Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference

A Two-Step Method to Learn Multidimensional Bayesian Network Classifiers Based on Mutual Information Measures

Julio H. Zaragoza, L. Enrique Sucar and **Eduardo F. Morales**
National Institute of Astrophysics, Optics and Electronics
Sta. Ma. Tonantzintla, Puebla, Mexico, 72840
{jzaragoza, esucar, emorales}@inaoep.mx

Abstract

Bayesian Network Classifiers are popular approaches for classification problems where instances have to be assigned to one of several classes. However, in many domains, it is necessary to assign instances to multiple classes at the same time. This task has been normally addressed either by (i) transforming the problem into a single-class scenario by defining a new class variable with all of the possible combinations of classes or, (ii) by building an independent classifier for each class variable. Either way, the resulting models do not capture all the relations and dependencies between classes and features resulting into unprecise multidimensional classifiers. In this paper, we introduce a two-step method for learning Multidimensional Bayesian Network Classifiers (MBC) from data based on mutual information measures. The first step of the method learns an initial MBC structure which then, in the second step, is refined. Our approach is simple and keeps all the interactions and dependencies among classes and features. The method was tested on three benchmark multidimensional data-sets. Preliminary experimental results show how our method outperforms state-of-the-art methods used in multidimensional classification.

Introduction

Bayesian Network Classifiers have gained considerable popularity for solving classification problems where an instance, described as a set of features, has to be assigned to one of several possible classes. However, many complex applications can be viewed as a classification problem where instances have to be assigned not only to one class, but to a set of many different classes at the same time, e.g., text classification, DNA's sequences analysis, HIV's mutations control, etc.

In order to tackle multidimensional classification with Bayesian networks two main approaches have been proposed. The first approach transforms the multidimensional problem into a single-class scenario by defining a new compound class variable whose possible values are all of the possible combinations of values of the original classes. However, the resulting new class variable might have too many possible values. The second approach decomposes the multidimensional classification problem into different single-class problems by building an independent classifier for each class variable. However, this approach is unable to capture the interactions between classes and features and, in general, the most likely class of each classifier will not match the most likely set of classes due to possible interactions among classes.

Some other, recent approaches, instead of modifying the multidimensional problem try to find structures that keep the interactions between all the classes and all the features variables. However, some of these methods are computationally expensive (Rodríguez and Lozano 2008), do not completely state how to learn the structure of the multidimensional classifiers (van der Gaag and de Waal 2006) or do not keep the interactions between classes and features (Qazi et al. 2007).

In this paper we introduce a two step method for learning Multidimensional Bayesian Network Classifiers from data based on the mutual information or dependency between the classes and the features variables. This method learns, in the first step, an initial structure of the MBC very fast, then, in the second step, this structure is refined in order to increase the accuracy of the classification process. Our method is simple to compute and keeps the relations or dependencies between classes and features. We tested our method on three benchmark data-sets and the classification rates of our classifiers outperform recent algorithms used for multidimensional classification.

This paper is organized as follows, in the next Section we define Multidimensional Bayesian Network Classifiers. Then we present some related work concerned with learning Multidimensional Classifiers. Afterwards we introduce our proposed method and present the results of the performed experiments to finally conclude the paper and suggest future research directions.

Multidimensional Bayesian Network Classifiers

Before introducing Multidimensional Bayesian Network Classifiers (MBC) (van der Gaag and de Waal 2006), we present some Bayesian Networks notation. A Bayesian network over a finite set $V = X_1, ..., X_n, n \geq 1$, of discrete random variables is a pair $B = (G, \Theta)$, where G is an acyclic directed graph whose vertices correspond to the random variables of V and Θ is a set of parameters $\Theta_{x|pa(x)} = p(x|pa(x))$, where $pa(x)$ is a value of the set of variables $Pa(X)$, parents of the X variable in the graphical structure G. B defines a joint probability distribution P_B over V given by:

$$P_B(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|pa(x_i)) \qquad (1)$$

An MBC is a Bayesian Network of restricted topology designed to solve classification problems which include multiple class variables, in which instances, described by a number of features, have to be assigned to a combination of classes.

