A Scalable Robust and Automatic Propositionalization Approach for Bayesian Classification of Large Mixed Numerical and Categorical Data

**Marc Boullé · Clément Charnay ·
Nicolas Lachiche**

Abstract Companies want to extract value from their relational databases. This is the aim of relational data mining. Propositionalization is one possible approach to relational data mining. Propositionalization adds new attributes, called features, to the main table, leading to an attribute-value representation, a single table, on which a propositional learner can be applied. However, current relational databases are large and composed of mixed, numerical and categorical, data. Moreover, the specificity of relational data is to involve one-to-many relationships. As an example of such data, consider customers purchasing products: each customer can purchase several products. Therefore, there is a need for techniques able to learn complex aggregates. Learning such features means to explore a combinatorial, possibly infinite, space and such an approach is prone to overfitting. We introduce a propositionalization approach dedicated to a robust Bayesian classifier. It efficiently samples a given number of features in the language bias, following a distribution over the complex aggregates. This distribution is also used to penalize complex aggregates in the regularization of the robust Bayesian classifier. Experiments show that it performs better than state-of-the-art methods on most investigated benchmarks and can deal with large datasets more easily. A new real, large, mixed relational dataset is introduced which confirms the ability of our approach to learn complex aggregates.

Marc Boullé
Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion, France
E-mail: marc.boulle@orange.com
`http://www.marc-boulle.fr`

Clément Charnay · Nicolas Lachiche
ICube, Université de Strasbourg, 300 Boulevard Brant, 67412 Illkirch, France
E-mail: {charnay,nicolas.lachiche}@unistra.fr
`http://icube-sdc.unistra.fr/en/index.php/{Cl%C3%A9ment_Charnay,Nicolas_`
`Lachiche}`

1 Introduction

Data mining encompasses a broad variety of practices. In a real industrial context, there are no domain experts, machine learners or data scientists to fine tune a new model every week. Additionally, there is no time to use wrappers or cross-validation to optimize the parameters. Therefore, a first requirement is that (1) the whole learning process, from data to predictions, should be automatic, parameter-free. Moreover, the data considered daily by companies are a mix of categorical and numerical columns in relational databases. A second requirement is (2) to use online (up-to-date) relational databases, hence to learn directly from different tables involving a mix of numerical and categorical data. In this context, discriminative features are complex aggregates, i.e. aggregation functions are applied to related objects selected according to conditions on possibly both their numerical and categorical columns. However, increasing the expressivity can increase overfitting. A third requirement is (3) to be able to learn complex aggregates without overfitting. Finally, more data are needed to learn more complex features. Fortunately, marketing databases are large. Therefore, a fourth requirement is that (4) the approach scales up to large databases.

Learning from data in a relational format is the area of relational data mining (Džeroski and Lavrač 2001). There are two approaches to relational data mining, either using a full-fledged relational data mining tool that works on the relational data directly, or transforming the relational data into an attribute-value dataset in order to use attribute-value learners the data scientists may be familiar with. The second approach is called propositionalization (Lachiche 2017). Few full-fledged relational data mining techniques scale up and are able to learn complex features. On the other hand, propositionalization enables one to run existing attribute-value software. In particular Boullé (2009) presents the Selective Naive Bayesian (SNB) classifier, that is parameter-free, scales up and does not overfit (Féraud et al 2010). However, it deals with attribute-value data only, so a scalable, robust and automatic propositionalization is needed. Few propositionalization approaches focused on using aggregation functions such as minimum, maximum or average in order to deal with both the numerical attributes and the one-to-many relationship. The first proposals were Knobbe et al (2001) and Krogel and Wrobel (2001). But these seminal works and more recent works could only generate simple aggregates with a single condition on the related objects: for instance, either the average duration of international calls, or the average duration of night calls, but not the average duration of international night calls which is a complex aggregate. The question summarizing the requirements listed above is whether existing relational data mining systems can be used on large mixed, numerical and categorical, data, and in particular whether increasing the expressivity by using complex features increases overfitting too and whether their runtimes and memory consumption are acceptable.

This paper presents a propositionalization approach combined with the SNB classifier. The key contributions are:

- a distribution over the complex aggregates,
- a sampling algorithm following this distribution to efficiently generate complex aggregates in spite of the combinatorial search space,
- an extensive comparison, with respect to the four requirements above, to the closest competitors: propositionalization approaches (Krogel and Wrobel 2001; Ahmed et al 2015) and full-fledged relational data mining systems (Van Assche et al 2006) able to generate aggregates, and first-order Bayesian classifiers (Flach and Lachiche 2004; Landwehr et al 2007), and the presentation of a new dataset, the Orange call detail records, that provides a real industrial challenge on a typical modern-industry scale (GB) to be considered by the academic community.

The approach was first sketched in (Boullé 2014). It is described further in this article with several examples illustrating the key points of the algorithms, and more extensive experiments, including the new dataset.

Section 2 presents an overview of the approach: using a regularization approach to deal with overfitting, a distribution over complex aggregates has to be defined. This distribution will be used both by the Bayesian classifier and the sampling algorithm generating the features. This distribution, detailed in Section 3, is based on the language bias only. Features are evaluated on the data when they are sampled, in particular they are kept only if they bring more information than the class only. Section 4 proposes an efficient sampling algorithm, generating a given number of features. Our approach is compared with respect to scalability and overfitting to the state-of-the-art methods detailed above on several benchmarks in Section 5. Section 6 extends these experiments to the new challenging relational datasets from Orange. This dataset illustrates the need for robust and scalable algorithms able to generate complex features on mixed, categorical and numerical, data. Section 7 describes related works. Section 8 gives a summary and discusses future work.

2 General settings

This section presents the general settings of our approach. We first define our learning task and the associated terminology. Then, we focus on the feature construction rules, explain why regularization is needed, and set the basis of the approach upon which the next sessions will be elaborated.

2.1 Learning task and terminology

We deal with classification: the objective consists in learning how to assign a class to any new individual. Here the notion of individual is obviously crucial since it is the object generalization is performed on. Hence a training set is a set of labeled individuals.

In relational data mining, individuals are described across several tables. One of the table enlists all the individuals: this table, referred to as the main
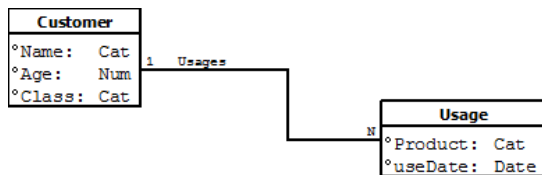
Fig. 1 Data structure for a problem of customer relationship management

table, associates one row to each individual, with intrinsic properties of the individual in its columns. Other tables, referred to as secondary tables, contain records related to the individual through foreign keys. For example, the main table may list customers whereas another table may present the products bought by the customers as illustrated in Figure 1. These secondary tables embed information that is relevant to the classification task. In Figure 2, molecules are represented as graphs with molecules (lumo, logp and the class feature) in a main table, atoms (element, type, charge) as vertices and bonds (bondtype) as edges in two secondary tables. From this relational schema, we extract and use the following relationships between the tables:

- every graph has a list of nodes (Atoms) and edges (Bonds)
- every edge (Bond) has two nodes (Atom1 and Atom2)
- every node (Atom) has a list of adjacent edges (Bonds it participates in). This relation AdjacentBonds is an additional language bias, built using the Atom1 and Atom2 relations between the Bond and Atom tables.

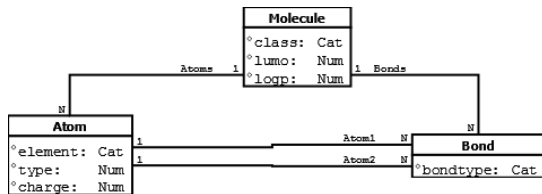This is actually the representation used for the Mutagenesis dataset in Section 5.

Fig. 2 Data structure for a molecule classification task

Propositionalization is the process of adding columns to the main table containing information extracted from the secondary tables (Lachiche 2017). Propositionalization can derive different properties of the individual such as the last date a product was used, the number of products used, etc. All these properties are called features of the individual. They are either original features which are intrinsic to the main table or constructed features which are derived from secondary tables. When there is a one-to-many relationship between the main table and the secondary table, e.g. a customer purchasing several products at different dates, many features can be derived, according

to the language bias. Indeed, many features can be imagined and no machine learning program generates all of them automatically. An example of such a feature would be "is the number of products used since 12 months greater than 10?". In relational data mining, the language bias is either hard-coded in the implementation, or defined by the user in the settings, for instance by feature construction rules.

2.2 Feature construction rules

In our framework a feature 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. The standard types, numerical or categorical, can be extended to other specialized types, such as date or time. For example, the *YearDay(Date)→Num* rule builds a numerical feature from a date feature. The operands can be a column of a table, the output of another rule, i.e. another feature, or a constant coming from the training set. In this paper, the construction rules used in the experiments of Section 5 are the following ones:

– *Selection(Table, selection criterion)→Table*: selection of records from the table according to a conjunction of selection terms (membership in a numerical interval or in a group of categorical values, on a column of the operand table or on a feature built from tables related to the operand table),
– *Count(Table)→Num*: count of records in a table,
– *Mode(Table, CatFeat)→Cat*: most frequent value of a categorical feature in a table,
– *CountDistinct(Table, CatFeat)→Num*: number of distinct values,
– *Mean(Table, NumFeat)→Num*: mean value of a numerical feature in a table,
– *Median(Table, NumFeat)→Num*: median value,
– *Min(Table, NumFeat)→Num*: min value,
– *Max(Table, NumFeat)→Num*: max value,
– *StdDev(Table, NumFeat)→Num*: standard deviation,
– *Sum(Table, NumFeat)→Num*: sum of values.

Using the data structure presented in Figure 1 and the previous construction rules (plus the *YearDay* rule for dates that denotes the day of the year, for example 32 for February 1st), one can construct the following features 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"))*.

Using the molecule data structure presented in Figure 2, one can construct more complex features exploiting the relationships between the atoms and the bonds at several depths of recursion, such as:

- *Mean(Atoms, Count(AdjacentBonds))*: mean over the atoms of the molecule of the number of bonds involving each atom,
- *Sum(Selection(Atoms, element = "c"), Min(AdjacentBonds, Atom1.type))*: sum, over the atoms of the molecule whose element is c, of the minimum of the types of the first atoms of the bonds involving each atom,
- *Sum(Bonds, Count(Selection(Atom2.AdjacentBonds where bondtype = 7)))*: sum, over the bonds of the molecule, of the number of bonds of type "7" involving their second atoms.

2.3 Regularization of feature construction

The issue is to exploit the language bias in order to efficiently drive the construction of features which are potentially informative for the prediction of the class attribute. In relational data mining, the data structure can have several levels of depth or even have a graph structure. Some of the constructed features can be used as operands of other rules, leading to the construction of features of any length. The space of constructed features is thus of potentially infinite size. This raises the two major following problems:

1. a combinatorial explosion for the exploration of this space of constructed features,
2. and a risk of overfitting.

Our proposition for solving these problems is the introduction of an evaluation criterion of the constructed features according to a Bayesian approach in order to penalize complex features. This is called regularization. In order to implement this evaluation criterion, we propose a prior distribution on the space of all features and an efficient sampling algorithm of the space of features according to their prior distribution.

Feature construction aims at enriching the main table with new features that will be taken as input of a classifier. Since usual classifiers take as input only numerical or categorical features, only these features will be considered.

2.4 Supervised preprocessing

Our approach builds upon the MODL supervised preprocessing methods (Boullé 2005, 2006). These methods consists in partitioning either a numerical features into intervals or a categorical feature into groups of values, through a piecewise constant class conditional density estimation. The parameters of a specific preprocessing model $M_P(X)$ of a feature X 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 preprocessing model parameters: the number of intervals, the partitions of the domain into intervals, and

the target distribution within each interval. The methods exploit a maximum a posteriori (MAP) technique to select the most probable preprocessing. In the Bayesian approach, the best model is found by maximizing the *posterior probability* $P(Model|Data)$ of the model given the data. Using the Bayes rule and since the probability $P(Data)$ is constant while varying the model, this is equivalent to maximizing $P(Model)P(Data|Model)$, that is the *prior probability* of the model times the *likelihood*. Taking the negative log of probabilities that are no other than coding lengths (Shannon 1948) in the minimum description length (MDL) approach (Rissanen 1978), this amounts to the description length L of a preprocessing model $M_P(X)$ (using a supervised partition) of a feature X plus the description length of the output data D_Y given the model and the input data D_X.

