Vers une Automatisation de la Construction de Variables pour la Classification Supervisée

Marc Boullé *, Dhafer Lahbib *

* Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion
marc.boulle@orange.com,
http://perso.rd.francetelecom.fr/boulle/

Résumé. Dans cet article, nous proposons un cadre visant à automatiser la construction de variables pour l'apprentissage supervisé, en particulier dans le cadre multi-tables. La connaissance du domaine est spécifiée d'une part en structurant les données en variables, tables et liens entre tables, d'autre part en choisissant des règles de construction de variables. L'espace de construction de variables ainsi défini est potentiellement infini, ce qui pose des problèmes d'exploration combinatoire et de sur-apprentissage. Nous introduisons une distribution de probabilité a priori sur l'espace des variables constructibles, ainsi qu'un algorithme performant de tirage d'échantillons dans cette distribution. Des expérimentations intensives montrent que l'approche est robuste et performante.

1 Introduction

Dans un projet de fouille de données, la phase de préparation des données vise à extraire une table de données pour la phase de modélisation (Pyle, 1999), (Chapman et al., 2000). La préparation des données est non seulement coûteuse en temps d'étude, mais également critique pour la qualité des résultats escomptés. La préparation repose essentiellement sur la recherche d'une représentation pertinente pour le problème à modéliser, recherche qui se base sur des étapes complémentaires de construction et de sélection de variables. La sélection de variables a été largement étudiée dans la littérature (Guyon et al., 2006). Dans ce papier, nous nous focalisons sur l'approche filtre, qui évalue la corrélation entre les variables explicatives et la variable cible indépendamment de la méthode de classification utilisée, et est adaptée à la phase de préparation des données dans le cas d'un grand nombre de variables descriptives.

La construction de variables (Liu et Motoda, 1998) est un sujet nettement moins étudié dans la littérature scientifique, qui représente néanmoins un travail considérable pour l'analyste de données. Celui-ci exploite sa connaissance du domaine pour créer de nouvelles variables potentiellement informatives. En pratique, les données initiales sont souvent issues de bases de données relationnelles et ne sont pas directement exploitables pour la plupart des techniques de classification qui exploitent un format tabulaire attributs-valeurs. La fouille de données relationnelle, en anglais Multi-Relational Data Mining (MRDM), introduit par (Knobbe et al., 1999) vise à exploiter directement le formalisme multi-tables, en transformant la représentation relationnelle. En programmation logique inductive (ILP) (Džeroski et Lavrač, 2001), les données sont recodées sous forme de prédicats logiques. D'autres méthodes, dénommées

propositionalisation Kramer et al. (2001) effectuent une mise à plat au format tabulaire de la représentation multi-tables par création de nouvelles variables. Par exemple, la méthode Relaggs (Krogel et Wrobel, 2001) exploite des fonctions de type moyenne, médiane, min, max pour résumer les variables numériques des tables secondaires ou des comptes par valeur pour les variables catégorielles secondaires. La méthode Tilde (Blockeel et al., 1998), (Vens et al., 2006) permet de construire des agrégats complexes exploitant des conjonctions de condition de sélection d'individus dans les tables secondaires. L'expressivité de ces méthodes se heurte néanmoins aux problèmes suivants : complexité du paramétrage de la méthode, explosion combinatoire du nombre de variables construites difficile à maitriser et risque de sur-apprentissage croissant avec le nombre de variables produites.

Dans ce papier, nous proposons un cadre visant à automatiser la construction de variables, avec un objectif de simplicité, de maîtrise de la combinatoire de construction des variables et de robustesse vis-à-vis du sur-apprentissage. La partie 2 présente un formalisme de description d'un domaine de connaissance, basé sur un format des données multi-tables en entrée et une liste des règles de construction de variables. La partie 3 introduit un critère d'évaluation des variables construites selon un approche Bayesienne, en proposant une distribution a priori sur l'espace des variables constructibles. La partie 4 analyse le problème d'échantillonnage dans cet espace et propose un algorithme efficace et calculable pour produire des échantillons de variables construites de taille désirée. La partie 5 évalue l'approche sur de nombreux jeux de données. Finalement, la partie 6 conclut cet article et propose des pistes de travaux futurs.

2 Spécification d'un domaine de construction de variables

Nous proposons dans cette section un cadre formel pour spécifier un domaine de connaissance permettant de piloter efficacement la construction de variables potentiellement utiles pour la classification. L'objectif n'est pas ici de proposer un nouveau formalisme expressif et généraliste de description de connaissance, mais simplement de préciser le cadre sur lequel s'appuieront les algorithmes de construction de variables présentés dans la partie 4. Ce domaine de connaissance se décline en deux parties : description de la structure des données et choix des règles de construction de variables utilisables.