In an MBC the graph $G = (V, A)$ has the set V of vertices partitioned into two sets $V_C = \{C_1, ..., C_d\}, d \geq 1$, of class variables and $V_F = \{F_1, ..., F_m\}, m \geq 1$, of feature variables ($d + m = n = |V|$). G also has the set A of arcs partitioned into three sets A_C, A_F and A_{CF} such that:

- The set $A_C \subseteq V_C \times V_C$ is composed of the arcs between the class variables, that is, the subgraph of G induced by V_C: $G_C = (V_C, A_C)$.
- The set $A_F \subseteq V_F \times V_F$ is composed of the arcs between the feature variables, that is, the subgraph of G induced by V_F: $G_F = (V_F, A_F)$.
- The set $A_{CF} \subseteq V_C \times V_F$ includes the arcs from the class variables to the feature variables such that for each $F_i \in V_F$ there is a $C_j \in V_C$ with $(C_j, F_i) \in A_{CF}$ and for each $C_i \in V_C$ there is an $F_j \in V_F$ with $(C_i, F_j) \in A_{CF}$.

The subgraph G_C of G is called the classifier's *classes subgraph*, the subgraph G_F is called the *features subgraph*, and the structure that connects the classes subgraph with the features subgraph is called the feature selector (van der Gaag and de Waal 2006) or *bridge* structure (Bielza, Li, and Larrañaga 2010) of the MBC. This bridge represents the features that are deemed relevant for classification in view of the variables $C_1, ..., C_n$. Figure 1 shows an MBC with its corresponding subgraphs and bridge structure. Different graphical structures for the classes and features subgraphs may produce different families of MBCs. In general, classes and features subgraphs may be: empty, directed trees, forest of trees, polytrees, and general directed acyclic graphs (DAG) (Bielza, Li, and Larrañaga 2010).

An MBC in essence serves to find a joint value assignment of highest posterior probability for its set of class variables. However finding such values for all feature variables involved is, in general, an NP-hard problem (Bielza, Li, and Larrañaga 2010). In view of the computational complexity involved, in (van der Gaag and de Waal 2006) the authors showed that the practicability of multidimensional classifiers is limited to models with restricted class and features subgraphs, such as the naïve, tree-augmented and polytree-augmented classifiers. Our method focuses on learning polytree-augmented[1] Multidimensional Bayesian Network classifiers.

[1]The classes and the features subgraphs are both polytrees.

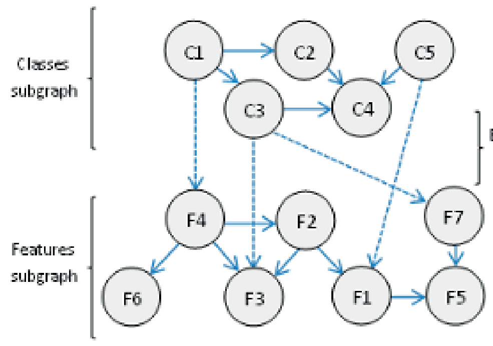

Figure 1: An example of a Multidimensional Bayesian Network Classifier with its corresponding classes and features subgraphs.

Related Work

In this section we present related work concerned with learning Multidimensional Classifiers.

In (Crammer and Singer 2001) the authors developed an approach for learning multidimensional support vector machines. They incorporate kernels with a compact set of constraints and decompose the problem into multiple optimization problems of reduced size. They also described a fixed-point algorithm for solving the reduced optimization problems. In (Rodríguez and Lozano 2008) the authors used a genetic algorithm called NSGA-II (Deb et al. 2000) for learning multidimensional polytrees. They used this genetic algorithm for learning the class, the features and the bridge structures independently. However, decomposing the problem or using genetic algorithms makes these methods complex, computationally expensive and slow which hampers their application to large datasets.

