A latent variable model for multivariate discretization

Stefano Monti[†]
[†]Intelligent Systems Program
University of Pittsburgh
901M CL, Pittsburgh, PA – 15260
smonti@isp.pitt.edu

Gregory F. Cooper[†‡]
[‡]Center for Biomedical Informatics
University of Pittsburgh
8084 Forbes Tower, Pittsburgh, PA – 15261
gfc@cbmi.upmc.edu

Abstract

We describe a new method for multivariate discretization based on the use of a latent variable model. The method is proposed as a tool to extend the scope of applicability of machine learning algorithms that handle discrete variables only.

1 Introduction

The discretization of continuous variables arises as an issue in machine learning because of the availability of machine learning algorithms that can handle discrete variables only. Furthermore, even when the learning algorithm at hand can directly model continuous variables, it may be possible to improve its predictive performance, as well as its induction time, by using discretized variables [6, 10].

While many of the available discretization algorithms search for the best discretization of each continuous variable individually, an approach that we refer to as *univariate* discretization, ideally the discretization of a continuous variable should be carried out so as to minimize the loss of information that the given variable may contain about other variables in the domain. True to this principle, current state-of-the-art discretization methods for classification search for the best discretization of a given feature variable by measuring its interaction with the class variable, in an attempt to maximize the discriminatory power of the former with respect to the latter (*class-based* discretization [7, 10, 11]). In general, when no class variable is provided[1], the discretization process should take into consideration the possible interaction of the variable being discretized with all the other variables in the domain of interest, an approach that we refer to as *multivariate* discretization.

[1]This is the case, for example, in exploratory data analysis, and Bayesian network learning.

Very few multivariate discretization methods have been proposed. The methods described in [8, 9] were both developed with the task of learning Bayesian network (BN) structures in mind. As such, they select a discretization conditional on the BN structure currently being evaluated, thereby calling for a re-evaluation of the selected discretization when the BN structure changes.

The discretization method described in this paper on the other hand, does not rely on a BN structure, and can be carried out as a preprocessing step to the application of any machine learning algorithm. The method is based on clustering: it looks for sub-populations of data points which are closest according to their probability conditioned on a cluster variable properly defined. The data points included in each cluster then determine the partition of the value range of the continuous variables. More specifically, following a Bayesian approach, the joint probability distribution over the observed variables is modeled by means of a finite mixture (FM) model as described, for example, in [2]. The induced FM model defines a latent cluster variable, as well as its conditional distribution given the observed variables. The cluster variable, and its conditional distribution, are then used to drive the discretization of the observed continuous variables.[2]

In the remainder of this paper, we illustrate in more detail our discretization approach. We do so by first briefly describing the fundamentals of class-based discretization (for classification), and by then showing that our multivariate discretization method can be interpreted as a generalization of the former to the case when the class variable is latent.

[2]The use of clustering techniques for multivariate discretization is also explored in [3]. In that work however, clusters are induced based only on the continuous variables, thus discounting the possible contribution to cluster formation of the discrete variables present in the domain. Furthermore, the ad-hoc distance metric there proposed to drive cluster formation is significantly different from the probabilistic approach we adopt.

2 Background

In general, we denote random variables with upper case letters, such as X, Y, and their instantiation or realization with the corresponding lower case letters, x, y, or $x^{(l)}$, $y^{(l)}$, where we use the latter notation when we need to distinguish between different instantiations. Similarly, we denote random vectors with bold upper case letters, such as $\boldsymbol{V}, \boldsymbol{W}$, and their instantiation or realization with the corresponding bold lower case letters, $\boldsymbol{v}$, $\boldsymbol{w}$.

Given a domain of interest, we denote with $\mathcal{X} = \{X_1, \ldots, X_n\}$ the complete set of variables in that domain, and with $\boldsymbol{x}$ or $\boldsymbol{x}^{(l)}$ the full instantiations of the variables in $\mathcal{X}$. In the context of classification, we consider the augmented set of domain variables $\{C\} \cup \mathcal{X}$, with C the class variable of interest, and $\mathcal{X}$ the set of feature variables.