2.1 Structure des données

La structure la plus simple est la structure tabulaire. Une instance est représentée par une liste de variables, chacune étant définie par son nom et son type. Les types usuels, numérique ou catégoriel, peuvent être étendus à d'autres types spécialisés, comme par exemple les dates, les heures ou les textes. Les données réelles étant souvent issues de bases de données relationnelles, il est naturel d'étendre cette structure au cas multi-tables. On propose ici de prendre en compte ces structures en s'inspirant des langages informatiques structurés ou orientés objets. L'objet d'étude statistique (instance) appartient à une table *principale*. Un objet principal est alors défini par une liste de variables, dont les types peuvent être simple (numérique, catégoriel...) comme dans le cas tabulaire, ou structuré : sous-objets (enregistrements d'une table secondaire en relation 0-1) ou tableau de sous-objets (enregistrement d'une table secondaire en relation 0-n). Dans le cas de la classification supervisée, la variable cible est une variable catégorielle de l'objet principal. La figure 1 présente un exemple de l'utilisation de ce formalisme.

L'objet principal est le client (Customer), avec des objets secondaires usages en relation 0-n. Les variables sont de type simples (Cat, Num ou Date) ou structurés (ObjectArray(Usage)). Les variables identifiant (ici préfixées par #) servent essentiellement à établir le lien avec une base relationnelle ; elles ne sont pas considérées comme des variables descriptives.

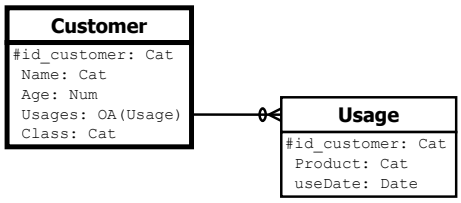

FIG. 1 – *Structure des données pour un problème de gestion de la relation client*

2.2 Règle de construction de variables

Une règle de construction de variable est une règle de calcul similaire à une fonction (ou méthode) d'un langage informatique. Elle est définie par son nom, la liste de ses opérandes, et son code retour. Les opérandes ainsi que le code retour sont typés, selon les mêmes types que pour les variables utilisées pour la spécification de la structure des données. Par exemple, la règle YearDay(Date)->Num permet de construire une variable numérique à partir d'une variable de type date. L'opérande peut soit provenir d'une variable native (présente dans la représentation de départ), soit être le résultat d'une autre règle de calcul, ou dans le cas particulier de la règle de sélection (TableSelection) provenir d'un intervalle ou d'une valeur issue de la base d'apprentissage. Dans cet article, les règles de construction prises en compte dans les expérimentations de la partie 5 sont les suivantes.

- TableSelection(ObjectArray, Num)->ObjectArray : sélection d'objets du tableau selon une conjonction de critères d'appartenance à des intervalles ou d'égalité à des valeurs
- TableCount(ObjectArray)->Num : effectif du tableau
- TableMode(ObjectArray, Cat)->Cat : valeur la plus fréquente
- TableCountDistinct(ObjectArray, Cat)->Num : nombre de valeurs différentes
- TableMean(ObjectArray, Num)->Num : valeur moyenne
- TableMedian(ObjectArray, Num)->Num : valeur médiane
- TableMin(ObjectArray, Num)->Num : valeur minimum
- TableMax(ObjectArray, Num)->Num : valeur maximum
- TableStdDev(ObjectArray, Num)->Num : écart type
- TableSum(ObjectArray, Num)->Num : somme des valeurs

En exploitant la structure des données présentée sur la figure 1 et les règles de construction précédentes (enrichies ici de la règle YearDay pour les dates), on peut par exemple construire les variables suivantes pour enrichir la description d'un client :

- MainProduct = TableMode(Usages, Product),
- LastUsageYearDay = TableMax(Usages, YearDay(useDate)),
- NbUsageProd1FirstQuarter = TableCount(TableSelection(Usages, YearDay(useDate) $\in$ [1 ;90] and Product = "Prod1").

3 Evaluation des variables construites

Il s'agit d'exploiter les connaissances du domaine pour piloter efficacement la construction de variables potentiellement utiles pour la prédiction de la variable cible. Dans le formalisme introduit en partie 2, la structure des données peut avoir plusieurs niveaux de profondeur, voire posséder la structure d'un graphe. Par exemple, une molécule est un graphe dont les noeuds sont les atomes et les arcs sont les liaisons entre atomes. Les règles de calcul peuvent être utilisées comme opérandes d'autres règles, conduisant à des formules de calcul de longueur quelconque. On est alors confronté à un espace de variables constructibles en nombre potentiellement infini. Cela pose les deux problèmes principaux suivants :

1. explosion combinatoire pour l'exploration de cet espace,
2. risque de sur-apprentissage.

