Propositionalisation of continuous attributes beyond simple aggregation

Soufiane El Jelali, Agnès Braud, and Nicolas Lachiche

University of Strasbourg, LSIIT
Pôle API, Bd Brant, 67400 Illkirch, France
{eljelali,agnes.braud,nicolas.lachiche}@unistra.fr

Abstract. Existing propositionalisation approaches mainly deal with categorical attributes. Few approaches deal with continuous attributes. A first solution is then to discretise numeric attributes to transform them into categorical ones. Alternative approaches dealing with numeric attributes consist in aggregating them with simple functions such as average, minimum, maximum, etc. We propose an approach dual to discretisation that reverses the processing of objects and thresholds, and whose discretisation corresponds to quantiles. Our approach is evaluated thoroughly on artificial data to characterize its behaviour with respect to two attribute-value learners, and on real datasets.

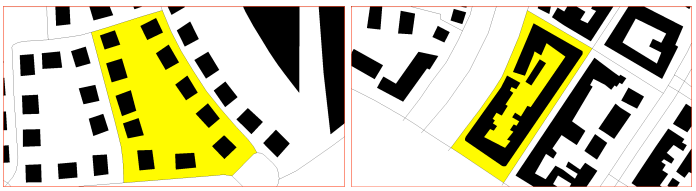
Keywords: Propositionalisation, Continuous attributes, Aggregation

1 Introduction

Relational Data Mining [4] considers data stored in at least two tables linked by a one-to-many relationship, as for example in the case of customers and their purchases, or molecules and their atoms. A way of mining these data consists in transforming them into a single attribute-value table. This transformation is called propositionalisation [11]. This paper focuses on propositionalisation of relational data involving continuous attributes.

A geographical problem motivated this work. This problem consists in predicting the class of urban blocks (see Fig. 1). The experts have defined 7 classes: Continuous urban fabric (city center), Discontinuous urban fabric with individual houses, Discontinuous urban fabric with housing blocks (blocks of flats), Mixed urban fabric (including individual housing and housing blocks), Mixed areas, High density of specialised areas (including industrial, commercial, hospital or scholar buildings), and Low density of specialised areas (containing few or no building). A urban block is characterised only by the geometrical properties of its polygon: area, elongation and convexity. The buildings contained in the urban block are represented as polygons characterised by the same geometrical properties. Density is an additional property of the urban block.

Example 1 *Below is a sample of the data, that we will use as a running example in the article. It contains the two urban blocks of Fig. 1:*



Individual houses Housing blocks

Fig. 1. Geographical example: prediction of the class of a urban block

idblock	*density*	*area*	*elong.*	*convex.*	*class*
9601	*0,194*	*6812*	*0,772*	*0,921*	*indiv*
9602	*0,455*	*11119*	*0,470*	*0,916*	*hous*
⋮	⋮	⋮	⋮	⋮	⋮

and their buildings:

idbuild	*area*	*elong.*	*convex.*	*idblock*
4519	*122*	*0,765*	*1,00*	*9601*
4521	*122*	*0,752*	*1,00*	*9601*
4528	*119*	*0,948*	*1,00*	*9601*
4537	*112*	*0,918*	*1,00*	*9601*
4545	*121*	*0,829*	*1,00*	*9601*
4556	*136*	*0,739*	*0,999*	*9601*
4564	*115*	*0,755*	*1,00*	*9601*
4568	*134*	*0,829*	*0,999*	*9601*

idbuild	*area*	*elong.*	*convex.*	*idblock*
4579	*125*	*0,745*	*1,00*	*9601*
4583	*98*	*0,935*	*0,999*	*9601*
4589	*113*	*0,909*	*1,000*	*9601*
4231	*1669*	*0,955*	*0,680*	*9602*
4866	*2239*	*0,772*	*0,595*	*9602*
4867	*229*	*0,818*	*0,999*	*9602*
4868	*164*	*0,795*	*0,936*	*9602*
4869	*559*	*0,451*	*0,894*	*9602*
4870	*205*	*0,271*	*0,999*	*9602*

Discussions with the experts showed that the class depends on conditions about the geometry of buildings and the number, or proportion, of buildings satisfying those conditions. For example, the class "individual houses" mainly depends on the presence of small buildings. So the learning task consists in determining relevant attributes and their thresholds, as well as the number of those buildings. Existing approaches are not optimised to search at the same time a threshold on an attribute and a threshold on the number of objects satisfying this condition, as we will see in the next section. We propose a dedicated approach, called cardinalisation. It is introduced in Section 3. Section 4 presents an experimental validation on artificial data and on real data.