2.1 Discretization

Informally, the discretization of a continuous variable specifies the set of cutpoints in the continuous range of the continuous variable that delimit the intervals to be mapped into the values of the discretized variable.

More formally, the discretization for a variable X_i, defined over the real interval $[a, b]$, with $a < b$, is uniquely defined by the ordered set of cutpoints $T = \{t_0, t_1, \ldots, t_{r_i}, t_{r_{i+1}}\}$ that partition the value range of X_i into the set the of subintervals $[t_0, t_1]$, $(t_1, t_2], \ldots, (t_{r_i}, t_{r_i+1}]$, with $a = t_0 < t_1 < \ldots < t_{r_i+1} = b$.[3] This partition defines a discrete variable taking values in the domain $\{0, 1, \ldots, r_i\}$, with the j-th value corresponding to the interval $(t_j, t_{j+1}]$.

Given a database $\mathcal{D}$ of $N_{\mathcal{D}}$ cases defined over $\mathcal{X}$, as candidate cutpoints we only consider the (at most) $N_{\mathcal{D}} - 1$ mid-points between contiguous data points in $\mathcal{D}$. Therefore, the number of values the discretized variable can take is upper-bounded by $N_{\mathcal{D}}$. Notice that even in the simpler univariate discretization problem, the search for the best discretization has combinatorial complexity in the number of datapoints, since the complete set of possible discretizations to be considered has cardinality $2^{N_{\mathcal{D}}-1}$.

2.2 Class-based discretization

The idea behind class-based discretization [7, 11] is to search for the partition of the value range of a continuous feature variable so as to minimize the uncertainty of the class variable conditioned on the discretized feature variable.

A commonly used measure of the uncertainty in a random variable is its entropy, which can be defined formally as follows. Let $\mathcal{D}$ be a database of $N_{\mathcal{D}}$ cases over $\{C\} \cup \mathcal{X}$. The *class entropy* $\mathrm{Ent}(\mathcal{D})$, measuring the uncertainty in the class variable with respect to the dataset $\mathcal{D}$, can then be defined as:

$$\mathrm{Ent}(\mathcal{D}) = -\sum_{k=1}^{K} \frac{N_{\mathcal{D}}(k)}{N_{\mathcal{D}}} \log \frac{N_{\mathcal{D}}(k)}{N_{\mathcal{D}}}\,, \tag{1}$$

where $N_{\mathcal{D}}(k)$ denotes the number of cases in $\mathcal{D}$ with $C = k$. That is, if we denote with $1_{\{\cdot\}}$ the indicator function ($1_{\{cond\}} = 1$, if *cond* holds, 0 otherwise), then $N_{\mathcal{D}}(k)$ can be expressed conveniently as follows:

$$N_{\mathcal{D}}(k) = \sum_{l=1}^{N_{\mathcal{D}}} 1_{\{c^{(l)}=k\}}\,, \tag{2}$$

a notation that will become useful in the next section.

Given a discretization T of the continuous variable X_i, the set of cutpoints in T determines the partition of the dataset $\mathcal{D}$ into the subsets $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{r_i}$, with the subset $\mathcal{D}_j$ containing all the datapoints in $\mathcal{D}$ for which variable X_i takes values within the j-th interval $(t_j, t_{j+1}]$. We can thus apply the above definition of class entropy to each of the sub-sets defined by the discretization, and measure their overall (weighted) contribution to the reduction of the uncertainty in C.

Following the notation in [7], we define the *class information entropy of the partition induced by* T, $\mathrm{Ent}(X_i, T; \mathcal{D})$ as

$$\mathrm{Ent}(X_i, T; \mathcal{D}) = \sum_{j=0}^{r_i} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \mathrm{Ent}(\mathcal{D}_j)\,, \tag{3}$$

and it can be interpreted as a measure of the uncertainty in the class variable C that is not accounted for by the discretized feature variable X_i.