On propose une solution à ces problèmes en introduisant un critère d'évaluation des variables selon une approche Bayesienne permettant de pénaliser les variables complexes. On introduit à cet effet une distribution a priori sur l'espace de toutes les variables, et un algorithme efficace d'échantillonnage de l'espace des variables selon leur distribution a priori.

3.1 Evaluation d'une variable

La construction de variables vise à enrichir la table principale par de nouvelles variables qui seront prises en entrée d'un classifieur. Les classifieurs usuels prenant en entrée uniquement des variables numériques ou catégorielles, on ne s'intéresse qu'à l'évaluation de ces variables.

Prétraitement supervisé. Le prétraitement supervisé MODL [1] consiste à partitionner une variable numérique en intervalles ou une variable catégorielle en groupes de valeurs, avec une estimation de densité conditionnelle constante par partie. Les paramètres d'un modèle de prétraitement sont le nombre de parties, la partition, et la distribution multinomiale des classes dans chaque partie. Dans l'approche MODL, le prétraitement supervisé est formulé en un problème de sélection de modèles, traité selon une approche Bayesienne en exploitant un a priori hiérarchique sur les paramètres de modélisation. Le meilleur modèle est le modèle MAP (maximum a posteriori). En prenant le log négatif de ces probabilités, qui s'interprète comme une longueur de codage (Shannon, 1948) dans l'approche minimum description length (MDL) (Rissanen, 1978), cela revient à minimiser la longueur de codage d'un modèle d'évaluation $M_E(X)$ (via une partition supervisée) d'une variable X plus la longueur de codage des données D_Y de classe connaissant le modèle et les données descriptives D_X.

$$c(X) = L(M_E(X)) + L(D_Y|M_E(X), D_X). \quad (1)$$

On a $c(X) \approx Nent(Y|X)$ où N est le nombre d'individus de l'échantillon d'apprentissage et $ent(Y|X)$ l'entropie conditionnelle de la variable de classe connaissant la variable descriptive. Le critère 1 et son optimisation sont détaillés dans (Boullé, 2006) pour la discrétisation supervisée et dans (Boullé, 2005) pour le groupement de valeurs supervisé.

1. Outil Khiops implémentant ces méthodes disponible en shareware sur `http://www.khiops.com`

Modèle nul et filtrage des variables. Le modèle nul $M_E(\emptyset)$ correspond au cas particulier d'une seule partie (intervalle ou groupe de valeurs) et donc de la modélisation directe des classes via une multinomiale, sans utiliser la variable descriptive. La valeur du critère $c(\emptyset)$ revient au cout d'encodage direct des classe cibles : $c(\emptyset) \approx Nent(Y)$. Le critère d'évaluation d'une variable est alors utilisé dans une approche filtre de sélection de variables Guyon et al. (2006) : seules les variables dont l'évaluation est meilleure que celle du modèle nul sont considérées comme informatives et retenues à l'issue de la phase de préparation des données.

Prise en compte de la construction des variables. Quand le nombre de variables natives ou construites devient important, le risque qu'une variable soit considérée à tort comme informative devient critique. Afin de prévenir ce risque de sur-apprentissage, on propose dans cet article d'exploiter l'espace de construction de variables décrit dans la partie 2 en introduisant une distribution a priori sur l'ensemble des modèles $M_C(X)$ de construction de variables. On obtient alors une régularisation Bayesienne pour l'obtention des variables, permettant de pénaliser a priori les variables les plus "complexes". Cela se traduit par un coût de construction $L(M_C(X))$ supplémentaire dans le critère d'évaluation des variables, qui devient celui de la formule 2.

$$c(X) = L(M_C(X)) + L(M_E(X)) + L(D_Y|M_E(X), D_X). \quad (2)$$

$L(M_C(X))$ est le log negatif de la probabilité a priori (longueur de codage) d'une variable X native ou construite, que nous définissons maintenant.

3.2 Distribution a priori des variables

Une variable à évaluer est une variable numérique ou catégorielle de la table principale, soit native, soit obtenue par application récursive des règles de construction de variables. L'espace des variables ainsi défini étant potentiellement infini, définir une probabilité a priori sur cet espace pose de nombreux problèmes et implique de nombreux choix. Afin de guider ces choix, on propose de suivre les principes généraux suivants :

1. la prise en compte de variables construites a un impact minimal sur les variables natives,
2. l'a priori est le plus uniforme possible, pour minimiser les biais,
3. l'a priori exploite au mieux les connaissances du domaine.

