Towards Automatic Feature Construction for Supervised Classification

Marc Boullé

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

Abstract. We suggest an approach 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 is 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.

Keywords: supervised learning, relational learning, feature construction, feature selection, regularization.

1 Introduction

In a data mining project, the data preparation phase aims at constructing a data table for the modeling phase [19,6]. The data preparation is both time consuming and critical for the quality of the mining results. It mainly consists in a search of an effective data representation, based on variable construction and selection. Variable selection has been extensively studied in the literature [12]. Two main approaches, filter and wrapper, have been proposed. Filter methods consider the correlation between the input variables and the output variable as a pre-processing step, independently of the chosen classifier. Wrapper methods search the best subset of variables for a given classification technique, used as a black box. Wrapper methods, which are time consuming, are restricted to the modeling phase of data mining, as a post-optimization of a classifier. Filter methods are better suited to the data preparation phase, since they can be combined with any data modeling approach and can deal with large numbers of input variables. In this paper, we focus on the filter approach, in the context of supervised classification.

Variable construction [18] has been less studied than variable selection in the literature. It implies a large amount of work for the data analyst and heavily relies on domain knowledge to construct new potentially informative variables. In practice, the initial raw data usually originate from relational databases. As most

T. Calders et al. (Eds.): ECML PKDD 2014, Part I, LNCS 8724, pp. 181–196, 2014.

classification techniques need a flat input data table with instances × variables tabular format, such relational data cannot be directly analyzed.

Learning from relational data has recently received an increasing attention in the literature. The term Multi-Relational Data Mining (MRDM) was initially introduced in [13] to address novel knowledge discovery techniques from multiple relational tables. The common point between these techniques is that they need to transform the relational representation. In Inductive Logic Programming (ILP) [9], data is recoded as logic formulas. In 1BC method [16] and its successor 1BC2 [17], simple predicates are used together with a naive Bayes classifier. More expressive approaches cause scalability problems especially with large-scale data. Other methods named by propositionalisation [14] try to flatten the relational data by constructing new variables. These variables aggregate the information contained in non target tables in order to obtain a classical tabular format. For example, the RELAGGS method [15] uses functions such as mean, median, min, max to summarize numerical variables from secondary tables in zero to many relationship, or counts per value for the categorical variables. The TILDE method [2,24] aims at constructing complex variables based on conjunctions of selection conditions of records in secondary tables. However, the expressiveness of such methods faces the following problems: complex parameter setting, combinatorial explosion of the number of potentially constructed variables and risk of over-fitting that increases with the number of constructed variables.

In this paper, we suggest an approach aiming at the automation of variable construction, with the three-fold following objective: simplicity of parameters, efficient control of the combinatorial search in the space of variable construction and robustness w.r.t. over-fitting. Section 2 presents a formal description of a variable construction domain. Section 3 introduces an evaluation criterion of the constructed variables exploiting a Bayesian approach, by suggesting a prior distribution over the space of variables that can be constructed. Section 4 studies the problem of drawing a sample from this space and describes an efficient and computable algorithm for drawing samples of constructed variables of given size. Section 5 evaluate the approach on several datasets. Finally, Section 6 gives a summary and discusses future work.

2 Specification of Variable Construction Domain

We suggest a formal specification of a variable construction domain in order to efficiently drive the construction of variables for supervised classification. The objective is not to propose a new expressive and general formalism for describing domain knowledge, but simply to clarify the framework exploited by the variable construction algorithms presented in Section 4. This framework consists in two parts: description of the data structure and choice of the construction rules.

2.1 Data Structure

The simplest data structure is the tabular one. Data instances are represented by a list of variables, each defined by its name and type. The standard types,

numerical or categorical, can be extended to other specialized types, such as date, time or text. As real data usually comes from relational databases, extending tabular format to multi-table looks natural. We suggest to describe these structures similarly to structured or object-oriented programming languages. The statistical unit (root instance) belongs to a *root* table. A root instance is then defined by a list of variables, whose type can be simple (numerical, categorical...) as in the tabular case, or structured: one record of a secondary table in zero to one relationship or several records of a secondary table in zero to many relationship. In the case of supervised classification, the output variable is a categorical variable in the root table. Figure 1 presents an example of the use of this formalism. The root instance is a Customer, with secondary records Usages in zero to many relationship. The variables are either of simple type (Cat, Num or Date) or structured type (Table(Usage)). The identifier variables (prefixed by #) are mainly used for practical purposes, in order to establish a matching with a relational database; they are not considered as input variables.