Therefore, the discretization T for a feature variable X_i should be selected so as to minimize the scoring metric $\mathrm{Ent}(X_i, T; \mathcal{D})$. Notice that the number of possible discretization is exponential in the number of datapoints in the dataset. Therefore, exhaustive search is in general computationally infeasible, and heuristics search techniques need to be used instead. A commonly used technique is a form of greedy search: a binary discretization for X_i is determined by selecting the cutpoint t for which the scoring metric $\mathrm{Ent}(X_i, \{t\}; \mathcal{D})$ is minimal amongst all the candidate cut points. Once a cut point is selected, the procedure can be recursively applied to each of the subsets determined by the cut point. Fayyad and Irani [7] use the

[3] In this definition, we have assumed the variable X_i to have a bounded domain. We can easily generalize the definition to an unbounded variable by setting $t_0 = -\infty$ and $t_{r_i+1} = +\infty$.

Minimum Description Length criterion to decide when to stop the recursive partition of X_i's value range.

3 Cluster-based discretization

The discretization method we propose in this paper is best described by analogy with the class-based discretization methods for classification described in the previous section.

For the purpose of multivariate discretization, where potentially every variable is the focus of prediction, and the discretization is aimed at capturing the interaction among all the variables, we can use a latent cluster variable as a *proxy* class variable, and use this variable to drive the partition of the value range of each continuous variable.

More specifically, let us assume that we have a database $\mathcal{D}$ defined over the set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ (notice that no class variable is included). We can use an FM model as described in [2] to model the probabilistic dependencies among the variables in $\mathcal{X}$. Figure 1 shows this model for the set of variables $\mathcal{X}$. The joint probability distribution over $\mathcal{X}$ is specified through the conditional distributions $p(X_i \,|\, H)$, and the prior $p(H)$. Notice that the FM model can model both continuous and discrete variables directly. A typical parametric form for the probability density of the continuous variables is the Normal distribution, but other parametric forms can easily be adopted.

It is important to emphasize that the latent variable has been specifically introduced in an attempt to model the interaction among the observable variables. As such, it can be thought of as a class variable whose values identify distinct sub-populations in the given database. Therefore, discretization with respect to this latent variable will take into consideration the variables' interaction.

Let us assume that we have applied some learning algorithm to learn a FM model for the set of variables $\mathcal{X}$ and the database $\mathcal{D}$ defined over $\mathcal{X}$.[4] Let us also assume that in the learned FM model the latent variable H has K categories. If we now treat the latent variable H as the unobserved class variable, we can apply the discretization method briefly outlined in the previous section, to carry out the discretization.

Since we do not observe H, we need to modify the scoring metric of Equation (3). In particular, we have to replace the sufficient statistics $N_{\mathcal{D}}(k)$ of Equation (2), defined over the variable C, with the expected sufficient statistics $\mathrm{E}[N_{\mathcal{D}}(k)]$, now defined over the latent variable H:

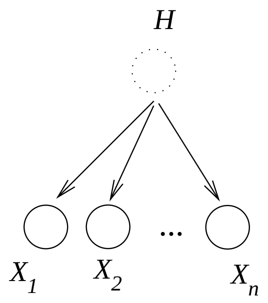

Figure 1: The finite mixture (FM) model, with the latent variable H modeling the interaction among the observed variables $\{X_i\}$, which can be both continuous and discrete.

