Construction par optimisation d'arbres de décision

R. Rakotomalala — S. Lallich

Laboratoire ERIC - Université Lyon 2
5 av. Pierre Mendès-France
F-69676 Bron cedex

RÉSUMÉ. *L'apprentissage par arbres se prête mal au traitement des très grandes bases de données dans la mesure où il nécessite à chaque noeud de scanner à nouveau l'ensemble de la base. Dans ce papier, nous proposons de limiter l'espace de recherche des arbres à celui des arbres par niveaux où les différents noeuds de chaque niveau sont segmentés par la même variable. Une telle simplification permet de transformer le problème d'apprentissage en un problème d'optimisation qui autorise une stratégie gloutonne ne nécessitant qu'une seule passe sur les données. Dans le cas de données booléennes, le critère d'optimisation retenu est celui de la maximisation du R^2, alors que dans le cas de données catégorielles multivaluées, on se ramène au cas booléen sans altérer la complexité de l'algorithme en raisonnant dans l'espace des paires d'individus décrites par les indicatrices de co-étiquetage. Notre stratégie d'optimisation pénalise la profondeur de l'arbre par le recours à la correction du R^2. Les expérimentations ont montré que la précision en généralisation des arbres par niveaux n'est pas détériorée par rapport aux arbres usuels tout en maintenant une relative intelligibilité des résultats, alors même que les arbres par niveaux ne nécessitent qu'une seule passe sur les données.*

ABSTRACT. *Induction of decision trees is not well adapted to handling of very large databases. It requires to access the whole database at each node. In this paper, we propose to limit the tree search space to oblivious decision trees The different nodes at the same level are splitted with the same attribute. Such a simplification allows to turn the learning problem into an optimization problem which relies on an hill-climbing strategy requiring only one pass over the database. In the case of boolean attributes, the optimization criterion is the R^2 maximisation. Multivalued attributes can be reduced to boolean attributes by rewriting the problem in the space of pairwise examples spanned with the co-labels of the original attributes, without increasing the complexity of the algorithm. Our strategy penalizes tree depth by using adjusted R^2. The experiments show that generalization error rate of our trees is as good as the one of C4.5, while maintaining the intelligibility of the results.*

MOTS-CLÉS : *Apprentissage, Arbre de décision par niveaux, Optimisation, Règles*

KEYWORDS: *Machine learning, Oblivious decision tree, Optimization, Rules*

1. Introduction

Les arbres de décision font partie des méthodes les plus populaires en apprentissage supervisé. Ils sont maintenant parfaitement maîtrisés et leur diffusion dans le monde industriel atteste de leur robustesse et de leurs performances. Dans la communauté scientifique, ils restent un des domaines privilégiés de publications, mais on constate généralement que si la réduction du taux d'erreur en prédiction est faible, la réduction de la taille de l'arbre, et donc sa lisibilité, est réelle dans les travaux récents [ZIG 00]. Bien souvent également, les arbres de décision, de par leur simplicité, sont mis en oeuvre dans l'étude de nouvelles voies de recherche : dans les premières études sur l'agrégation des classifieurs par exemple [BRE 96], ou encore dans le nettoyage des ensembles de données préalable à l'apprentissage [SEB 00].

Pourtant, fondamentalement, les algorithmes de construction des arbres de décision n'ont pas varié depuis les premiers travaux génériques [QUI 79]. Il s'agit toujours de trouver, à chaque noeud de l'arbre, la segmentation la plus intéressante au regard d'un critère donné. L'arbre est ainsi construit récursivement de la racine vers les feuilles. Les points les plus importants sont : le choix du critère de segmentation, la détermination de la bonne taille de l'arbre, et l'assignation d'une conclusion à une feuille [BRE 84]. Malgré les nombreux travaux qui sont venus par la suite [BRE 97], il y a deux points sur lesquels les algorithmes de construction d'arbre ont très peu évolué ces dernières années : la capacité à prédire à travers le taux d'erreur en généralisation) et la rapidité de construction de l'arbre. Le premier point est inhérent au biais de représentation adopté, un arbre de décision induit des partitionnements parallèles aux axes, il paraît difficile de l'améliorer. Le second point tient à la nécessité de revenir à chaque fois sur la base de données afin d'effectuer les croisements nécessaires aux calculs lors de la segmentation de chaque noeud de l'arbre.

Dans cet article, nous nous intéressons à une famille particulière d'arbre de décision que nous nommerons "arbre de décision par niveau" (que nous opposerons donc aux arbres de décisions "classiques" bien connus dans la communauté de l'apprentissage automatique [QUI 93]). Son principe est simple, nous segmentons chaque noeud situé sur un même niveau de l'arbre à l'aide de la même variable, la construction s'effectuant toujours dans le sens de la racine vers les feuilles. Certes, le biais de représentation est plus rigide ici par rapport un arbre de décision "classique" [POM 01], la spécialisation de l'arbre est réalisée variable par variable et non plus noeud par noeud. En revanche, de cette famille d'arbres nous tirons deux avantages que nous mettrons à profit lors de sa construction : (1) il est possible de transformer le problème d'apprentissage en problème d'optimisation ; (2) les informations nécessaires à l'élaboration de l'arbre sont calculées de proche en proche sans avoir à accéder aux données, une seule passe sur la base est nécessaire pour le calcul des coefficients initiaux. Nous lèvons ainsi un des principaux goulots d'étranglement de l'élaboration des arbres sur de grandes bases de données, lorsque l'ensemble d'apprentissage ne peut pas tenir en mémoire.

La procédure que nous utilisons dans le cadre de la construction d'arbre par niveau possède une forte analogie avec la régression multiple bien connue en statistique, notamment la régression "pas à pas" qui permet à la fois de construire le modèle et de choisir les variables explicatives les plus pertinentes. Il est à noter que la construction d'un arbre par niveaux suivant une recherche exhaustive n'est pas envisageable dans la mesure où elle exige de comparer les performances de 2^p sous-ensembles d'attributs. L'heuristique que nous avons adoptée est une stratégie "pas à pas" descendante gloutone : à chaque niveau de l'arbre, nous introduisons l'attribut prédictif qui minimise le critère global retenu. Pour définir ce critère, nous avons retenu différentes exigences : éviter les attributs redondants ou non pertinents, déceler les interactions complexes, travailler en une seule passe sur les données. Les propriétés de la régression pas à pas que nous allons rappeler brièvement par la suite nous ont paru satisfaire ces exigences.

Notre article est organisé de la manière suivante : dans la section 2, nous détaillerons les caractéristiques de la régression pas à pas pour les variables numériques. Ensuite, dans le section 3, nous adapterons cette procédure au cas des arbres par niveaux. Dans la section 4, nous décrirons l'algorithme que nous mettons en oeuvre pour construire un arbre par niveau. Dans la section 5, nous comparerons nos résultats avec les méthodes classiques d'arbre de décision sur des bases de données synthétiques et réelles. Dans la section 6, nous discuterons des avantages et inconvénients de notre approche, et nous essayerons de la situer au sein des travaux dans le même domaine. Enfin, nous conclurons dans une septième et dernière section en essayant de mettre en exergue la suite que nous pouvons donner à cette approche.