In (van der Gaag and de Waal 2006) the authors introduced the family of Multidimensional Bayesian Networks Classifiers (MBC) and proposed a method to learn tree-augmented MBCs structures. They first learn the classes and features subgraphs with Chow and Liu's algorithm (Chow and Liu 1968) and Chu and Liu's algorithm (Chu and Liu 1968). The bridge structure is learnt in a greedy wrapper way. On (de Waal and van der Gaag 2007) the authors extend their previous work for learning tree-augmented MBCs developing a method for finding the conditions for the optimal recovery of *polytree* structures in both subgraphs.

In (Bielza, Li, and Larrañaga 2010) the authors developed three strategies (filter, wrapper and hybrid) for learning any type of Bayesian network structure for the class and feature subgraphs of the MBC. Unfortunately, these strategies can lead, in some cases, to local maxima because of the greedy approaches they use. They also proposed a gray code for enumerating the joint configurations of all the class variables in a special order, which reduces the computational costs. Finally they presented two accuracy evaluation metrics for MBCs, the same that will be used to evaluate and compare our proposed method.

Our method designed to learn polytree-augmented MBCs structures does not assume a fixed bridge structure, keeps all of the dependencies between classes and features and reduces local maxima problems. Besides, the method is easy to compute and outperforms state-of-the-art methods. In the next section we introduce our method for learning MBCs.

Learning Multidimensional Bayesian Networks Classifiers

In this section we introduce our two-step method to learn Multidimensional Bayesian Networks Classifiers (MBC) from data. The learning problem is to find an MBC that best fits the available data. To find this structure we have developed a method based on mutual information. The mutual information between a pair of variables is calculated using the following equation:

$$I(X_i, X_j) = \sum_{i=0}^{n}\sum_{j=0}^{n} P(X_i, X_j) log(\frac{P(X_i, X_j)}{P(X_i)P(X_j)}) \quad (2)$$

By using equation 2, Chow and Liu (Chow and Liu 1968) developed an algorithm (shown in Algorithm 1) for learning maximum weight spanning trees from data. We use this algorithm for learning the class and the feature subgraphs independently.

Algorithm 1: Chow and Liu's algorithm for learning Maximum Weight Spanning Trees.

```
  Input: V: Set of Variables
  Output: G: Acyclic Undirected Tree Structure
1 Build an empty graph $G$ with $V$ as the set of vertices;
2 Calculate the mutual information value between all
  pairs of variables ($I(V_i, V_j)$);
3 Sort the mutual information values from the highest to
  the lowest;
4 Add the arc with the highest mutual information value
  to $G$;
5 repeat
6     Add the next arc iif no cycles are generated;
7 until All variables in $V$ are considered;
8 Return $G$;
```

The remaining task is then to learn the bridge structure of the MBC. To achieve this, we use the concept of mutual information between feature and class variables. If a class-feature pair has a "high" mutual information value, an arc from the class to the feature (i.e., in the bridge), is added to the MBC structure. Once this process is completed we refine the resulting structure.

Algorithm 2 presents the first step of our method to learn MBCs. In this first step we learn the classes and the features subgraphs through Chow and Liu's algorithm. Then we assign direction to the arcs of each of these subgraphs by using Rebane and Pearl's algorithm (Rebane and Pearl 1987). Then we build the bridge. In order to do this we define a *threshold*. If the mutual information between a class variable and a feature variable is larger than this $threshold$, an arc from this class to this feature is added to the bridge structure of the MBC. The idea of this first step is to quickly develop an initial MBC structure by avoiding accuracy or score tests that slow the learning process. Through this first step we can achieve a good approximation to the "best" MBC structure, the remaining work is to refine this initial MBC structure.