Cas des variables natives. Dans le cas où il n'y a pas de variables constructibles, le problème se limite au choix d'une variable à évaluer parmi K variables numériques ou catégorielles de la table principale. En utilisant un a priori uniforme pour ce choix, on a $p(M_C(X)) = 1/K$, d'où $L(M_C(X)) = \log K$.

Cas des variables construites. Dans le cas où des variables sont constructibles, il faut d'abord choisir si l'on utilise une variable native ou construite. Un a priori uniforme ($p = 1/2$) sur ce choix implique un surcoût de $\log 2$ pour les variables natives, ce qui contrevient au principe d'impact minimal sur les variables natives. On propose alors de considérer le choix d'une variable construite comme une possibilité de choix supplémentaire en plus des K variables

natives. Le coût d'une variable native devient alors $L(M_C(X)) = \log(K+1)$, avec un impact minimal de $\log(1+1/K) \approx 1/K$ par rapport au cas sans construction de variables.

Choisir une variable construite repose alors sur la hiérarchie suivante de choix :

- choix de construire une variable,
- choix de la règle de construction parmi les R règles de construction applicables (de code retour numérique ou catégoriel),
- pour chaque opérande de la règle, choix d'une variable native ou construite, dont le type est compatible avec le type de l'opérande attendu.

En utilisant un a priori hiérarchique, uniforme à chaque niveau de la hiérarchie, le coût d'une variable construite se décompose sur les opérandes de la règle $\mathcal{R}$ utilisée selon la formule récursive 3, où les variable X_{op} sont les variables natives ou construites applicables en opérandes de la règle.

$$L(M_C(X)) = \log(K+1) + \log R + \sum_{op \in \mathcal{R}} L(M_C(X_{op})). \tag{3}$$

Le cas de la règle TableSelection qui extrait une sous-table d'objets selon une conjonction de critères exploite de façon similaire une hiérarchie de choix : nombre de variables de sélection, liste des variables de sélection et pour chaque variable de sélection, choix de l'opérande de sélection (intervalle ou valeur), avec dans le cas des intervalles, choix de la granularité (nombre de partiles) et de l'intervalle (index du partile).

La figure 2 donne un exemple d'une telle distribution a priori sur l'ensemble des variables constructibles à l'aide des règles de construction TableMode, TableMin, TableMax et YearDay, pour le problème de gestion de la relation client présenté sur la figure 1. Par exemple, le coût de sélection de la variable native Age est $L(M_C(Age)) = \log 3$. Celui de la variable construite $TableMin(Usages, YearDay(Date))$ exploite une hiérarchie de choix, conduisant à $L(M_C(TableMin(Usages, YearDay(Date)))) = \log 3 + \log 3 + \log 1 + \log 1 + \log 1$.

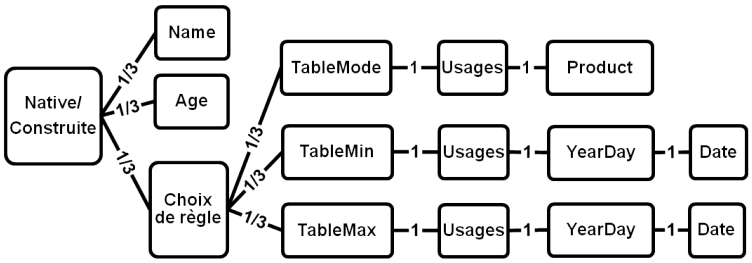

FIG. 2 – *Distribution de probabilité a priori pour la construction de variables, pour un problème de gestion de la relation client*

L'a apriori de construction de variables ainsi défini correspond à une hiérarchie de distributions multinomiales de profondeur potentiellement infinie (HDMPI). Les variables natives sont obtenues dès le premier niveau de la multinomiale, alors que les variables construites ont des probabilités a priori d'autant plus faibles qu'elles sont complexes et exploitent les parties profondes du prior HDMPI.

4 Construction d'un échantillon de variables

L'objectif est ici de construire un nombre donné de variables pour créer une représentation des données potentiellement informative pour la classification supervisée. Nous proposons de construire un échantillon de variables par tirage selon leur distribution a priori. Nous présentons un premier algorithme "naturel" de construction d'échantillon, et démontrons qu'il n'est ni efficace ni calculable. Nous proposons alors un second algorithme répondant au problème.