2. Régression pas à pas pour des variables numériques

Considérons le modèle formé par une variable numérique Y à expliquer et p variables numériques explicatives $X_j, j = 1, 2, ..., p$. Pour choisir les variables pertinentes et éliminer les multicolinéarités entre variables explicatives, on a recours classiquement à une procédure de sélection progressive (*forward selection*) des variables explicatives [MAD 92] . On note ϕ_i une mesure de l'imprécision associée au modèle M_i comportant i variables explicatives et ϕ_0 la valeur de cette mesure en l'absence de variables explicatives.

Pour choisir les variables du modèle M_i, la sélection progressive consiste à garder les variables du modèle M_{i-1} et à leur ajouter celle des variables restantes qui minimise ϕ_i. Par construction : $\phi_0 \geq \phi_1 \geq \phi_2 ... \geq \phi_{i-1} \geq \phi_i$. Le modèle M_i est sans erreur en apprentissage lorsque $\phi_i = 0$. On peut alors définir le critère de qualité global par $C_i = \frac{\phi_0 - \phi_i}{\phi_0}$, où par construction C_i est compris entre 0 et 1. On définit encore un critère partiel $c_i = \frac{\phi_{i-1} - \phi_i}{\phi_{i-1}} = \frac{C_i - C_{i-1}}{1 - C_{i-1}}$, lui ausi compris entre 0 et 1, qui indique quel est l'apport de la variable introduite au i^e niveau. Enfin, on appelle critère simple la quantité c_1 qui exprime la valeur du critère global pour un arbre à un seul niveau, c'est-à-dire la mesure de l'association prédictive entre Y et l'attribut introduit au pas 1.

En économétrie, ϕ_i est le plus souvent définie comme la somme des carrés des résidus (ou erreurs en apprentissage) associés à l'explication de Y linéairement suivant les i variables retenues, ce qui fait que ϕ_0 correspond à la variation totale de Y (ajustement de Y par une constante). Dans ce cas, C_i est le coefficient de détermination totale du modèle M_i noté $R_i^2 = R^2(Y; X_1, X_2, ..., X_i)$, alors que c_i est le coefficient de détermination partielle $r_i^2 = r^2(Y; X_i/X_1, X_2, ..., X_{i-1})$, où l'on note X_i la variable introduite au pas i. Le coefficient de détermination partielle entre Y et X_i conditionnellement à $X_1, X_2, ..., X_{i-1}$ s'interpréte comme le coefficient de détermination entre les résidus de Y et les résidus de X_i, une fois que l'on a retiré à chacune de ces deux variables l'effet linéaire de toutes les autres variables explicatives. Le critère simple c_1 correspond au coefficient de détermination simple entre Y et X_1 la variable introduite au 1^{er} pas. La sélection progressive des variables amène à choisir au 1^{er} pas la variable la plus correlée à Y et à introduire au pas i la variable qui présente le meilleur coefficient de détermination partielle r_i^2.

Une propriété remarquable du coefficient de corrélation partielle est l'existence d'une formule de passage qui permet de calculer le coefficient de corrélation partielle, de proche en proche à partir des coefficients simples [SAP 75].

Au premier ordre :

$$r(Y; X_2/X_1) = \frac{r(Y; X_2) - r(Y; X_1)r(X_2; X_1)}{\sqrt{(1 - r^2(Y; X_1))\,(1 - r^2(X_2; X_1))}}$$

Aux ordres supérieurs :

$$r(Y; X_i/X_1, X_2, ..., X_j) = \frac{r(Y;X_i/X_1,X_2,...,X_{j-1}) - r(Y;X_j/X_1,X_2,...,X_{j-1})r(X_i;X_j/X_1,X_2,...,X_{j-1})}{\sqrt{(1-r^2(Y;X_j/X_1,X_2,...,X_{j-1}))(1-r^2(X_i;X_j/X_1,X_2,...,X_{j-1}))}}$$

Cette formule de passage, peu utilisée en econométrie, montre bien que pour passer d'un pas à l'autre, il n'est pas nécessaire de revenir aux données. En effet, il suffit de tenir à jour la table des corrélations entre toutes les variables, y compris Y. La table de départ est celle des corrélations simples. On recherche dans la colonne correspondant à Y la variable qui maximise la corrélation simple, notée X_1. On met à jour la table en calculant les corrélations partielles par rapport à cette variable suivant la formule de passage au 1^{er} ordre. On cherche à nouveau dans la colonne correspondant à Y la variable qui procure la corrélation partielle maximum, notée X_2, et on met à jour la table, et ainsi de suite ... Une fois choisie la variable introduite au pas i, on calcule la valeur correspondante du critère global par la formule :

$$1 - R_i^2 = \prod_{j=1}^{i} \left(1 - r_j^2\right)$$

Il reste encore à choisir un critère d'arrêt. Pour des variables numériques, dans le cadre des hypothèses du modèle linéaire général, on dispose d'une formule intéres-

sante [MAD 92] qui relie le coefficient partiel $r_i^2 = r^2(Y;\, X_i/X_1, X_2..., X_{i-1})$ au ratio de Student de la variable X_i, noté t_i :

$$r_i^2 = \frac{t_i^2}{t_i^2 + n - i - 1}$$

Au risque de 5%, pourvu que $n - i - 1 > 30$, un ratio de Student n'est pas significativement différent de 0 lorsque $|t_i| < 2$. Dans le cadre du modèle linéaire général, un critère d'arrêt peut donc être :

$$r_i^2 < \frac{4}{n - i + 3}$$

Ce critère n'est pas entièrement satisfaisant car d'une part il suppose réalisées les hypothèses du modèle linéaire général, notamment le caractère gaussien de l'erreur, d'autre part il ne tient pas compte du fait que l'on pratique des tests répétés.

Un point de vue complémentaire purement empirique est apporté par le recours au R^2 corrigé (ou ajusté) qui pénalise le modèle retenu en fonction du nombre i de variables explicatives :

$$1 - \overline{R_i}^2 = \frac{n-1}{n-i-1}\left(1 - R_i^2\right)$$

On arrête alors la procédure lorsque le R^2 corrigé diminue. On a : $\frac{1-\overline{R_i}^2}{1-\overline{R_{i-1}}^2} = \frac{n-i}{n-i-1}\left(1 - r_i^2\right)$, dont on déduit le critère d'arrêt :

$$r_i^2 < \frac{1}{n-i}$$

On peut aussi avoir recours à un critère d'arrêt du type Akaïke [CEL 94], ce qui nous amène à introduire de nouveaux prédicteurs tant que la quantité $n\Delta R^2 - 2$ reste positive, soit le critère d'arrêt : $r_i^2 < \frac{2}{n\left(1-R_{i-1}^2\right)}$. Si l'on veut que le nombre de prédicteurs apparaisse explicitement et pas seulement à partir de la qualité du modèle déjà obtenu, on peut utiliser le critère *SBIC* (critère *BIC* de Schwartz).

