	<i>LibSVM</i>	6,95%	8,60%	10,24%
	<i>LibSVM norm</i>	3,20%	3,85%	4,51%
PG	<i>kNN</i>	4,03%	5,01%	6,00%
	<i>kNN norm</i>	4,27%	5,37%	6,47%
	<i>MLP norm</i>	5,34%	6,43%	7,51%
	<i>SMO</i>	6,01%	7,00%	7,98%
	<i>SMO norm</i>	4,90%	5,88%	6,87%
	<i>LibSVM</i>	3,60%	4,49%	5,38%
	<i>LibSVM norm</i>	4,88%	5,94%	7,01%
Brutes + PG	<i>kNN</i>	2,49%	3,17%	3,85%
	<i>kNN norm</i>	2,61%	3,26%	3,90%
	<i>MLP norm</i>	3,09%	3,81%	4,52%
	<i>SMO</i>	3,41%	4,06%	4,70%
	<i>SMO norm</i>	3,36%	4,00%	4,65%
	<i>LibSVM</i>	2,13%	2,87%	3,60%
	<i>LibSVM norm</i>	3,16%	3,81%	4,47%
AG	<i>kNN</i>	5,29%	5,89%	6,49%
	<i>kNN norm</i>	4,78%	5,50%	6,21%
	<i>MLP norm</i>	5,93%	6,88%	7,83%
	<i>SMO</i>	6,83%	7,86%	8,89%
	<i>SMO norm</i>	5,14%	5,92%	6,70%
	<i>LibSVM</i>	8,17%	10,34%	12,52%
	<i>LibSVM norm</i>	4,90%	5,65%	6,40%
ACP	<i>kNN</i>	2,60%	3,19%	3,78%
	<i>kNN norm</i>	3,29%	3,94%	4,58%
	<i>MLP norm</i>	4,02%	4,67%	5,33%
	<i>SMO</i>	3,90%	4,51%	5,11%
	<i>SMO norm</i>	2,93%	3,62%	4,31%
	<i>LibSVM</i>	2,33%	2,97%	3,61%
	<i>LibSVM norm</i>	4,67%	5,45%	6,23%

Tableau 17

Intervalles de confiance de l'espérance de la performance d'un système
(base des chiffres manuscrits)

		Borne inférieure de l'intervalle de confiance	Valeur moyenne	Borne supérieure de l'intervalle de confiance
Brutes	<i>kNN</i>	3,19%	3,29%	3,40%
	<i>kNN norm</i>	3,17%	3,25%	3,33%
	<i>MLP norm</i>	3,77%	3,88%	3,98%
	<i>SMO</i>	4,13%	4,22%	4,32%
	<i>SMO norm</i>	3,95%	4,05%	4,15%
	<i>LibSVM</i>	8,34%	8,60%	8,86%
	<i>LibSVM norm</i>	3,75%	3,85%	3,96%
PG	<i>kNN</i>	4,86%	5,01%	5,17%
	<i>kNN norm</i>	5,20%	5,37%	5,55%
	<i>MLP norm</i>	6,25%	6,43%	6,60%
	<i>SMO</i>	6,84%	7,00%	7,15%
	<i>SMO norm</i>	5,73%	5,88%	6,04%
	<i>LibSVM</i>	4,35%	4,49%	4,63%
	<i>LibSVM norm</i>	5,77%	5,94%	6,11%
Brutes + PG	<i>kNN</i>	3,07%	3,17%	3,28%
	<i>kNN norm</i>	3,15%	3,26%	3,36%
	<i>MLP norm</i>	3,69%	3,81%	3,92%
	<i>SMO</i>	3,95%	4,06%	4,16%
	<i>SMO norm</i>	3,90%	4,00%	4,10%
	<i>LibSVM</i>	2,75%	2,87%	2,98%
	<i>LibSVM norm</i>	3,71%	3,81%	3,92%
AG	<i>kNN</i>	5,79%	5,89%	5,98%
	<i>kNN norm</i>	5,38%	5,50%	5,61%
	<i>MLP norm</i>	6,73%	6,88%	7,03%
	<i>SMO</i>	7,70%	7,86%	8,02%
	<i>SMO norm</i>	5,80%	5,92%	6,04%
	<i>LibSVM</i>	10,00%	10,34%	10,69%
	<i>LibSVM norm</i>	5,53%	5,65%	5,77%
ACP	<i>kNN</i>	3,10%	3,19%	3,28%
	<i>kNN norm</i>	3,83%	3,94%	4,04%
	<i>MLP norm</i>	4,57%	4,67%	4,78%
	<i>SMO</i>	4,41%	4,51%	4,60%
	<i>SMO norm</i>	3,51%	3,62%	3,73%
	<i>LibSVM</i>	2,87%	2,97%	3,07%
	<i>LibSVM norm</i>	5,33%	5,45%	5,57%

