Numbers in Multi-Relational Data Mining

Arno J. Knobbe[1,2], Eric K. Y. Ho[1]

[1]Kiminkii, Postbus 171, NL-3990 DD, Houten, The Netherlands
a.knobbe@kiminkii.com, e.ho@kiminkii.com
[2]Utrecht University, P.O. box 80 089, NL-3508 TB Utrecht, The Netherlands

Abstract Numeric data has traditionally received little attention in the field of Multi-Relational Data Mining (MRDM). It is often assumed that numeric data can simply be turned into symbolic data by means of discretisation. However, very few guidelines for successfully applying discretisation in MRDM exist. Furthermore, it is unclear whether the loss of information involved is negligible. In this paper, we consider different alternatives for dealing with numeric data in MRDM. Specifically, we analyse the adequacy of discretisation by performing a number of experiments with different existing discretisation approaches, and comparing the results with a procedure that handles numeric data dynamically. The discretisation procedures considered include an algorithm that is insensitive to the multi-relational structure of the data, and two algorithms that do involve this structure. With the empirical results thus obtained, we shed some light on the applicability of both dynamic and static procedures (discretisation), and give recommendations for when and how they can best be applied.

1 Introduction

Whereas numeric data is at the core of the majority of propositional Data Mining systems, it has been largely overlooked in Multi-Relational Data Mining (MRDM). Most MRDM systems assume that the data is a mixture of symbolic and structural data, and if the source database contains numbers, they will either have to be filtered out or pre-processed into symbolic values. Apart from historical reasons – symbolic representations are popular in the logical roots of MRDM –, the full treatment of numeric data comparable to propositional approaches is mostly ignored for reasons of simplicity and efficiency. MRDM is characterised by large hypothesis spaces, and the inclusion of continuous domains that offer a large range of (very similar) refinements is thought to make MRDM intractable. Most multi-relational systems rely on so-called discretisation procedures to reduce the continuous domains to more manageable symbolic domains of low cardinality, such that the search remains realistic. The resulting loss of precision is assumed to be negligible.

In this paper, we survey a number of existing approaches to dealing with numeric data in MRDM, with the aim of empirically determining the value of each of these approaches. These approaches include a number of pre-processing procedures suggested recently [5, 2], as well as one of the few MRDM algorithms that deal with numbers dynamically, developed by the authors of this paper [2, 3]. The discretisation procedures include a simple algorithm that considers each table in isolation, and

discretises each numeric attribute on the basis of the distribution of its values, regardless of any other tables connected to the current table. Two further discretisation procedures do involve the multi-relational structure of the database, and aim at finding good intervals, keeping in mind that the resulting symbolic attributes will be used in the context of the other tables in the database. The algorithm that deals with numbers dynamically does not require any pre-processing of the data. Rather than fixing a number of intervals prior to the analysis, it will consider the numeric data for a hypothesis at hand, and determine thresholds that are optimal for the given context. Especially at deeper levels of the search, where reasonably specific subgroups are considered, relevant thresholds will differ significantly from those determined on the whole dataset.

We test the four approaches experimentally on four well-known multi-relational datasets where numeric attributes play an important role: Mutagenesis (two varieties), Financial and Musk. With these experiments, we aim to shed some light on when and how each approach can best be applied. Furthermore, we hope to get some guidelines for important parameters of the discretisation procedures, such as the coarseness of the discretisation and the choice of representation. The experimental results are compared to those obtained on databases where all numeric information is removed, in order to get a baseline for the procedures that do (to some extent) involve the continuous domains.

2 Foundations

In the class of discrete patterns that we aim at (decision trees, rules, etc.), dealing with numeric data comes down to choosing numeric thresholds that form useful subgroups. Clearly, the distribution of numeric values, and how the target concept depends on this distribution is essential. In propositional data mining, choosing thresholds is fairly straightforward, as there is a one-to-one correspondence between occurring values and individuals. In MRDM however, we are dealing with non-determinate (i.e. one-to-many) relations between tables. In many cases, numeric attributes do not appear in the target table, and multiple values of the attribute are associated with a single structured individual. Whereas in propositional data mining, we can think of the whole database as a 'cloud' of points, in MRDM each individual forms a cloud (see Figure 1). The majority of pattern languages in MRDM characterise such individuals by testing for the presence of values that exceed a given threshold. As the following lemma shows, only the largest and smallest values within each individual are relevant to include or exclude an individual on the basis of a single numeric test. Only these values will therefore be candidates for numeric thresholds.