$$\mathrm{E}[N_{\mathcal{D}}(k)] = \sum_{l=1}^{N} p(h_k \,|\, \boldsymbol{x}^{(l)}) \,, \tag{4}$$

where each of the terms $p(h_k \,|\, \boldsymbol{x}^{(l)})$ is computed as follows:

$$p(h_k \,|\, \boldsymbol{x}^{(l)}) = \frac{p(\boldsymbol{x}^{(l)} \,|\, h_k) p(h_k)}{\sum_{k'} p(\boldsymbol{x}^{(l)} \,|\, h_{k'}) p(h_{k'})} \,, \tag{5}$$

and all the probabilities are as specified by the induced FM model. As a consequence, the class entropy $\mathrm{Ent}(\mathcal{D})$ defined in Equation (1) must be replaced by the *expected* class entropy $\hat{\mathrm{Ent}}(\mathcal{D})$

$$\hat{\mathrm{Ent}}(\mathcal{D}) = -\sum_{k=1}^{K} \frac{\mathrm{E}[N_{\mathcal{D}}(k)]}{N_{\mathcal{D}}} \log \frac{\mathrm{E}[N_{\mathcal{D}}(k)]}{N_{\mathcal{D}}} \,. \tag{6}$$

Finally, Equation (3) must be modified by replacing the class entropy terms $\mathrm{Ent}(\mathcal{D}_j)$ with the corresponding expected class entropy terms $\hat{\mathrm{Ent}}(\mathcal{D}_j)$.

To summarize, to adapt the class-based discretization method described in the previous section to the problem of multivariate discretization, all that is needed is the replacement of the *hard* partition determined by the indicator function of Equation (2), with the *soft* partition determined by the posterior probability of class membership of Equation (4). In fact, since we do not observe the class variable H, we have to assign each case in $\mathcal{D}$ probabilistically, rather than deterministically.

Notice that despite the fact that we do not observe the variable H, the selection of a cut-point for a given variable X_i does not affect the selection of the cut-points for the other variables. That is, we can still perform a local search. In fact, the "goodness" of a

[4]See, for example, [2], for techniques to learn a FM model from data.

given cut-point for a specific variable X_i does not depend on how we discretize other variables, but only on how the selected cut-point influences the expected sufficient statistics $\mathrm{E}[N_{\mathcal{D}}(k)]$. These in turn are influenced by the posterior distribution $p(H \mid \mathcal{X})$. The distribution $p(H \mid \mathcal{X})$, as it is based on the *non-discretized* data, is independent of the discretization chosen.

Each of the terms $p(\boldsymbol{x}^{(l)} \mid h_k)$ and $p(h_k)$ of Equation (5) is easily computed based on the FM model of Figure 1. The computation required is not significantly more than the computation required had the variable H been observed. In fact, each of the $p(h_k \mid \boldsymbol{x}^{(l)})$ needs to be computed only once and then stored. This can be done as a preprocessing step to the search for the best discretization. We thus need to compute a total of NK terms, i.e., K terms for each of the N cases. Clearly, to the time required by the discretization, we need to add the time required to learn the FM model.

4 Evaluation

The empirical evaluation of multivariate discretization methods faces some of the same problems encountered in the evaluation of Bayesian network learning algorithms, in that it is difficult to set up real-data experiments. Moreover, when considering multivariate discretization methods, there is the additional issue of choosing the task for which the discretization will be needed, knowing that different tasks will privilege different features of a discretization.

In this paper, we use simulated data to test our discretization method. Since one of our main reasons for the development of multivariate discretization techniques is Bayesian network (BN) learning from mixed data (i.e., data containing both continuous and discrete variables), we use simulated data generated from a Bayesian network. Furthermore, since our main focus is on structure learning (i.e., learning of the BN structure), we use the structural differences of the learned BN relative to the generating gold standard BN to assess the discretization method performance.

In the remainder of this section, we first detail our experimental design. We then present and discuss the experimental results.

4.1 Experimental design

The experimental design is as follows.