Il faut bien noter que la procédure pas à pas n'est pas globalement optimale. Plus précisément, sa faiblesse est qu'elle est liée indissolublement aux premières variables choisies, ce qui peut écarter une combinaison de variables performantes. Les logiciels économétriques introduisent donc dans la procédure pas à pas la possibilité de revenir en arrière. Après l'introduction de la variable X_i, le statut de chaque variable du modèle est réexaminé et on élimine celle dont le t de Student est inférieur au seuil critique. Pour ce faire, sans revenir aux données, il faut repartir de la table des corrélations simples et pour chaque variable $X_j, j = 1, 2, ...i$ il faut opérer le calcul des

coefficients partiels puis du R_i^2 en gardant la variable X_j pour la fin, ce qui permet de calculer t_j et d'éliminer la variable X_j si elle n'est plus significative.

3. Procédure pas à pas pour construire des arbres par niveaux

La régression pas à pas offre donc pour des variables numériques les qualités que nous avons identifiées pour une stratégie de construction d'arbres par niveaux : éviter les attributs redondants ou non pertinents, déceler des interactions complexes, travailler en une seule passe sur les données. Suivant une méthode que nous avons élaboré pour la sélection de variables [LAL 00b], nous allons montrer que la régression pas à pas s'applique directement à la construction d'arbres par niveaux dans le cas de variables booléennes, puis nous indiquerons comment ramener commodémment le cas multivalué au cas booléen.

3.1. *Le cas booléen*

Dans le cas de variables booléennes, nous proposons de construire l'arbre par niveaux suivant une procédure pas à pas qui maximise le R^2 corrigé entre l'étiquette Y et les attributs $X_j, j = 1, 2, ...p$. En effet, les variables booléennes ont des propriétés très remarquables liées au fait que l'on peut les traiter comme des variables numériques codées $0-1$, sans perte de généralité, puisque tous les codages se déduisent les uns les autres par transformation affine. Si l'on considère deux variables Y et X dont la proportion conjointe de 1 est notée p_{11} et les proportions marginales p_{1+} et p_{+1}, alors X est de moyenne p_{+1} et de variance $p_{+1}(1-p_{+1})$, Y est de moyenne p_{1+} et de variance $p_{1+}(1-p_{1+})$ et la covariance de X et de Y s'écrit $p_{11}-p_{+1}p_{1+}$. On calcule ainsi le coefficient de corrélation linéaire entre Y et X (appelé coefficient de corrélation de point dans ce cas précis), invariant par transformation affine, qui se confond dans le cas 2×2 avec le ϕ de contingence signé obtenu par normalisation du *khi 2* d'indépendance entre Y et X.

$$r(Y;X) = \varphi = \frac{p_{11}p_{22}-p_{12}p_{21}}{\sqrt{p_{1+}p_{2+}p_{+1}p_{+2}}}$$

L'intérêt de raisonner sur le coefficient de corrélation de point r est multiple : il est naturellement signé, il vérifie une formule de passage en tant que cas particulier du coefficient de corrélation de Pearson et il est nul en cas d'indépendance conditionnelle. Pour montrer ce dernier point, on effectue une démonstration directe sur le r partiel en utilisant le fait que lorsque l'on régresse une variable quelconque sur une variable booléenne, la régression est linéaire. On notera que, comme le coefficient de Tschuprow partiel (sec. 6.1), $r(Y;X/Z)$ peut être nul sans qu'il y ait indépendance. Il en est de même de tout coefficient vérifiant la relation de passage. Il suffit de considérer une table $2 \times 2 \times 2$ pour laquelle le nouveau prédicteur X est de corrélation nulle avec l'ancien prédicteur Z et avec la variable d'intérêt Y. Les coefficients de corrélation partielle fondés sur les coefficients du type *Proportional Reduction in Er-*

Individus	Y	X
1	1	1
2	1	2
3	2	2
4	3	3

Paires	I_Y	I_X
1,1	1	1
1,2	1	0
1,3	0	0
1,4	0	0
2,1	1	0
2,2	1	1
2,3	0	1
2,4	0	0
3,1	0	0
3,2	0	1
3,3	1	1
3,4	0	0
4,1	0	0
4,2	0	0
4,3	0	0
4,4	1	1

Tableau 1. *Transformation de données multivaluées en données booléennes sur l'espace des paires.*

ror [LAL 00a] satisfont à la propriété d'indépendance conditionnelle mais ne vérifient pas de formule de passage.

3.2. *Le cas multivalué*

Pour traiter le cas des variables catégorielles multivaluées, on propose le plus souvent de se ramener au cas booléen en mettant les variables sous forme disjonctive complète, ce qui présente ici plusieurs difficultés :

– le nombre de variables augmente considérablement, ce qui accroît le nombre de tableaux croisés à construire ainsi que la taille de la matrice de corrélations gérée en mémoire, augmentant de façon importante la complexité de la procédure (sec. 4.4) ;

– le cas où la variable à expliquer est elle aussi multivaluée ne peut pas être pris en compte.

Pour échapper à ces deux difficultés, nous proposons une solution plus originale, qui consiste à travailler dans l'espace des paires d'individus décrites par les indicatrices de coétiquetage attachées à chacune des variables de départ, à l'image des travaux de Michaud et Marcotorchino. On recherche ainsi pas à pas, sur la base du coefficient de corrélation de point, les indicatrices de coétiquetage associées aux variables exogènes qui expliquent le mieux l'indicatrice de coétiquetage associée à l'endogène. On passe donc d'une matrice de variables exogènes multivaluées de taille $n \times p$ à une matrice booléenne de taille $\frac{n(n-1)}{2} \times p$ (Table 1).

Pour calculer le coefficient de corrélation de point entre les indicatrices de coétiquetage, plutôt que de se limiter à la formule usuelle de r sur la table associée aux deux indicatrices des variables Y et X, le plus commode est d'établir une formule contingentielle à partir de la table de départ des variables Y et X. Il suffit pour cela d'exprimer les effectifs de la table des indicatrices, notés $g_{kl}, k = 1, 2; l = 1, 2$, en fonction des effectifs de la table de départ des variables Y et X, notés n_{kl}. Nous avons choisi de définir les paires d'individus avec répétitions, prenant donc l'ordre en considération. En effet, on peut dans ce cas démontrer que l'indépendance des variables catégorielles multivaluées Y et X implique l'indépendance des indicatrices de coétiquetage associées et donc la nullité du coefficient de corrélation associé, $r(I_Y, I_X)$. A l'inverse, lorsque les paires sont définies sans répétitions et sans ordre, $r(I_Y, I_X)$ a une valeur négative.