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

We asymptotically have $cost_P(X) \approx Nent(Y|X)$ where N is the number of training instances and $ent(Y|X)$ the conditional entropy (Cover and Thomas 1991) of the output Y given the input feature X. Formula (1) and the related optimisation algorithms are fully detailed in (Boullé 2006) for supervised discretization and (Boullé 2005) for supervised value grouping.

2.5 Null model and feature 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 modelling of the output values using a multinomial distribution, without using the input feature. Hence, the criterion $cost_P(\emptyset)$ can be estimated through the direct coding of the output values: the null cost is $cost_P(\emptyset) \approx Nent(Y)$, where $ent(Y)$ is the entropy of Y (Shannon 1948). The evaluation criterion of a feature is then exploited according to a filter approach (Guyon et al 2006): only those features whose evaluation is better than the null cost are considered informative and retained at the end of the data preparation phase.

2.6 Accounting for the feature construction process

As the number of original or constructed features increases, the chance for a feature to be wrongly considered informative becomes critical. In order to prevent this risk of overfitting, we suggest in this paper to exploit the space of constructed features described above by proposing a prior distribution over the set of all feature construction models $M_C(X)$. From this distribution, we derive a Bayesian regularization of the constructed features, which allows to penalize the most "complex" features. More formally, we consider a prior probability for $(M_C(X), M_P(X))$, a couple of feature construction and preprocessing model.

Assuming that the preprocessing model is independent of the construction model, we have

$$p(M_C(X), M_P(X)) = p(M_C(X))p(M_P(X)). \qquad (2)$$

Let $L(M_C(X)) = -\log p(M_C(X))$ be the negative log of the prior probability (coding length) of a construction model. This translates into an additional construction cost $L(M_C(X))$ in the evaluation criterion of the features, which becomes as in Formula (3).

$$cost_{CP}(X) = L(M_C(X)) + L(M_P(X)) + L(D_Y|M_P(X), D_X). \qquad (3)$$

2.7 Overall learning process

The overall learning process consists of the following steps:

1. create the formulas of features (e.g. the number of calls longer than the third quartile of phone call durations),
2. instantiate the variables (first scan of the data, e.g. find what the value of the third quartile of phone call durations),
3. evaluate the features (second scan of the data, e.g. find for each customer the actual number of calls longer than the third quartile of phone call durations),
4. preprocess the features values (discretize numeric values and cluster categorical values, e.g. identify several intervals of the number of such calls),
5. evaluate their cost with Formula (3),
6. filter out the features whose cost is worse than the null model,
7. train the SNB classifier using the remaining features.

3 Prior distribution of the original and constructed features

In this section, we describe how to evaluate the prior distribution of features. A feature is either a numerical or categorical feature. It appears in the main table and it is either original or built using construction rules recursively. Since the space of such features is of virtually infinite size, defining a prior probability on this space raises many problems. In order to solve these problems, our choices were guided by the following general principles:

1. *A priori* prefer the original features rather than constructed features.
2. Without additional information, the prior is uniform.
3. The prior exploits the feature construction domain.

The prior distributions over original and over constructed features are introduced in this section and the combinatorial step of the selection rule is detailed.

3.1 Case of original features

In the case where no feature can be constructed, the problem boils down to the choice of an original feature among the K numerical or categorical features of the main table. Using a uniform prior for this choice, we obtain $p(M_C(X)) = 1/K$, thus $L(M_C(X)) = \log K$.

3.2 Case of constructed features

In the case where features can be constructed, one must first choose whether to use an original feature or to construct a new feature. Using a uniform prior $(p = 1/2)$ for this choice implies an additional cost of $\log 2$ and violates our first general principle: *a priori* prefer the original features rather than constructed features. Then, we suggest considering the choice of constructing a new feature as an additional possibility beyond the K original features. The cost of an original feature becomes $L(M_C(X)) = \log(K + 1)$, with an additional cost of $\log(1 + 1/K) \approx 1/K$ with respect to the case of original features only.

Then, constructing a new feature relies on the following hierarchy of choices:

- choice of constructing a new feature,
- 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 feature or to construct a new feature 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 feature is decomposed on the operands of the used construction rule according to the recursive Formula (4), where the features X_{op} are the original or constructed features used as operands op of the rule $\mathcal{R}$.

$$L(M_C(X)) = \log(K + 1) + \log R + \sum_{op \in \mathcal{R}} L(M_C(X_{op})). \qquad (4)$$

3.3 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 features (original or constructed) and for each selection feature, choice of the selection part (numerical interval or group of categorical values). The selection part is itself chosen hierarchically, first with the choice of a granularity of the partitioned feature into a set of quantiles and second with the index of the quantile in the partition. In definitions 1 and 2, we provide a precise definition of the notion of quantile partitions both for numerical and categorical features.

Definition 1 (Numerical quantile partition) Let D be a dataset of N instances and X a numerical feature. 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 feature 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 (Rissanen 1983). This prior distribution is as flat as possible, with larger probabilities for small integer numbers. Each selection feature (original or constructed) is distributed using the prior defined previously in this section. As for the granularities, we consider only powers of two $2^1, 2^2, \ldots 2^p, \ldots$ 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 training 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. This reading step requires collecting the values of each secondary feature, which might raise some scalability problems in case of very large secondary tables. This can be mitigated by using the Reservoir Sampling algorithm (Vitter 1985), which collects a representative sample of limited size in one pass.

3.4 Synthesis

Figure 3 presents an example of such a prior distribution over the set of features 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 feature *Age* is $L(M_C(Age)) = \log 3$. That of constructing the feature with formula $Min(\ Usages, YearDay(Date))$ exploits 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 feature construction corresponds to a Hierarchy of Multinomial Distributions with potentially Infinite Depth (HMDID). The original features are obtained from the first level of hierarchy of the prior, whereas the constructed features get lower prior probabilities as they exploit deeper parts of the HMDID prior with complex formulas.
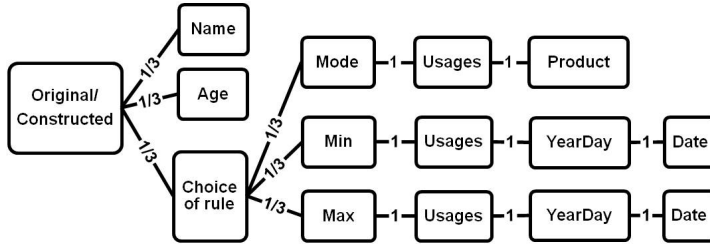
Fig. 3 Prior distribution of feature construction in the case of the customer relationship management dataset

4 Building a random sample of features

The objective is to build a given number of features, potentially informative for supervised classification, in order to create an input tabular representation. We suggest building a sample of features by drawing them according to their prior distribution. We present a first algorithm for building samples of features: while it may seem quite straightforward and "natural", we demonstrate that it is neither efficient nor even computable. Then, we 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 features}
 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 features 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 feature, 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.

4.1.1 Algorithm 1 is not efficient

Let us consider a construction domain with V original features that can be evaluated in the main 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 features is $V(1 - e^{-K/V})$ (Efron and Tibshirani 1993). In the case where $K = V$, this corresponds to the size of a bootstrap sample, that is $(1 - 1/e) \approx 63\%$ features obtained using V draws. To obtain 99% of the original features, one needs $K \approx 5V$ draws, which is not efficient. We informally define efficiency as the number of distinct features with respect to the number of draws. Furthermore, in the case of construction rules, the multinomial at the root of the HMDID now consists in $V + 1$ equidistributed values. The draws result in the construction of a new feature in only $\frac{1}{V+1}$ 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.

4.1.2 Algorithm 1 is not computable

Let us consider a construction domain with one single numerical feature x and one single construction rule $f(Num, Num) \rightarrow Num$. The features 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 feature 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 $s(F)$ of a constructed feature F (size defined by the number of factors in the constructed formula) can then be computed. Using the above approximation of the Catalan number, Formula (5) states that the expectation of the size of the feature is infinite.

$$E(s(F)) = \sum_{n=1}^{\infty} n 2^{-(2n-1)} C_{n-1} = \infty. \qquad (5)$$

This means that if one draws a random feature 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 features cannot be drawn individually as in Algorithm 1, we suggest drawing 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:

$$\frac{n!}{n_1! n_2! \ldots n_K!} p_1^{n_1} p_2^{n_2} \ldots p_K^{n_K}. \qquad (6)$$

Algorithm 2 Simultaneous random draws

Require: $K > 0$ {Number of draws}
Ensure: $\mathcal{V} = \{V\}, |\mathcal{V}| \leq K$ {Sample of features}
 1: $\mathcal{V} = \emptyset$
 2: Start from the root of the hierarchy of the HMDID prior
 3: Compute the number of draws $K_i = p_i K$ per branch of the prior tree (original feature, rule, operand...), where $(p_1, p_2, \ldots, p_i, \ldots)$ are the parameters of the multinomial distribution locally to the root node
 4: **for all** branches of the prior tree with $K_i > 0$ **do**
 5:     **if** terminal leaf of the prior tree (original feature or feature 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 this branch on the multinomial distribution at the sub-level of the prior tree
 9:     **end if**
10: **end for**

The most probable sample is obtained by maximizing Formula (6), which results into counts $n_k = p_k n$ per value according to maximum likelihood. For example, in the case of an equidistributed multinomial distribution with $p_k = 1/K$ and $n = K$ draws, Formula (6) 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. The draws are assigned to the original or constructed features 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 features), the draw is chosen randomly using a uniform distribution, priority being given to original features when both original and constructed features 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.

Algorithm 3 Construction of a sample of features

Require: N_f {Number of features to construct}
Ensure: $\mathcal{V} = \{V\}, |\mathcal{V}| \leq N_f$ {Sample of features}
 1: $\mathcal{V} = \emptyset$
 2: $K = N_f$ {Number of draws}
 3: **repeat**
 4:     $n_v = |\mathcal{V}|$ {Number of features in the current sample}
 5:     $\mathcal{V} = $ call Algorithm 2 with K draws
 6:     $K = K * 2$
 7:     {Stop if enough features or if no new features}
 8: **until** $|\mathcal{V}| \geq N_f$ or $|\mathcal{V}| = n_v$
 9: {Filter in case of too many constructed features}
10: **if** $|\mathcal{V}| > N_f$ **then**
11:     Sort $\mathcal{V}$ by decreasing probabilities
12:     Keep top N_f features
13: **end if**

In Algorithm 2, the number of draws may be greater than 1 in some leaves of the prior hierarchy. This implies that the final number of features can be inferior to the number of initial draws. Algorithm 3 is then used to perform iterative runs of Algorithm 2 with increasing numbers of draws. To obtain a given number of features N_f, Algorithm 2 is first called with N_f draws, then called again successively with twice the number of draws at each call, until the number of required features is reached or until no additional feature is built in the last call. Last, if too many features have been sampled, these are sorted by decreasing prior probability and only the top N_f features are kept.

This iterative doubling of the number of features to construct raises the issue of its efficiency. In theory, estimating the right number will take as long as actually iterating. But in practice this computation is negligible compared to the feature generation because it does not involve any scan of the data. Overall, Algorithm 2 is more efficient than Algorithm 1. It could still be improved but this is not a priority since it only represents a small fraction of the overall computing time.

More generally, considering the computational efficiency of the approach, the overhead of the construction algorithm is negligible with respect to the overall training time. Actually, Algorithm 3 consists in drawing a sample of constructed features 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 feature construction process, and is itself dominated by the data preparation and modelling steps of the classifier.

4.3 Illustration of the algorithm

We illustrate the Algorithm 3 on a toy example, where $N_c = 4$ features must be sampled from the prior tree displayed in Figure 3.
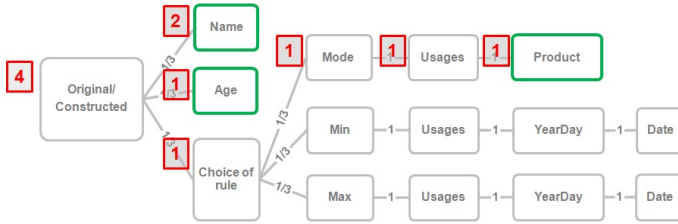

Fig. 4 First call to Algorithm 2 with 4 draws.

In a first step, Algorithm 3 calls Algorithm 2 with $K = 4$ simultaneous draws, as shown in Figure 4. At the root of the prior tree, there are three possibilities with equal probability:

- selecting original feature *Name*,
- selecting original feature *Age*,
- selecting a rule to construct a new feature.