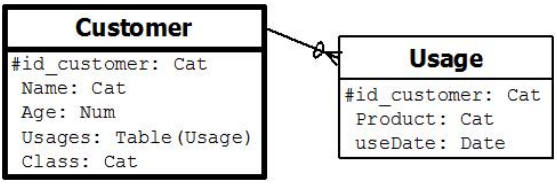

Fig. 1. Data structure for a problem of customer relationship management

2.2 Variable Construction Rules

A variable construction rule is similar to a function (or method) in a programming language. It is defined by its name, the list of its operands and its return value. The operands and the return value are typed, with the types defined in Section 2.1. For example, the *YearDay(Date)→Num* rule builds a numerical variable from a date variable. The operands can originate from an *original* variable (in the initial data representation), from the return value of another rule, or from values coming from a train dataset. In this paper, the construction rules used in the experiments of Section 5 are the following ones:

- *Selection(Table, Num)→Table*: selection of records from the table according to a conjunction of selection terms (membership in a numerical interval or in group of categorical values),
- *Count(Table)→Num*: count of records in a table,
- *Mode(Table, Cat)→Cat*: most frequent value of a variable in a table,
- *CountDistinct(Table, Cat)→Num*: number of distinct values,
- *Mean(Table, Num)→Num*: mean value,
- *Median(Table, Num)→Num*: median value,
- *Min(Table, Num)→Num*: min value,
- *Max(Table, Num)→Num*: max value,

- *StdDev(Table, Num)→Num*: standard deviation,
- *Sum(Table, Num)→Num*: sum of values.

Using the data structure presented in Figure 1 and the previous construction rules (plus the *YearDay* rule for date variables), one can construct the following variables to enrich the description of a customer:

- *MainProduct = Mode(Usages, Product)*,
- *LastUsageYearDay = Max(Usages, YearDay(useDate))*,
- *NbUsageProd1FirstQuarter = Count(Selection(Usages, YearDay(useDate) ∈ [1;90] and Product = "Prod1"))*.

3 Evaluation of Constructed Variables

The issue is to exploit the variable construction domain in order to efficiently drive the construction of variables which are potentially informative for the prediction of the output variable. In the framework introduced in Section 2, the data structure can have several level of depth or even have a graph structure. For example, a molecule is a graph where the vertices are the atoms and the edges are the bounds between atoms. The constructed rules can be used as operands of other rules, leading to computation formulas of any length. The space of constructed variables is then of potentially infinite size. This raises the two major following problems:

1. combinatorial explosion for the exploration of this construction space,
2. risk of over-fitting.

We suggest to solve these problems by introducing an evaluation criterion of the constructed variables according to a Bayesian approach in order to penalize complex variables. For this purpose, we propose a prior distribution on the space of all variables and an efficient sampling algorithm of the space of variables according to their prior distribution.

3.1 Evaluation of a Variable

Variable construction aims to enrich the root table with new variables that will be taken as input of a classifier. As usual classifiers take as input only numerical or categorical variables, only these variables need to be evaluated.

Supervised Preprocessing. The MODL supervised preprocessing methods [3,4] consist in partitioning a numerical variable into intervals or a categorical variable into groups of values, with a piecewise constant class conditional density estimation. The parameters of a specific preprocessing model are the number of parts, the partition and the multinomial distribution of the classes within each part. In the MODL approach, supervised preprocessing is turned into a model selection problem and solved in a Bayesian way. A prior distribution is proposed

on this model space. This prior exploits the hierarchy of the parameters, with a uniform distribution at each stage of the hierarchy. The methods exploit a maximum a posteriori (MAP) technique to select the most probable preprocessing model given the input data. Taking the negative log of probabilities that are no other than coding lengths [22] in the minimum description length (MDL) approach [20], this amounts to the description length of a preprocessing model $M_P(X)$ (using a supervised partition) of a variable X plus the description length of the output data D_Y given the model and the input data D_X.

$$cost_P(X) = L(M_P(X)) + L(D_Y|M_P(X), D_X). \quad (1)$$

We asymptotically have $cost_P(X) \approx Nent(Y|X)$ where N is the number of train instances and $ent(Y|X)$ the conditional entropy [7] of the output given the input variable. Formula (1) and the related optimisation algorithms are fully detailed in [4] for supervised discretization and [3] for supervised value grouping.

Null Model and Variable Filtering. The null model $M_P(\emptyset)$ corresponds to the case of a preprocessing model with one single part (interval or group of values) and thus to the direct modeling of the output values using a multinomial distribution, without using the input variable. The value of criterion $cost_P(\emptyset)$ amounts to a direct coding of the output values: the null cost is $cost_P(\emptyset) \approx Nent(Y)$, where $ent(Y)$ is the entropy of Y. The evaluation criterion of a variable is then exploited according to a filter approach [12]: only variables whose evaluation is better than the null cost are considered informative and retained at the end of the data preparation phase.

Accounting for the Variable Construction Process. When the number of original or constructed variables increases, the chance for a variable to be wrongly considered as informative becomes critical. In order to prevent this risk of over-fitting, we suggest in this paper to exploit the space of constructed variables described in Section 2 by proposing a prior distribution over the set of all variable construction models $M_C(X)$. We then get a Bayesian regularization of the constructed variables, which allows to penalize the most "complex" variables. This translates into an additional construction cost $L(M_C(X))$ in the evaluation criterion of the variables, which becomes that of Formula (2).

$$cost_{CP}(X) = L(M_C(X)) + L(M_P(X)) + L(D_Y|M_P(X), D_X). \quad (2)$$

$L(M_C(X))$ is the negative log of the prior probability (coding length) of an original or constructed variable X, defined below.