Y\X	$=$ (même étiquette sur X)	$\neq$ (étiquettes différentes sur X)
$=$	$2g_{11} = \sum_{k=1}^{K} \sum_{l=1}^{L} n_{kl} [n_{kl} - 1]$	$2g_{12} = \sum_{k=1}^{K} \sum_{l=1}^{L} n_{kl} [n_{k+} - n_{kl}]$
$\neq$	$2g_{21} = \sum_{k=1}^{K} \sum_{l=1}^{L} n_{kl} [n_{+l} - n_{kl}]$	$2g_{22} = \sum_{k=1}^{K} \sum_{l=1}^{L} n_{kl} [n - n_{k+} - n_{+l} + n_{kl}]$

$$r(I_Y, I_X) = \frac{g_{11}g_{22} - g_{12}g_{21}}{\sqrt{g_{1+}g_{2+}g_{+1}g_{+2}}}$$

Une fois établie la table des coefficients de corrélation de points bruts, il ne reste plus qu'à calculer de proche en proche les coefficients de corrélation partielle qui permettent de sélectionner pas à pas les variables. En passant de l'espace des individus dans l'espace des paires, on ne prédit plus l'étiquette sur Y que présente un objet compte tenu de ses étiquettes sur les $X_j, j = 1, 2, ...p$, mais on prédit le fait que deux objets présentent ou non la même étiquette sur Y compte tenu du fait que leurs étiquettes sur les $X_j, j = 1, 2, ...p$, sont identiques ou non. C′est ainsi que l'on travaille dans une situation booléenne particulière où les rôles des deux étiquettes de chaque indicatrice ne sont pas interchangeables. On doit donc maximiser r et non pas r^2, dans la mesure où une forte corrélation négative est l'indice d'une aptitude élevée de X à mal prédire Y. De la même façon, aux pas suivants, on maximisera r_{part} et non pas r^2_{part}. En effet, prenons l'exemple du second pas de la procédure, où l'on cherche la variable X qui prédit le mieux Y à Z fixé, où Z est la variable introduite au pas 1. A Z fixé, soit toutes les paires sont concordantes en Z, soit toutes les paires sont discordantes en Z ; dans chacun de ces deux cas, on veut que la concordance en X aille de pair avec la concordance en Y.

4. Algorithme de construction de l'arbre

La construction de l'arbre de décision repose sur l'optimisation du critère $\overline{R}^2$ en explorant les arbres de différents niveaux. Dans la figure 1, nous montrons un exemple d'arbres par niveau pour le fichier AUTOS [BAY 99]. Nous adoptons une démarche descendante en partant de la racine de l'arbre vers les feuilles. Cette démarche gloutonne est déjà habituellement rapide par rapport aux méthodes d'optimisation plus

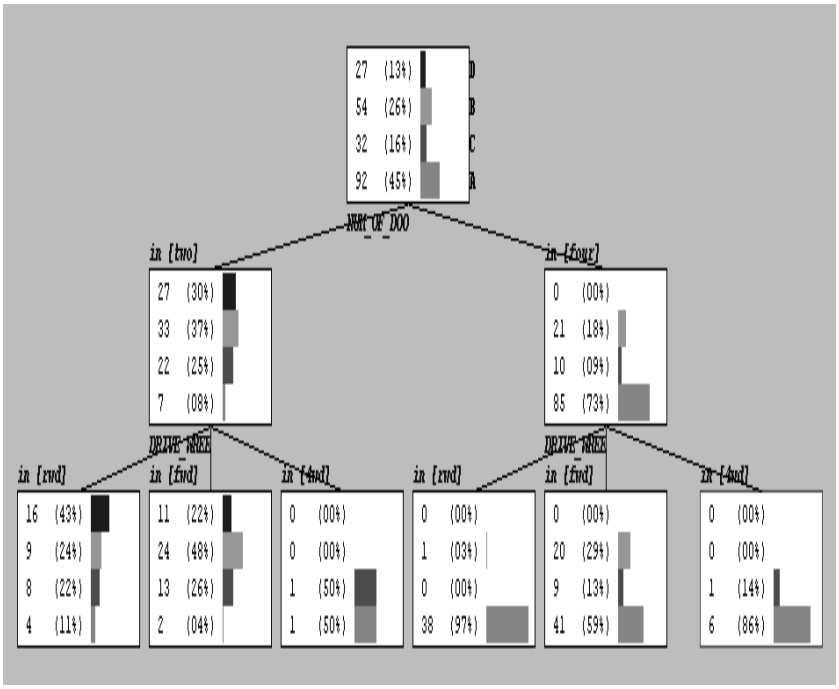

Figure 1. *Arbre de décision par niveaux, à deux niveaux, sur la base AUTOS*

sophistiquées. Nous la rendons encore plus rapide en limitant l'accès à la base de données à une seule passe, au début de l'apprentissage. Tous les calculs qui viennent par la suite s'effectuent en mémoire.

4.1. *Principe générique*

Le principe est simple, nous ajoutons des variables (des niveaux) dans l'arbre tant que le critère à optimiser augmente. Nous remarquons que dans ce cadre, une variable ne peut apparaître qu'une seule fois dans l'arbre, sur un niveau donné.

Si l'on adopte la démarche classique de construction d'un arbre de décision, pour un niveau de l'arbre, la décision de l'introduction d'une nouvelle variable X_{j^*} passe par le calcul du gain qu'apporte la segmentation de tous les sommets sur le dernier niveau à l'aide de cette variable [ZIG 00]. Cette approche "naïve" pose problème pour plusieurs raisons : un arbre par niveau a tendance à produire plus de feuilles qu'un arbre classique, si l'on a déjà introduit 10 variables binaires dans l'arbre, nous disposerons alors de $2^{10} = 1024$ feuilles sur lesquelles nous devons calculer tous les croisements nécessaires en scannant la base de données pour évaluer les distributions conditionnelles et déduire ainsi le gain induit par l'introduction d'une nouvelle va-

riable, ce qui devient rapidement prohibitif à mesure que la profondeur de l'arbre augmente ; toujours due à la prolifération des feuilles de l'arbre, il se produit très rapidement une fragmentation des données, plusieurs sommets sont vides, ce qui rend impossible le calcul effectif des gains induits par une segmentation supplémentaire.

4.2. *Un algorithme "rapide"*

Notre objectif est de construire un arbre de décision par niveaux en limitant l'accès à la base de données à une seule passe, tout le reste des calculs s'effectuant alors en mémoire. L'algorithme repose en grande partie sur la possibilité de calculer les coefficients de corrélations partielles à partir des coefficients marginaux et de déduire à partir de ces coefficients la mesure $\overline{R}^2$ évaluant globalement la qualité de l'arbre.