ANNEXE 6

INTERVALLES DE CONFIANCE : BASE DES CÉRÉALES

Tableau 18

Intervalles de confiance pour une réalisation du système
(base des céréales)

		Borne inférieure de l'intervalle de confiance	Valeur moyenne	Borne supérieure de l'intervalle de confiance
Brutes	<i>kNN</i>	8,61%	9,74%	10,88%
	<i>kNN norm</i>	4,23%	5,26%	6,30%
	<i>MLP norm</i>	3,29%	4,79%	6,28%
	<i>SMO</i>	5,93%	6,93%	7,92%
	<i>SMO norm</i>	10,19%	11,27%	12,34%
	<i>LibSVM</i>	65,02%	65,13%	65,25%
	<i>LibSVM norm</i>	13,98%	15,58%	17,19%
PG	<i>kNN</i>	3,46%	4,81%	6,17%
	<i>kNN norm</i>	3,51%	4,89%	6,26%
	<i>MLP norm</i>	3,95%	5,18%	6,40%
	<i>SMO</i>	3,27%	4,77%	6,26%
	<i>SMO norm</i>	3,53%	5,35%	7,17%
	<i>LibSVM</i>	3,02%	4,43%	5,84%
	<i>LibSVM norm</i>	5,89%	8,01%	10,13%
Brutes + PG	<i>kNN</i>	3,62%	4,51%	5,40%
	<i>kNN norm</i>	3,58%	4,63%	5,67%
	<i>MLP norm</i>	3,54%	4,86%	6,18%
	<i>SMO</i>	3,49%	4,60%	5,72%
	<i>SMO norm</i>	3,54%	4,66%	5,77%
	<i>LibSVM</i>	3,02%	4,15%	5,29%
	<i>LibSVM norm</i>	6,51%	8,42%	10,34%
ACP	<i>kNN</i>	4,33%	5,48%	6,62%
	<i>kNN norm</i>	5,20%	6,42%	7,64%
	<i>MLP norm</i>	2,70%	4,36%	6,02%
	<i>SMO</i>	11,23%	12,12%	13,02%
	<i>SMO norm</i>	11,66%	12,62%	13,59%
	<i>LibSVM</i>	16,33%	17,23%	18,14%
	<i>LibSVM norm</i>	24,10%	26,83%	29,55%

Tableau 19

Intervalle de confiance de l'espérance de la performance d'un système
(base des céréales)

		Borne inférieure de l'intervalle de confiance	Valeur moyenne	Borne supérieure de l'intervalle de confiance
Brutes	<i>kNN</i>	9,56%	9,74%	9,92%
	<i>kNN norm</i>	5,10%	5,26%	5,42%
	<i>MLP norm</i>	4,55%	4,79%	5,02%
	<i>SMO</i>	6,77%	6,93%	7,08%
	<i>SMO norm</i>	11,09%	11,27%	11,44%
	<i>LibSVM</i>	65,12%	65,13%	65,15%
	<i>LibSVM norm</i>	15,33%	15,58%	15,84%
PG	<i>kNN</i>	4,60%	4,81%	5,03%
	<i>kNN norm</i>	4,67%	4,89%	5,10%
	<i>MLP norm</i>	4,98%	5,18%	5,37%
	<i>SMO</i>	4,53%	4,77%	5,00%
	<i>SMO norm</i>	5,06%	5,35%	5,64%
	<i>LibSVM</i>	4,21%	4,43%	4,65%
	<i>LibSVM norm</i>	7,67%	8,01%	8,34%
Brutes + PG	<i>kNN</i>	4,37%	4,51%	4,65%
	<i>kNN norm</i>	4,46%	4,63%	4,79%
	<i>MLP norm</i>	4,65%	4,86%	5,07%
	<i>SMO</i>	4,43%	4,60%	4,78%
	<i>SMO norm</i>	4,48%	4,66%	4,83%
	<i>LibSVM</i>	3,97%	4,15%	4,33%
	<i>LibSVM norm</i>	8,12%	8,42%	8,72%
ACP	<i>kNN</i>	5,30%	5,48%	5,66%
	<i>kNN norm</i>	6,23%	6,42%	6,61%
	<i>MLP norm</i>	4,10%	4,36%	4,62%
	<i>SMO</i>	11,98%	12,12%	12,26%
	<i>SMO norm</i>	12,47%	12,62%	12,77%
	<i>LibSVM</i>	17,09%	17,23%	17,38%
	<i>LibSVM norm</i>	26,39%	26,83%	27,26%