3.2 Prior Distribution of the Original and Constructed Variables

A variable to evaluate is a numerical or categorical variable in the root table, either original or built using construction rules recursively. The space of such variables being of virtually infinite size, defining a prior probability on this space

raises many problems and involves many choices. To guide these choices, we will stick to the following general principles:

1. taking into account the constructed variables has a minimal impact on the original variables,
2. in order to have a minimum bias, the prior is as uniform as possible,
3. the prior exploits at best the variable construction domain.

Case of Original Variables. In the case where no variable can be constructed, the problem reduces to the choice of an original variable to evaluate among the K numerical or categorical variables of the root table. Using a uniform prior for this choice, we obtain $p(M_C(X)) = 1/K$, thus $L(M_C(X)) = \log K$.

Case of Constructed Variables. In the case where variables can be constructed, one must first choose whether to use an original variable or to construct a new variable. Using a uniform prior ($p = 1/2$) for this choice implies an additional cost of $\log 2$, which violates the principle of minimal impact on the original variables. We then suggest to consider the choice of constructing a new variable as an additional possibility beyond the K original variables. The cost of an original variable becomes $L(M_C(X)) = \log(K + 1)$, with an additional cost of $\log(1 + 1/K) \approx 1/K$ w.r.t. the case of original variables only.

Constructing a new variable then relies on the following hierarchy of choices:

- choice of constructing a new variable,
- choice of the construction rule among the R applicable rules (with the required return value type and available operands of the required types),
- for each operand of the rule, choice of using an original variable or to construct a new variable with a rule whose return value is compatible with the expected operand type.

Using a hierarchical prior, uniform at each level of the hierarchy, the cost of a constructed variable is decomposed on the operands of the used construction rule according to the recursive Formula (3), where the variables X_{op} are the original or constructed variables used as operands *op* of the rule $\mathcal{R}$.

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

Case of the *Selection* Rule. The case of the *Selection* rule that extracts records from a secondary table according to a conjunction of selection terms is treated similarly. The hierarchy of choices is extended in the following way: number of selection operands, list of selection variables (original or constructed) and for each selection variable, choice of the selection part (numerical interval or group of categorical values). The selection part is itself chosen hierarchically with first a choice of granularity of the partitioned variable into a set of quantiles and second the index of the quantile in the partition. In definitions 1 and 2, we precisely define quantile partitions both for numerical and categorical variables.

Definition 1 (Numerical quantile partition). *Let D be a dataset of N instances and X a numerical variable. Let $x_1, x_2, \ldots, x_N$ be the N sorted values of X in dataset D. For a given number of parts P, the dataset is divided into P equal frequency intervals* $]-\infty, x_{\lfloor 1+\frac{N}{P} \rfloor}[$, $[x_{\lfloor 1+\frac{N}{P} \rfloor}, x_{\lfloor 1+2\frac{N}{P} \rfloor}[$, $\ldots$, $[x_{\lfloor 1+i\frac{N}{P} \rfloor}, x_{\lfloor 1+(i+1)\frac{N}{P} \rfloor}[$, $\ldots$, $[x_{\lfloor 1+(P-1)\frac{N}{P} \rfloor}, +\infty[$.

Definition 2 (Categorical quantile partition). *Let D be a dataset of N instances and X a categorical variable with V values. For a given number of parts P, let $N_P = \lceil \frac{N}{P} \rceil$ be the expected minimum frequency per part. The categorical quantile partition into (at most) P parts is defined by singleton parts for each value of X with frequency above the threshold frequency N_P and a "garbage" part consisting of all values of X below the threshold frequency.*

The number of selection terms is chosen according to the universal prior for integer numbers of Rissanen [21]. This prior distribution is as flat as possible, with larger probabilities for small integer numbers. Each selection variable (original or constructed) is distributed using the prior defined previously in this section. As for the granularities, we consider only powers of two $2^0, 2^1, 2^2, \ldots 2^p, \ldots$ for the sizes of the partitions, with the exponent p distributed according to the universal prior for integer numbers. Finally, the index of each quantile is distributed uniformly among the 2^p parts.

Whereas all the other rules exploit only the data structure and the set of construction rules, the *Selection* rule exploits the values of the train dataset to build the actual definition of the selection parts. This requires one reading step of each secondary table to instantiate the formal definition of each part (granularity and part index) into an actual definition, with numerical boundaries for intervals and categorical values for groups of values.