4.1 Tirages aléatoires successifs

Algorithm 1 Tirages aléatoires successifs

Require: K {Nombre de tirages}
Ensure: $\mathcal{V} = \{V\}, |\mathcal{V}| \leq K$ {Echantillon des variables construites}
1: $\mathcal{V} = \emptyset$
2: **for** $k = 1$ to K **do**
3: Tirer V selon le prior HDMPI
4: Ajouter V dans $\mathcal{V}$
5: **end for**

L'algorithme 1 consiste à tirer successivement K variables selon le prior HDMPI. Chaque tirage consiste à partir de la racine de l'arbre du prior et à descendre dans la hiérarchie par des tirages multinomiaux successifs, jusqu'à l'obtention d'une variable native ou construite, ce qui correspond à une feuille de l'arbre du prior. Cet algorithme naturel ne peut être utilisé dans le cas général, car il n'est ni efficace, ni calculable, ce que nous démontrons ci-dessous.

L'algorithme 1 n'est pas efficace. Soit un domaine de connaissance comportant V variables natives évaluables dans la table principale et pas de variable constructible. Le prior HDMPI se réduit à une multionomiale avec V valeurs équidistribuées. Si l'on effectue K tirages selon cette multinomiale, l'espérance du nombre de variables différentes obtenues est $V(1-e^{-K/V})$ (Efron et Tibshirani, 1993). Dans le cas ou $K = V$, on retrouve la taille d'un échantillon boostrap, à savoir $1 - 1/e \approx 63\%$ des variables obtenues avec V tirages. Pour obtenir 99% des variables, il faut $K \approx 5V$ tirages, ce qui n'est pas efficace. Si de plus, il y a des variables constructibles, la multinomiale à la racine du prior HDMPI comporte cette fois $K + 1$ choix équidistribués. Les tirages ne permettent de construire des variables que dans $1/(K + 1)\%$ des cas. Il est à noter que ce problème d'innefficacité se produit à tous les niveaux de profondeur du prior HDMPI pour le tirage des opérandes des règles en cours de construction.

L'algorithme 1 n'est pas calculable. Soit un domaine de connaissance comportant une seule variable native numérique x et une seule règle de construction f(Num, Num)->Num. Les variables que l'on peut construire sont x, $f(x, x)$, $f(x, f(x, x))$, $f(f(x, x), f(x, x))$, $f(f(x, x),$-$f(f(x, x), x))$... Le nombre de Catalan $C_n = \frac{(2n)!}{(n+1)!n!}$ permet de dénombrer de telles expressions. C_n correspond au nombre de façons différentes de placer des parenthèses autour de $n+1$ facteurs ou au nombre d'arbres binaires à $n + 1$ feuilles. Chaque variable correspondant à un arbre binaire à n feuilles (nombre de x dans la formule) vient en C_{n-1} exemplaires différents,

chacun avec un probabilité a priori de $2^{-(2n-1)}$ selon le prior HDMPI. On peut alors calculer l'espérance de la longueur $s(V)$ d'une variable calculée (en nombre de feuilles dans son arbre binaire de calcul). En utilisant la formule de Stirling pour approximer les nombres de Catalan, on établit dans la formule 4 que cette espérance est infinie.

$$E(s(V)) = \sum_{n=1}^{\infty} n 2^{-(2n-1)} C_{n-1} = \infty. \tag{4}$$

Cela signifie que si l'on tire au hasard une variable selon le prior HDMPI, parmi l'ensemble des expressions faisant intervenir f et x, l'algorithme 1 mettra en moyenne un temps infini avant de produire une variable. L'algorithme 1 n'est donc pas calculable dans le cas général.

4.2 Tirages aléatoires simultanés

Algorithm 2 Tirages aléatoires simultanés

Require: K {Nombre de tirages}
Ensure: $\mathcal{V} = \{V\}, |\mathcal{V}| \leq K$ {Echantillon des variables construites}
1: $\mathcal{V} = \emptyset$
2: Partir de la racine de la hiérarchie du prior HDMPI
3: Calculer le nombre de tirages K_i par branche du prior (variable native, règle, opérande...)
4: **for all** branche du prior **do**
5: **if** feuille terminale du prior (variable construite depuis une règle complète) **then**
6: Ajouter V dans $\mathcal{V}$
7: **else**
8: Propager la construction de façon récursive en distribuant les K_i tirages sur la multinomiale du sous-niveau
9: **end if**
10: **end for**

Comme il n'est pas possible de tirer les variables individuellement, nous proposons de créer un échantillon directement sur la base de plusieurs tirages simultanés. Dans le cas d'une loi multinomiale simple ayant K événements de probabilités $(p_1, p_2, \ldots, p_K)$, la probabilité qu'un échantillon de n tirages ait pour effectifs $n_1, n_2, \ldots, n_K$ par évènement est :

$$\frac{n!}{n_1! n_2! \ldots n_K!} p_1^{n_1} p_2^{n_2} \ldots p_K^{n_K}. \tag{5}$$