The related multinomial probabilities are $(p_1, p_2, p_3) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The most probable multinomial sample has counts $K_i = p_i K$ per outcome. To get integer counts, we first take the integer part $K_i = \lfloor p_i K \rfloor$, and the remaining counts are chosen randomly, with a priority for original features. In our example, we get $(K_1, K_2, K_3) = (2, 1, 1)$ (assuming that the remaining count is assigned to *Name*).

The construction process is propagated in the three branches of the prior tree, with the numbers of draws $(2, 1, 1)$. The first branch receives two draws, but as it is a terminal leaf, one single feature (*Name*) is added to the sample of features to construct or select. Similarly, the second branch ends with the additional feature *age*. One draw ($K_3 = 1$) is propagated in the last branch, which relates to the choice of a rule. In this node of the prior tree, three construction rules can be applied according to the types of their operands, with equal probability:

- *Mode*,
- *Min*,
- *Max*.

The last remaining draw ($K_3 = 1$) in this branch is chosen randomly among the three rules, and we assume that *Mode* is chosen. The *Mode* rule needs a Table as a first operand. As the only available Table feature in the data structure (see Figure 1) is *Usages*, it gets a probability 1 in the prior tree and is chosen. The second operand of *Mode* is a Categorical feature, and the only available one in *Usages* is *Product*, that is thus chosen. At this point, we are in a leaf of the prior tree, and the constructed feature *Mode(Usages, Product)* is added to the set of features. Overall, the $K = 4$ initial draws have been exploited to obtain three features:
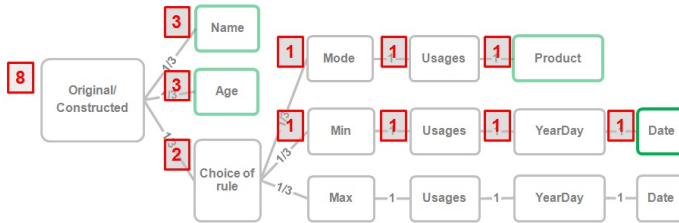
- *Name*,
- *Age*,
- *Mode(Usages, Product)*.

Fig. 5 Second call to Algorithm 2 with 8 draws.

Since the first sampling step did not yield enough features, the number of draws is doubled and we iterate. In a second step, Algorithm 3 then calls Algorithm 2 with $K = 8$ simultaneous draws, as illustrated in Figure 5. At the root of the prior tree, the most probable sample for the multinomial distribution with three equiprobable choices receives counts $(K_1, K_2, K_3) = (3, 3, 2)$.

Again, the construction process is propagated in the three branches of the prior tree. The two first branches are terminal leaves that each receive 3 draws and each branch returns one single feature (*Name* and *Age*). Two draws ($K_3 = 2$) are propagated in the last branch, which relates to the choice of a rule among the three following rules: *Mode*, *Min*, *Max*. The best multinomial sample of size $K_3 = 2$ is $(K_{3.1}, K_{3.2}, K_{3.3}) = (1, 1, 0)$ The first branch is related to the rule *Mode*, and as previously, it ends with the constructed feature *Mode(Usages, Product)*. The last remaining draw ($K_{3.2} = 1$) is exploited in the *Min* sub-branch. Like *Mode*, the *Min* rule takes the *Usages* for its first operand. The second operand of *Min* is a Numerical feature, and the only available rule that could produce one is the *YearDay*. This rule, the only one available, gets a prior probability 1 in the prior tree and is thus chosen. In this node of the prior tree, the *YearDay* rule needs a Date operand, and it uses the *Date* feature of *Usages*, which is the only one available. At this point, we are in a leaf of the prior tree, and the constructed feature *Mean(Usages, YearDay(Date))* is added to the set of sampled features. Overall, the $K = 8$ draws in the second call to Algorithm 2 have been exploited to build four features:

– *Name*,
– *Age*,
– *Mode(Usages, Product)*,
– *Mean(Usages, YearDay(Date))*.

Algorithm 3 then stops, since the required number of features has been constructed. Being a propositionalization method, it has built a flat representation, that could be used as input by any classification method. Still, in the suggested approach (cf. Section 2.3), the construction prior is exploited again for regularization purposes so as to prevent overfitting.

4.4 Illustration with the *Selection* rule

We now illustrate the case of the *Selection* rule described in Section 3.3. We reuse the same data structure as in the previous section, but assume that the set of available construction rules is extended with the *Selection* rule, and that $N_c = 100$ features must be sampled using Algorithm 3.

In a first step, Algorithm 3 calls Algorithm 2 with $K = 100$ simultaneous draws, as shown in Figure 6. Like in Figure 4, at the root of the prior tree, there are three possibilities with equal probability $(p_1, p_2, p_3) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The most probable multinomial sample has counts $K_i = p_i K$ per outcome, and we get $(K_1, K_2, K_3) = (34, 34, 33)$, resulting in selecting the two original features
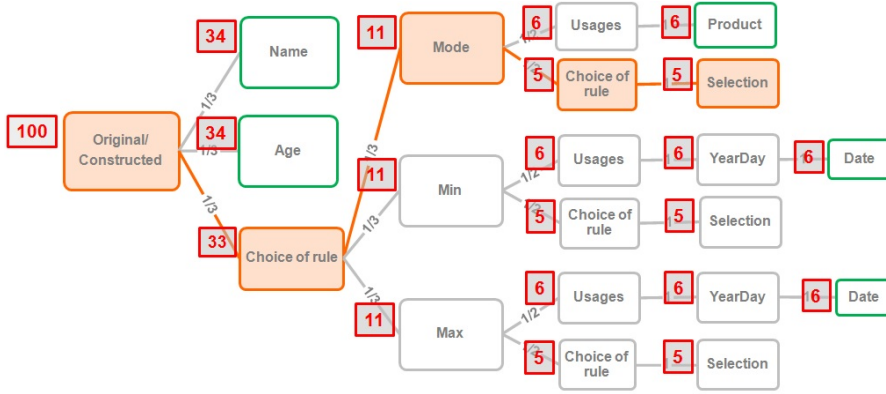
Fig. 6 Call to Algorithm 2 with 100 draws, using the *Selection* rule.

Name and *Age*, and propagating 33 draws to construct a new feature. In this last branch, three construction rules (*Mode*, *Min*, *Max*) are available with equal probability, and each construction rule receives 11 draws.

4.4.1 Choosing between a Table original feature and the Selection rule

We now focus on the *Mode* rule, which has two operands: a *Table* feature and a *Categorical* feature within the table operand. The first operand can be either an original feature or a constructed feature. As the *Customer* data structure contains a single *Table* feature (*Usages*: see Figure 1), there are two possibilities with equal probabilities for the first operand of *Mode*:

– selecting original feature *Usages*,
– selecting a rule to construct a new *Table* feature.

In the *Mode* node of the prior tree pictured in Figure 6, the best multi-nomial sample of size $K_{3.1} = 11$ is $(K_{3.1.1}, K_{3.1.2}) = (6, 5)$. In the first sub-branch, the *Usages* is selected, and the 6 draws end in constructing the feature *Mode(Usages, Product)*, like in Section 4.3. In the second sub-branch, 5 draws are propagated to build the first operand of *Mode* using the *Selection* rule.
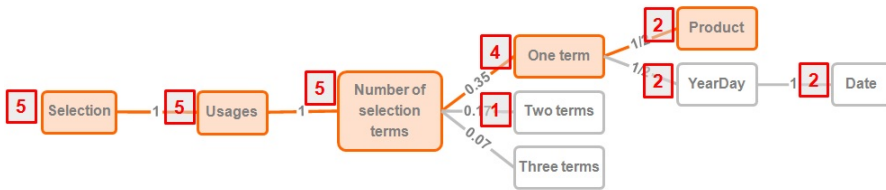

Fig. 7 Choice of selection terms in the *Selection* rule.

4.4.2 Choosing the selection terms

We continue focusing on this branch of the prior tree in Figure 7, that illustrates the next step in constructing a feature with the *Selection* rule. The *Selection* rule has two operands: a *Table* and a selection operand. The only available *Table* feature is *Usages*, which is chosen as first operand and receives 5 draws. The second operand is a conjunction of selection terms. According to Section 3.3, the number of selection terms is distributed according to the universal prior for integer numbers of Rissanen (Rissanen 1983).
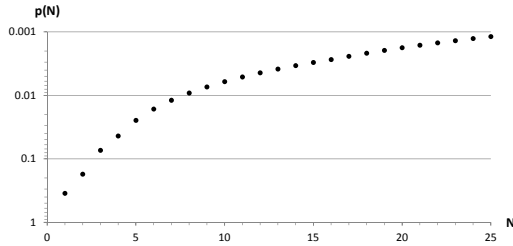

Fig. 8 Universal prior for integer numbers.

We have $p_1 \approx 0.349027, p_2 \approx 0.174513, p_3 \approx 0.073404, p_4 \approx 0.043628, \ldots$, with $\sum_{i=1}^{\infty} p_i = 1$. This prior is very flat, and for example, we have $p_{25} \approx 0.0012$, $p_{100} \approx 0.00013$. Back to our feature construction algorithm illustrated in Figure 7, we have 5 draws to choose the number of selection terms, distributed according to the universal prior for integer numbers of Rissanen. The maximum likelihood estimate ($n_k = n p_k$) is valid only asymptotically, and for very small n, we use a greedy optimization heuristic to maximize the likelihood using the exact Formula (6). This heuristic starts from the asymptotic solution and iteratively adds, removes or moves draws across the multinomial values while the criterion improves. In our case with 5 draws for the number of selection terms, we get 4 draws for selection operands having one selection term and one draw for selection operands with two selection terms.

Each selection term is either of numerical or categorical type and is chosen using the construction prior as before. In the case illustrated in Figure 7 for one selection term, there is one single original feature with one of the required types (*Product*). Otherwise, a rule must be chosen to construct a feature, and here, the *YearDay* rule is the only available one to build an additional feature, of numerical type. In the end, the constructed selection operand can be *Product* or *YearDay(Date)* with equal probability, and each possibility receives two draws. The case of multiple selection terms is treated in the same way by constructing the feature involved in each selection term.
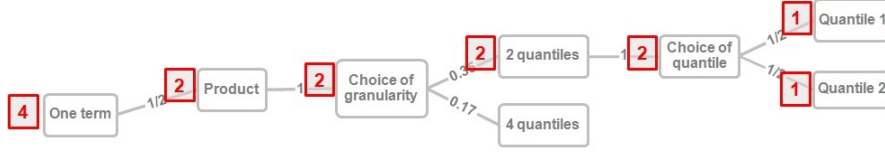
Fig. 9 Choice of granularity and quantile in a selection term.

4.4.3 Choosing the selection quantiles

We remind that every selection term is of the form $(feature \in quantile)$ (see Section 3.3). Once the features in the selection terms have been chosen, the selection terms must be completed by choosing the quantile. This is illustrated in Figure 9 by focusing on the branch $Mode(Selection(Usage, Product \in \ldots),$ $\ldots)$. The size $2^1, 2^2, \ldots 2^p, \ldots$ of the partition of the $Product$ feature is chosen with the exponent p distributed according to the universal prior for integer numbers. With only two draws at this stage of the prior tree, we get two draws for partitions with 2 quantiles, and zero draw for finer partitions. The last choice is the quantile, given the partition size. Given a partition of size 2, two quantiles are available with equal probabilities $(\frac{1}{2}, \frac{1}{2})$, and here, each of them gets one draw. Altogether, in this branch of the prior tree, the $Selection$ rule is completed in two ways:

– $Mode(Selection(Usage, Product \in quantile_{1/2}), \ldots),$
– $Mode(Selection(Usage, Product \in quantile_{2/2}), \ldots).$

Once the first operand of the $Mode$ rule is completed with a $Selection$, the second operand can be chosen using again the prior tree, as in Figure 6.

4.4.4 Getting the actual definition of the selection quantiles

Let us notice that the Algorithm 2 exploits only the data structure and the set of available construction rules to build new features, without reading the database. As explained in Section 3.3, the $Selection$ rules are constructed using formal definitions of quantiles (such as $quantile_{i/s}$: quantile of index i in a partition of size s). Once the required number of features has been sampled by Algorithm 3, there is one reading step on the whole database to instantiate the formal definition of each selection quantile into an actual definition, with numerical boundaries for intervals and categorical values for categorical quantiles.

5 Experimental evaluation

This section details the experimental setup, analyses the results of 9 algorithms on 14 benchmarks, and discusses further aspects.