Synthesis. Figure 2 presents an example of such a prior distribution over the set of variables that can be built using the construction rules *Mode*, *Min*, *Max* and *YearDay*, in the case of the customer relationship management dataset of Figure 1. For example, the cost of selecting the original variable *Age* is

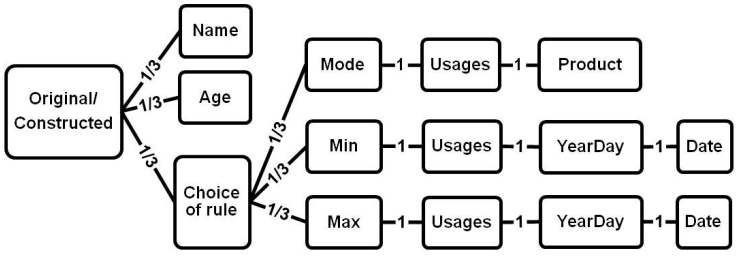

Fig. 2. Prior distribution of variable construction in the case of the customer relationship management dataset

$L(M_C(Age)) = \log 3$. That of constructing the variable with formula $Min(Usages, YearDay(Date))$ exploits of a hierarchy of choices leading to

$$L(M_C(Min(Usages, YearDay(Date)))) = \log 3 + \log 3 + \log 1 + \log 1 + \log 1.$$

This prior distribution on the space of variable construction corresponds to a Hierarchy of Multinomial Distributions with potentially Infinite Depth (HMDID). The original variables are obtained from the first level of hierarchy of the prior, whereas the constructed variables get all the more lower prior probabilities as they exploit deeper parts of the HMDID prior with complex formulas.

4 Building a Random Sample of Variables

The objective is to build a given number of variables, potentially informative for supervised classification, in order to create an input tabular representation,. We suggest to build a sample of variables by drawing them according to their prior distribution. We present a first "natural" algorithm for building samples of variables, and demonstrate that it is neither efficient nor even computable. We then propose a second algorithm that solves the problem.

4.1 Successive Random Draws

Algorithm 1. Successive random draws

Require: K {Number of draws}
Ensure: $\mathcal{V} = \{V\}, |\mathcal{V}| \leq K$ {Sample of constructed variables}
1: $\mathcal{V} = \emptyset$
2: **for** $k = 1$ to K **do**
3: Draw V according to HMDID prior
4: Add V in $\mathcal{V}$
5: **end for**

Algorithm 1 consists in successively drawing K according to the HMDID prior. Each draw starts from the root of the prior and goes down in the hierarchy until obtaining an original or constructed variable, which corresponds to a leaf in the prior hierarchy. This natural algorithm cannot be used in the general case, because it is neither efficient nor computable, as we demonstrate below.

Algorithm 1 Is not Efficient. Let us consider a construction domain with V original variables that can be evaluated in the root table and no construction rule. The HMDID prior reduces to a standard multinomial distribution with V equidistributed values. If K draws are performed according to this multinomial distribution, the expectation of the number of distinct obtained variables is $V(1-e^{-K/V})$ [11]. In the case where $K = V$, this corresponds to the size of a bootstrap

sample, that is $(1 - 1/e) \approx 63\%$ variables obtained using V draws. To obtain 99% of the original variables, one needs $K \approx 5V$ draws, which is not efficient. Furthermore, in the case with construction rules, the multinomial at the root of the HMDID consists now into $K + 1$ equidistributed values. The draws result in the construction of a new variable in only $\frac{1}{K+1}$ of the cases. It is noteworthy that this problem of inefficiency occurs at all levels of depth of the HMDID prior for the draw of the operands of the rules under construction.

Algorithm 1 Is not Computable. Let us consider a construction domain with one single numerical variable x and one single construction rule $f(Num, Num) \rightarrow Num$. The variables that can be constructed are x, $f(x, x)$, $f(x, f(x, x))$, $f(f(x, x), f(x, x))$, $f(f(x, x), f(f(x, x), x))$... In combinatorial mathematics, the Catalan number $C_n = \frac{(2n)!}{(n+1)!n!} \approx \frac{4^n}{\pi n^{3/2}}$ counts the number of such expressions. C_n is the number of different ways $n+1$ factors can be completely parenthesized or the number of full binary trees with $n + 1$ leaves. Each variable represented by a binary construction tree with n leaves (repetitions of x in the formula) comes into C_{n-1} formally distinct copies, each with a prior probability of $2^{-(2n-1)}$ according to the HMDID prior. The expectation of the size $s(V)$ of a constructed variable V (size defined by the number of leaves in the binary construction tree) can then be computed. Using the above approximation of the Catalan number, Formula (4) states that the expectation of the size of the variable is infinite.

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

This means that if one draws a random variable according to the HMDID prior among all expressions involving f and x, the drawing process will never stop on average. Algorithm 1 is thus not computable in the general case.

4.2 Simultaneous Random Draws

As variables cannot be drawn individually as in Algorithm 1, we suggest to draw a complete sample using several draws simultaneously. For a multinomial distribution $(n; p_1, p_2, \ldots, p_K)$ with n draws and K values, the probability that a sample results in counts $n_1, n_2, \ldots, n_K$ per value is:

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

The most probable sample is obtained by maximizing Formula (5), which results into counts $n_k = p_k n$ per value according to maximum likelihood. For example, in the case of an equidistributed multionomial distribution with $p_k = 1/K$ and $n = K$ draws, Formula (5) is maximized for $n_k = 1$. As a consequence, all the values are drawn, which solves the inefficiency problem described in Section 4.1. Algorithm 2 exploits this drawing process using maximum likelihood recursively.