Algorithm 2: First Step, learning an initial structure of the MBC.

```
   Input: $V_C$ and $V_F$: Sets of classes and features
          variables
   Output: $G$: Initial MBC structure, and a set of arcs
           whose mutual information value $\ngtr threshold$
1  Learn, separately, the classes and features subgraphs
   ($G_C$ and $G_F$) through Chow and Liu's Algorithm;
2  Assign direction to the arcs in $G_C$ and $G_F$ with Rebane
   and Pearl's Algorithm;
3  Build a graph $G = G_C \bigcup G_F$;
4  Calculate the mutual information between classes and
   features ($I(V_{C_i}, V_{F_j})$);
5  Sort the mutual information values from the highest to
   the lowest;
6  Define a $threshold$ between the highest and lowest
   mutual information values;
7  Add the arc with the highest mutual information value
   to $G$;
8  while The mutual information value of the arc
   $> threshold$ do
9      Add the next arc iif no cycles are generated;
10 Return $G$ and the set of arcs whose mutual information
   value $\ngtr threshold$;
```

The second step of our method refines the initial MBC structure. This initial MBC structure has some of all of the possible arcs on its bridge structure. And those arcs represent the strongest dependencies (highest mutual information values) between classes and features. However, there could be arcs that, although they have a *low* mutual information value, contribute to improve the classification accuracy. That is why, in this second step of our method, shown in Algorithm 3, we add an arc to the MBC's bridge structure (from the set of arcs that were not previously added in the first step) and we test if the accuracy of this MBC is better that the accuracy of the MBC without this arc. If the accuracy of the MBC with this arc is better we keep this new MBC structure and we go for the next arc in the same way. If there is no improvement in the accuracy then our method finishes returning the last MBC.

As accuracy measures for learning the MBCs structures on Algorithm 3 we can use:

- The *Mean accuracy* over the d class variables (accuracy per label):

$$\overline{Acc_d} = \frac{1}{d}\sum_{j=1}^{d} Acc_j = \frac{1}{d}\sum_{j=1}^{d}\frac{1}{N}\sum_{i=1}^{N}\delta(c'_{ij}, c_{ij}) \quad (3)$$

  where N is the training set, $\delta(c'_{ij}, c_{ij}) = 1$ if $c'_{ij} = c_{ij}$ and 0 otherwise. Note that c'_{ij} denotes the C_j class value outputted by the model for case i and c_{ij} is its true value.

Algorithm 3: Second Step, refinement of the structure of the MBC.

```
  Input: $G$: Initial MBC structure, and the set of arcs
         whose mutual information value $\not> threshold$
  Output: G: Final (refined) MBC Structure
1 Build a graph $G' = G$;
2 repeat
3     $G = G'$;
4     Add the next arc to $G'$ iif no cycles are generated;
5 until Accuracy($G'$) $\not>$ Accuracy($G$);
6 Return $G$;
```

- Or the *Global accuracy* over the d-dimensional class variable (accuracy per example):

$$Acc = \frac{1}{N} \sum_{i=1}^{N} \delta(c_i', c_i) \qquad (4)$$

where $\delta(c_i', c_i) = 1$ if $c_i' = c_i$ and 0 otherwise. Therefore, we call for a total coincidence on all of the components of the vector of predicted classes and the vector of real classes.

With this two-step method we achieve a good compromise between learning time and complexity of the models. If we define a low $threshold$ value in the first step of our method we can learn a model very fast because the second step does not have to perform many accuracy tests. But, the resulting model can also be very complex (with a lot arcs that do not represent strong or relevant dependencies on the bridge structure). On the other hand, if we define a high $threshold$ value in the first step of the method we are going to have simple models where the dependencies or arcs in the bridge structure are going to belong to strong dependencies only. However, the second step of the method is going to spend a lot of time refining the model searching for the remaining relevant dependency relations. So, defining an optimal *threshold* is not an easy task. However we experimentally observed that a *threshold* that allows 30% of the arcs to be added to the MBC structure in the first step of the method quickly generates a good initial structure of the MBC which is then refined in the second step of the method in relatively short time.

Once the structure is learned through our two-step method, the parameters of the MBCs can be estimated as in standard Bayesian networks.

In the next section we present the experimental results.

Experiments and Results

The proposed method was tested for developing MBCs for three benchmark data sets.