Lemma 1 Let B be a bag of real numbers, and t some real, then

$$\exists\, v \in B: v \geq t \;\; \textit{iff} \;\; \max(B) \geq t,$$
$$\exists\, v \in B: v \leq t \;\; \textit{iff} \;\; \min(B) \leq t.$$

Example Consider the **account** and **transaction** table in the Financial dataset (see Section 5). Every individual (account) consists of a large number of transactions. Although the distribution of values for the numeric attributes *amount* and *balance* in the **transaction** table is interesting, not every occurring value can act as a threshold to

include or exclude individuals. Figure 1 presents the transactions of two individuals (black and grey dots respectively). To include an individual on the basis of an existential test involving one of these attributes, only the minimum and maximum value per individual is relevant. These thresholds are represented by the dashed lines in the graph.

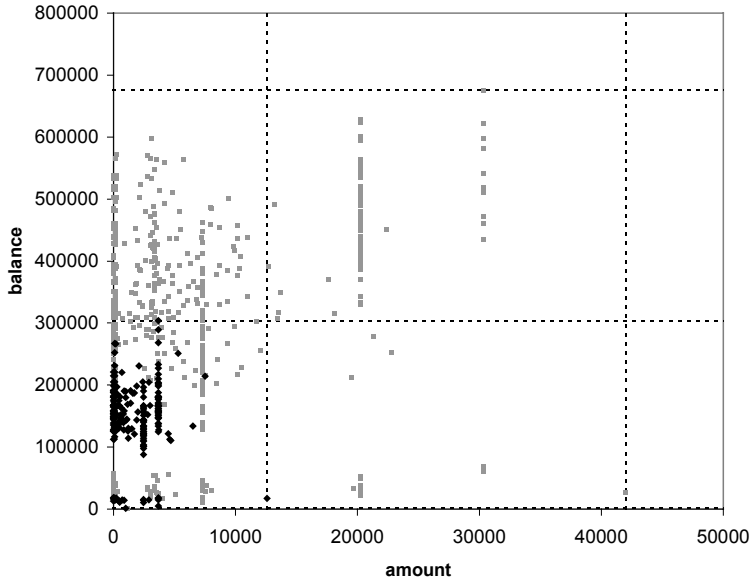

Figure 1 The amount and balance of transactions related to two accounts.

Lemma 1 furthermore demonstrates that there is a difference between the set of thresholds appropriate for the ≤and the ≥ operator. This means that any procedure that selects thresholds will have to be performed separately for each operator.

Choosing thresholds can roughly be done in two ways: dynamically and statically. A *dynamic* approach (see Section 3) considers the hypothesis at hand, and determines a collection of thresholds on the basis of the information contained in the individuals covered by the hypothesis in question. A *static* approach (see Section 4) on the other hand considers the entire database prior to analysis and determines a collection of thresholds once and for all. Typically these thresholds are then used to pre-process the data, replacing the numeric data with symbolic approximations. We refer to such a pre-processing step as *discretisation*. Clearly, a dynamic approach is preferable from an accuracy standpoint, as optimal thresholds are computed for the situation at hand. On the other hand, dynamic computation of thresholds makes algorithms more complex, and less efficient.

In the context of discretisation, we refer to numeric thresholds as *cut points*. A collection of n-1 cut points splits the continuous domain into n intervals. A group of values falling in a specific interval is referred to as a *bin*.

In MRDM, it makes sense to not just consider the available numeric values in the computation of cut-points, by also the multi-relational structure of the database. In general, a table is connected to other tables by associations, some of which may be non-determinate (a single record in one table corresponds to multiple records in another table). The effect of such associations is thus that records in a table can be divided into *groups*, depending on the relation to records in the associated table. Considering the multi-relational structure in the computation of cut points is hence tantamount to considering the numeric value, as well as the group the value belongs to. In the remainder of this paper, we refer to groups as the sets implied by this multi-relational structure.