Algorithm 2. Simultaneous random draws

Require: K {Number of draws}
Ensure: $\mathcal{V} = \{V\}, |\mathcal{V}| \leq K$ {Sample of constructed variables}
1: $\mathcal{V} = \emptyset$
2: Start from the root of the hierarchy of the HMDID prior
3: Compute the number of draws K_i per branch of the prior tree (original variable, rule, operand...)
4: **for all** branch of the prior tree **do**
5: **if** terminal leaf of the prior tree (original variable or variable constructed with a complete formula) **then**
6: Add V in $\mathcal{V}$
7: **else**
8: Propagate the construction process recursively by assigning the K_i draws of each branch on the multinomial distribution at the sub-level of the prior tree
9: **end if**
10: **end for**

The draws are assigned on the original or constructed variables at each level of depth of the HMDID prior, which results in a number of draws that decreases with the depth of the prior hierarchy. In case of even choices (for example, one single draw among K variables), the draw is chosen randomly uniformly, with a priority for original variables when both original and constructed variables are possible. By assigning recursively the draws according to multinomial distributions at each branch of the hierarchy of the HMDID prior, with numbers of draws that decrease with the depth of the hierarchy, Algorithm 2 is both efficient and computable.

In Algorithm 2, the number of draws may be greater than 1 in some leaves of the prior hierarchy. This implies that the number of obtained variables can be inferior to the number of initial draws. To reach a given number of variables K, Algorithm 2 is first called with K draws, then called again successively with twice the number of draws at each call, until the number of required variables is reached or until no additional variable is built in the last call.

5 Experiments

The proposed method is evaluated by focusing on the following points: ability to construct large numbers of variables without problem of combinatorial explosion, robustness w.r.t. over-fitting and contribution to the classification performance.

5.1 Experimental Setup

The experiment performs comparisons with alternative relational data mining methods based on propositionalisation and with inductive logic programming Bayesian classifiers. The compared methods are:

- MODL is the method described in this paper. It exploits the following construction rules (cf. Section 2.2): *Selection*, *Count*, *Mode*, *CountDistinct*, *Mean*, *Median*, *Min*, *Max*, *StdDev*, *Sum*. The only parameter (see Section 4.2) is the number of variables to construct: 1, 3, 10, 30, 100, 300, 1000, 3000 and 10000 in the experiments. The variables are constructed using Algorithm 2 then filtered using Formula (2), which accounts both for construction and preprocessing cost. The filtered variables are used as input of a selective naive Bayes classifier with variable selection and model averaging (SNB) [1] [5], which is both robust and accurate in the case of very large numbers of variables.
- RELAGGS is a method similar to the Relaggs propositionalisation method [15]. It exploits the same construction rules as MODL and exhaustively constructs all the possible variables, except for the *Selection* rule that raises combinatorial problems. Instead, RELAGGS constructs all the rules based on counts per categorical value in the secondary tables. The data preprocessing and the SNB classifier are the same as for MODL, without accounting the construction of the variables (Formula (1) is used for filtering).
- 1BC is the first-order Bayesian classifier described in [16]. It can be considered as a propositionalisation method, with one variable per value in a secondary table. To preprocess the numerical values of each table, all numerical variables are discretized into equal frequency intervals. In the experiments, we use discretisation into 1, 2, 5, 10, 20, 50, 100 and 200 bins.
- 1BC2 is the successor of 1BC described in [17]. While 1BC applies propositionalisation, 1BC2 is a true first-order classifier. [2]

Fourteen relational datasets are considered in the experiments. The Auslan, CharacterTrajectories, JapaneseVowels, OptDigit and SpliceJunction datasets come from the UCI repository [1] and are related to the recognition of Australian sign language, characters from pen tip trajectories, Japanese speakers from cepstrum coefficients of two uttered vowels, handwritten digits from a matrix of 32*32 black and white pixels, and boundaries between intron and exon in gene sequences (DNA). These sequential or time series datasets are represented with one root table and a secondary table in zero to many relationship. The Diterpenes [10], Musk1, Musk2 [8] and Mutagenesis [23] datasets are related to molecular chemistry. The Mutagenesis dataset is a graph with molecules (*lumo, logp* plus the class variable) in a root table, atoms (*element, type, charge*) as vertices and bonds (*bondtype*) as edges. The Miml dataset [25] is related to image recognition, with five different target variables. Table 1 gives a summary of these datasets.

All the experiments are performed using a stratified 10-fold cross validation. In each train fold, the variables are constructed and selected and the classifier is trained, while the test accuracy is evaluated in the test fold.

[1] The SNB classsifier is available as a shareware at `http://www.khiops.com`.