5.1 Experimental setup

We present the algorithms, the datasets and the experiments.

5.1.1 Competing algorithms

The experiments perform comparisons with alternative relational data mining methods based on propositionalization and with Inductive Logic Programming (ILP) Bayesian classifiers. We compare the following methods:

– 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*. Its single parameter is the number of features to construct (see Section 4.2). It was set to 1, 3, 10, 30, 100, 300, 1,000, 3,000 and 10,000 in the experiments. The features are generated using Algorithm 3 then filtered using Formula (3), which accounts for both construction and preprocessing cost. The filtered features are used as input of a selective naive Bayesian classifier with feature selection and model averaging (SNB[1]) (Boullé 2007), which is both robust and accurate in the case of very large numbers of features.
– RELAGGS is our implementation of the Relaggs propositionalization method (Krogel and Wrobel 2001). It exploits the same construction rules as MODL and exhaustively constructs all the possible features, except for the *Selection* rule that raises combinatorial problems. Instead RELAGGS adopts a systematic approach for retrieving information from secondary tables: it constructs all the rules based on counts for categorical values and it derives rules from all other numerical aggregation functions for numerical attributes. The constructed features are then fed into the SNB classifier.
– Cardinalisation and Quantilisation (Ahmed et al 2015) are propositionalization techniques to deal with numerical features in a secondary table. Let us denote i an instance of the primary table, associated to n_i records of the secondary table, A a numerical feature of the secondary table, and $Q - 1$ the number of features we want to build. Let us assume that the values $v_{i,j}$ of A for the n_i associated records are ordered. For $1 \leq q \leq Q - 1$, quantilisation outputs $v_{i,j}$ such that $j = \frac{q}{Q} * n_i$ and cardinalisation outputs $v_{i,j}$ such that $j = min(n_i, \frac{q}{Q} * N)$ where $N = max_i(n_i)$. Again these propositionalization techniques generate features that are used with the SNB classifier. No filtering is used.
– 1BC is the first-order Bayesian classifier described in (Lachiche and Flach 1999). It can be considered as a propositionalization method, with one feature per value in a secondary table. These features are counted as words in a document in the wordification approach (Perovsek et al 2015). In 1BC they are used as features in a propositional naive Bayesian classifier. To preprocess the numerical values of each table, all numerical features

---

[1] The SNB classifier, with the MODL feature construction method, is available as a shareware at `http://www.khiops.com`.

are discretized into equal frequency intervals. In the experiments, we use discretization into 1, 2, 5, 10, 20, 50, 100 and 200 bins.

- 1BC2 is the successor of 1BC described in (Lachiche and Flach 2002). While 1BC applies propositionalization, 1BC2 is a true first-order classifier. It estimates the probabilities of sets given the probabilities of their elements. Numerical features were dealt with using the same discretizations as $1BC^2$.

- NFOIL (Landwehr et al 2007) is the most recent approach combining naive Bayes and propositionalization. In NFOIL, feature construction is more tightly integrated with the Bayesian classifier than in our propositionalization approach. However, it does not use aggregation functions other than the existential quantifier and does not deal explicitly with numerical attributes. We use discretization into 1, 2, 5, 10, 20, 50, 100 and 200 bins as for the other classifiers. We use the same parameters as in the experiments reported in (Landwehr et al 2007): a beam search of size 5, a sensitivity of 0.1%, but we set the maximum number of features to 100 because the actual number of features exceeded the default 25. Indeed, the number of categorical values increases with the number of discretization bins. Increasing the maximum number of features increases the search space hence the runtime. NFOIL is able to select a much lower number of features than the maximum number of features allowed. The few times the number of features reached or exceeded 100 (namely on the MIML datasets), we ran NFOIL again raising the maximum number of features to 500.

- Tilde is a logical decision tree learner described in (Blockeel and De Raedt 1998). Based on Inductive Logic Programming, it constructs and refines first-order rules as decision tree splits. To do so, Tilde introduces features using existential quantification, and evaluates the quality of the rules using the same metrics as Quinlan's C4.5, described in (Quinlan 1993). In the same fashion as for 1BC, we use discretization into 1, 2, 5, 10, 20, 50, 100 and 200 bins.

- FORF is the extension of Tilde to a Random Forest model, as described in (Van Assche et al 2006). Moreover, it uses a subset of the construction rules of MODL, namely *Count*, *Mean*, *Min* and *Max* construction rules, and the *Selection* rule limited to a maximum of one term in the conjunction. We used the same discretization levels as for Tilde, and Random Forests were constructed with 33 trees.

Those techniques represent the variety of ILP techniques, from propositionalization to full-fledged first-order learners. They include the closest competitors, i.e. first-order Bayesian classifiers or techniques that can learn complex features for real datasets involving a mix of numerical and categorical feature. Other relevant approaches are discussed in Section 7.

The Relaggs, Cardinalisation and Quantilisation are propositionalization approaches. The features they generate are compared to those introduced in this article following the MODL approach using exactly the same robust naive Bayesian classifier (SNB).

---

[2] 1BC and 1BC2 are available on the online data mining service `http://clowdflows.org/`.

1BC and 1BC2 are considered as simple baselines. They are quick but less expressive naive relational Bayesian classifiers. They represent similar approaches, Ceci et al (2003); Liu et al (2005) that are discussed in Section 7. The aim of our approach is to be able to generate more complex features, still efficiently, to get a better accuracy without overfitting. As a consequence, our approach has to beat such simple baselines. NFOIL is the state-of-the-art in first-order Bayesian classifiers. It is able to use more complex features than 1BC, 1BC2 and similar approaches.

The data we aim to deal with (cf. Section 6) involve a mix of numerical and categorical data and require to be able to generate hypotheses containing complex aggregates, for example the number of calls of a given type and duration. For this reason, we consider full-fledged relational data mining techniques that are able to deal as naturally as possible with numerical attributes and to generate such complex aggregates. To the best of our knowledge, the only available techniques meeting these criteria are the relational decision tree learner Tilde and its random forest variant FORF.

5.1.2 Benchmarks

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

Dataset	Instances	Records	Cat.cols	Num.cols	Classes	Maj.
Auslan	2,565	146,949	1	23	96	0.011
CharacterTraject.	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

Fourteen relational datasets are considered in these experiments. They were chosen because they contain a mix of numerical and categorical columns, are likely to need complex aggregates and are among the largest ILP benchmarks. The Auslan, CharacterTrajectories, JapaneseVowels, OptDigits and SpliceJunction datasets come from the UCI repository (Bache and Lichman 2013) and are respectively 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). Each of these sequential or time series datasets is represented with one main table and a secondary table in zero-to-many relationship. The Diterpenes (Džeroski et al 1998), Musk1, Musk2 (De Raedt 1998) and Mutagenesis (Srinivasan et al 1994) datasets are related to molecular chemistry. The Mutagenesis dataset is a graph with molecules (*lumo, logp* plus the class feature) in a main table, atoms (*element, type, charge*) as vertices and bonds (*bondtype*) as edges, cf. Figure 2. It is a dataset involving an arbitrary depth, i.e. an arbitrary number of imbrications of secondary tables. The Miml dataset (Zhou and Zhang 2007) is related to image recognition, with five different target features. Table 1 gives main characteristics of these datasets: the number of instances in the main table, the number of records in the secondary table and its numbers of categorical and numerical columns, the number of classes and the proportion of the majority class.

Some datasets, in particular images, may be represented by an exact propositional version, whereas truly relational datasets do not have such an attribute-value representation. However, a relational representation of these non truly relational datasets allows learning relational features, such as the number of black pixels on the horizontal middle line, that cannot be learned by an attribute-value learner. This is investigated in 5.2.3.

5.1.3 Experiments

All the experiments are performed using a stratified 10-fold cross validation. In each training fold, the features are generated and selected and the classifier is trained, while the test accuracy is evaluated on the test fold.

For each dataset, we are interested in the accuracy with respect to the expressivity. In our experiments, the expressivity of each algorithm depends on the number of features it can consider. RELAGGS has no control over the number of generated features which depends only on the number of values of categorical features and on the number of aggregation functions. The number of features considered by cardinalisation, quantilisation, 1BC, 1BC2, NFOIL, Tilde and FORF can be tuned by varying the number of discretization bins in a wrapping approach, e.g. (Ahmed et al 2015). NFOIL, Tilde and FORF only output features that are used in the model. They do not output a set of features as a propositionalization approach. But they test numerous features when building the model. We use this number of queries as the number of features. The MODL method explicitly controls the number of generated features.

Figure 10 shows the average accuracies of each technique with respect to the original number of features generated (i.e. before filtering). The mean test accuracy with respect to the number of generated features per dataset is reported in Figure 10, with the standard deviation represented by error bars. The baseline (horizontal gray dashed line) is the accuracy of the majority classifier. The performance of MODL is reported for each number of actually generated features. The performance of RELAGGS is reported only once, with
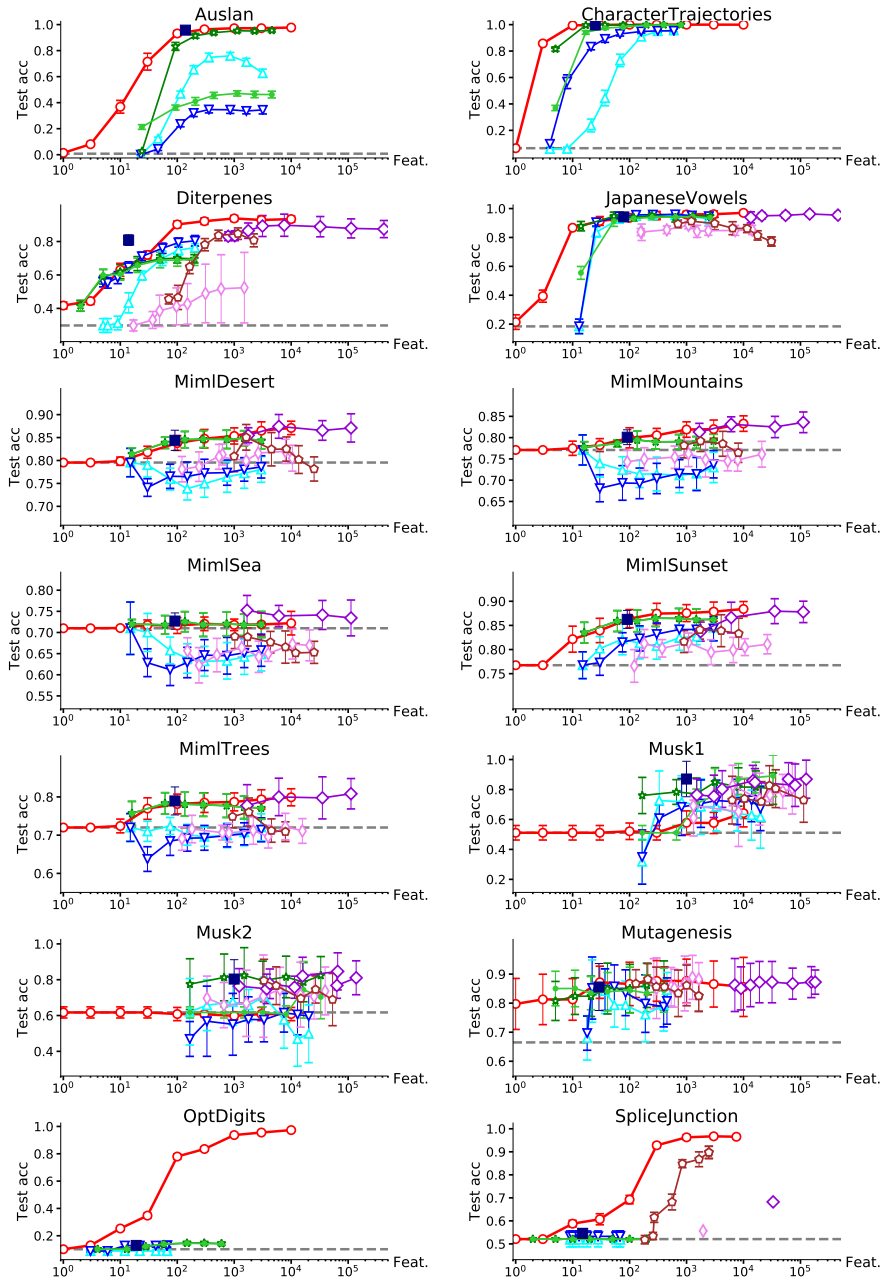
Fig. 10 Test accuracy versus original number of generated features per dataset.