BIBLIOGRAPHIE

- [1] Craven, Mark W. et Jude W. Shavlik. 1995. « Investigating the Value of a Good Input Representation ». *Computing Learning Theory and Natural Learning Systems*, vol. 3, forthcoming 1995. MIT Press
- [2] Cowley, James L. 2001. *Reconnaissance de Forme Statistique : La discriminante Linéaire de Fisher*. Note de cours de deuxième année. Grenoble, France : École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble (ENSIMAG), 9 p.
- [3] LeCun, Yann et Corinna Cortes. 1998. « MNIST handwritten digit database, Yann LeCun and Corinna Cortes ». In *Yann LeCun's Home*. En ligne. < <http://yann.lecun.com/exdb/mnist/> />. Consulté le 27 juin 2007.
- [4] Inza, Inaki, Pedro Larranaga et Jorge Quiroga. 2001. « Feature Subset Selection by Bayesian networks : a comparison with genetic and sequential algorithms ». *Artificial Intelligence in Medicine*, vol. 3, n°2, p. 187-205.
- [5] Kudo, Mineichi et Jack Sklansky. 2000. « Comparison of Algorithms that Select Features for Pattern Classifiers ». *Pattern Recognition*, vol. 33, n°1, p. 25-41.
- [6] Benahmed, Nadia. 2002. « Optimisation de réseaux de neurones pour la reconnaissance de chiffres manuscrits isolés : sélection et pondération de primitives par algorithmes génétiques ». Mémoire de maîtrise en génie de la production automatisée, Montréal, École de technologie supérieure, 137p.
- [7] Fondation Wikimedia. 2007. « Analyse en composantes principales ». In *Wikipedia : L'encyclopédie libre*. < http://fr.wikipedia.org/wiki/Analyse_en_composantes_principales />. Consulté le 27 juin 2007.

- [8] Markovitch, Shaul et Dan Rosenstein. 2002. « Feature Generation Using Constructor Fonctions ». *Machine Learning*, 40, 59-98.
- [9] Shafti, Leila S., Eduardo Pérez. 2003. « Constructive Induction Using Non-algebraic Feature Representation ». *3rd IASTED International Conference on Artificial Intelligence and Applications* (Benalmadena, Spain, 2003). pp. 134-139. Acta Press.
- [10] Banzhaf W., Nordin P., Keller R.E. et Francone F.D. 1998. *Genetic Programming : An introduction*, San Francisco (USA) : Morgan Kaufmann
- [11] Bensusan, Hilan and Ibrahim Kescu. 1996. « Constructive Induction using Genetic Programming ». In *Evolutionary Computing and Machine Learning Workshop* (ICML-96). 6 p. San Francisco (USA) : T. Fogarty and G. Venturini.
- [12] Poli, Riccardo. 1996. « Genetic Programming for Feature Detection and Image Segmentation ». *Evolutionary Computing*, n° 1143, p 110-125.
- [13] Otero, Fernando E.B., Monique M. S. Silva, Alex A. Freitas et Julio C. Nievola. 2003. « Genetic Programming for Attribute Construction in Data Mining ». *European Conference on Genetic Programming* (Essex, UK, 2003), 10 p. Springer-Verlag.
- [14] Krawiec Krzystof et Bhanu Bir. 2003. « Coevolutionary Computation for Synthesis of Recognition Systems ». In *Proceedings of Computer Vision and Pattern Recognition Conference* (Madisson, Wisconsin, June 16-22 2003), 7 p.
- [15] Ross T., Worell S., Veltten V., Mossing J. et Bryant M. 1998. « Standard SAR ATR Evaluation Experiments using the MSTAR Public Release Data Set ». *SPIE*