Dans un premier temps, nous calculons les coefficients de corrélations marginaux, un tableau $T_{r(Y,X)}$ de taille $(p+1) \times (p+1)$ est construit (p est le nombre de descripteurs dans la base), contenant les corrélations $r_{Y,X_j} (j = 1, \ldots, p)$ et $r_{X_i,X_j} (i, j = 1, \ldots, p)$. La mesure étant symétrique, le calcul est mené uniquement sur la partie triangulaire supérieure du tableau. Notons que le calcul de ce tableau de coefficients ne nécessite qu'une seule passe sur les données.

Dans un deuxième temps, nous partons de la racine de l'arbre, et nous cherchons dans le tableau $T_{r(Y,X)}$ la variable X^* qui est la plus corrélée avec Y afin d'induire une première segmentation. Si cette corrélation est significativement non nulle, nous l'introduisons dans l'arbre de décision, et nous recalculons notre tableau de corrélation qui, cette fois-ci, contient les corrélations partielles $r(Y, X/X^*)$. En utilisant la formule permettant de déduire les coefficients partiels des coefficients bruts, on constate aisément qu'il n'est plus nécessaire à partir de cette étape de lancer les calculs à partir des données. Dans l'étape suivante, nous recherchons la variable X^{**} qui est la plus corrélée avec Y conditionnellement à X^*, nous en déduisons la qualité de l'arbre $\overline{R}^2_{\{Y;X^*,X^{**}\}}$. Et nous continuons ainsi jusqu'à ce que le critère $\overline{R}^2_{\{Y;\ldots\}}$ commence à décroître, ce qui constitue la règle d'arrêt de l'algorithme. Ainsi, par une exploration gloutonne très simple, le tableau $T_{r(Y,X)}$ est mis à jour de proche en proche, au fur et à mesure que l'on introduit des variables dans l'arbre. L'algorithme correspondant est décrit en pseudo-code dans le tableau 2. On remarque qu'à la règle d'arrêt de l'optimisation de $\overline{R}^2$, l'indicateur commence à décroître, on a substitué la règle d'arrêt $r_i(Y, X/S) > \frac{1.0}{\sqrt{n-i}}$, les deux règles d'arrêt sont totalement équivalentes (sec. 2).

4.3. *Assignation de conclusions aux feuilles*

A la différence des méthodes classiques d'induction d'arbres de décision, nous nous affranchissons du calcul explicite des distributions conditionnelles $P(Y = y_k/\text{chemin du sommet})$ sur chaque sommet lors de la construction de l'arbre de décision. Cela nous pose problème lorsqu'il s'agit d'assigner une conclusion à une des feuilles de

Construire la racine de l'arbre - $S = \emptyset$, $i = 0$
Calculer les coefficients marginaux $T_{r_0}(Y, X)$
Répéter
  Trouver $X^* = \arg\max_X r_i(Y, X/S)$
  Si $r_i(Y, X/S) > \frac{1.0}{\sqrt{n-i}}$
   Alors
     Ajouter un niveau dans l'arbre - $S = S \cup \{X^*\}$
     $i = i + 1$
     Recalculer $T_{r_i}(Y, X)$ à partir de $T_{r_{i-1}}(Y, X)$
  Fin Si
Jusqu'à *"Dernier ajout refusé"*
Renvoyer l'arbre contenant les variables S

Tableau 2. *Algorithme induction rapide d'arbre par niveau - S représente l'ensemble des variables déjà introduites dans l'arbre*

l'arbre. Cette opération nécessite une estimation de cette probabilité, et généralement lorsque l'on utilise une matrice de coût de mauvaise affectation unitaire, on choisira la conclusion y_{k^*} telle que :

$$y_{k^*} = \arg\max_k \ P(Y = y_k / \text{ chemin de la feuille}) \tag{1}$$

Une première solution simple consisterait à re-scanner une seconde fois la base de données pour calculer réellement les effectifs sur chaque sommet de l'arbre, en déduire les distributions conditionnelles, et ainsi, assigner les bonnes conclusions sur chaque feuille. Nous lui avons préféré une autre solution qui donne des performances équivalentes en prédiction mais qui ne nécessite pas de passage sur les données. En nous appuyant sur l'hypothèse d'indépendance conditionnelle, nous pouvons approcher la probabilité conditionnelle sur chaque feuille de l'arbre en nous appuyant sur les tableaux de comptage qui ont servi à estimer les corrélations marginales (r_{Y,X_j}). En effet, pour un chemin de l'arbre décrit par la conjonction ($X_1 = x_1, \ldots, X_p = x_p$), la probabilité $P(Y = y_k / X_1 = x_1, \ldots, X_p = x_p)$ est proportionnel à [MIT 97]

$$P(Y = y_k) \times \prod_{j=1}^{p} P(X_j = x_j / Y = y_k) \tag{2}$$

Pour une feuille de l'arbre, il suffit de calculer cette quantité (équation 2), et de lui assigner la conclusion y_{k^*} correspondant au maximum.

Malgré les critiques que l'on pourrait formuler sur la crédibilité de l'hypothèse introduite (indépendance conditionnelle), de nombreuses études ont montré que les résultats obtenus à l'aide de cette procédure sont de très bonne qualité [LEW 98]. Certains études empiriques ont montré la robustesse de cette solution sur des exemples

synthétiques où manifestement l'hypothèse était erronée [RIS 95]. Des auteurs attribuent cette bonne tenue au fait que l'équation 2 peut, dans certains cas, mal estimer la probabilité conditionnelle $P(Y = y_k/X \ldots)$; en revanche, le maximum de cette équation est très souvent un bon estimateur du maximum de la probabilité conditionnelle (équation 1). Or, l'affectation de la conclusion à une feuille repose bien sur une recherche de la probabilité conditionelle maximum sur la feuille [DOM 97].

4.4. *Complexité de calcul*

L'élaboration de l'arbre de décision repose sur deux étapes distinctes : dans un premier temps, l'évaluation des coeffcients de corrélations bruts, la complexité associée est de l'ordre de $O(n \times p^2)$, n est le nombre d'individus dans la base ; dans un deuxième temps, la recherche de l'arbre de décision optimal est, dans le pire des cas, de complexité $O(p^2)$. Comparé à la complexité d'un algorithme de construction d'arbre de décision qui est de $O(p \times n \times \log n)$, dans l'hypothèse où le nombre de sommets dans l'arbre est de $\log n$ [WIT 00], notre algorithme de construction d'arbres contraints ne semble pas réellement se démarquer.

En réalité, la simple évaluation des algorithmes par les complexités de calcul donne une indication fallacieuse des performances de l'algorithme. En effet, il faut distinguer le cas où l'on doit lire les données sur le disque, et le cas où l'on accède à une information en mémoire. Généralement, l'accès à une information dans une base de données est réalisé enregistrement par enregistrement. Lors de l'accès à un enregistrement, le moteur de base de données lit toute la ligne dans un tampon mémoire. De fait, nonobstant les aspects spécifiques aux systèmes de caches, on peut raisonnablement schématiser en disant que le passage d'un enregistrement à l'autre correspond à un accès disque ; alors que le passage d'une variable à une autre, pour un enregistrement donné, s'effectue en mémoire. Il en ressort ainsi que notre approche recquiert n accès à la base (au disque), alors qu'un algorithme d'induction d'arbre classique effectue $n \times \log n$ accès, ce qui est très pénalisant lorsque le coût d'accès au disque est comparativement très élevé par rapport à l'accès mémoire, ou lorsque les données doivent transiter par un réseau.