L'échantillon le plus probable est obtenu en maximisant l'expression 5, qui par maximum de vraisemblance correspond aux effectifs $n_k = p_k n$. Dans le cas par exemple d'une multinomiale équidistribuée avec $p_k = 1/K$ et de $n = K$ tirages, on remarque que l'expression 5 atteint son maximum pour $n_k = 1$ et que donc tous les évènements sont tirés, ce qui remédie au problème d'efficacité décrit en partie 4.1. L'algorithme 2 exploite ce tirage par maximum de vraisemblance récursivement. Les tirages sont répartis selon les variables natives ou construites à chaque niveau du prior HDMPI, ce qui fait que le nombre de tirages demandés diminue en descendant dans la hiérarchie du prior. En cas d'égalité entre plusieurs choix (par exemple, 1 tirage pour K variables) le choix est fait au hasard, avec priorité aux variables natives en cas

de choix entre variables natives ou construites. En répartissant les tirages par branche de la hiérarchie du prior HDMPI, avec des effectifs décroissants selon la profondeur de la hiérarchie, l'algorithme 2 est à la fois efficace et calculable. La taille de l'échantillon obtenu étant potentiellement inférieure au nombre de tirages demandés, on réitère l'appel à l'algorithme 2 en doublant le nombre de tirage demandés à chaque appel, jusqu'à obtenir le nombre de variables désiré ou qu'aucune variable supplémentaire ne soit produite entre deux appels successifs.

5 Evaluation

Nous évaluons la méthode proposée en nous focalisant sur les axes suivants : capacité à générer de grands nombres de variables sans problème d'explosion combinatoire, résistance au sur-apprentissage et apport pour la prédiction.

5.1 Bases multi-tables de petite taille

Dans cette première évaluation, nous utilisons 20 bases issues de la communauté du data mining multi-tables, en ignorant les variables de la table principale et nous focalisant sur la construction de variables par propositionalisation au moyen des règles de construction présentées dans la partie 2.2. Afin d'avoir un résultat de référence, on utilise une méthode apparentée à Relaggs (Krogel et Wrobel, 2001), basée sur l'utilisation des mêmes règles de construction (celles de la partie 2.2, excepté TableSelection), et de l'effectif par valeur pour chaque variable secondaire catégorielle. Suite à cette construction de variable, on exploite un prédicteur Bayesien naif avec sélection de variables et moyennage de modèles (SNB) (Boullé, 2007), qui est à la fois robuste et performant dans le cas de très grands nombres de variables.

Les bases utilisées [2] sont des bases issues du traitement d'image (bases Elephant, Fox, Tiger, et base Miml, avec les variables cibles Desert, Mountains, Sea, Sunset, Trees), de la chimie moléculaire (Diterpenses , Musk1, Musk2, et Mutagenesis avec trois représentations), de la médecine (Stulong, avec les variables cibles Cholrisk, Htrisk, Kourisk, Obezrisk et Rarisk), ainsi que la base TicTacToe (Asuncion et Newman, 2007) traitée en multi-tables avec les neuf cases du jeu en table secondaire. Ces bases sont de petite taille, avec de 100 à 2000 individus.

En utilisant l'algorithme 2, nous contrôlons la taille de la représentation en générant 1, 10, 100, 1000, 10000 et 100000 variables par jeu de donnée dans les échantillons d'apprentissage dans un processus en validation croisée stratifiée à 10 niveaux, ce qui représente au total environ 20 millions de variables construites.

Evaluation de la performance. Dans une première analyse, nous collectons l'AUC moyenne en test pour chaque nombre de variables générées. La méthode Relaggs, qui s'appuie sur une construction de variables par application systématique de règles, ne permet pas de contrôler la combinatoire du nombre de variables produites, qui ici varie d'une dizaine à environ 1400 variables selon la base. Elle reste néanmoins applicable dans le cas de petites bases et fournit ici une performance de base compétitive. Les résultats présentés sur la figure 3 montrent que la performance de notre approche croit systématiquement avec le nombre de variables construites,

2. Miml : `http://lamda.nju.edu.cn/data_MIMLimage.ashx`, Fox, Elephant, Tiger, Mutagenesis, Musk1, Musk2 : `http://www.uco.es/grupos/kdis/mil/dataset.html`, Diterpenses : `http://cui.unige.ch/~woznica/rel_weka/`, Stulong : `http://euromise.vse.cz/challenge2003`

et atteint ou dépasse la performance de Relaggs sur 17 des 20 bases quand le nombre de variables construites est suffisant. Pour trois des bases (Fox, Musk1 et Musk2), les performance de notre approche sont nettement moins bonnes que celles de Relaggs. Ces trois bases sont soit très bruitées (Fox avec AUC=0.7) soit très petites (Musk1 et Musk2 avec moins de 100 instances) et correspondent exactement aux trois bases avec la plus grande variance des résultats pour Relaggs (entre 10 et 15% d'écart type de l'AUC). Pour ces trois bases, le nombre d'instances est insuffisant pour compenser le coût de régularisation (critère 2) : la plupart des variables construites sont alors éliminées, ce qui fait chuter la performance.