2 Existing approaches

Relational data mining approaches mainly come from Inductive Logic Programming (ILP) [12]. These approaches deal with categorical attributes rather than with numeric ones, and generally rely on a discretisation to transform numeric attributes into categorical ones. Other approaches use relational databases and propose to employ aggregate functions available in SQL. Finally, some approaches

in full-fledged relational data mining systems integrate aggregate operators to the hypothesis construction. We detail those three approaches in this section.

In what follows, the main table is the table that contains the target column. Its rows are the individuals. In our running example, the individuals are the urban blocks. The secondary table describes objects linked to, or components of, the individual. E denotes the objects linked to an individual i, for example the buildings of a urban block. The cardinality of this set is denoted by $|E|$. In our example, it is the number of buildings. For reason of clarity, the attributes built by propositionalisation are called *features* to distinguish them from the columns of the original tables.

Discretisation The search space of relational data mining is larger than the one, already exponential, of the attribute-value learning. Therefore, most works in ILP focus on an efficient exploration of this search space rather than handling numeric attributes. In particular, many propositionalisation algorithms deal with numeric attributes as with categorical ones, including recent algorithms as for example RSD [18], Hifi [9] and RelF [10].

Thus, in order to deal with numeric attributes, a first approach consists in discretising them. This discretisation is made *a priori*, regardless of the model construction. The advantage is that the whole set of objects is considered, *i.e.* components of all individuals are taken together. For example, the areas of the buildings of all urban blocks can be discretised at the same time into small buildings (individual houses), medium (housing blocks) and big (industrial, commercial). Appropriate thresholds might be found more easily than with smaller samples, in particular than with a single urban block. The drawback is that the choice of the thresholds is not optimised with respect to a subset of the examples, as in the case of decision trees [15].

Example 2 *We discretise the areas of buildings into 20 intervals of equal frequency. So each interval contains one twentieth of the buildings. This produces 19 thresholds and thus 38 features of the form "number of buildings whose area is lower (respectively greater) than such threshold":*

idblock	...	|*area*<*66*|	|*area*>*66*|	|*area*<*101*|	|*area*>*101*|	|*area*<*110*|	...
9601	...	*0*	*11*	*0*	*11*	*1*	...
9602	...	*0*	*6*	*0*	*6*	*0*	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Features are generated for the elongation and convexity in the same way.

Once the continuous attribute is turned into categorical ones (two for each threshold), propositionalisation generates features. We generate the simpler features built by a state-of-the-art system such as Relaggs [7, 8]: the number of buildings for each elementary condition. An elementary condition is a single constraint on one attribute. Let us notice that other propositionalisation systems dealing with categorical attributes, such as Hifi [9], would be able to generate a conjunction of attribute-value constraints, using an existential quantifier, and

thus to build more complex features than the propositionalisation systems dedicated to numeric attributes do. Nevertheless the number of features increases exponentially, and it is difficult in practice to generate exhaustively all the features.

Aggregation The systems Polka [6] and Relaggs [7, 8] explicitly deal with numeric attributes. They apply the usual SQL aggregate functions to the values v_A of each numeric attribute A for the set E of tuples linked to the current individual,

$$\text{aggregation}(A, f, E) = f(\{v_A(t), t \in E\})$$

where f is an aggregate function. The available functions are minimum, maximum, average, standard deviation, sum, count, first quartile, median, third quartile, and the difference between the maximum and the minimum.

The difference between Polka and Relaggs can just be seen for problems with a secondary table linked to a third table by a one-to-many relationship. We call nested tables this sequence of tables linked by one-to-many relationships. In this case, Polka applies successively the aggregate operators in a depth-first order. Relaggs makes joins between the secondary table and the tables depending on it, and then apply the aggregate operators to the columns obtained.

Example 3 *We apply Relaggs that computes the sum, the average, the minimum, the maximum and the standard deviation to each numeric attribute:*

idblock	...	*sum(area)*	*avg(area)*	*min(area)*	*max(area)*	*stddev(area)*	...
9601	...	*1317*	*119*	*98*	*136*	*10,5*	...
9602	...	*5065*	*844*	*164*	*2239*	*889*	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

The interest of those approaches is that they summarize the data and they are not subject to combinatorial explosion, even when applied to nested tables.