- Proceedings : Algorithms for Synthetic Aperture Radar Imagery V*, vol. 3370, pp 566-573, April 1998.
- [16] Intel Corporation. 2000. *Intel Image Processing Library : Reference Manual*. USA : Intel Corporation. 244 p.
 - [17] Intel Corporation. 2001. *Open Source Computer Vision Library : Reference Manual*. USA : Intel Corporation. 377 p.
 - [18] Quinlan, Ross. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA : Morgan Kaufmann, 302p.
 - [19] Delatte, Thibaut. 2006. « Les algorithmes ID3 et C4.5 ». In *SUPINFO Projects – Tous les projets des élèves ingénieurs de Supinfo*. En ligne. < <http://www.supinfo-projects.com/fr/2006/algo%5Fid3%5Fc45/introduction/> />. Consulté le 1^{er} juillet 2007.
 - [20] Eibe Frank, Mark Hall et Len Trigg. 1999. *WEKA*, version 3.4.8. Logiciel. Hamilton, New Zeland : University of Waikato.
 - [21] Krawiec Krzysztof. « Krzysztof Krawiec Home Page ». In *Zakład Inteligentnych Systemów Wspomagania Decyzji (IDSS)*. En ligne. < <http://www-idss.cs.put.poznan.pl/~krawiec/> />. Consulté le 27 juin 2007.
 - [22] Krawiec Krzysztof. 2006. « Evolutionary Learning of Primitive-Based Visual Concepts ». In *IEEE Congress on Evolutionary Computation* (Vancouver, BC, Canada, july 16-21, 2006).
 - [23] GECCO. 2007. « GECCO 2007 : Genetic and Evolutionary Computation Conference ». In *Special Interest Group for Genetic and Evolutionary*

Computation (SIGEVO). En ligne. < <http://www.sigevo.org/gecco-2007/index.html> />. Consulté le 27 juin 2007.

- [24] Koza, John R. 1992. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*, Cambridge, MA : MIT Press. 835 p.
- [25] Kantschik, Wolfgang et Wolfgang Banzhaf. 2002. « Linear-Graph GP - A new GP Structure ». *4th European Conference on Genetic Programming* (Kinsale, Ireland, 2002). Springer-Verlag.
- [26] Kantschik, Wolfgang et Wolfgang Banzhaf. 2001. « Linear-Tree GP and its comparison with other GP structures ». *3th European Conference on Genetic Programming* (Milan, Italy, 2001). Springer-Verlag.
- [27] Nordin, Peter, Wolfgang Banzhaf et Franck Francone. 1999. « Efficient Evolution of Machine Code for CISC Architectures using Blocks and Homologous Crossover ». In *Advances in Genetic Programming, Vol. 3 : Complex Adaptive Systems*, sous la dir. de Spector, Lee, William B. Langdon, Una-May O'Reilly et Peter J. Angeline, chapitre 6. The MIT Press.
- [28] Herrera, Francisco et Manuel Lozano. 2000. « Gradual Distributed Real-Coded Genetic Algorithms ». In *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 1 (april 2000), p. 43-63..
- [29] GNU Project. À jour en juin 1991. « GNU General Public License ». Document en ligne. GNU Project. < <http://www.gnu.org/licenses/gpl.html> />. Consulté le 27 juin 2007.
- [30] Witten Ian H. et Frank Eibe. 2000. *Data Mining : Practical machine learning tools and techniques with JAVA implementations*. USA, Academic Press. 372p.

- [31] Robert P.W. Duin. 1998. « External Home Page Robert P.W. Duin ». In *ICT - Information and Communication Theory Group*. En ligne. < <http://www-ict.ewi.tudelft.nl/~duin/> />. Consulté le 07 mai 2007.
- [32] Eaton, John W. 2004. *MATLAB*, version 7.0.0.19920 (R14). Logiciel. Natick (Ma.) : MathWorks.
- [33] Platt, John C. 1998. *Sequential Minimal Optimization : A Fast Algorithm for Training Support Vector Machines*. « Technical Report MSR-TR-98-14 ». Microsoft Research. 21 p.
- [34] Yasser EL-Manzalawy et Vasant Honavar. 2005. *WLSVM : Integrating LibSVM into Weka Environment*, Software at < <http://www.cs.iastate.edu/~yasser/wlsvm> />
- [35] Hamamura, Tomoyuki, Hiroyuki Mizutani et Bunpei Irie. 2003. « A Multiclass Classification Method Based on Multiple Pairwise Classifiers ». *Seventh International Conference on Document Analysis and Recognition*, 2003. IEEE Computer Society.
- [36] Kohavi, Ron. 1995. « A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection ». In *International Joint Conference on Artificial Intelligence (IJCAI)* (Montréal, Qc, août 1995).
- [37] Le Guen, Monique. 2001. *La boîte à moustaches de Tukey : un outil pour initier à la statistique*. Paris : CNRS-MATISSE, 14 p.
- [38] Tukey, J.W. 1977. *Exploratory Data Analysis*, EDA Reading, MA, Addison-Wesley.