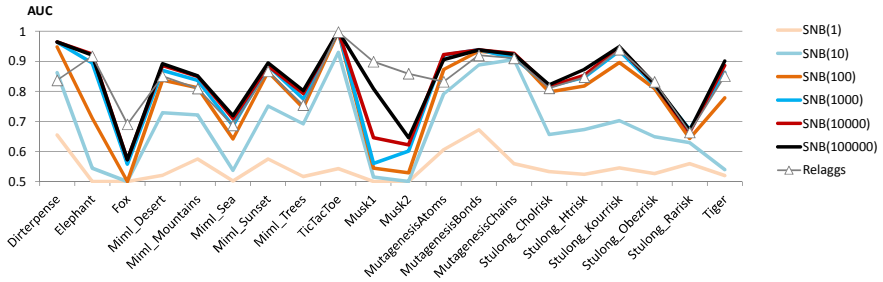

FIG. 3 – *AUC en test en fonction du nombre variables construites*

Evaluation de la robustesse. Dans une seconde analyse, l'expérience est réalisée après réaffectation aléatoire des classes pour chaque jeu de données, afin d'évaluer la robustesse de l'approche. Nous collectons le nombre de variables sélectionnées, avec (critère 2) ou sans (critère 1) prise en compte du coût de construction des variables. Sans régularisation de construction, en moyenne 0.2% des variables sont sélectionnées à tort, avec des pourcentages diminuant avec la taille de la base (de 0.8% pour les plus petites bases à 0.02% pour les plus grandes). Par exemple, pour la base Musk1 qui comprend 92 instances, plus de 700 variables parmi 100000 sont identifiées à tort comme informatives. Avec prise en compte de la régularisation de construction, la méthode est extrêmement robuste : les 20 millions de variables générées sont identifiées comme étant non informatives, sans aucune exception.

5.2 Bases de taille moyenne

Dans une seconde évaluation, nous utilisons trois bases de taille moyenne : Connect4, PokerHands et Digits (Asuncion et Newman, 2007). La base Connect4 correspond à l'ensemble des positions légales après 8 coups du jeu "Puissance 4". Il s'agit de prédire l'issue de la partie (victoire, nul, ou défaite) pour deux joueurs jouant les coups parfaits (le jeu a été "cassé"). Le format tabulaire initial est représenté sous forme multi-tables, avec une table secondaire comportant les $6 * 7 = 42$ cellules du jeu avec trois variables secondaires X, Y et Val. 70% des 67557 instances sont utilisées en apprentissage et 30% en test. La base PokerHand correspond à une main de cinq cartes au poker, avec le rang et la couleur (variables numériques) de chaque carte. Il s'agit de reconnaitre le type de main parmi 10 classes (rien, paire, double paire, brelan...). Le format tabulaire initial est représenté sous forme multi-tables, avec une table secondaire comportant les cinq cartes d'une main. 25000 instances sont utilisées en apprentissage et 1000000 en test. La base Digits est une base d'images décrites par $28 * 28 = 784$

pixels, avec pour objectif la reconnaissance d'un chiffre manuscrit de 0 à 9. Le format tabulaire initial est représenté sous forme multi-tables, avec une table secondaire comportant les 784 pixels avec trois variables secondaires X, Y et $GrayLevel$. 60000 instances sont utilisées en apprentissage et 10000 en test.

Nous évaluons le taux de bonne prédiction en test, plus discriminant ici que l'AUC. Les trois problèmes sont difficiles dans leur représentation initiale pour le prédicteur SNB, qui obtient des performances de 72.4%, 50.1% et 87.5%. La figure 4 montre que les performances de notre approche croissent avec le nombre de variables construites, et dépassent significativement celle du classifieur SNB pour la représentation initiale.