Complex Aggregates Full-fledged relational data mining systems do not dissociate the propositionalisation step from the construction of the model. Most of the full-fledged relational data mining systems, as for example Progol [13], do not explicitly deal with numeric attributes. Nevertheless, Tilde [2] has been modified to construct complex aggregates [16]. A complex aggregate is the application of an aggregate function to a conjunction of elementary conditions, for example the number of buildings whose areas are lower than 180 and their elongations are lower than 0.8. A simple aggregate is the application of an aggregate function to the objects selected by a single elementary condition, such as the number of buildings with an area lower than 180. The features considered in this paper are simple aggregates. Their generation with Tilde is not easy because the language bias has to be defined by hand, in particular the values to test for the thresholds on the number of buildings must be enumerated.

In theory, Tilde is able to generate complex aggregates. In practise, the combinatorial exceeds in a few minutes the memory capacity of the program. The expressible features are thus restricted to an aggregate and an inequality, for example "at least 2 buildings with an area greater than 180". The functions minimum, maximum, average, sum, mode, count and count-distinct are implemented, but standard deviation, first quartile, median and third quartile are missing. Tilde can also build features implying a proportion, such as "at least 30% of small buildings", but it is not optimised to do so. So, we will use a dedicated implementation in a propositionalisation approach rather than Tilde.

The contribution of this paper consists in proposing new ways to relate a continuous attribute of a secondary table to the main table. It is easier to present and evaluate in a propositionalisation approach, but it could be used in full-fledged relational data mining systems as well.

3 Cardinalisation

We propose a new approach for propositionalising in presence of numeric attributes, without discretising in advance, and without restricting ourselves to a fixed number of features limited by the number of aggregate functions. We call this approach cardinalisation to highlight the fact that it sets the cardinality between the main table and the secondary one, on the contrary to the existing approaches based on a discretisation that sets a threshold on the domain of the numeric attribute.

This duality is the subject of the first part of this section. In the second part, we will see that discretising the cardinalities correspond to quantiles. So it is a generalisation of aggregate functions, allowing to generate a greater number of features.

An approach dual to discretisation Aggregate functions provide a simple solution for propositionalising numeric attributes. Nevertheless, the number of functions proposed is small, and does not allow choosing both a threshold on the numeric attributes and a minimum number of the corresponding objects.

An alternative approach consists in discretising the numeric attribute so as to transform it into a categorical attribute. Thus thresholds are introduced on the numeric attribute and they will not be changed during the model construction. Then, the propositionalisation is made by applying an aggregate function to count the corresponding number of objects per individual. Given a numeric attribute A of the secondary table, a feature is built for each threshold s coming from the discretisation. The value of this feature for an individual linked to a set of tuples E of the secondary table is the number of tuples t whose value $v_A(t)$ for the attribute A is lower than the threshold s:

$$\text{aggregationAfterDiscretisation}(A, s, E) = |t \in E \text{ such that } v_A(t) \leq s|$$

This produces a numeric feature that the attribute-value learner can use. The value of this feature is a number of objects. During its contruction process,

a classifier such as a decision tree learner will only have the possibility to choose the threshold on a numeric attribute by choosing the most discriminant feature, since the feature is defined by the threshold.

Cardinalisation tries to reverse the roles of the thresholds set on the numeric attribute and on the cardinality. It sets a threshold on the cardinality, and then lets the attribute-value learner choose the threshold on the numeric attribute. Actually, given a numeric attribute A of the secondary table, a feature is built for each possible threshold k of the cardinality, between 1 and the maximal number of objects per individual. The value of this feature is the minimal value of the threshold on the numeric attribute such that at least k tuples t have a value $v_A(t)$ for the numeric attribute A less than this threshold:

$$\begin{aligned} &\text{cardinalisation}(A, k, E) \\ &= min(s \in Domain(A) \text{ such that } |t \in E \text{ such that } v_A(t) \leq s| \geq k) \\ &= min(s \in Domain(A) \text{ such that aggregationAfterDiscretisation}(A, s, E) \geq k) \end{aligned}$$

When a urban block has less than k objects, an infinite threshold is assigned to the feature.