[2] 1BC et 1BC2 are available at `http://www.cs.bris.ac.uk/Research/MachineLearning/1BC/`. I am grateful to Nicolas Lachiche for providing support and advices regarding their use.

Table 1. Relational datasets: number of instances, records in the secondary tables, categorical and numerical variables, classes, and accuracy of the majority class

Dataset	Instances	Records	Cat.vars	Num.vars	Classes	Maj.
Auslan	2,565	146,949	1	23	96	0.011
CharacterTrajectories	2,858	487,277	1	4	20	0.065
Diterpenes	1,503	30,060	2	1	23	0.298
JapaneseVowels	640	9961	1	13	9	0.184
MimlDesert	2,000	18,000	1	15	2	0.796
MimlMountains	2,000	18,000	1	15	2	0.771
MimlSea	2,000	18,000	1	15	2	0.710
MimlSunset	2,000	18,000	1	15	2	0.768
MimlTrees	2,000	18,000	1	15	2	0.720
Musk1	92	476	1	166	2	0.511
Musk2	102	6,598	1	166	2	0.618
Mutagenesis	188	10,136	3	4	2	0.665
OptDigits	5,620	5,754,880	1	3	10	0.102
SpliceJunction	3,178	191,400	2	1	3	0.521

As for computational efficiency, the overhead of the construction algorithm is negligible w.r.t. the overall training time. Actually, Algorithm 2 consists in drawing a sample of constructed variables with their construction formulas. This algorithm mainly relies on the exploration of the construction domain (data structure and set of construction rules). The *Selection* rule requires one reading step of each secondary table to build the actual selection operands. This reading step dominates the time of the variable construction process, and is itself dominated by the data preparation and modeling steps of the classifier.

5.2 Results

The mean test accuracy versus the number of constructed variables per dataset is reported in Figure 3, with the standard deviation represented by error bars. The baseline (horizontal gray dashed line) is the accuracy of the majority classifier. The MODL performance is reported for each number of actually constructed variables. The RELAGGS performance is reported only once, with the number of constructed variables resulting from an exhaustive application of the construction rules. The 1BC and 1BC2 performance are reported for each bin number of the discretization preprocessing, with a number of constructed variables based on the total number of actually different values per variable.

Control of Variable Construction. The RELAGGS, 1BC and 1BC2 methods have little control on the size of the constructed representation space, which strongly varies with the complexity of the data structure and the number of values in the dataset. For example, the size of the representation space goes from a