5. Expérimentations

Nous poursuivons deux objectifs dans cette expérimentation. D'une part, nous souhaitons vérifier si la qualité de la prédiction de nos arbres par niveaux est comparable à celle des arbres de décision classiques, nous avons choisi comme référence la méthode C4.5 bien connue [QUI 93]. D'autre part, nous souhaitons vérifier que la procédure d'optimisation aboutit bien à un apprentissage, à savoir l'arbre que nous construisons se démarque suffisamment de l'arbre complet, incluant tous les attributs. Cet arbre complet, pour lequel nous estimons les distributions de probabilité sur les feuilles en utilisant l'hypothèse d'indépendance conditionnelle, correspond au modèle bayesien naïf [MIT 97].

Base	Examples	Err. init	Arbre complet	Arbre par niveaux	C4.5
Adult	48842	0.24	14 (0.161)	7 (0.143)	13 (0.142)
Autos	205	0.55	25 (0.248)	7 (0.203)	5 (0.247)
Breast noisy	699	0.345	18 (0.037)	8 (0.030)	6 (0.063)
Dermatology	366	0.70	34 (0.102)	16 (0.059)	8 (0.0386)
Heart	270	0.44	13 (0.169)	7 (0.172)	7 (0.2296)
Iris	150	0.66	4 (0.060)	2 (0.028)	2 (0.040)
Monks-1	556	0.50	6 (0.254)	1 (0.254)	6 (0.075)
Monks-3	554	0.48	6 (0.036)	2 (0.036)	4 (0.011)
Mushroom	8416	0.47	22 (0.004)	13 (0.007)	7 (0.001)
Segmentation	2310	0.86	11 (0.163)	6 (0.110)	4 (0.194)
Wave noisy	5000	0.67	40 (0.224)	14 (0.219)	8 (0.268)

Tableau 3. *Caractéristiques des bases utilisées et résultats - Nombre de niveaux dans l'arbre (Taux d'erreur en validation croisée)*

Les bases utilisées sont issues du serveur UCI [BAY 99]. Nous utilisons un ensemble assez hétérogène de bases pour évaluer le comportement de notre algorithme dans différents cas de figure. Certaines d'entre elles sont issues de problèmes réels (adult, autos, dermatology, heart, iris) ; d'autres sont des données artificielles construites (monks-1, monks-3, mushroom) ; d'autres enfin sont des bases sur lesquelles nous avons additionné des attributs construits aléatoirement (wave noisy, breast cancer noisy). Afin de mettre tous les algorithmes sur un pied d'égalité, tous les attributs continus ont été discrétisés à l'aide de l'algorithme FUSINTER [ZIG 98]. Le taux d'erreur est évalué en utilisant une procédure de validation croisée répétée (20 fois une 10-validation croisée) [DIE 98].

Plusieurs points retiennent notre attention :

– par rapport à l'arbre complet, la réduction du nombre de variables introduites dans l'arbre est significatif, sans déterioration de la qualité de la prédiction. Dans certains cas même (adult, dermatology, iris), l'erreur diminue ;

– sur les bases bruitées (wave noisy, breast cancer noisy), les attributs qui n'ont aucun rapport avec le problème étudié sont évacuées ;

– par rapport à l'arbre classique (C4.5), notre algorithme se comporte honorablement mis à part sur les bases monks et mushroom. On constate néamoins que notre arbre est dans la plupart des cas plus complexe (la profondeur pour C4.5 correspond à la profondeur de la feuille la plus basse dans l'arbre). Un dispositif de post-élagage réduirait vraisemblablement cette complexité, reste à savoir si ceci se fera au détriment des performances (sec. 6.2).

Ces résultats semblent indiquer en tous les cas que le critère $\overline{R}^2$ que nous voulons optimiser permet de trouver un bon compromis entre la complexité de l'arbre et sa précision, ceci malgré une procédure d'optimisation gloutonne. Par ailleurs, l'arbre que nous construisons, malgré le biais supplémentaire introduit par la construction par niveaux, se comporte très honorablement en termes de performances, au prix certes

d'une complexité plus élevée. Ces éléments nous indique d'ores et déjà les prochaines pistes que nous devons explorer dans le futur : quelle sera l'efficacité de procédures d'optimisation plus élaborées (*backward*, recherche en avant, etc.) ; est-il possible de trouver des procédures de post-élagage drastiques, qui permettront de réduire le taille de l'arbre sans en altérer les performances en prédiction.

6. Discussion et travaux similaires

6.1. *Travaux similaires*

La construction d'arbres de décision classiques par optimisation est un domaine bien connu. Depuis le début des années 90, plusieurs approches ont été proposées. La plupart reposent toujours sur l'algorithme générique de construction d'arbres mais introduisent des mesures d'évaluations globales de l'arbre mettant en balance la complexité de l'arbre et sa capacité à "coller" aux données. Les approches les plus cohérentes s'appuient sur des formulations bayesiennes [BUN 92], d'autres intègrent la théorie de la description minimale des messages [QUI 89]. Notre approche se démarque sur plusieurs points par rapport à ces travaux : elle prend son essence dans la régression pas à pas et utilise un arbre avec un biais de représentation plus fort (la construction par niveau) ; sa construction repose sur un calcul de proche en proche de la mesure à optimiser, il ne requiert pas de passages supplémentaires sur la base de données ; l'assignation de la conclusion sur les feuilles repose sur une hypothèse supplémentaire d'indépendance conditionnelle. Il n'en reste pas moins qu'au final, il s'agit de trouver l'arbre optimal au sens d'un critère tentant de trouver le juste milieu entre la complexité de la structure de l'arbre, et la spécialisation réalisée sur les données.

Concernant la construction d'arbres par niveau, les motivations ont été diverses. Pormorski et Perche [POM 01] ont largement étudié différentes manières de construire un arbre de décision, de manière descendante ou ascendante, par niveau ou par noeud (classique). Leur principal apport a été de s'appuyer sur les approches par niveau pour améliorer les performances des arbres classiques. Aucune expérimentation approfondie ne vient cependant confirmer ou infirmer la supériorité des arbres classiques, les auteurs pensent tout simplement que le biais de représentation plus rigide des arbres par niveau les rendent moins avantageux.

Dans le cadre de la sélection de variables, FOCUS [ALM 91] et FGMIS [LEE 99] proposent de filtrer les variables prédictives de manière à ne retenir que les plus intéressantes en utilisant un arbre de décision par niveau. FGMIFS, par exemple, choisit la variable dont le gain moyen sur l'ensemble des segmentations sur chaque feuille est le meilleur. Cette méthode bute très rapidement sur un problème rédhibitoire, la fragmentation des données : beaucoup de sommets sont vides, il est impossible de calculer réellement le gain induit par une segmentation.