Example 4 *For each attribute, new features compute the minimal thresholds to select k buildings, k varying between 1 and the maximal number of buildings per urban block. For example, min_3_area indicates the minimal area such that the urban block contains at least 3 buildings with an area lower than or equal to that threshold.*

idblock	...	*min_1_area*	*min_2_area*	*min_3_area*	*min_4_area*	*min_5_area*	...
9601	...	*98*	*112*	*113*	*115*	*119*	...
9602	...	*164*	*205*	*229*	*559*	*1669*	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

We note that the thresholds depend on each urban block, in contrary to the discretisation that chooses each threshold independently from the urban blocks.

The interest of cardinalisation is to not set the threshold on the numeric attribute during propositionalisation, thus allowing the attribute-value program to choose the relevant threshold on the numeric attribute, in particular by taking into account the context as in the branches of a decision tree for example.

Cardinalisation is an approach dual to discretisation followed by an aggregation: a discretisation theoretically equivalent to cardinalisation consists in introducing a threshold between each value of the numeric attribute. The drawback is that the number of thresholds produced by this discretisation (in intervals containing a single object each) is the maximal number of values taken by the attribute on all objects from all individuals together. In the worst case, the number of distinct values is of the order of the total number of objects of the secondary table, for each of its numeric attributes. Moreover the aggregation produces two features per threshold: the number of objects below and above each threshold. On the other hand, the cardinalisation produces a number of

features that is exactly the maximal number of objects per individual. In general, the maximal number of objects per individual is smaller than the double of the number of distinct values of a continuous attribute. Therefore, the discretisation that is theoretically equivalent to the cardinalisation, *i.e.* an interval per value, is often not tractable: the number of columns exceeds the maximal number of columns of the RDBMS (1600 columns in PostgreSQL and 4096 in MySQL, respectively). So, the cardinalisation is a more tractable way to let the attribute-value learner see all distinct numeric values.

If the number of features is too large in the case of a discretisation followed by an aggregation, it is possible to discretise the numeric attribute in larger intervals. Nevertheless, the drawback remains that the thresholds are set with respect to all the objects, all individuals together, for example the areas of all the buildings from all the urban blocks together. Thus, this setting is not as expressive as the cardinalisation.

If the cardinalisation generates too many features as well, the cardinality can be discretised.

Quantiles Instead of building a feature per possible value of the cardinality, it is possible to discretise the cardinality in order to obtain a fixed number of features, in a similar way to the discretisation of the numeric attribute itself followed by the aggregation. Instead of setting an absolute number for the threshold on the cardinality, the discretised cardinalisation uses a relative number. Actually, this corresponds to quantiles. If four intervals are chosen, the features correspond to the first quartile, the median, and the third quartile. The k^{th} n-quantile is the threshold such that a proportion of at least k/n objects are selected:

$$\begin{aligned}\text{quantile}(A, k, n, E) = min(s \in Domain(A) \text{ such that} \\ \text{agregationAfterDiscretisation}(A, s, E) \geq k \times |E|/n)\end{aligned}$$

Example 5 *For each attribute of the buildings table, the features compute the minimal thresholds to select at least k twentieth of the buildings. The number of intervals, 20 in this case, is a parameter of the program.*

idblock	...	*min_1_20_area*	*min_2_20_area*	*min_3_20_area*	*min_4_20_area*	...
9601	...	*98*	*98*	*112*	*112*	...
9602	...	*164*	*164*	*164*	*164*	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮

While quartiles are implemented in some propositionalisation approaches, none of them generalise quartiles to quantiles. Moreover, no full-fledged relational system but Tilde accepts a language bias allowing to construct aggregates corresponding to quantiles, and they are not really intended to. We propose to use quantiles in propositionalisation, so as to be able to configure the number of generated features, and thus to obtain a greater expressivity than in the existing propositionalisation approaches.

4 Experimental evaluation

Four propositionalisation approaches, aggregationAfterDiscretisation[1], simple numeric aggregates using Relaggs, cardinalisation, and quantiles, are evaluated with respect to two attribute-value learners (classifiers).

The attribute value learners are the decision tree learner J48, and the support vector machine with a gaussian kernel using weka [17]. The width of the gaussian, γ, and the complexity, C, are tuned by a grid search on the training data: We select the combination of $\gamma \in \{10^{-5}, 10^{-3}, 10^{-1}\}$ and $C \in \{10^{-4}, 10^{-2}, 1, 10^{2}, 10^{4}\}$ that gets the best accuracy on the training set.