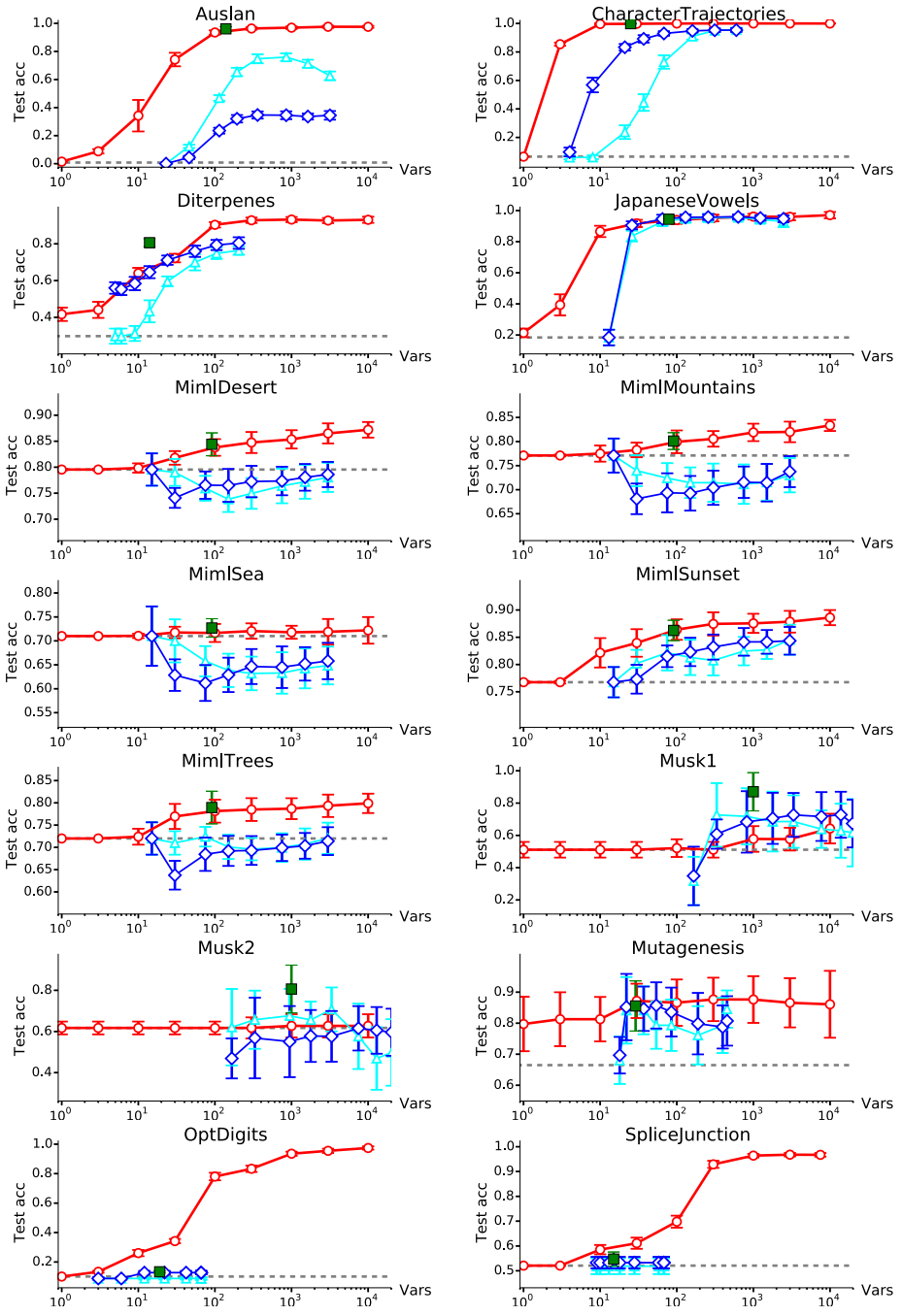

Fig. 3. Test accuracy versus number of constructed variables per dataset. MODL: ○ red RELAGGS: ■ green 1BC: △ cyan 1BC2: ◇ blue.

few tens of variables (SpliceJunction dataset) to around 20,000 variables (Musk2 dataset) for the 1BC method. The MODL method is much more expressive than the alternative methods, with the *Selection* rule which can build conjunctions of selection terms at any granularity. Still, Algorithm 2 allows to control the combinatorial exploration of this huge space and to construct the requested number of variables, as shown in Figure 3.

Examples of Constructed Variables. For the Mutagenesis dataset, in one train fold, the null cost ($\approx NEnt(Y)$) is 115.08 for the encoding of the classes of the 177 train instances. Among the 10,000 generated variables, only 618 are identified as informative. The original variable *lumo* in the root table has a low construction cost of 2.08. The simplest informative constructed variable involves one construction rule having one operand: *Count(Atoms)* with a construction cost of 5.08. The most complex informative constructed variable involves three construction rules, including the *Selection* rule with two selection terms. This rule *Sum(Selection(Atoms, (type* $\leq$ *23.5 and charge* $\leq$ *-0.0685)), CountDistinct(AdjacentBonds, bondtype))* has a construction cost of 17.66, which is not negligible compared to the null cost for this small dataset. This illustrates the ability of Algorithm 2 to build rather complex variables and of Formula (2) to filter the constructed variables.

Test Accuracy. The 1BC and 1BC2 often obtain similar performance, except for Auslan where 1BC is better and CharacterTrajectories and Diterpenes where 1BC2 dominates 1BC. Both methods get generally better performance as the number of bins increases, but they suffer from over-fitting, especially in the Miml image datasets where their performance is under the baseline. The RELAGGS propositionalisation method is used together with the SNB classifier (same as for MODL) and inherits from its accuracy and robustness. It always dominates the 1BC and 1BC2 methods. In the Musk1 and Musk2 datasets, the MODL method is not better than the baseline. Actually, these datasets are very small (around 100 instances), have a complex data structure (166 variables in the secondary table) and the classes are not well separable. Altogether, the construction penalization in Formula (2) results in rejecting almost all constructed variables. More instances would be necessary to learn the concept. In all other datasets, the MODL method benefits from its expressive construction language and obtains better test accuracy than the alternative methods. Remarkably, it often achieves good test accuracy with fewer variables than the alternative methods, and its performance never decreases with the number of constructed variables, which is the only user parameter in the approach.

Robustness. In order to evaluate the robustness of our approach, the classes have been randomly shuffled in each dataset before performing the experiment again. Two experiments are performed, one using criterion $cost_{CP}$ of Formula (2) (accounting for the construction cost of the variables), the other using $cost_P$ of Formula (1) (not accounting for the construction cost). The number of selected

variables is collected in both cases. The used preprocessing methods [3,4] are very robust. However, when 10,000 variables are constructed, on average 5 variables per dataset are wrongly selected, with more than 20 variables for the JapaneseVowels dataset. When the construction regularization is used (criterion $cost_{CP}$), the method is extremely robust: the overall 1.4 millions of constructed variables over all the datasets and folds of the cross-validation are all identified as information-less, without any exception.