MODL: ○ red   RELAGGS: ■ navy   Cardinal.: ● lime   Quantil.: ★ green
1BC: △ cyan   1BC2: ▽ blue   NFOIL: ⬠ brown   Tilde: ◇ pink   FORF: ◇ violet

the number of generated features resulting from an exhaustive application of the construction rules. The performances of 1BC, 1BC2, NFOIL, Tilde, FORF, cardinalisation and quantilisation are reported for each bin number of the discretization preprocessing. All numerical features were discretized into 1, 2, 5, 10, 20, 50, 100 and 200 bins to generate an increasing number of generated features. Only 1 to 10 bins were evaluated for NFOIL and FORF on the MIML datasets because the run time of a cross-validation is longer than 4 days, and did not show an increase of accuracy. NFOIL, Tilde and FORF could not be run on the largest datasets, Auslan, CharacterTrajectories and SpliceJunction, due to memory limitations.

Figure 11 shows the average accuracies of each technique with respect to the training time while varying the number of features. Training durations lower than 1 second are not shown. For this reason only MODL, NFOIL, Tilde and FORF can be seen on the mutagenesis dataset. Let us notice that the algorithms are implemented in different programming languages and the training time can vary according to the load and power of the computers where the experiments were run. Therefore, we focus on the order of magnitude, using a logarithmic scale on the x axis, on the one hand, and on the shape of the learning curve, i.e. how quickly the accuracy improves with more training time in order to build more complex features.

5.2 Analysis

Machine learning aims at maximizing the accuracy or other similar performance measure. One can check on Figures 10 and 11 that MODL produces models ranking among the best in term of accuracy on most of the datasets, at least when there are enough data to learn complex hypotheses, cf. 5.3.2. Conversely, learning complex hypotheses makes sense if the learner avoids overfitting, is tractable and finally reaches its best accuracy automatically.

5.2.1 Robustness

As can be seen on Figures 10 and 11, most methods get better performance as the number of features increases. But they suffer from overfitting: their accuracies decrease, after having reached a maximum, when the number of features increases. All the propositionalization methods, RELAGGS, cardinalisation and quantilisation, are used with the SNB classifier (the same as for MODL) and inherit from its accuracy and robustness. With its simple expressivity, RELAGGS provides a strong baseline that gets a very good accuracy with few features. More expressive techniques can hardly outperform this baseline. Only FORF and MODL succeed. MODL makes an impressive difference on OptDigits and SpliceJunction, actually on datasets with numerous instances. On the contrary, in the Musk1 and Musk2 datasets, the MODL method does not improve over the baseline.
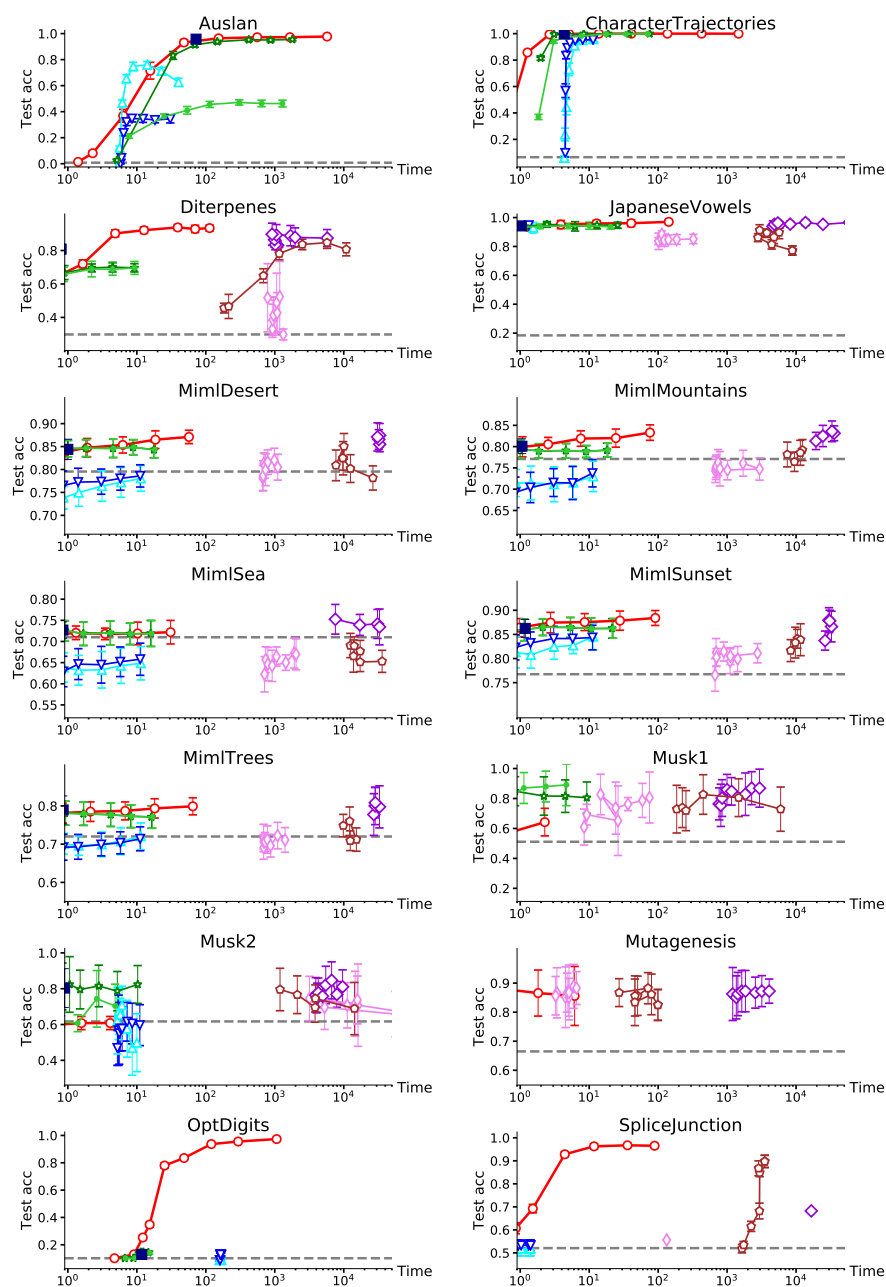
Fig. 11 Test accuracy versus training time per dataset.

MODL: ○ red   RELAGGS: ■ navy   Cardinal.: ● lime   Quantil.: ★ green
1BC: △ cyan   1BC2: ▽ blue   NFOIL: ⬠ brown   Tilde: ◇ pink   FORF: ◇ violet

We pointed out that all approaches but MODL are prone to overfitting. Indeed, sometimes their accuracies happen to decrease as the number of features decreases. In order to further evaluate the robustness of the MODL approach, the class labels have been randomly shuffled in each dataset before performing the experiment again. Two experiments are performed, one using criterion $cost_{CP}$ of Formula (3) (accounting for the construction cost of the features), the other using $cost_P$ of Formula (1) (not accounting for the construction cost). The number of selected features is collected in both cases. The used preprocessing methods (Boullé 2005, 2006) are very robust. However, when 10,000 features are generated, on average 5 features per dataset are wrongly selected, with more than 20 features for the JapaneseVowels dataset. When the construction regularization is used (criterion $cost_{CP}$), the method is extremely robust: the overall 1.4 million generated features, over all the datasets and folds of the cross-validation, are all identified as information-less, without any exception.

5.2.2 Scalability

Figures 10 and 11 illustrate that MODL provides a better accuracy with a smaller number of features and with a smaller run time (often by several orders of magnitude: the x-axis scales are logarithmic). This frugality allows MODL to tackle large datasets such as auslan, character trajectories and optical digits while its competitors with high expressivity, in particular NFOIL, Tilde and FORF, cannot be run.

Table 2 Accuracies (%) at the maximal expressivity

Dataset	MODL	REL.	Quant.	Card.	1BC	1BC2	NFOIL	Tilde	FORF	Published
Auslan	*97.7±0.9*	95.8±1.3	95.5±1.2	46.2±2.6	62.7±3.1	34.4±3.1	NA	NA	NA	**97.9±0.2** (Burke 2013)
Character Trajectories	**100±0.1**	99.7±0.3	**100±0.1**	*99.8±0.2*	95.7±1.3	95.3±1.3	NA	NA	NA	98.7±0.4 (Burke 2013)
Diterpenes	93.4±1.8	80.8±3.2	69.7±2.5	69.6±3.9	76.4±2.3	80.4±3.2	80.7±3.9	52.4±21.1	87.4±5.1	**97.2** (Gärtner et al 2004)
Japanese Vowels	97.0±1.5	94.2±2.1	94.7±2.1	93.6±1.6	93.0±3.6	94.7±2.2	77.2±3.1	85.2±3.2	96.6±2.2	**99.0** (Burke 2013)
Miml Desert	**87.1±1.5**	84.4±2.2	84.3±1.7	84.3±1.7	78.1±2.8	78.6±2.4	78.2±2.6	80.5±2.8	**87.1±3.1**	*86.9±1.4* (Zhou and Zhang 2007)
Miml Mountains	*83.3±1.8*	80.1±1.7	79.2±1.6	79.2±1.6	73.0±3.6	73.8±3.2	76.4±2.3	76.1±3.0	**83.6±2.4**	*82.0±2.2* (Zhou and Zhang 2007)
MimlSea	*72.2±2.8*	*72.7±1.9*	*71.9±3.1*	*71.9±3.1*	64.9±3.9	65.8±3.8	65.3±2.6	67.0±3.7	**73.4±4.2**	*73.0±3.0* (Zhou and Zhang 2007)
Miml Sunset	**88.4±1.5**	86.3±1.9	86.3±2.0	86.3±2.0	84.6±2.8	84.4±2.5	83.3±3.2	81.1±2.0	87.8±2.2	*88.3±2.3* (Zhou and Zhang 2007)
Miml Trees	*79.9±2.2*	*79.0±3.7*	*77.1±2.9*	*77.1±2.9*	72.1±3.5	71.4±3.1	70.9±2.6	71.1±3.2	**80.8±4.0**	*80.1±1.5* (Zhou and Zhang 2007)
Musk1	64.2±9.2	*87±11.8*	80.6±10.5	*89.1±13.9*	61.7±20.9	67.3±14.8	72.9±14.8	*80.6±17.0*	*86.9±12.7*	**92.4** (Gärtner et al 2004)
Musk2	60.8±3.7	*80.4±10.8*	*82.4±10.6*	70.4±12.2	49.8±16.1	59.6±11.4	68.9±14.6	73.7±13.4	*81.1±9.5*	**89.2** (Gärtner et al 2004)
Mutagenesis	85.6±10.2	85.6±8.1	86.1±8.4	84.5±7.6	84.6±6.0	80.7±8.0	82.5±5.3	88.3±5.7	87.2±4.2	**95.8±3.3** (Lodhi and Muggleton 2005)
OptDigits	97.3±0.5	12.9±1.1	14.3±1.2	14.1±1.0	8.8±0.8	12.8±1.2	NA	NA	NA	**98.0** (Meena and Devi 2015)
Splice Junction	**96.5±0.8**	54.6±2.1	52.1±0.1	52.1±0.1	51.9±3.2	53.3±2.3	89.7±2.7 c	55.6±0.1	68.2±0.1	91.7 (Zhou 2015)

5.2.3 Automaticity

Ideally, the user of a learning system should not be required to tune any parameter. On the contrary, she/he might expect the learning system to provide the best model automatically. Whereas RELAGGS has no parameters, the other algorithms are not automatic. In particular, all algorithms that cannot deal with numeric attributes require to choose a discretization. MODL has a single parameter: the number of features. So our approach could be used in an anytime setting where the program is run iteratively, doubling the number of features at each step, until the user stops it. The relevance of such an anytime setting is backed by Figures 10 and 11: they illustrate that MODL selects the most informative features first, and that it is monotonic: its accuracy increases when the number of features, and runtime, increase.

An additional issue is whether such automatically built features can compete with the state-of-the art methods. Table 2 shows the accuracies and standard deviations on every dataset of each approach at the maximal number of features considered in our experiments, cf. Figure 10. We assume that the maximal number of features corresponds to the best expressivity of each approach, even though we already pointed out before that some approaches, e.g. 1BC on Auslan, overfit, therefore we call it maximal expressivity. We saw previously that MODL is monotonic and outperforms other relational data mining techniques on most datasets. We focus now on the best published results on each dataset, to the best of our knowledge. Table 2 refers to the publications where each best accuracy was published and reports the standard deviation when it is available. Even though the exact experimental setup might be different from ours, most published accuracies were estimated averaging several runs of cross-validation, so we consider they provide a sound basis for comparison. Table 2 highlights in bold face the best accuracy for each dataset and in italic accuracies that are not significantly lower, less than one standard deviation lower than the best accuracy.