- [39] Gathercole, Chris. 1998. « An Investigation of Supervised Learning in Genetic Programming ». Thèse de doctorat, UK, University of Edinburgh, 195 p.
- [40] Eggermont, Jeroen, Agoston E. Eiben et Jano I. van Hemert. 1999. « A Comparison of Genetic Programming Variants for Data Classification ». In *Advances in Intelligent Data Analysis: Third International Symposium*. (Amsterdam, The Netherlands, 1999), IDA-99, vol. 1642, p. 281-290. Berlin, Allemagne : Springer-Verlag.
- [41] Bellman, R.E. 1961. *Adaptive Control Processes*. Princeton, NJ : Princeton University Press, 274 p.
- [42] Fondation Wikimedia. 2007. « Fléau de la dimension ». ». In *Wikipedia : L'encyclopédie libre*. En ligne. < http://fr.wikipedia.org/wiki/Fl%C3%A9au_de_la_dimension />, consulté le 16 juin 2007.
- [43] Bélisle Claude. 2004. *La loi du Khi-Deux, la loi de Student, la loi de Fisher*, document pédagogique, Québec : Université Laval, Département de mathématiques et de statistique, 20 p.
- [44] Fondation Wikimedia. 2007. « Loi de Student ». In *Wikipedia : L'encyclopédie libre*. En ligne. < http://fr.wikipedia.org/wiki/Loi_de_Student />. Consulté le 18 juin 2007.
- [45] Bouckaert, Remco R. « Estimating Replicability of Classifier Learning Experiments ». *21th International Conference on Machine Learning* (Banff, Canada, 2004).

- [46] Fondation Wikimedia. 2007. « Erreur (métrologie) ». In *Wikipedia : L'encyclopédie libre*. En ligne. < http://fr.wikipedia.org/wiki/Erreur_%28m%C3%A9trologie%29 />. Consulté le 24 juin 2007.
- [47] Salzberg Steven L. 1999. « On Comparing Classifiers: A Critique of Current Research and Methods ». *Data Mining and Knowledge Discovery*, 1, 1-12.
- [48] Dealemans, Walter et Véronique Hoste. 2002. « Evaluation of Machine Learning Methods for Natural Language Processing Tasks ». In *the Third International Conference on Language Resources and Evaluation* (Las Palmas, Spain, may 2002), p. 755-760.
- [49] Dietterich, Thomas G. 1998. « Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms ». *Neural Computation*, vol. 10, p. 1895-1923.
- [50] Nadeau, Claude et Yoshua Benjio. 2000. « Inference for the Generalization Error ». In *Advances in Neural Information Processing Systems 12* (Cambridge, MA, 2000). The MIT Press.
- [51] Bouckaert, Remco R. 2003. « Choosing between two learning algorithms based on calibrated tests ». In *20th International Conference on Machine Learning* (Washington, DC USA, august 2003), p. 51-58. Morgan Kaufmann.
- [52] Zhang, Byoung-Tak et Je-Gun Joung. 2000. « Building Optimal Committees of Genetic Programs ». *Parallel Problem Solving from Nature*, PPSN VI, p. 231-240.
- [53] Gathercole, Chris et Peter Ross. 1994. « Dynamic Training Subset Selection for Supervised Learning in Genetic Programming ». *Parallel Problem Solving from Nature III*, vol. 866, p. 312-321.

- [54] Alpaydin, Ethem. 1999. « Combined 5x2 cv F Test for Comparing Supervised Classification Learning Algorithms ». Neural Computation, vol. 11, n°8, p. 1885-1892.