The *Emotions* data set includes 593 sound clips (examples) from a 30-seconds sequence after the initial 30 seconds of a song. The 72 features extracted fall into two categories: 8 rhythmic features and 64 timbre features. Songs are categorized by 6 class variables: amazed-surprised, happy-pleased, relaxing- calm, quiet-still, sad-lonely, and angry-aggressive. The *Scene* data set has 2407 pictures, and their semantic scenes have to be classified into 6 class binary variables: beach, sunset, foliage, field, mountain, and urban. The 294 features correspond to spatial color moments in the *LUV* space. The *Yeast* data set is about predicting 14 functional classes of 2417 genes in the *Saccharomyces Cerevisae Yeast*. Each gene is described by the concatenation of some microarray expression data and a phylogenetic profile given by 103 features. All class variables for the three data sets are binary, however, feature variables are numeric; we used a static, global, supervised and top-down discretization algorithm (Cheng-Jung, Chien-I, and Wei-Pang 2008). The datasets can be found at `mulan.sourceforge.net/datasets.html`[2].

We compared our method against eight different algorithms to learn MBCs (Bielza, Li, and Larrañaga 2010). From these eight algorithms, five were explicitly designed for learning MBCs: *tree-tree* (van der Gaag and de Waal 2006), *polytree-polytree* (de Waal and van der Gaag 2007) and *pure filter*, *pure wrapper* and *hybrid* (Bielza, Li, and Larrañaga 2010). Two of the algorithms use greedy search approaches that learn a general Bayesian network, one guided by the K2 metric (Cooper and Herskovits 1992) (filter approach), and the other guided by a performance evaluation metric, as defined in (Bielza, Li, and Larrañaga 2010) (wrapper approach). The first one will be denoted *K2 BN*, while the second one will be denoted *wrapper BN*. And the last one of the eight algorithms is a multi-label lazy learning approach named *ML-KNN* (Zhang and Zhou 2006), derived from the traditional *K*-nearest neighbor algorithm. In this case, *K* was set to 3 in the *Emotions* and *Scene* data sets, and 5 in the *Yeast* data set. Since it is unfeasible to compute the mutual information of two features given all the class variables, as required in (van der Gaag and de Waal 2006), the implementation of the *polytree-polytree* learning algorithm uses the marginal mutual information of pairs of features.

For our method, named *Mi-MBC*, we choose (as mentioned before) a *threshold* which adds to the bridge nearly 30% of the arcs in the first part of Algorithm 2. Then in the second part the accuracy measures described before are used to improve the model. Our method was developed in C++. For parameter estimation and for evaluation we have integrated and used the *R package* (Gentleman and Ihaka 1997).

Table 1 summarizes the results using first the Mean accuracy measure and then the Global accuracy (from Equations 3 and 4) for each data set.

As shown in Table 1, in the case of the *Emotions* data set our method did not achieve as high accuracy as the *Hybrid* and *tree-tree* methods. The reason is mainly that the number of instances is relatively low compared with the number of features. So the mutual information that those examples provide is not enough for achieving higher classification rates as the other two approaches; however, our approach is very competitive.

For the *Scene* and the *Yeast* data sets, the performance of our approach was significantly better than the other approaches because these data sets have more examples than the *Emotions* data set. When we have more examples, we

[2]Last time visited: February 07, 2011.

Table 1: Performance metrics (mean ± std. deviation) and rank (in brackets) of the algorithms over the three data sets. For all of the algorithms 10-fold cross validation was used.