We observe that MODL and FORF, when it is tractable, get close and sometimes better than the best published accuracies so far, whereas those results often correspond to dedicated approaches, in particular image or signal processing for Auslan, CharacterTrajectories, JapaneseVowels and OptDigits. Thus, generic automatic feature construction techniques can compete with expert approaches.

5.2.4 Comparative analysis of accuracy

We apply the Friedman test coupled with a post-hoc Nemenyi test as suggested by Demšar (2006) for comparisons of classifiers over multiple datasets (at significance level $\alpha = 0.05$ for both tests). The null-hypothesis is rejected, which indicates that the compared methods are not equivalent in terms of accuracy. The result of the Nemenyi test is represented by the critical difference chart shown in Figure 12 with $CD \approx 3.21$ and where the mean rank of each method is plotted. First of all, we observe that there is no critical difference
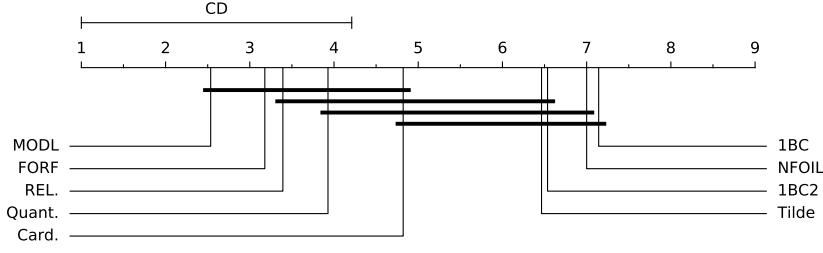
Fig. 12 Critical difference chart

of performance between the top five competitors MODL, FORF, RELAGGS, Cardinalisation and Quantilization. Secondly, even though MODL is not statistically singled out, only MODL and FORF get a significant advantage on Tilde, NFOIL, 1BC and 1BC2.

5.3 Further investigations

In this section, we investigate two further aspects, namely the impact of the initial representation, i.e. the sensitivity to a clever representation, and the impact of regularization, i.e. why more data are needed to learn more complex hypotheses.

5.3.1 Impact of the initial representation
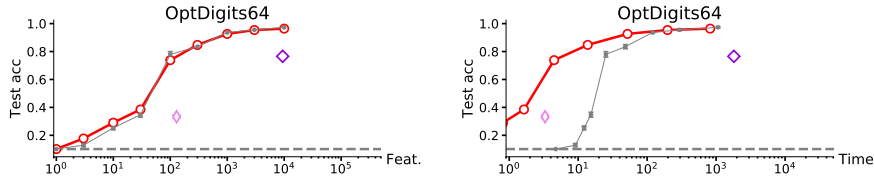

Fig. 13 Test accuracy result for the OptDigits dataset, using 4*4 blocks of pixels.
MODL: ○ red   MODL(pixel): ● gray   Tilde: ◇ pink   FORF: ◇ violet

This section evaluates the impact of the data representation, in particular whether a preprocessing step according to human expertise is needed. We consider the OptDigits dataset from UCI machine learning repository. The raw representation consists of 32×32 bitmaps. The human experts suggested an improved representation (Bache and Lichman 2013) dividing the 32×32 bitmaps " into non overlapping blocks of 4x4 and the number of activated pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions." Figure 13 shows the test accuracy with

respect to the number of features on the left and with respect to the training time on the right. We focus on 4 approaches: MODL, Tilde and FORF using the expert representation, and MODL(pixel) using the raw 32×32 bitmaps. Let us first notice that the raw representation is too large to run Tilde and FORF. Hence, the reduction of dimensionality is useful. MODL also benefits from it as can be seen with its better learning curve with respect to time. However MODL does not actually need the expert representation to achieve good accuracy. On the right-hand side of Figure 13, only training time greater than 1 second are shown. MODL(pixel) takes 100 seconds to generate complex enough features to finally reach the same accuracy as MODL using the expert representation. Moreover, the curves of MODL and MODL(pixel) with respect to the number of features, on the left-hand side of Figure 13, are close, showing that MODL is able to build relevant features in both representations.

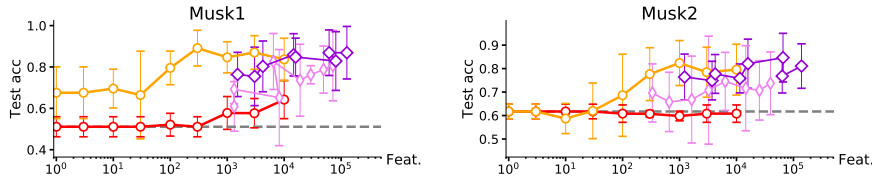### 5.3.2 Impact of feature construction regularization

Fig. 14 Test accuracy result for the Musk1 en Musk2 datasets, with or without (NR) regularization of feature construction.
MODL: ○ red   MODL(NR): ○ orange   Tilde: ◇ pink   FORF: ◇ violet

In this section, we further study the impact of the construction cost (term $L(M_C(X))$ in Formula (3)) exploited in the MODL approach to penalize complex constructed features. Using this construction cost, the approach is regularized and very resilient to overfitting, at the expense of underfitting for noisy tiny datasets such as Musk1 or Musk2. Indeed, Figure 10 illustrates that the accuracy of MODL increases when the number of features increases: there is no overfitting. Moreover, we checked in Subsection 5.2.1 on robustness that our regularization detects when there is nothing to learn. Therefore, regularization is effective on all datasets. This section focuses rather on why regularization prevents MODL to reach accuracies as good as other expressive approaches like Tilde or FORF on small datasets (Musk1 and Musk2), and finally whether this behaviour is desirable.

In a new experiment, the MODL approach is applied again on Musk1 and Musk2 dataset, with or without (NR) regularization, and compared to the most accurate alternative methods, *Tilde* and *FORF*. The results, presented in Figure 14, show that without regularization, the MODL approach obtains competitive results.

Table 3 Test accuracy results for the Musk1 and Musk2 datasets.

Dataset	MODL	MODL(NR)	Tilde	FORF
Musk1	0.642 ± 0.092	0.837 ± 0.102	0.806 ± 0.170	0.869 ± 0.127
Musk2	0.608 ± 0.037	0.795 ± 0.108	0.737 ± 0.134	0.811 ± 0.095

Table 3 reports the detailed accuracy results obtained by each method using the largest number of generated features. Despite the large standard deviations, the results obtained by MODL are better without than with regularization, and equivalent to the results of FORF.

Let us now focus on the Musk1 dataset to illustrate how the regularization actually works. In the ten-fold experiments with 10,000 generated features, MODL keeps only two features after filtering the features whose cost is beyond the cost of the null model (see Section 2.3). Without regularization, the construction cost $L(M_C(X))$ is not exploited and around 1,650 features are kept after filtering; these many features explain the improvement in test accuracy.

Table 4 Null cost for the Musk1 dataset.

Cost term	Formula	Approximation	nats
Construction model	$L(M_C(\emptyset))$	0	0
Preparation model	$L(M_P(\emptyset))$	$\log N$	4.4
Data	$L(D_Y|M_P(\emptyset))$	$Nent(Y)$	55.4
Total	$cost_P(\emptyset)$		59.8

Let us compute an approximation of the costs for Musk1, using a training dataset containing $N = 80$ instances with two equidistributed targets ($N_{.1} = N_{.2} = 40$, $ent(Y) = \log 2$). Using the notations of Section 2.3, the null cost is evaluated in Table 4. It takes about 59.8 nats[3] to encode the class values without using any input feature.

With regularization, the most informative generated feature X is *Min (Conformations.f129)*, where *Min* is chosen among the set {*Min, Max, Mean, Median, StdDev, Sum*} of $n_c = 6$ construction rules and *f129* is a numerical secondary feature chosen among the $n_s = 166$ available ones. The optimal discretization of X consists in two intervals, with the first one containing $N_{1.} = 20$ molecules of the same class ($N_{11} = 20, N_{12} = 0$, $ent(Y|X_{Int_1}) = 0$) and the second one containing $N_{2.} = 60$ molecules with the remaining classes ($N_{21} = 20, N_{22} = 40$, $ent(Y|X_{Int_2}) = -\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3} \approx 0.64$).

---

[3] the nat is the natural unit of information based on natural logarithm rather than base 2 logarithm, which defines the bit

Table 5 Cost of the most informative feature for the Musk1 dataset.

Cost term	Formula	Approximation	nats
Construction model	$L(M_C(X))$	$\log n_c + \log n_s$	6.9
Preparation model (bound position)	$L(M_{P_{Bound}}(X))$	$\log N$	4.4
Preparation model (interval 1)	$L(M_{P_{Int_1}}(X))$	$\log N_{1.}$	3.0
Preparation model (interval 2)	$L(M_{P_{Int_2}}(X))$	$\log N_{2.}$	4.1
Data (interval 1)	$L(D_{Y_{Int_1}}|M_P(X), D_{X_{Int_1}})$	$N_{1.}ent(Y|X_{Int_1})$	0.0
Data (interval 2)	$L(D_{Y_{Int_2}}|M_P(X), D_{X_{Int_2}})$	$N_{2.}ent(Y|X_{Int_2})$	38.2
Total	$cost_{CP}(X)$		56.6

The cost of X is detailed in Table 5. It is 3.2 nats below the cost of the null model and the generated feature is thus detected as informative. However, the gain in nats compared to the null model is very small, and more complex features such as *Mean (Conformations.f152) where f82 ≥ 2.5* are filtered since their construction cost is larger and not counterbalanced by a gain in the data cost.

Interestingly, Table 5 shows that the construction and preparation costs grow logarithmically with the complexity of the construction domain and the sample size, whereas the data cost is proportional to the gain in conditional entropy and decreases linearly with the sample size. In the case of the Musk1 and Musk2 datasets, the classes are not fully separable even with complex constructed features, and the overhead in construction or preparation cost does not compensate for the gain in data cost. A few tenths additional instances (like in the mutagenesis dataset) would be necessary to keep more constructed features so as to improve the test accuracy.

Does it mean that the MODL method is over-regularized? As a matter of fact, trading robustness for a better sensitivity to pattern detection does not look interesting in an industrial context, where the high resilience to overfitting is a key advantage. Instead of introducing a user parameter to decrease the weight of the regularization terms, a more promising direction is to work on the design of more parsimonious priors for feature construction or data preparation. Still, in case of tiny datasets and complex construction domains, there is a limit to what can be learned robustly.

For example, the Michalski's train dataset presented in Figure 15 is a toy problem of size ten that aims at finding rules that distinguish Eastbound trains from Westbound trains. Overall, 10 bits (6.9 nats) are sufficient to encode the class labels whereas the construction domain allows to construct a huge number of features, far beyond 1,000. As $\log_2 1,000 \approx 10$, this means that whatever be the labels of the ten trains, it is likely to find rules that separate the labels perfectly. Furthermore, even when restricting to "simple" rules, there are certainly many candidate rules that get a perfect accuracy on the ten trains. Given new trains from a test datasets, among those "perfect"
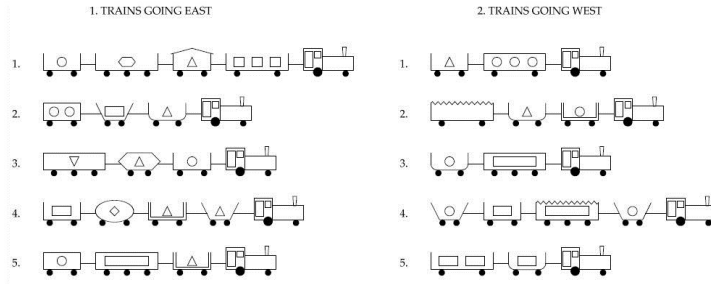
Fig. 15 Michalski's original set of 10 trains.

rules, some may fit the new data well whereas others are likely to make errors: choosing one best rule among all candidates is quite close to making a random guess. In this case, expressive methods that are able to fit any data are not likely to be resilient to overfitting. And the role of regularization is to penalize over-complex rules that can be selected only if they are supported by enough data.

6 Orange Call Detail Records Dataset

We introduce the Orange call detail records (CDR) dataset, present experimental results for three classification tasks and discuss open questions for future research directions.