3 Dynamic Handling of Numbers

An MRDM algorithm that handles numbers dynamically considers a range of cut points for a given numeric attribute, and determines how each of these tentative cut points influences the quality of a multi-relational hypothesis under consideration. As the optimal cut point depends on the current hypothesis, and many hypotheses are considered by an MRDM algorithm, the set of relevant cut points cannot be determined from the outset. Rather, we will have to consider the subgroup at hand, and query the database for a list of relevant cut points, and associated statistics.

In general, all values for the numeric attribute that occur in the individuals covered by the hypothesis at hand can act as candidate cut points. In theory, this set of values can be quite large, which can make the dynamic generation of cut points very inefficient. The MRDM system Safarii [2, 3] uses an approach that considers only a subset of these values, thus reducing some of the work. It relies on the observation from Lemma 1 that only the extreme values within a bag of numbers are relevant in order to test the presence of values above or below a certain cut point. Safarii uses a database primitive (a predefined query template) called NumericCrossTable [2] that selects the minimum (maximum) value within each individual covered by the current hypothesis, and then groups over these extreme values to produce the desired counts. We thus get a more reasonable number of candidate refinements.

Unfortunately it is still not realistic to continue the search on the basis of each of these refinements. Safarii therefore selects from the reduced set of candidate refinements only the optimal one for further examination. Because the operators $\leq$and $\geq$ produce two different sets of candidate refinements, we essentially get two refinements per hypothesis and numeric attribute encountered. Note that keeping only the optimal refinements introduces a certain level of greediness into the algorithm.

4 Discretisation

In this section, we present three methods for discretising numeric data. Conceptually, discretisation entails defining a number of consecutive intervals on the domain of a numeric attribute, and replacing this attribute with a nominal attribute that represents the interval values fall into. The three methods are identical in how numeric attributes are transformed based on the intervals defined. The essential

difference between the methods lies in how the cut points between intervals are computed.

The first method presented computes a (user-determined) number of cut points based on the distribution of values of the numeric attribute. It ignores the fact that data in a particular table will generally be considered in the context of that in other tables. The remaining two methods do consider the multi-relational structure of the data, and compute cut points assuming that discretised values will be considered after joining with tables that are directly attached to the table at hand.

Because the numeric data typically appears in tables other than the target table, it is not always straightforward to assign a class (which is related to the target table) to the value. All three methods are therefore class-blind (or *unsupervised*): the methods do not consider a predefined target concept. As a result, the transformed data can be used on a range of class-definitions.

Equal Height Histogram The first algorithm computes cut points regardless of any multi-relational structure. It simply considers every numeric attribute in every table in turn and replaces it by a nominal attribute that preserves as much of the information in the original attribute as possible. In other words, it constructs a new feature x that maximises the entropy

$$H(x) = -\sum_i p(x_i) \lg p(x_i)$$

Information theory dictates that $H(x)$ is maximal if the probability distribution $p(x_i)$ is uniform. Therefore a collection of cut points is computed that produces bins of (approximately) equal size. Such a procedure is known as *equal interval frequency*, or *equal height histogram*, which is the term we will adopt.

The algorithm EqualHeight takes as input an array d of tuples (v, g) indicating the numeric value, and the group it belongs to (actually ignored in this algorithm). The second argument b of the algorithm indicates the desired number of bins. We denote the size of the array by $|d|$. The notation $d[s].v$ refers to the first argument of the sth tuple in d. The algorithm essentially skips through the sorted numeric values with steps $|d|/b$, and adds the values encountered as cut points. The result is a collection of b bins of roughly equal size. Note that the algorithm is class-blind: it does not consider any class-distribution over the numeric values.