The experimental evaluation consists of two parts. The first part concerns artificial data where the class is chosen according to two known target functions. The second test consists in comparing the approaches on real data.

The 20-quantiles were used and the agregationAfterDiscretisation used 20 equal-frequency intervals, in order to use the same number of intervals.

Artificial data Artificial data have a structure similar to the geographical data that motivated this work. The main table describes some urban blocks with a Boolean class and one numeric attribute, the area. The building table also has a single numeric attribute, the area, and each building belongs to one and only one urban block. The values of the numeric attributes have been generated randomly between 0 and 100 with a uniform distribution. The number of buildings per urban block has been randomly chosen between 0 and 40 according to a uniform distribution.

Let us point out that the cardinalisation and the agregationAfterDiscretisation produce the same number of features related to the areas of the buildings in this setting: 40.

The number of training instances was increased from 100 to 1000 by step of 100, according to the same uniform distribution. An independent test set of 2000 urban blocks was generated. 10 training sets were generated for each number of training instances. The figures report the average of the accuracies on the test set obtained by models built from the 10 training sets, with error-bars corresponding to the standard deviation. Two target functions were tested:

Absolute number: urban blocks whose areas are greater than 50 and have less than 14 buildings with areas lower than 73, or whose areas are lower than 50 and have more than 4 buildings with areas lower than 19.

Relative number: at least 40% of buildings have an area lower than 40.

A first round of experiments concerns the absolute number target function defined above. Figure 2 reports the accuracies of a decision tree learner and of a support vector machine with a gaussian kernel, with respect to the four propositionalisation approaches. On the left hand side of Figure 2, we observe that the cardinalisation enables the decision tree to learn the target model more easily than the agregationAfterDiscretisation. Both of them get a higher accuracy

[1] The aggregationAfterDiscretisation is denoted Discretisation in the figures.

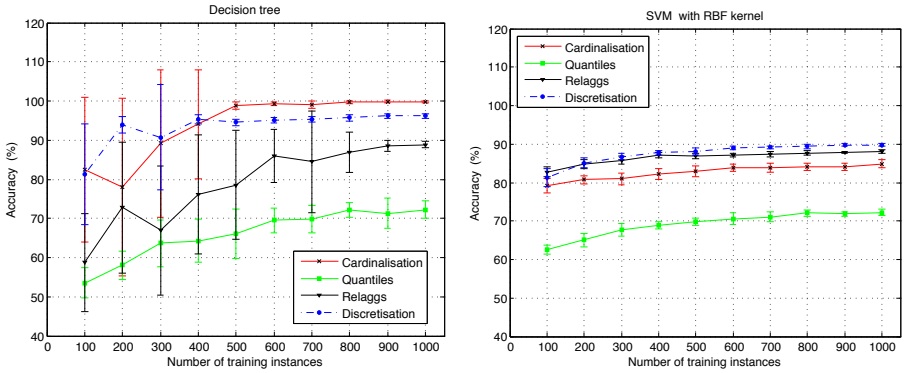


Fig. 2. Accuracy of two attribute-value learners on artificial data for a target involving absolute numbers

than the simple aggregates of Relaggs. The quantiles are not designed to learn such a target function involving absolute numbers of objects. Let us notice that the decision tree learner is sensitive to the training set: small training sets (below 500 training instances) lead to a high standard deviation of the accuracy. The standard deviation is considerably reduced once there are enough training data to learn a good decision tree. On the right hand side, we note that the support vector machine is not able to learn models as accurate as the decision tree, for this target function.

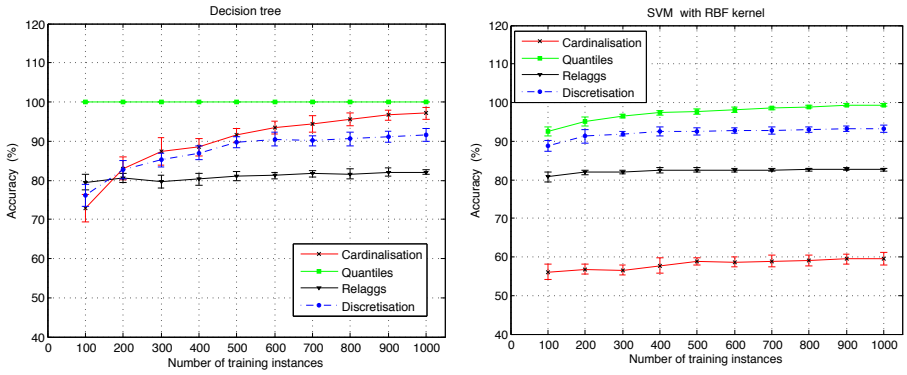


Fig. 3. Accuracy of two attribute-value learners on artificial data for a target involving a relative number