6.1 Description of the dataset

The Orange CDR dataset originates from Orange, a major french telecommunication company. The objective of releasing these data is to provide the multi-relational data mining academic community with a challenging dataset, representative of real world multi-table classification problems.

The data comes from a real Orange database (from year 2007), with raw uncleaned data. It is structured as a star schema, with a main table containing 100,000 customers and a secondary table containing six months of CDRs per customer, with a total amount of 37 million CDRs and 1.3 GB storage in a tabular format.

The customer table consists in two fields:

- Categorical Id: identifier of the customer,
- Categorical Target: target column.

Three targets are provided, which are representative of the difficulty of marketing tasks, such as churn, fraud or up-selling. The proposed targets are related to classification tasks that deal with the problem of filling missing values in real world datasets:

- Gender: (*Male*; *Female*),
- AgeRange: (*20-30*; *30-40*; *40-50*; *50-60*; *60-90*),
- Region: (*DOM*, *IdF*, *North-East*, *South-East*, *South-West*, *West*).

The CDR table contains nine fields:

- Categorical Id: identifier of the customer,
- Numerical Weekday,
- Time (format *HH:MM:SS*),
- Numerical Duration,
- Numerical InVolume,
- Numerical OutVolume,
- Categorical TypeA,
- Categorical TypeB,
- Categorical TypeC.

For proprietary, privacy and scalability reasons, only a small fraction of the tables and fields in the original Orange dataset has been released, with the main following transformation rules:

- the Id is a categorical column, between *I000001* and *I100000*,
- the empty (categorical) or missing (numerical) values are preserved,
- the numerical columns are slightly shuffled
  (multiplied by $(1 + random() * 10^{-3})$),
- the InVolume and OutVolume are normalized between 0 and 10^9,
- all numerical values are then truncated to integer precision,
- the Time column is truncated with 10 minutes precision,
- the three columns TypeA, TypeB and TypeC, which represent different categorizations of the CDRs, are recoded with values $\{a_1, a_2, a_3, a_4\}$, $\{b_1, b_2, b_3\}$ and $\{c_1, c_2, c_3, c_4\}$.

The data are divided into training and test datasets, each containing 50,000 customers. Altogether, there are three customer files (one per target) and one CDR file, for training and for testing. [4].

There are on average 375 CDRs per customer in the training dataset, with a standard deviation of 630 and a maximum of 40,000. Figure 16 shows the distribution of the numbers of CDRs per customer using a log scale on each axis, with more than 80% of the customer having between 100 and 1,000 CDRs.

6.2 Experimental results

Among the methods evaluated in Section 5, only the MODL, RELAGGS, cardinalisation and quantilisation methods could be applied to the Orange CDR dataset without scalability issue. The ranking/ordering of objects is more important than the prediction as it is usual in marketing tasks to select a relevant number of predicted positive. Hence, the Area under the ROC curve (AUC)

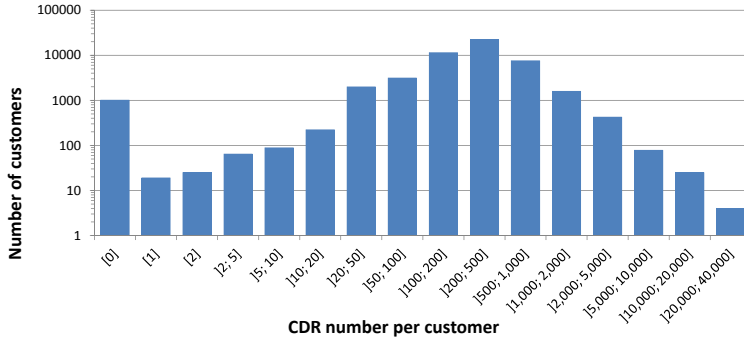[4] The dataset will be available in the UCI repository

Fig. 16 Number of CDRs per customer.

(Fawcett 2003) is used rather than the accuracy to assess the performance of each method. For multi-class AUC, we use the approach of (Provost and Domingos 2001) by computing each one-against-the-others two-class AUC and weighting them by the class prior probabilities.
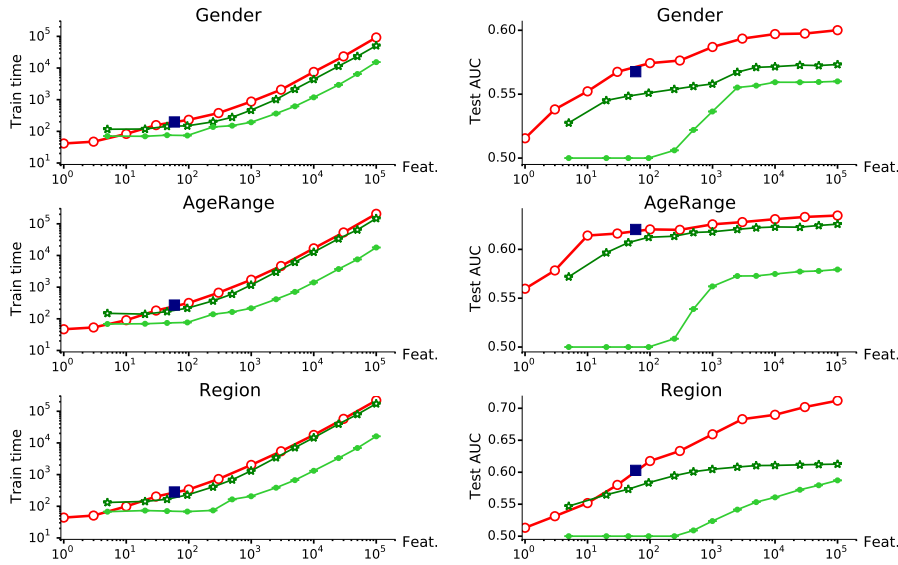

Fig. 17 Training time (left) and test AUC (right) versus the number of generated features per dataset.

MODL: ○ red   RELAGGS: ■ navy   Quantil.: ⋆ green   Cardinal.: ● lime

We train each classifier on the training dataset for the three targets and report both the training time and the test AUC in Figure 17.

6.2.1 Training time results

The overall training time of the MODL method includes the feature construction, feature preprocessing (discretization and value grouping) and the feature selection time in the Selective Naive Bayesian (SNB) classifier. The feature construction time depends on the number of secondary records and the complexity of the construction formulas; overall, it is roughly linear with the number of generated features. The feature preprocessing time is $O(KN \log(N) \, C)$, where K is the number of features, N the number of instances and C the number of classes of the target feature (see time complexity of preprocessing methods (Boullé 2006, 2005)). The feature selection time is $O(K_\star N \log(KN) \, C)$ (see time complexity of the SNB classifier (Boullé 2007)), where $K_\star$ is the number of features kept after filtering the non-informative features (see Section 2.5). Figure 17 (left) confirms that the overall training time grows almost linearly with the number of generated features. For small numbers of features, typically less than 100, a constant training time is needed, because the whole dataset including all training instances and secondary records must be read at least once. Beyond this threshold, the training time is dominated by the feature selection time, especially when the number $K_\star$ of informative features is high. The quantilisation and especially the cardinalisation methods require less training time because they produce many non-informative features that are filtered before the feature selection step. The largest representation with 100,000 generated features involves a dataset with five billion values (50,000 instances × 100,000 features), each computed using a complex formula involving around 400 CDRs. In this extreme case, the MODL method needs around one day training time [5] for the *Gender* target and two days for the other two targets that have 4 and 5 classes.

6.2.2 Test AUC results

Figure 17 (right) shows that the MODL method dominates the other evaluated methods, whatever be the number of generated features.

The RELAGGS method is quite competitive, getting good performance with few generated features. However its expressivity is limited to features involving at most one secondary feature and this prevents the method from obtaining top performance.

The quantilisation and cardinalisation methods construct features according to one user parameter, the number of intervals per secondary feature. The performance of both methods increases with the number of generated features, but they reach a plateau as soon as the number of generated features per secondary numerical feature reaches the number of values per instance. Because of the unbalanced distribution of number of CDRs per customer (see Figure 16), the cardinalisation method needs to construct more features to discretize efficiently the secondary numerical features, and does not even reach

---

[5] Training time on a PC Intel Xeon L5640, Windows, using one single processor and 40 GB RAM

the performance of the quantilisation method. The quantilisation method performs only slightly better than the RELAGGS method, with at most 0.01 better test AUC, in spite of far larger numbers of generated features. Overall, the quantilisation and cardinalisation generated features are too simple to reach superior performance.

The MODL method, which has a strict control over the number of generated features, is able to construct complex features such as *"StdDev(CDRs.Duration) where OutVolume in]150, 6350] and Time in]09:40, 14:10]"* and to assess their informativeness efficiently and reliably. The SNB classifier exploits this representation space and obtains test performance that significantly increases with the number of features and dominates the other evaluated methods. However, whereas the MODL method attained a plateau of performance within a few hundred or a thousand features in the benchmark experiments presented in Section 5, this is not the case for the Orange CDR dataset. Especially in the case of the *Region* target, the AUC is 0.659 for 1,000 features, 0.690 for 10,000 features and 0.712 for 100,000 features. It still increases significantly beyond 10,000 generated features and the plateau of performance has not been reached.

6.3 Discussion

The experiments performed with the Orange CDR dataset show that the related classification tasks are challenging for the objective of automatic feature construction, with both scalability and performance issues. This suggests that human expert tuned methods could obtain superior performance and that there is room for improvement in automatic data mining.

The MODL approach presented in this paper has demonstrated appealing properties for automation, scalability, robustness and performance on classification tasks from relational data, but many questions remain open, that may stimulate future research directions:

– beyond the random sampling algorithm exploited to construct features (cf. Section 4.2), how to better explore the huge space of potential representations to achieve better performance with fewer features?
– how to construct features that are both informative individually and non redundant, so as to get a set of features that are informative collectively?
– how to extend the specification of the feature construction domain, with more complex data structures and larger number of construction rules, while keeping the robustness, scalability and performance of the approach?
– how to efficiently address the problem of anytime learning, with potentially large and complex multi-table databases and infinite number of potential representations?
– how to efficiently parallelize the whole training process, considering feature construction, preprocessing and selection in an anytime setting?

7 Related works

We propose an efficient propositionalization approach for Bayesian classification of mixed numerical and categorical data. Therefore, related works are first-order Bayesian classifiers and propositionalization approaches able to deal with mixed numerical and categorical data, in particular apply typical aggregation functions such as min, max, etc. to the numerical columns of secondary tables.

This work follows a long history of propositionalization approaches. Several years before Kramer et al (1998) defined the term propositionalization itself, LINUS (Lavrac et al 1991) was already generating elementary features: features relating a single property of a secondary table through one or several one-to-many relationships using existential quantifiers. This is similar to the elementary features generated exhaustively by 1BC. Many approaches use frequent patterns or queries for propositionalization, e.g. (Zelezný and Lavrac 2006; Landwehr et al 2007). However, most propositionalization approaches only deal with the set of objects related to the main individual, for example all the calls of one customer, using the existential quantifier only, e.g. "there exists a phone call such that..." This includes recent works, for instance (Kuzelka and Zelezný 2011), which can very efficiently generate complex features, involving several properties of several kinds of related objects but through existential quantifiers only. Moreover, they mainly deal with numerical attributes by discretising them into categorical attributes. Similarly, most full-fledged relational data mining systems, directly learning from relational data without propositionalizing and using an external attribute-value learner, focus on the existential quantifier and categorical data too. Only the ACE system integrates complex aggregates (Van Assche et al 2006) into first-order trees, Tilde, and into first-order random forests, FORF. However they do not scale up well.

Knobbe et al (2001) and Krogel and Wrobel (2001) with their respective systems POLKA and RELAGGS were the first to propose to use aggregation functions other than the existential quantifier to build features. POLKA and RELAGGS mainly differ in the way they deal with nested tables: tables related to secondary tables through further one-to-many relationships. POLKA applies aggregation operators recursively, from the deepest relationship to the main table. RELAGGS joins the secondary table to the lower level tables depending on it, and then applies aggregation operators to the columns of the joined table. These joins behave like the identifier propagation of Cross-mine (Yin et al 2004). Whereas POLKA and RELAGGS consider simple aggregates, applying one aggregation function to a set of related objects, satisfying a single condition, we consider complex aggregates, applying one aggregation function to a set of related objects satisfying complex conditions, possibly involving conditions on complex aggregates of nested tables if any.