- A dataset $\mathcal{D}$ is generated from the gold standard (GS) Bayesian network of choice.
- The discretization method of choice is applied to the dataset $\mathcal{D}$, yielding the discretized dataset $\mathcal{D}'$.
- A BN learning algorithm for discrete domains is applied to the dataset $\mathcal{D}'$, and the structural differences of the learned BN from the GS, are measured.

We used the ALARM network [1] as our gold standard, since it models a real domain and considerable effort has been put into its development. The ALARM network contains 37 nodes, and 46 arcs, with each node taking between 2 and 4 discrete values. Since all the variables in ALARM are discrete, some of these variables needed to be made continuous. To this purpose we adopted a very simple approach. We identified those variables corresponding to naturally continuous measures that were originally discretized in the construction of the ALARM network. For each of these variables, we considered each of its discrete values as the mean of a continuous distribution with known variance. That is, given an n-valued variable, we transformed it into a continuous variable by mapping its n values into the integers $0, \ldots, n-1$, and by considering these integers as the means of Normal distributions with common standard deviation σ to be set. Generating a value from a continuous variable thus obtained can be done in two steps: i) a discrete value in the range $0, \ldots, n-1$ is generated according to the distribution specified by the ALARM network; and ii) the generated discrete value is perturbed by white noise with standard deviation σ. Of the 37 discrete variables of ALARM, 22 variables were transformed into continuous variables.

To learn a BN structure from data, we used the K2 algorithm described in [4], which requires a node ordering as input. Since our objective is the assessment of discretization methods, we believe that reducing the possible sources of uncertainty in the learning process helps us to better focus on the discretization task, and to make the interpretation of the experimental results easier. For this reason, we did not adopt more general learning algorithms that do not need a node ordering as input.

For the discretization of the continuous dataset, we compared our cluster-based discretization method with a simple *constant-density* discretization.

In constant-density discretization, the continuous range of a variable is partitioned into sub-intervals each approximately containing an equal number of datapoints. Since the number of values needs to be specified, we considered constant-density discretizations with a number of values ranging from 2 to 5.

For the cluster-based discretization, we used the scoring metric described in Section 3, with an MDL penalty term as described in [7].

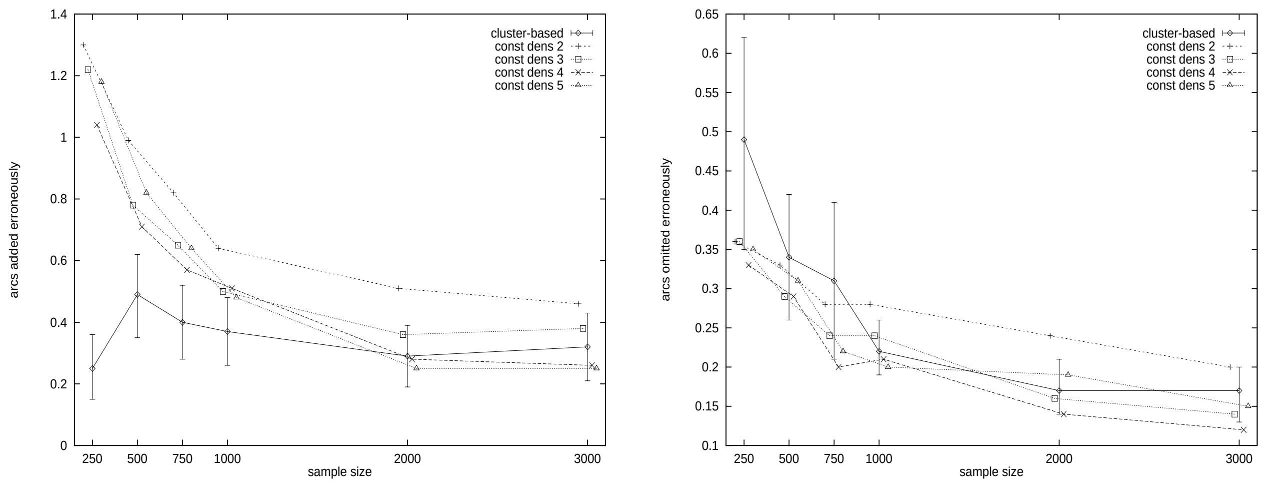

Figure 2: Fraction of arcs added (left graph) and omitted (right graph) based on the different discretization methods, for different sample sizes, averaged over the different values of the perturbation σ.