6 Conclusion

In this paper, we have suggested a framework aiming at automating variable construction for supervised classification. On the basis of a description of a multi-table schema and a set of construction rules, we have suggested a prior distribution over the space of all variables that can be constructed. We have demonstrated that drawing variables according to this prior distribution raises critical problems of inefficiency and non-computability, then proposed an efficient algorithm that is able to perform many simultaneous draws in order to build a tabular representation with the required number of variables. The experiments indicate that the proposed method solves the problem of combinatorial explosion that usually limits the approaches which construct variables by a systematic application of construction rules, and the problem of over-fitting that occurs in case of representations with very large numbers of variables. The obtained classification performance are promising. In future work, we plan to extend the description of the variable construction domain by providing additional construction rules with potential specialization per application domain. Another research direction consists in drawing constructed variables according to their posterior distribution rather than their prior distribution. Finally, accounting for correlations between the constructed variables so as to avoid the risk of constructing many variants of the same variables raises another challenge.

References

1. Bache, K., Lichman, M.: UCI machine learning repository (2013), `http://archive.ics.uci.edu/ml`
2. Blockeel, H., De Raedt, L., Ramon, J.: Top-Down Induction of Clustering Trees. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 55–63 (1998)
3. Boullé, M.: A Bayes optimal approach for partitioning the values of categorical attributes. Journal of Machine Learning Research 6, 1431–1452 (2005)
4. Boullé, M.: MODL: a Bayes optimal discretization method for continuous attributes. Machine Learning 65(1), 131–165 (2006)
5. Boullé, M.: Compression-based averaging of selective naive Bayes classifiers. Journal of Machine Learning Research 8, 1659–1685 (2007)
6. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: CRISP-DM 1.0: step-by-step data mining guide. Tech. rep., The CRISP-DM consortium (2000)

7. Cover, T., Thomas, J.: Elements of information theory. Wiley-Interscience, New York (1991)
8. De Raedt, L.: Attribute-Value Learning Versus Inductive Logic Programming: The Missing Links (Extended Abstract). In: Page, D.L. (ed.) ILP 1998. LNCS, vol. 1446, pp. 1–8. Springer, Heidelberg (1998)
9. Džeroski, S., Lavrač, N.: Relational Data Mining. Springer-Verlag New York, Inc. (2001)
10. Džeroski, S., Schulze-Kremer, S., Heidtke, K.R., Siems, K., Wettschereck, D., Blockeel, H.: Diterpene Structure Elucidation From 13C NMR Spectra With Inductive Logic Programming. Applied Artificial Intelligence, Special Issue on First-Order Knowledge Discovery in Databases 12(5), 363–383 (1998)
11. Efron, B., Tibshirani, R.: An introduction to the bootstrap. Monographs on Statistics and Applied Probability, vol. 57. Chapman & Hall/CRC, New York (1993)
12. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. (eds.): Feature Extraction: Foundations And Applications. Springer (2006)
13. Knobbe, A.J., Blockeel, H., Siebes, A., Van Der Wallen, D.: Multi-Relational Data Mining. In: Proceedings of Benelearn 1999 (1999)
14. Kramer, S., Flach, P.A., Lavrač, N.: Propositionalization approaches to relational data mining. In: Džeroski, S., Lavrač, N. (eds.) Relational Data Mining, ch. 11, pp. 262–286. Springer (2001)
15. Krogel, M.-A., Wrobel, S.: Transformation-based learning using multirelational aggregation. In: Rouveirol, C., Sebag, M. (eds.) ILP 2001. LNCS (LNAI), vol. 2157, pp. 142–155. Springer, Heidelberg (2001)
16. Lachiche, N., Flach, P.: Ibc: A first-order bayesian classifier. In: Proceedings of the 9th International Workshop on Inductive Logic Programming, pp. 92–103. Springer (1999)
17. Lachiche, N., Flach, P.A.: 1bc2: A true first-order bayesian classifier. In: Matwin, S., Sammut, C. (eds.) ILP 2002. LNCS (LNAI), vol. 2583, pp. 133–148. Springer, Heidelberg (2003)
18. Liu, H., Motoda, H.: Feature Extraction, Construction and Selection: A Data Mining Perspective. Kluwer Academic Publishers (1998)
19. Pyle, D.: Data preparation for data mining. Morgan Kaufmann Publishers, Inc., San Francisco (1999)
20. Rissanen, J.: Modeling by shortest data description. Automatica 14, 465–471 (1978)
21. Rissanen, J.: A universal prior for integers and estimation by minimum description length. Annals of Statistics 11(2), 416–431 (1983)
22. Shannon, C.: A mathematical theory of communication. Tech. Rep. 27, Bell Systems Technical Journal (1948)
23. Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Mutagenesis: ILP experiments in a non-determinate biological domain. In: Wrobel, S. (ed.) Proceedings of the 4th International Workshop on Inductive Logic Programmin (ILP 1994). GMD-Studien, vol. 237, pp. 217–232 (1994)
24. Vens, C., Ramon, J., Blockeel, H.: Refining aggregate conditions in relational learning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 383–394. Springer, Heidelberg (2006)
25. Zhou, Z.H., Zhang, M.L.: Multi-instance multi-label learning with application to scene classification. In: Schölkopf, B., Platt, J., Hofmann, T. (eds.) Advances in Neural Information Processing Systems (NIPS 2006), vol. i, pp. 1609–1616. MIT Press, Cambridge (2007)