CILP++ (França et al 2014) is similar to our approach because it proposes a propositionalization approach dedicated to an attribute-value learner. However, a neural network is considered whereas we focus on the naive Bayesian classifier for scalability. Indeed, the reported run times go up to hours for

datasets containing hundreds of individuals only. Moreover, their software deals with categorical attributes mainly and the Bottom Clause Proposition-alization cannot generate complex aggregates.

Focusing on first-order Bayesian classifiers now, MR-SBC (Ceci et al 2003) is an alternative to 1BC and 1BC2. It builds exactly the same features as 1BC, called elementary features in 1BC, which link a property of a related table to the main table through a foreign key path. The main differences are that MR-SBC connects to a Relational Database Management System whereas 1BCs use prolog facts from a file, MR-SBC estimates the probabilities differently and MR-SBC discretises numerical attributes differently. Thus, MR-SBC considers exactly the same features as 1BC and the difference comes mainly from the implementation of the propositional naive Bayesian classifier. Ceci et al (2003) compared 1BC to MR-SBC and reports one win for 1BC and one win for MR-SBC. Therefore, we consider they have an equivalent expressivity.

Graph-NB (Liu et al 2005) reduces the run time of Cross-mine (Yin et al 2004). It estimates probabilities of tables linked through a semantic relation-ship graph, which acts as a language bias. The identifiers from the main ta-ble are propagated to secondary tables, hence probabilities of related objects necessarily take into account the repetitions of similar objects. Actually, the estimated probabilities are similar to those of 1BC2. The accuracies reported in (Liu et al 2005) are indeed similar to those of MR-SBC, 1BC and 1BC2.

The sampling algorithm is similar to the approach considered by Schietgat et al (2011) in a graph context but has never been used in propositionalization.

8 Conclusion

Most companies nowadays claim that they want to get value out of the vast amount of data they collect, but these data come in the form of potentially complex relational schema. Mining these data requires to be able to generate complex aggregates over mixed, numerical and categorical, data. The availabil-ity of large datasets may be seen as an opportunity, since it may enable more expressive learners to produce more accurate models. However such learners can only seize this opportunity if they can scale up and do so without any risk of overfitting. Our contribution is a new propositionalization approach for a bayesian classifier that outperforms the state-of-the-art on most inves-tigated datasets. It is robust: it does not overfit when the number of fea-tures is increased. It is efficient: it can generate a given number of features, in a breadth-first approach driven by the prior distribution of the features. Whereas NFOIL proposes a dynamic propositionalization selecting the best features for the bayesian classifiers, and Tilde and FORF similarly perform a dynamic depth search, the MODL approach proposes a static propositional-ization generating a large set of features through a breadth search. The curse of dimensionality is kept under control by the regularization of the robust classifier (SNB).

The accuracy of the proposed approach generally increases with the number of features, thanks to its robustness, mainly due to regularization based on the definition of a prior distribution over complex aggregates. The single remaining parameter of our approach, the number of features, could be tuned automatically. A wrapping script could implement a loop doubling the number of features until the user stops it, in an anytime fashion. Also, the loop could be stopped when the increase in accuracy falls below some threshold.

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. Additionally, accounting for correlations between the constructed variables so as to avoid the risk of constructing many variants of the same variables raises another challenge.

Our approach (regularization, prior distribution, and breadth-first search) could be investigated for other propositional learners, as long as they are robust, for example SVM. It could even be integrated into full-fledged first-order logic learners.

References

Ahmed CF, Lachiche N, Charnay C, Jelali SE, Braud A (2015) Flexible propositionalization of continuous attributes in relational data mining. Expert Syst Appl 42(21):7698–7709, DOI 10.1016/j.eswa.2015.05.053, URL `http://dx.doi.org/10.1016/j.eswa.2015.05.053`

Bache K, Lichman M (2013) UCI machine learning repository. URL `http://archive.ics.uci.edu/ml`

Blockeel H, De Raedt L (1998) Top-down induction of first-order logical decision trees. Artif Intell 101(1-2):285–297

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

Boullé M (2009) A parameter-free classification method for large scale learning. Journal of Machine Learning Research 10:1367–1385

Boullé M (2014) Towards automatic feature construction for supervised classification. In: Calders T, Esposito F, Hüllermeier E, Meo R (eds) Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I, Springer, Lecture Notes in Computer Science, vol 8724, pp 181–196, DOI 10.1007/978-3-662-44848-9_12, URL `http://dx.doi.org/10.1007/978-3-662-44848-9_12`

Burke M (2013) Multilinear function factorisation for time series feature extraction. In: 18th International Conference on Digital Signal Processing (DSP), IEEE, pp 1–8, DOI 10.1109/ICDSP.2013.6622721

Ceci M, Appice A, Malerba D (2003) Mr-sbc: A multi-relational naïve bayes classifier. In: Lavrac N, Gamberger D, Blockeel H, Todorovski L (eds) Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings, Springer, Lecture Notes in Computer Science, vol 2838, pp 95–106, DOI 10.1007/978-3-540-39804-2_11, URL https://doi.org/10.1007/978-3-540-39804-2_11

Cover T, Thomas J (1991) Elements of information theory. Wiley-Interscience, New York, NY, USA

De Raedt L (1998) Attribute-Value Learning Versus Inductive Logic Programming: The Missing Links (Extended Abstract). In: Page D (ed) Proceedings of the 8th International Workshop on Inductive Logic Programming, ILP '98, Springer-Verlag, pp 1–8

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Reseasrch 7:1–30

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

Džeroski S, Schulze-Kremer S, Heidtke KR, Siems K, Wettschereck D, Blockeel H (1998) 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

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

Fawcett T (2003) ROC graphs: Notes and practical considerations for researchers. Tech. Rep. HPL-2003-4, HP Laboratories

Féraud R, Boullé M, Clérot F, Fessant F, Lemaire V (2010) The orange customer analysis platform. In: Proceedings of the 10th Industrial Conference on Data Mining, Springer Verlag, pp 584–594

Flach PA, Lachiche N (2004) Naive bayesian classification of structured data. Machine Learning 57(3):233–269, DOI 10.1023/B:MACH.0000039778.69032.ab, URL https://doi.org/10.1023/B:MACH.0000039778.69032.ab

França MVM, Zaverucha G, d'Avila Garcez AS (2014) Fast relational learning using bottom clause propositionalization with artificial neural networks. Machine Learning 94(1):81–104, DOI 10.1007/s10994-013-5392-1, URL https://doi.org/10.1007/s10994-013-5392-1

Gärtner T, Lloyd JW, Flach PA (2004) Kernels and distances for structured data. Machine Learning 57(3):205–232, DOI 10.1023/B:MACH.0000039777.23772.30, URL http://dx.doi.org/10.1023/B:MACH.0000039777.23772.30

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

Knobbe AJ, de Haas M, Siebes A (2001) Propositionalisation and aggregates. In: Raedt LD, Siebes A (eds) Principles of Data Mining and Knowledge Discovery, 5th European Conference, PKDD 2001, Freiburg, Germany, September 3-5, 2001, Proceedings, Springer, Lecture Notes in Computer Science, vol 2168, pp 277–288, DOI 10.1007/3-540-44794-6_23, URL https://doi.org/10.1007/3-540-44794-6_23

Kramer S, Pfahringer B, Helma C (1998) Stochastic propositionalization of non-determinate background knowledge. In: Page D (ed) Inductive Logic Programming, 8th International Workshop, ILP-98, Madison, Wisconsin, USA, July 22-24, 1998, Proceedings, Springer, Lecture Notes in Computer Science, vol 1446, pp 80–94, DOI 10.1007/BFb0027312, URL https://doi.org/10.1007/BFb0027312

Krogel MA, Wrobel S (2001) Transformation-based learning using multirelational aggregation. In: ILP, Springer, pp 142–155

Kuzelka O, Zelezný F (2011) Block-wise construction of tree-like relational features with monotone reducibility and redundancy. Machine Learning 83(2):163–192, DOI 10.1007/s10994-010-5208-5, URL https://doi.org/10.1007/s10994-010-5208-5

Lachiche N (2017) Propositionalization. In: Sammut C, Webb GI (eds) Encyclopedia of Machine Learning and Data Mining, Springer US, Boston, MA, pp 1025–1031, DOI 10.1007/978-1-4899-7687-1_686, URL http://dx.doi.org/10.1007/978-1-4899-7687-1_686

Lachiche N, Flach P (1999) 1BC: A first-order bayesian classifier. In: Proceedings of the 9th International Workshop on Inductive Logic Programming, Springer-Verlag, pp 92–103

Lachiche N, Flach P (2002) 1BC2: A true first-order bayesian classifier. In: Proceedings of the 12th International Conference on Inductive Logic Programming, Springer-Verlag, pp 133–148

Landwehr N, Kersting K, Raedt LD (2007) Integrating naïve bayes and FOIL. Journal of Machine Learning Research 8:481–507, URL `http://dl.acm.org/citation.cfm?id=1314516`

Lavrac N, Dzeroski S, Grobelnik M (1991) Learning nonrecursive definitions of relations with LINUS. In: Kodratoff Y (ed) Machine Learning - EWSL-91, European Working Session on Learning, Porto, Portugal, March 6-8, 1991, Proceedings, Springer, Lecture Notes in Computer Science, vol 482, pp 265–281, DOI 10.1007/BFb0017020, URL `https://doi.org/10.1007/BFb0017020`

Liu H, Yin X, Han J (2005) An efficient multi-relational naÏve bayesian classifier based on semantic relationship graph. In: Proceedings of the 4th International Workshop on Multi-relational Mining, ACM, New York, NY, USA, MRDM '05, pp 39–48, DOI 10.1145/1090193.1090200, URL `http://doi.acm.org/10.1145/1090193.1090200`

Lodhi H, Muggleton S (2005) Is mutagenesis still challenging? In: Late-breaking papers of the international conference on inductive logic programming, pp 35–40

Meena L, Devi VS (2015) Prototype selection on large and streaming data. In: Arik S, Huang T, Lai WK, Liu Q (eds) Neural Information Processing - 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part I, Springer, Lecture Notes in Computer Science, vol 9489, pp 671–679, DOI 10.1007/978-3-319-26532-2_74, URL `http://dx.doi.org/10.1007/978-3-319-26532-2_74`

Perovsek M, Vavpetic A, Kranjc J, Cestnik B, Lavrac N (2015) Wordification: Propositionalization by unfolding relational data into bags of words. Expert Syst Appl 42(17-18):6442–6456, DOI 10.1016/j.eswa.2015.04.017, URL `http://dx.doi.org/10.1016/j.eswa.2015.04.017`

Provost F, Domingos P (2001) Well-trained pets: Improving probability estimation trees. Tech. Rep. CeDER #IS-00-04, New York University

Quinlan J (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann

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

Rissanen J (1983) A universal prior for integers and estimation by minimum description length. Annals of Statistics 11(2):416–431

Schietgat L, Costa F, Ramon J, Raedt LD (2011) Effective feature construction by maximum common subgraph sampling. Machine Learning 83(2):137–161, DOI 10.1007/s10994-010-5193-8, URL `https://doi.org/10.1007/s10994-010-5193-8`

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

Srinivasan A, Muggleton S, King R, Sternberg M (1994) Mutagenesis: ILP experiments in a non-determinate biological domain. In: Wrobel S (ed) Proceedings of the 4th International Workshop on Inductive Logic Programmin (ILP'94), volume 237 of GMD-Studien, pp 217–232

Van Assche A, Vens C, Blockeel H, Džeroski S (2006) First order random forests: Learning relational classifiers with complex aggregates. Machine Learning 64(1-3):149–182

Vitter J (1985) Random sampling with a reservoir. ACM Transactions on Mathematical Software 11(1):37–57

Yin X, Han J, Yang J, Yu PS (2004) Crossmine: Efficient classification across multiple database relations. In: Özsoyoglu ZM, Zdonik SB (eds) Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA, IEEE Computer Society, pp 399–410, DOI 10.1109/ICDE.2004.1320014, URL `https://doi.org/10.1109/ICDE.2004.1320014`

Zelezný F, Lavrac N (2006) Propositionalization-based relational subgroup discovery with RSD. Machine Learning 62(1-2):33–63, DOI 10.1007/s10994-006-5834-0, URL `https://doi.org/10.1007/s10994-006-5834-0`

Zhou C (2015) Conformal and venn predictors for multi-probabilistic predictions and their applications. PhD thesis, Royal Holloway, University of London

Zhou Zh, Zhang ML (2007) 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'06), Cambridge, MA: MIT Press, pp 1609–1616