Algorithm **EqualHeight(*d*, *b*)**

$r = \emptyset$
sort d on v
for l **in** [1..b-1]
    $s = \lfloor l \cdot |d|/b \rfloor$
    add $d[s].v$ to r
return r

Equal Weight Histogram The second discretisation procedure involves an idea proposed by Van Laer et al. [5]. The algorithm considers not only the distribution of numeric values present, but also the groups they appear in. It is observed that larger

groups have a larger impact on the choice of cut points because they have more contributing numeric values. In order to compensate for this, numeric values are weighted with the inverse of the size of the group they belong to:

$$w(d[i]) = 1 / \mathit{freq}\ (d[i].g)$$

Rather than producing bins of equal size, we now compute cut points to obtain bins of equal weight:

$$\sum_{i \in B} w(d[i]) = \sum_{i \in B} 1 / \mathit{freq}\ (d[i].g) \approx |d.g| / b$$

where B indicates a bin, and $|d.g|$ denotes the number of groups occurring in d. Note that the sum of all weights equals the number of groups.

The algorithm starts by first computing an array f of counts for each group g. It then moves through the sorted array of values, keeping a running sum of weights s. Whenever s reaches a current target threshold (i.e. a bin is full), the current numeric value is added as a cut point, and the target is increased.

Algorithm **EqualWeight(*d*, *b*)**

```
r = ∅
for all unique g in d.g
        f[g] = frequency of g in d.g
sort d on d.v
t = |f|/b
s = 0
for l in [0..|d|-1]
        s = s + 1/f[d[l].g]
        if s ≥ t
                add d[l].v to r
                t = t + |f|/b
return r
```

Unlike the EqualHeight algorithm, the EqualWeight algorithm may have to be applied more than once for a single attribute. Because tables can conceivably be involved in multiple one-to-many associations, there can be multiple groupings of a table. Therefore, the EqualWeight algorithm should be applied once for each combination of grouping and attribute, potentially resulting in several new nominal attributes per original numeric attribute. Note that many-to-one (as opposed to one-to-many) associations cause only groups of single elements, in which case the result of EqualWeight is equivalent to EqualHeight.

Aggregated Equal Height Histogram Like the EqualWeight algorithm, the AggregatedEqualHeight algorithm proposed in [2] takes the multi-relational structure of the database into account in the computation of the cut points. The algorithm is centred around the idea that not all values within a group are relevant when inquiring about the presence of numeric values above or below some threshold. As was outlined in Section 2, it suffices to consider the minimum and maximum value within a group. The idea of the AggregatedEqualHeight algorithm is hence to take the minimum

value per group and compute an equal height histogram on these values, in order to discretise all values. The process is then repeated for the maximum per group. We thus get two new attributes per numeric attribute. As with the EqualWeight algorithm, we need to consider multiple one-to-many associations, such that we can get multiple pairs of attributes for each original attribute.

The argument *a* indicates the desired aggregate function, either **min** or **max**.

Algorithm **AggregatedEqualHeight(*d*, *b*, *a*)**

sort *d* on *d.g*
aggregate *d* into *d´* using *a*
r = EqualHeight(*d´*, *b*)
return *r*

Representation In our discussion of the different discretisation procedures, we have assumed that the outcome is a collection of nominal attributes, where each value represents one of the computed intervals. In fact when we produce *n* nominal values, we do not only lose some amount of precision (which we assume to be minimal), but also the inherent order between intervals. Although the inability to handle ordered domains (numeric or ordinal) is part of our motivation for applying discretisation, we can choose a representation that preserves the order information without having to accommodate for it explicitly. This representation involves *n*-1 binary attributes per original numeric attribute, one for each cut point. Rather than representing each individual interval, the binary attributes represent overlapping intervals of increasing size. By adding such attributes as conjuncts to the hypothesis through repeated refinements, a range of intervals can be considered. A further advantage of this representation is that the accuracy is less sensitive to the number of intervals as the size of the intervals does not decrease with the number of intervals. An important disadvantage of this representation is the space it requires. Especially with larger numbers of intervals, having *n*-1 new binary attributes per original attribute can become prohibitive.

In our experiments, we will consider both the nominal and the binary representation, and compare the results to determine the optimal choice. We will refer to the latter representation as *cumulative binary*.

5 Experiments

Although we have multiple approaches to dealing with numeric data to test, we have chosen to apply a single mining algorithm. This allows us to sensibly compare results. The algorithm of choice is the Rule Discovery algorithm contained in the Safarii MRDM package produced by the authors [2, 3]. This algorithm produces a set of independent multi-relational rules. The algorithm includes the dynamic strategy for dealing with numbers described in Section 3. In order to test the discretisation procedures, we have pre-processed the different databases by generating the desired discretised attributes, and removing the original numeric attributes. The different discretisation procedures were implemented in the pre-processing companion to Safarii, known as ProSafarii.