	Mean Acc.	Global Acc.
Emotions		
tree-tree	0.8300 ±0.0151(2)	**0.3844 ±0.0398(1)**
polytree-polytree	0.8209 ±0.0243(5)	0.3776 ±0.0622(3)
Pure filter	0.7548 ±0.0280(8)	0.2866 ±0.0495(7)
Pure wrapper	**0.8333 ±0.0123(1)**	0.3708 ±0.0435(4)
Hybrid	0.8210 ±0.0170(4)	0.3557 ±0.0435(5)
K2 BN	0.7751 ±0.0261(7)	0.2812 ±0.0799(8)
Wrapper BN	0.7985 ±0.0200(6)	0.3033 ±0.0752(6)
ML-KNN	0.6133 ±0.0169(9)	0.0254 ±0.0120(9)
Mi-MBC	0.8250 ±0.0159(3)	0.3793 ±0.0631(2)
Scene		
tree-tree	0.7324 ±0.0359(8)	0.1857 ±0.0977(7)
polytree-polytree	0.7602 ±0.0663(7)	0.2643 ±0.1915(5)
Pure filter	0.7726 ±0.0700(5)	**0.3067 ±0.1991(1)**
Pure wrapper	0.7765 ±0.0580(3)	0.2688 ±0.1642(4)
Hybrid	0.7229 ±0.0442(9)	0.1570 ±0.1018(8)
K2 BN	0.7689 ±0.0692(6)	0.2883 ±0.1995(2)
Wrapper BN	0.7739 ±0.0492(4)	0.2277 ±0.1372(6)
ML-KNN	0.8196 ±0.0092(2)	0.0311 ±0.0147(9)
Mi-MBC	**0.8457 ±0.0412(1)**	0.2876 ±0.1050(3)
Yeast		
tree-tree	0.7728 ±0.0071(4)	**0.1953 ±0.0207(1)**
polytree-polytree	0.7336 ±0.0182(8)	0.1431 ±0.0257(3)
Pure filter	0.7480 ±0.0119(6)	0.0989 ±0.0342(7)
Pure wrapper	0.7845 ±0.0131(2)	0.1410 ±0.0989(4)
Hybrid	0.7397 ±0.0114(7)	0.1200 ±0.0268(6)
K2 BN	0.7686 ±0.0112(5)	0.1299 ±0.0204(5)
Wrapper BN	0.7745 ±0.0049(3)	0.0550 ±0.0212(8)
ML-KNN	0.6364 ±0.0196(9)	0.0062 ±0.0029(9)
Mi-MBC	**0.8046 ±0.0112(1)**	0.1889 ±0.0442(2)

Table 2: Average Accuracy values and global ranking (in brackets) of the algorithms over the three data sets.

	Mean Acc.	Global Acc.	Average Acc.
tree-tree	4.6667	3.0000	3.8333(3)
polytree-polytree	6.6667	3.6667	5.1666(4)
Pure filter	6.3333	5.0000	5.6666(7)
Pure wrapper	2.0000	4.0000	3.0000(2)
Hybrid	6.6667	6.3333	6.5000(8)
K2 BN	6.0000	5.0000	5.5000(6)
Wrapper BN	4.3333	6.6667	5.5000(5)
ML-KNN	6.6667	9.0000	7.8333(9)
Mi-MBC	**1.6667**	**2.3333**	**2.0000(1)**

are able to obtain mutual information values that allow our method to accurately represent the relations and dependencies between variables and, as consequence, we have better performance in the classification tasks.

Table 2 shows the average rankings of the algorithms. As shown in this table, in general, our method outperforms the other methods designed for multidimensional classification.

The methods that we used to compare ours were previously coded[3] in Matlab and they were programmed by using parallel methods. Thus, a direct comparison in terms of processing times of our method against the others is not possible, we can only provide a qualitative one. For a comparison of the execution times between those methods please refer to (Bielza, Li, and Larrañaga 2010). So far we can only say that the execution time of our method ranges between the limits of the *tree-tree* parallel implementation but further tests are needed in order to accurately measure and compare the execution times. In Table 3 we present the time that our method took to build the MBCs structures and learn the parameters on the three benchmark data sets. The table shows if the MBC model was learnt using the Mean or the Global accuracy measure, the time that took every step of our method, the time that took to learn the parameters of the MBC (through the R package) and the total time. Our method was tested on a computer equipped with 2 Gb of RAM Memory and with a *Pentium Dual Core* processor at 1.8 Ghz.

Table 3: Time (in minutes) per type of accuracy measure and per step, that our method took to build the MBCs structures.

	Step 1	Step 2	Param. Est.	**Total**
Emotions				
Mean accuracy	13	20	18	51
Global accuracy	13	16	15	44
Scene				
Mean accuracy	21	30	42	93
Global accuracy	21	23	45	89
Yeast				
Mean accuracy	31	30	55	116
Global accuracy	27	26	64	117

As we can see in Table 3, in average the parameter estimation step is the most expensive computationally, followed by the second phase (step 2) of our method.