D'autres travaux ont exploité une des caractéristiques marquantes des arbres par niveau, la possibilité de le construire à partir des feuilles, de manière ascendante.

L'arbre, intégrant toutes les variables, est maximal au départ, l'induction consiste à supprimer au fur et à mesure les niveaux (les variables prédictives) qui ne sont pas pertinentes. En partant du bas, c'est-à-dire lorsque toutes les variables predictives sont présentes dans l'arbre, il est plus aisé de tenir compte des interactions entre variables. OBLIVION aboutit ainsi à un arbre réduit [LAN 94], OODG en revanche propose au final un graphe de décision en fusionnant certains sommets de l'arbre [KOH 94].

D'autres algorithmes sont proches du nôtre dans leur esprit bien que leurs auteurs ne fassent pas mention, explicitement, de la construction d'arbre par niveau. Les algorithmes MIFS [BAT 94] et CFS [HAL 00] cherchent à calculer l'apport d'information sur la classe Y d'une nouvelle variable X par rapport à un sous-ensemble de variables déjà sélectionnées. L'approche repose sur le principe suivant : un bon sous-ensemble de variables prédictives est un ensemble de variables fortement corrélées avec la variable à prédire mais qui sont très peu corrélées entre elles. Les principales différences avec nos travaux tiennent alors sur deux mesures utilisées dans le processus de sélection : celle utilisée pour calculer le lien entre les variables deux à deux, et celle qui est mise en oeuvre pour évaluer le pouvoir de prédiction d'un ensemble de variables, c.-à-d. la mesure que l'on aura à optimiser dans le processus de sélection de variables. MIFS s'appuie sur l'information mutuelle et propose une évaluation agrégée qui malheureusement repose sur un paramétrage assez difficile ; CFS propose une mesure d'entropie normalisée que son auteur interprète comme une corrélation, et la mesure agrégée, qu'il baptise MERIT, propose un compromis entre la corrélation des prédictives avec l'attribut à prédire et leurs corrélations reciproques.

Le travail de [SAP 75] est assez ancien mais également proche du nôtre. Son idée principale est toujours de calculer les coefficients partiels à partir des coefficients bruts. Mais à la différence de notre approche où l'on cherche un coefficient qui vérifie certaines propriétés, notamment une formule de passage de proche en proche, son auteur a défini un coefficient à partir de la formule de passage. Le principal reproche que l'on peut lui adresser dans ce cadre est que la mesure ainsi construite ne vérifie pas les propriétés désirées (non-nulle dans le cas de l'indépendance conditionnelle), et surtout de produire des coefficients partiels qui sortent du domaine de définition du coefficient brut (le coefficient de Tschuprow est défini entre 0 et 1 ; le Tschuprow partiel, tel qu'il est défini par son auteur, peut prendre des valeurs négatives), cas difficilement interprétables.

Enfin, l'amélioration de la capacité de l'algorithme à appréhender les grandes bases de données (la *scalabilité*) a été une de nos principales préoccupations dans ce travail. Cette question, cruciale dans le cadre de la Fouille de Données où la volumétrie des données est très importante, a donné lieu à peu de développements dans le cadre des arbres de décision classiques qui, nécessitant un accès répété aux données, sont par nature peu *scalables*. Les principaux travaux recensés ont surtout visé à réduire la nécessité de trier plusieurs fois les données et préparer des structures efficaces d'accès aux attributs prédictifs [SHA 96]. Notre algorithme se dégage naturellement de cette contrainte, il est naturellement *scalable* dans la mesure où le principal goulot d'étranglement, l'accès aux données, est entièrement levé car, d'une part, une seule

passe sur les données est nécessaire, d'autre part, ce passage se fait naturellement par une lecture séquentielle de la base de données. La contrepartie de cette rapidité est l'obligation de nous restreindre à une famille particulière des arbres de décision, les arbres de décision par niveau. Nous avons vu cependant plus haut que cette restriction a peu de repercussions sur la qualité de la prédiction du classifieur induit.

6.2. *Travaux futurs*

L'algorithme que nous présentons dans cet article laisse entrevoir une excellente tenue face aux algorithmes classiques d'induction d'arbre. Il n'en reste pas moins que la méthode proposée ici est très simple compte tenu des spécifications qui ont été proposées. Plusieurs pistes inspirées des résultats obtenus dans le cadre de l'induction d'arbres dits "classiques" sont susceptibles d'en améliorer les performances. Il est d'ores et déjà envisageable de les recenser et d'effectuer une projection sur les résultats que l'on pourrait obtenir :

– réduction du temps de calcul en estimant les coefficients bruts à partir d'un échantillon de la base, cette approche a donné des résultats très intéressants dans le cadre de l'induction d'arbres classiques [CHA 00]. L'argument est moins décisif dans notre cas dans la mesure où l'on ne procède qu'à un seul passage sur la base de données ;

– amélioration des capacités de recherche en introduisant des méthodes d'exploration plus puissantes que la simple optimisation gloutonne. Parmi les différentes approches proposées, la recherche en avant a souvent été citée [MUR 95]. Cette variante ne pose aucun problème pour notre part, la recherche en avant rend l'algorithme plus complexe et requiert plus de mémoire mais n'induit en rien un accès supplémentaire à la base de données. Les premiers essais que nous avons menés dans ce domaine montre que la recherche en avant à deux niveaux permet effectivement d'améliorer les résultats sur les bases artificielles (monks), le gain est en revanche moins convaincant sur les bases réelles. Des études supplémentaires sont nécessaires pour explorer la portée de cette piste ;

– réduction de l'arbre de décision via un dispositif de post-élagage. Cette approche est très facilement implémentable dans notre procédure, au prix néanmoins d'un second passage sur la base de données. Il est possible dès lors de procéder en trois étapes : (1) injection des individus de la base dans la structure d'arbre, il est ainsi possible de calculer les "vraies" distributions de probabilités sur chaque sommet, (2) éliminer les sommets ne contenant aucun individu, et (3) réduire la taille de l'abre en utilisant les techniques de post-élagage répandues [BRE 84]. Les principaux résultats dans l'étude des arbres de décision classiques ont montré que la réduction de l'arbre, et donc sa lisibilité, est réelle avec ce type de technique, en revanche, la réduction du taux d'erreur en généralisation risque de ne pas être décisive [BRE 97]. En ce qui concerne notre approche, le vrai goulot d'étranglement en termes de temps de calcul est le second accès à la base de données, la recherche de l'arbre élagué est relativement rapide.