Although a range of evaluation measures and search strategies is available in Safarii, we have opted for rules of high *novelty*, discovered by means of *beam search* (beam width 100, maximum depth 6). A time limit of 30 minutes per experiment was selected. The algorithm offers filtering of rules by means of a computed convex hull in ROC space [2]. The area under the ROC curve gives a good measure of the quality of the discovered rule set, as it is insensitive to copies or redundant combinations of rules. We will use this measure (values between 0.5 and 1) to compare results.

We will test the different algorithms on the following three well-known multi-relational databases:

- **Mutagenesis**. This database describes molecules falling in two classes, *mutagenic,* and *non-mutagenic* [4]. The description consists of the atoms and the bonds that make up the compound. In particular, the database consists of 3 tables that describe directly the graphical structure of the molecule (**molecule**, **atom**, and **bond**). We have used the so-called 'regression-friendly' dataset that consists of 188 molecules, of which 125 (66.5%) are *mutagenic*. The database comes in a number of 'backgrounds'. Different backgrounds contain different sets of features. We will use two backgrounds B2 and B3. B2 contains the symbolic and structural information as well as a single numeric attribute describing the charge of each atom. B3 contains two additional attributes on the molecule-level. Although these two attributes are very informative, their contribution to the results does not give any insight into how the algorithms deal with numbers in a multi-relational setting.
- **Financial**. This database is taken from the Discovery Challenge organised at PKDD '99 and PKDD 2000 [6]. The database is based on data from a Czech bank. It describes the operations of 5369 clients holding 4500 accounts. The data is stored in seven tables, three of which describe the activities of clients. Three further tables describe client and account information, and the remaining table contains demographic information about 77 Czech districts. We have chosen the **account** table as our target table. Clearly the numeric data (for example in tables **transaction**, **order**, and **district**) is crucial for obtaining good results.
- **Musk**. This database describes molecules occurring in different conformations [1] . Each molecule is either *musk* or *non-musk*, and one of the conformations determines this property. The molecules will be modelled by two tables, **molecule** and **conformation**, joined by a one-to-many association. **conformation** contains a molecule identifier plus 166 continuous features. **molecule** just contains the identifier and the class. The database comes in two versions. We have selected the largest, MuskLarge, containing 102 molecules and 6598 conformations.

Table 1 gives a detailed overview of the performance of the different procedures tested.

Discretisation Procedures Let us begin by considering how well the discretisation procedures perform. Table 2 summarises how often each procedure is involved in a win or a tie (no other procedure is superior). Procedures are compared per setting of the number of bins, in order to get comparable results. It turns out that

AggregatedEqualHeight is clearly the best choice for Financial and Musk. Surprisingly, the propositional procedure EqualHeight performs quite well on Mutagenesis B2. The results for EqualHeight and EqualWeight on Mutagenesis B3

	# bins	Mutagenesis B2	Mutagenesis B3	Financial	Musk
no numbers		0.596	0.596	0.632	0.5
dynamic		0.762	0.848	0.908	0.896
Equal Height nominal	2	0.717	0.837	0.635	0.806
	4	0.773	0.773	0.64	0.828
	8	0.731	0.731	0.711	0.832
	16	0.701	0.701	0.675	0.787
Equal Height cumulative binary	2	0.717	0.837	0.635	0.806
	4	0.765	0.898	0.641	0.827
	8	0.768	0.902	0.662	0.788
	16	0.77	0.909	0.685	0.853
Equal Weight nominal	2	0.724	0.837	0.636	0.809
	4	0.749	0.76	0.647	0.862
	8	0.731	0.731	0.714	0.792
	16	0.701	0.701	0.677	0.752
Equal Weight cumulative binary	2	0.724	0.837	0.636	0.809
	4	0.765	0.898	0.647	0.85
	8	0.744	0.902	0.67	0.81
	16	0.701	0.909	0.687	0.857
Aggregated Equal Height nominal	2	0.724	0.837	0.844	0.879
	4	0.747	0.76	0.673	0.867
	8	0.73	0.73	0.676	0.861
	16	0.718	0.718	0.804	0.822
Aggregated Equal Height cumulative binary	2	0.724	0.837	0.844	0.879
	4	0.753	0.898	0.845	0.81
	8	0.705	0.902	0.907	0.896
	16	0.707	0.902	0.907	0.812

Table 1 Overview of results.

are virtually identical, which should come as no surprise, as this database contains two powerful attributes in the target table. The multi-relational data is mostly ignored.