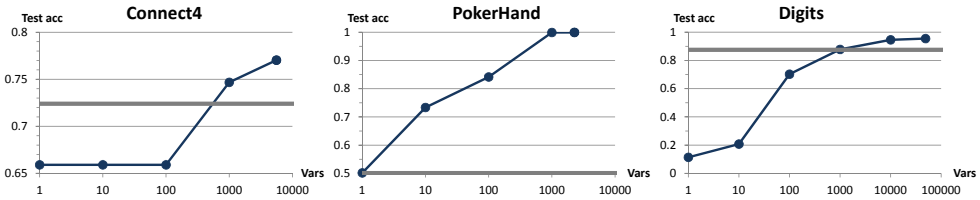

FIG. 4 – *Taux de bonne prédiction en test pour les bases Connect4, PokerHand et Digits en fonction du nombre de variables construites*

6 Conclusion

Nous avons proposé dans cet article un cadre visant à automatiser la construction de variables pour la classification supervisée. Sur la base d'une description d'un format multi-table des données et d'un choix de règles de construction de variables, nous avons défini une distribution a priori sur l'ensemble de toutes les variables constructibles au moyen de ce formalisme. Nous avons démontré qu'effectuer des tirages successifs selon cette distribution a priori pose des problèmes critiques d'efficacité et de calculabilité, puis proposé un algorithme efficace permettant d'effectuer simultanément un grand nombre de tirages de façon à construire une représentation comportant le nombre de variables désiré. Les expérimentations montrent que l'approche résout à la fois le problème de l'explosion combinatoire qui intervient dans les approches de construction de variables par application systématique de règles, et le problème de sur-apprentissage quand les représentations comportent de très grands nombres de variables. Les performances obtenues en classification sont très prometteuses. Dans des travaux futurs, nous prévoyons d'améliorer la prise en compte des connaissances du domaine, en premier lieu en étendant la liste des règles de construction disponibles avec des spécialisations potentielles par domaine applicatif. Une autre piste de travail consiste à rechercher les meilleures variables à construire, en échantillonnant leur distribution a posteriori plutôt que leur distribution a priori.

Références

Asuncion, A. et D. Newman (2007). UCI machine learning repository.

Blockeel, H., L. De Raedt, et J. Ramon (1998). Top-Down Induction of Clustering Trees. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 55–63.

Boullé, M. (2005). A Bayes optimal approach for partitioning the values of categorical attributes. *Journal of Machine Learning Research 6*, 1431–1452.

Boullé, M. (2006). MODL : a Bayes optimal discretization method for continuous attributes. *Machine Learning 65*(1), 131–165.

Boullé, M. (2007). Compression-based averaging of selective naive Bayes classifiers. *Journal of Machine Learning Research 8*, 1659–1685.

Chapman, P., J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, et R. Wirth (2000). CRISP-DM 1.0 : step-by-step data mining guide. Technical report, The CRISP-DM consortium.

Džeroski, S. et N. Lavrač (2001). *Relational Data Mining*. Springer-Verlag New York, Inc.

Efron, B. et R. Tibshirani (1993). *An introduction to the bootstrap*. Monographs on Statistics and Applied Probability ; 57. New York : Chapman & Hall/CRC.

Guyon, I., S. Gunn, M. Nikravesh, et L. Zadeh (Eds.) (2006). *Feature Extraction : Foundations And Applications*. Springer.

Knobbe, A. J., H. Blockeel, A. Siebes, et D. Van Der Wallen (1999). Multi-Relational Data Mining. In *Proceedings of Benelearn '99*.

Kramer, S., P. A. Flach, et N. Lavrač (2001). Propositionalization approaches to relational data mining. In S. Džeroski et N. Lavrač (Eds.), *Relational data mining*, Chapter 11, pp. 262–286. Springer-Verlag.

Krogel, M.-A. et S. Wrobel (2001). Transformation-based learning using multirelational aggregation. In *ILP*, pp. 142–155. Springer.

Liu, H. et H. Motoda (1998). *Feature Extraction, Construction and Selection : A Data Mining Perspective*. Kluwer Academic Publishers.

Pyle, D. (1999). *Data preparation for data mining*. Morgan Kaufmann Publishers, Inc. San Francisco, USA.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica 14*, 465–471.

Shannon, C. (1948). A mathematical theory of communication. Technical Report 27, Bell systems technical journal.

Vens, C., J. Ramon, et H. Blockeel (2006). Refining aggregate conditions in relational learning. In *Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases (PKDD 06)*, pp. 383–394. Springer-Verlag.

Summary

In this paper, we suggest a framework to automate variable construction for supervised learning, especially in the multi-relational setting. Domain knowledge is specified by describing the structure of data by the means of variables, tables and links across tables, and choosing construction rules. The space of variables that can be constructed if virtually infinite, which raises both combinatorial and over-fitting problems. We introduce a prior distribution over all the constructed variables, as well as an effective algorithm to draw samples of constructed variables from this distribution. Experiments show that the approach is robust and efficient.