7. Conclusion

Dans cet article, nous avons présenté un algorithme rapide d'induction d'arbres de décision, arbres qui présentent la spécificité d'être construits niveaux par niveaux. Nos expérimentations montrent que ce biais supplémentaire ne détériore en rien la qualité de la prédiction, bien au contraire, tandis que le temps de calcul est considérablement réduit dans la mesure où un seul passage sur les données est nécessaire pour initier la construction de l'arbre, rendant notre algorithme particulièrement bien adapté pour le traitement de grandes bases de données.

Notre travail s'appuie en grande partie sur l'analogie avec la régression pas à pas bien connue en statistique, elle nous a permis de traduire la problématique de l'induction en une problématique d'optimisation. Notre originalité réside dans l'existence d'une formule de passage nous permettant de calculer de proche en proche les coefficients partiels sans avoir à revenir sur la base de données, et la proposition de nouvelles formulations du coefficient de corrélation brut sur données catégorielles multivaluées.

Ce premier travail ouvre plusieurs pistes intéressantes que nous explorerons dans le futur. En nous inspirant des résultats obtenus, tant du côté de la régression, que du côté de l'induction d'arbres de décision classiques, notamment en introduisant la possibilité de retour en arrière lors du choix des variables, nous pourrons enrichir les résultats présentés ici.

8. Bibliographie

[ALM 91] ALMUALLIM H., DIETTERICH T., « Learning with many irrelevant features », *Proceedings of the 9^{th} National Conference on Artificial Intelligence*, 1991, p. 547-552.

[BAT 94] BATTITI R., « Using mutual information for selecting features in supervised neural net learning », *IEEE Transactions on Neural Networks*, vol. 5, n° 4, 1994, p. 537-550.

[BAY 99] BAY S., « The UCI KDD Archive [http ://kdd.ics.uci.edu] », Irvine, CA : University of California, Department of Computer Science, 1999.

[BRE 84] BREIMAN L., FRIEDMAN J., OLSHEN R., STONE C., *Classification and Regression Trees*, California : Wadsworth International, 1984.

[BRE 96] BREIMAN L., « Bagging predictors », *Machine Learning*, vol. 24, 1996, p. 123-140.

[BRE 97] BRESLOW L. A., AHA D. W., « Simplifying decision trees : a survey », *Knowledge Engineering Review*, vol. 12, n° 1, 1997, p. 1–40.

[BUN 92] BUNTINE W., « Learning classification trees », *Statistics and Computing*, vol. 2, 1992, p. 63-73.

[CEL 94] CELEUX G., NAKACHE J., *Analyse discriminante sur variables qualitatives*, Polytechnica, 1994.

[CHA 00] CHAUCHAT J., RAKOTOMALALA R., « A new sampling strategy for building decision trees from large databases », *Proceedings of the 7th Conference of the International Federation of Classification Societies, IFCS'2000*, Springer Verlag, 2000, p. 199-204.

[DIE 98] DIETTERICH T., « Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms », *Neural Computation*, vol. 10, n° 7, 1998, p. 1895–1924.

[DOM 97] DOMINGOS P., PAZZANI M., « On the optimality of the simple bayesian classifier under zero-one loss », *Machine Learning*, , n° 29, 1997, p. 103-130.

[HAL 00] HALL M. A., « Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning », *Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 2000, p. 359–366.

[KOH 94] KOHAVI R., « Bottom-up induction of oblivious read-once decision graphs », *Proceedings of the European Conference on Machine Learning*, 1994.

[LAL 00a] LALLICH S., « Diversité, association brute et association partielle », *Proceedings of $XXXII^{mes}$ Journées de Statistique*, May 2000.

[LAL 00b] LALLICH S., RAKOTOMALALA R., « Fast feature selection using partial correlation for multi-valued attributes », *Proceedings of the 4th European Conference on Knowledge Discovery in Databases, PKDD'2000*, September 2000, p. 221-231.

[LAN 94] LANGLEY P., SAGE S., « Oblivious decision trees and abstract cases », *Working Notes of the AAAI-94, Workshop on Case-Based Reasoning*, 1994.

[LEE 99] LEE K.-C., « A technique of dynamic feature selection using the feature group mutual information », *Proceedings of the Third PAKDD-99*, 1999, p. 138-142.

[LEW 98] LEWIS D. D., « Naive (Bayes) at forty : The independence assumption in information retrieval. », NÉDELLEC C., ROUVEIROL C., Eds., *Proceedings of ECML-98, 10th European Conference on Machine Learning*, n° 1398, Chemnitz, DE, 1998, Springer Verlag, Heidelberg, DE, p. 4–15.

[MAD 92] MADDALA G., *Introduction to Econometrics*, Mac Millan, London, 2nd édition, 1992.

[MIT 97] MITCHELL T., *Machine learning*, McGraw Hill, 1997.

[MUR 95] MURTHY S., SALZBERG S., « Lookahead and pathology in decision tree induction », *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'95*, 1995.

[POM 01] POMORSKI D., PERCHE P., « Inductive learning of decision trees : application to fault isolation of an induction motor », *Engineering Applications of Artificial Intelligence*, vol. 14, n° 2, 2001, p. 155-166.

[QUI 79] QUINLAN J., « Discovering rules by induction from large collections of examples », MICHIE D., Ed., *Expert Systems in the Microelectronic Age*, Edinburgh, 1979, Edinburgh University Press, p. 168–201.

[QUI 89] QUINLAN J., RIVEST R., « Inferring decision trees using the minimum description length principle », *Information and Computation*, vol. 80, 1989, p. 227-248.

[QUI 93] QUINLAN J., *C4.5 : Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.

[RIS 95] RISH I., « An empirical study of the naive bayes classifier », *Proceedings of IJCAI'01 - Workshop "Empirical Methods in Artificial Intelligence"*, 1995.

[SAP 75] SAPORTA G., « Liaisons entre plusieurs ensembles de variables et codage de données qualitatives », PhD thesis, 1975.

[SEB 00] SEBBAN M., NOCK R., CHAUCHAT J., RAKOTOMALALA R., « Impact of Learning Set Quality and Size on Decision Tree Performances », *International Journal of Computers*,

Systems and Signals (IJCSS), vol. 1, n° 1, 2000, p. 85-105.

[SHA 96] SHAFER J. C., AGRAWAL R., MEHTA M., « SPRINT : A Scalable Parallel Classifier for Data Mining », VIJAYARAMAN T. M., BUCHMANN A. P., MOHAN C., SARDA N. L., Eds., *Proc. 22nd Int. Conf. Very Large Databases, VLDB*, Morgan Kaufmann, 3–6 1996, p. 544–555.

[WIT 00] WITTEN I., FRANK E., *Data Mining : practical machine learning tools and techniques with JAVA implementations*, Morgan Kaufmann, 2000.

[ZIG 98] ZIGHED D., RABASEDA S., RAKOTOMALALA R., « FUSINTER : a method for discretization of continuous attributes for supervised learning. », *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, n° 33, 1998, p. 307-326.

[ZIG 00] ZIGHED D., RAKOTOMALALA R., *Graphes d'Induction - Apprentissage et Data Mining*, Hermes, 2000.