In every case, the use of discretised attributes is better than not using the numeric information altogether, although in a few cases the advantage was minimal. Consider for example the performance of EqualHeight with few bins on the Financial database.

	EqualHeight	EqualWeight	AggregatedEqualHeight
Mutagenesis B2	62.5%	50.0%	37.5%
Mutagenesis B3	75.0%	87.5%	75.0%
Financial	0%	12.5%	87.5%
Musk	0%	25%	75.0%

Table 2 Relative frequency of wins or ties of each algorithm on each database.

Discretisation vs. Dynamic Handling So can the discretisation procedures compete with the dynamic approach to numeric data, or is it always best to use the latter? In the Table 3, we compare the performance of the collection of discretisation procedures to dynamic handling of numbers. Each row shows in how many of the $3\times 4\times 2=24$ runs discretisation outperforms the dynamic approach. In the majority of cases, the dynamic approach outperforms the discretisation procedures, as was expected. However, for every database, there are a number of choices of algorithm, representation and number of bins, for which discretisation can compete, or even give slightly better results. For example in Mutagenesis B3, the score of discretisation approaches using cumulative binary attributes with sufficient numbers of bins (0.898 and higher) is clearly better than the score of the dynamic algorithm (0.848).

If the set of cut points considered by the dynamic approach in theory is a superset of that considered by any discretisation procedure, how can we explain the moderate performance of the dynamic algorithm in such cases? The main reason is that the dynamic algorithm is more greedy than the discretisation procedures, because of the way numeric attributes are treated. Of the many refinements made possible by the numeric attribute, only the optimal pattern is kept for future refinements. Therefore, good rules involving two or more numeric conditions may be overlooked. On the other hand, the nominal attributes resulting from discretisation produce a candidate for each occurring value, rather than only the optimal one. Because beam search allows several candidates to be considered, it may occur that sub-optimal initial choices may lead to optimal results in more complex rules.

	discretisation	dynamic
Mutagenesis B2	5	19
Mutagenesis B3	9	15
Financial	6	18
Musk	1	23

Table 3 Performance of the discretisation procedures compared to the dynamic approach.

Choice of representation The comparison between the two proposed representations is clear-cut: the cumulative binary representation generally gives the best results (see Table 4). The few cases where the nominal representation was (slightly) superior can be largely attributed to lower efficiency caused by the larger hypothesis space of the cumulative binary approach. Every attribute gives rise to $2n$-2 refinements for n bins, compared to n refinements for the nominal case. The ties are largely caused by the equivalence of the two representations if only 2 bins are considered.

Although the cumulative binary representation is very desirable from an accuracy point of view, in terms of computing resources and disk space, the cumulative binary approach can become quite impractical, especially with many bins. Particularly in the Musk database, which contains 166 numeric attributes, several limits of the database technology used were encountered.

	nominal	cumulative binary	ties
Mutagenesis B2	3	5	4
Mutagenesis B3	0	9	3
Financial	2	6	4
Musk	5	4	7

Table 4 Performance of the discretisation procedures for the two representations.

Effect of Number of Bins As has become clear, the number of bins is an important parameter of the discretisation procedures considered. Can we say something about the optimal value for this parameter? It turns out that the answer to this question depends on the choice of representation. Let us consider the cumulative binary representation. The performance roughly increases as more cut points are added (see Table 5 and Figure 2). This is because extra cut points just add extra opportunities for refinement and thus extra precision. The only exception to this rule is when severe time constraints are present. Because of the larger search space, there may be no time to reach the optimal result. For the nominal representation, there appears to be an optimal number of cut points that depends on specifics of the database in question. Having fewer cut points has a negative effect on the precision, whereas too many cut points results in rules of low support, because each nominal value only represents a small interval. For the Mutagenesis and Musk database, the optimal value is relatively low: between 2 and 4. The optimal value for Financial is less clear.