A second round of experiments concerns the relative number target function defined above. Figure 3 reports the accuracies of the two attribute-value learners

with respect to the four propositionalisation approaches. We observe that the quantiles enable both the decision tree and the support vector machine with a gaussian kernel to learn a target function involving a proportion of objects. Moreover, when the number of training instances increases, the decision tree using the cardinalisation successfully models this target function, but it does not work with the support vector machine.

Additional experiments, not detailled in this article, show that increasing the number of intervals for discretisation allows it to increase its accuracy with respect to the decision tree learner, but it leads to overfitting for the support vector machine with a gaussian kernel. Moreover, the propositionalisation and the learning time grow with respect to the number of intervals.

These experiments show that the different combinations of propositionalisation and attribute-value learner get different accuracies, depending on the target problem. Thus, the different propositionalisations provide different inductive biases that can be beneficial on some datasets, but not necessarily on all datasets.

Real data Let us evaluate how the different propositionalisation approaches perform on real datasets. Few ILP datasets mainly involve continuous attributes. We consider six benchmarks from three application domains. Musk1 and Musk2 deal with properties of molecules [3]. Each molecule is described by a set of conformations, each conformation is described by 166 continuous attributes. Tiger, Elephant and Fox are content-based image retrieval problems [1]. Each image is described by a set of segments, each segment is described by 230 continuous attributes about color, texture and shape. Continuous attributes of images with low standard deviation have been filtered out. 105 attributes were kept. The geographical data correspond to suburbs of the city of Strasbourg. The aim is to predict the classes of urban blocks from their geometrical properties (density, area, elongation, convexity) and the geometrical properties (area, elongation, convexity) of their buildings. The five datasets on musk and images are available on `http://www.uco.es/grupos/kdis/mil/dataset.html`. Those datasets have no dedicated test set. The reported accuracy is the average of 10 runs of 10-fold cross-validations. For the geographical dataset, all urban areas corresponding to year 2002 were used for the training set, and all urban areas from other years for the test set. The number of quantiles and of intervals for the agregationAfterDiscretisation were restricted to 4 for the musk and the images datasets in order to keep the number of features below 1600. Cardinalisation produced more than 1600 features on those datasets. So, cardinalisation was only run on the geographical data.

The accuracies of the two standard attribute-value learners with respect to the four propositionalisation approaches are reported in Figure 4 and compared to the accuracy, denoted Reference, of the original dedicated multi-instance learning algorithms for the musk [3] and image [1] datasets.

First of all, we observe that the accuracies vary with respect to the propositionalisation and to the attribute-value learner, in a similar way to the results on the artificial data. No single propositionalisation and attribute-value learner

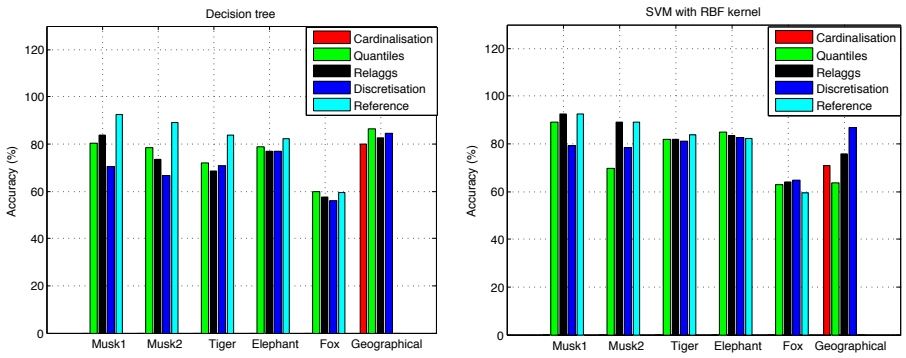


Fig. 4. Accuracy of two attribute-value learners on real data

wins on all datasets. On the opposite, each propositionalisation leads to the maximal accuracy on some dataset: quantiles on all datasets but Musk1 for J48, and on Tiger and Elephant for the SVM, simple aggregates of Relaggs on Musk1 and Musk2 for the SVM, and the agregationAfterDiscretisation on Fox and Geographical for the SVM. Indeed, the different propositionalisation approaches provide different biases for different attribute-value learners on different domains.