Conclusions and Future Work

In this paper we introduced a two-step method for learning Multidimensional Bayesian Network Classifiers (MBC). In the first step, our method learns the subgraphs for the class and for the feature variables by using Chow and Liu's algorithm and also learns a "first" bridge structure that connects these two subgraphs. In the second step, our method refines the MBC structure, learned in the first step, by using the mutual information values between classes and features variables. Our method adds only those arcs belonging to highly dependant variables guiding the second step and avoiding testing possible irrelevant arcs. Our method was able to outperform state-of-the-art methods designed for multidimensional classification. As future work we would like to incorporate the possibility of deleting previously added arcs from the bridge in order to evaluate the complexity and accuracy of the MBC structure; to perform experiments on some "real-world" multidimensional data-sets (e.g., the Slashdot `slashdot.org` or the

[3]Developed by (Bielza, Li, and Larrañaga 2010).

IMDB imdb.com/interfaces#plain data-sets), and to compare our method with some other recent approaches for Multidimensional Classification (e.g., Classifier Chains (Read et al. 2009)).

Acknowledgments

The authors wish to acknowledge **FONCICYT** for the support provided through Project **DyNaMo**, No. 95185 (www.dynamopro.org).
The authors would also like to thank Dr. Pedro Larrañaga and Dr. Concha Bielza from the Polytechnical University of Madrid for their help and comments about this work.

References

Bielza, C.; Li, G.; and Larrañaga, P. 2010. Multi-dimensional classication with bayesian networks. *Technical Report UPM-FI/DIA/2010-1.*

Cheng-Jung, T.; Chien-I, L.; and Wei-Pang, Y. 2008. A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences* (178):714–731.

Chow, C., and Liu, C. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* 14:462–467.

Chu, C., and Liu, T. H. 1968. On the shortest arborescence of a directed graph. *Science Sinica* 14:1396–1400.

Cooper, G. F., and Herskovits, E. 1992. A bayesian method for the induction of probabilistic networks from data. *Machine Learning* (9):309–347.

Crammer, K., and Singer, Y. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Machine Learning Research* 2:265–292.

de Waal, P. R., and van der Gaag, L. C. 2007. Inference and learning in multi-dimensional bayesian network classifiers. In *In European Conference on Symbolic and Quantitative Approaches to Reasoning under Uncertainty, Lecture Notes in Artificial Intelligence*, volume 4724, 501–511.

Deb, K.; Agrawal, S.; Pratap, A.; and Meyarivan, T. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: Nsga-ii. *Parallel Problem Solving from Nature. Lecture Notes in Computer Science* 849–858.

Gentleman, R., and Ihaka, R. 1997. The r project. www.r-project.org.

Qazi, M.; Fung, G.; Krishnan, S.; Rosales, R.; Steck, H.; Rao, R. B.; Poldermans, D.; and Chandrasekaran, D. 2007. Automated heart wall motion abnormality detection from ultrasound images using bayesian networks. *International Joint Conference on Articial Intelligence* 519–525.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2009. Classifier chains for multi-label classification. In *Proceedings ECML/PKDD*, 254–269.

Rebane, G., and Pearl, J. 1987. The recovery of causal poly-trees from statistical data. *Proceedings of the Third Annual Conference on Uncertainty in Articial Intelligence (UAI-87)* 3:222–228.

Rodríguez, J. D., and Lozano, J. A. 2008. Multi-objective learning of multi-dimensional bayesian classifiers. *Proceedings of the Eighth International Conference on Hybrid Intelligent Systems* 501–506.

van der Gaag, L. C., and de Waal, P. R. 2006. Multi-dimensional bayesian network classifiers. In *Third European Conference on Probabilistic Graphic Models*, 107–114.

Zhang, M. L., and Zhou, Z. H. 2006. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering* 18(10):1338–1351.