For the induction of the FM model used in the cluster-based discretization, we used the EM algorithm [5] for the estimation of the model's parameters, and we used the BIC approximation for model selection (i.e., for the determination of the number of values of the latent variable). The conditional distributions of the continuous variables were modeled as Gaussians.

Given the above design, we ran several experiments, where we varied the number of samples generated by the GS, and the scale of the perturbation σ of the continuous variables. In particular, we considered sample sizes of 250, 500, 750, 1000, 2000, and 3000 cases; and standard deviations σ of .25, .35, and .50.

4.2 Results

A summary of the experimental results is presented in Table 1 where, for each discretization algorithm (namely, cluster-based discretization, and constant density discretization with 2, 3, 4, and 5 values), we report mean and standard deviation of the fraction of arcs erroneously added and omitted by the learned BN with respect to the total number of arcs in the gold standard (i.e., $\frac{\#\ \textit{arcs added/omitted}}{46}$, where 46 is the number of arcs in ALARM). Mean and standard deviation were computed over 15 independent runs. From Table 1 it is clear that the constant-density discretization algorithms lead to the induction of BN structures that add a considerably larger number of incorrect arcs than the BN structure induced based on the cluster-based discretization method. On the other hand, the cluster-based discretization leads to the omission of more correct arcs than the constant-density algorithms, although the difference is not as significant as in the number of arcs incorrectly added. However, since the constant-density discretization tends to add more arcs, this also increases the possibility that some correct arcs are added by chance.

Discretization method	*arcs added erroneously*	*arcs omitted erroneously*
const dens 2	(36.2) 0.79 ± 0.30	(13.1) 0.28 ± 0.06
const dens 3	(29.8) 0.65 ± 0.31	(10.9) 0.24 ± 0.08
const dens 4	(25.9) 0.56 ± 0.28	(9.9) 0.22 ± 0.09
const dens 5	(27.8) 0.60 ± 0.34	(10.8) 0.24 ± 0.09
cluster-based	(16.2) 0.35 ± 0.12	(13.1) 0.28 ± 0.13

Table 1: Arcs erroneously added and omitted based on the different discretization methods. Each entry reports the mean and standard deviation of the fraction of arcs added/omitted with respect to the GS (the actual number of arcs added/omitted is reported between parentheses).

The plots of Figure 2 decompose the results of Table 1, and they report mean and standard deviation of the fraction of arcs added and omitted by the different discretization algorithms for different sample sizes. As in Table 1, the benefits of using the cluster-method discretization are mainly evident in the significantly lower number of arcs erroneously added, in particular for samples of smaller size.

The plots of Figure 3 decompose the results of Table 1 as a function of the perturbation σ. As in the previous graphs, the advantage of use of the cluster-based discretization is mainly evident in the reduced number of arcs erroneously added.