Moreover, the best accuracies of two standard attribute-value learners with those propositionalisation approaches reach, and in some cases outperform, the accuracies of the original dedicated learning algorithm.

Finally, we notice that the quantiles enable the attribute-value decision tree learner to produce an accurate model and solve the real seven classes geographical problem that motivated that work. The experts preferred the decision tree learner to the support vector machine because the models are easier to interpret [14].

5 Conclusion

This paper proposed two new propositionalisation techniques dealing with continuous attributes. Experiments on artificial and on real data showed that those propositionalisation techniques offer a different expressivity than existing propositionalisation approaches. This new expressivity can be beneficial on some target problems, according to the used attribute-value learner. Since the accuracy depends on the propositionalisation and on the applied attribute-value learner, we plan in future work to select the best combination on the training set.

The image and musk datasets contain too many columns to generate more than 4 quantiles in a propositionalisation approach. Indeed, propositionalisation is not tractable on all datasets. Therefore another perspective concerns the generation of complex aggregates on-the-fly, *i.e.* in a full-fledged relational data mining system, such as BET [5].

Acknowledgments. The geographical data were prepared by the LIVE laboratory of the University of Strasbourg and the COGIT laboratory of the French National Geographic Institute.

References

1. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Becker, S., Thrun, S., Obermayer, K. (eds.) NIPS. pp. 561–568. MIT Press (2002)
2. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artif. Intell. 101(1-2), 285–297 (1998)
3. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artif. Intell. 89(1-2), 31–71 (1997)
4. Džeroski, S., Lavrač, N. (eds.): Relational data mining. Springer (2001)
5. Kalgi, S., Gosar, C., Gawde, P., Ramakrishnan, G., Gada, K., Iyer, C., Kiran, T.V.S., Srinivasan, A.: BET : An inductive logic programming workbench. In: Frasconi, P., Lisi, F.A. (eds.) ILP. Lecture Notes in Computer Science, vol. 6489, pp. 130–137. Springer (2010)
6. Knobbe, A.J., de Haas, M., Siebes, A.: Propositionalisation and aggregates. In: De Raedt, L., Siebes, A. (eds.) PKDD. LNCS, vol. 2168, pp. 277–288. Springer (2001)
7. Krogel, M.A., Wrobel, S.: Transformation-based learning using multirelational aggregation. In: Rouveirol, C., Sebag, M. (eds.) ILP. LNCS, vol. 2157, pp. 142–155. Springer (2001)
8. Krogel, M.A., Wrobel, S.: Facets of aggregation approaches to propositionalization. In: Work-in-progress session of the 13th Int. Conf. on ILP (2003)
9. Kuželka, O., Železný, F.: Hifi: Tractable propositionalization through hierarchical feature construction. In: Železný, F., Lavrač, N. (eds.) Late Breaking Papers, the 18th Int. Conf. on ILP (2008)
10. Kuzelka, O., Zelezný, F.: Block-wise construction of acyclic relational features with monotone irreducibility and relevancy properties. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) ICML. ACM Int. Conf. Proceeding Series, vol. 382, p. 72. ACM (2009)
11. Lachiche, N.: Encyclopedia of Machine Learning, chap. Propositionalization. C. Sammut and G. I. Webb (eds.), Springer (2010)
12. Lavrač, N., Džeroski, S.: Inductive Logic Programming: Techniques and Applications. Ellis Horwood (1994)
13. Muggleton, S.: Inverse entailment and progol. New Generation Computing 13(3-4), 245–286 (1995)
14. Puissant, A., Skupinski, G., Lachiche, N., Braud, A., Perret, J.: Classification et évolution des tissus urbains à partir de données vectorielles. Revue internationale de géomatique 21(4), 513–532 (Dec 2011)
15. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
16. Vens, C., Ramon, J., Blockeel, H.: Refining aggregate conditions in relational learning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD. Lecture Notes in Computer Science, vol. 4213, pp. 383–394. Springer (2006)
17. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, second edn. (2005)
18. Zelezný, F., Lavrac, N.: Propositionalization-based relational subgroup discovery with RSD. Machine Learning 62(1-2), 33–63 (2006)