6 Conclusion

In general, we can say that the dynamic approach to dealing with numbers outperforms discretisation. This should come as no surprise, as the dynamic approach is more precise in choosing the optimal numeric cut points. It is surprising however to observe that in some cases, it is possible to choose parameters and set up the discretisation process such that it is superior. Unfortunately, it is not immediately clear when faced with a new database what choice of algorithm, representation and coarseness produces the desired result. Essentially, it is a matter of some experimentation to come up with the right settings. Even then, there is no guarantee that the extra effort of pre-processing the data provides a substantial improvement over the dynamic approach. Of course, when working with a purely symbolic MRDM system, discretisation is mandatory.

For discretisation, we recommend that the AggregatedEqualHeight procedure be tried first, as it has proven to give good results. It is worth the effort to consider EqualHeight as an alternative. The added value of the EqualWeight procedure over EqualHeight is negligible, and can therefore be ignored.

Our experimentation shows that in general, the simple nominal representation commonly used in MRDM projects is sub-optimal. Moreover, this representation is rather sensitive to the selected number of bins. In most cases the cumulative binary representation is preferable. This representation should be applied with as many bins

as is realistic, given space and time limitations. Only when time restrictions can be expected to have a detrimental effect on the search depth, should lower numbers be considered.

	nominal				cumulative binary			
	2	4	8	16	2	4	8	16
Mutagenesis B2	0.722	0.756	0.731	0.707	0.722	0.761	0.739	0.726
Mutagenesis B3	0.837	0.764	0.731	0.707	0.837	0.898	0.902	0.907
Financial	0.705	0.653	0.7	0.719	0.705	0.711	0.746	0.759
Musk	0.831	0.852	0.828	0.787	0.831	0.829	0.831	0.841

Table 5 Average performance of the discretisation procedures for different numbers of bins.

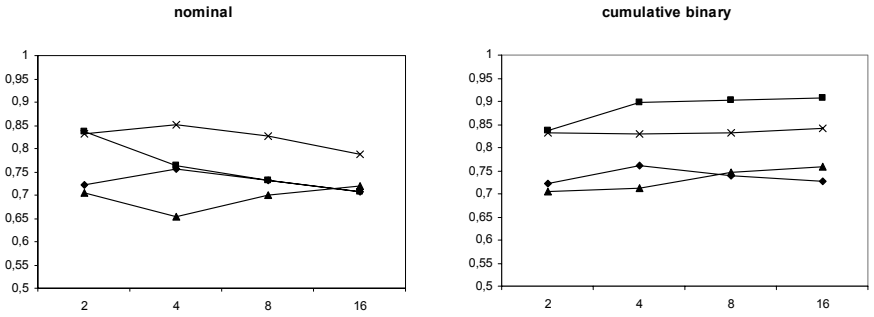

Figure 2 Effect of number of bins on performance.

References

1. Dietterich, T., Lathrop, R., Lozano-Pérez, T. *Solving the multiple-instance problem with axis-parallel rectangles*, Artificial Intelligence, 89(1-2):31-71, 1997
2. Knobbe, A.J., *Multi-Relational Data Mining*, Ph.D. dissertation, 2004, http://www.kiminkii.com/thesis.pdf
3. *Safarii, the Multi-Relational Data Mining engine*, Kiminkii, 2005, http://www.kiminkii.com/safarii.html
4. Srinivasan, A., Muggleton, S.H., Sternberg, M.J.E., King, R.D., *Theories for mutagenicity: A study in first-order and feature-based induction*, Artificial Intelligence, 85(1,2), 1996
5. Van Laer, W., De Raedt, L., Džeroski, S., *On multi-class problems and discretization in inductive logic programming*, In Proceedings of the Tenth International Symposium on Methodologies for Intelligent Systems, LNAI 1325, Springer-Verlag, 1997
6. Workshop notes on Discovery Challenge PKDD '99, 1999