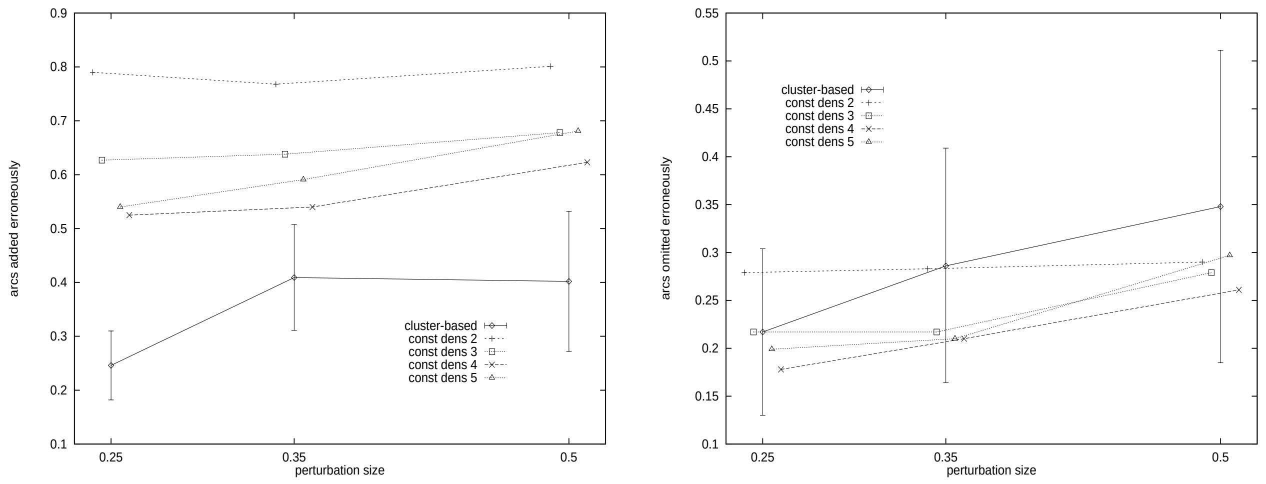

Figure 3: Fraction of arcs erroneously added (left graph) and omitted (right graph) based on the different discretization methods, for different values of the pertubation σ, averaged over different sample sizes.

5 Conclusions and future work

In this paper, we have introduced a new method for multivariate discretization based on the use of a latent variable model. The method uses a latent variable as a proxy class variable. This allows for the adoption of any of the available class-based discretization algorithms, with the natural modifications needed to account for the fact that the class variable is in this case latent.

The preliminary evaluation presented in this paper provides evidence that the new method has definite merits, but more experiments are needed to better assess its strengths and limitations.

For simplicity and clarity of exposition, in this paper we have focused on the entropy-based approach to class/cluster-based discretization. However, this is by no means the only possibility, and other scoring metrics for cutpoint selection can be used. In particular, it would be worth exploring the use of the Bayesian scoring metric for discretization proposed in [9], as well as the MDL-based scoring metric described in [8].

Acknowledgements

The research presented here was supported in part by grants BES-9315428 and IRI-9509792 from the National Science Foundation and by grant LM05291 from the National Library of Medicine.

References

[1] I. Beinlich, H. Suermondt, H. Chavez, and G. F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *2nd Conference of AI in Medicine Europe*, pages 247–256, London, England, 1989.

[2] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[3] M. R. Chmielewski and J. W. Grzymala-Busse. Global discretization of continuous attributes as preprocessing for machine learning. *International Journal of Approximate Reasoning*, 15:319–331, 1996.

[4] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society*, 39:398–409, 1977.

[6] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervized discretization of continuous features. In A. Prieditis and S. Russel, editors, *Machine Learning: Proocedings of the 12th International Conference*, San Francisco, CA, 1995.

[7] U. M. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13-th International Joint Conference on Artificial Intelligence*, pages 1022–1027, Chambéry, France, 1993.

[8] N. Friedman and M. Goldszmidt. Discretization of continuous attributes while learning Bayesian networks. In L. Saitta, editor, *Proceedings of 13-th International Conference on Machine Learning*, pages 157–165, 1996.

[9] S. Monti and G. F. Cooper. A multivariate discretization method for learning Bayesian networks from mixed data. In *Proceedings of 14th Conference of Uncertainty in AI*, pages 404–413, 1998.

[10] B. Pfahringer. Compression-based discretization of continuous attributes. In *Proceedings of 12th International Conference on Machine Learning*, pages 456–463, 1995.

[11] J. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, Los Altos, CA, 1993.