Bayesian Chain Classifiers for Multidimensional Classification

Julio H. Zaragoza, L. Enrique Sucar,
Eduardo F. Morales, Concha Bielza* and **Pedro Larrañaga***
Computer Science Department, National Institute for Astrophysics, Optics and Electronics, Puebla, Mexico
{jzaragoza, esucar, emorales}@inaoep.mx
* Computational Intelligence Group, Technical University of Madrid, Madrid, Spain
{mcbielza, pedro.larranaga}@fi.upm.es

Abstract

In multidimensional classification the goal is to assign an instance to a set of different classes. This task is normally addressed either by defining a compound class variable with all the possible combinations of classes (label power-set methods, LPMs) or by building independent classifiers for each class (binary-relevance methods, BRMs). However, LPMs do not scale well and BRMs ignore the dependency relations between classes. We introduce a method for chaining binary Bayesian classifiers that combines the strengths of classifier chains and Bayesian networks for multidimensional classification. The method consists of two phases. In the first phase, a Bayesian network (BN) that represents the dependency relations between the class variables is learned from data. In the second phase, several chain classifiers are built, such that the order of the class variables in the chain is consistent with the class BN. At the end we combine the results of the different generated orders. Our method considers the dependencies between class variables and takes advantage of the conditional independence relations to build simplified models. We perform experiments with a chain of naïve Bayes classifiers on different benchmark multidimensional datasets and show that our approach outperforms other state-of-the-art methods.

1 Introduction

In contrast with traditional (one–dimensional) classifiers, multidimensional classifiers (MDCs) assign each instance to a set of d classes. MDCs have gained a lot of attention in recent years, as several important problems can be seen as multidimensional classification [Zhang and Zhou, 2007; Vens *et al.*, 2008], such as text classification (assigning a document to several topics), HIV drug selection (determining the optimal set of drugs), among others.

Two main types of approaches have been proposed for solving a MDC problem with binary classes: *binary relevance* and *label power-set* [Tsoumakas and Katakis, 2007]. In the binary relevance approach [Zhang and Zhou, 2007], an MDC problem is transformed into d binary classification problems, one for each class variable, $C_1, \ldots, C_d$. A classifier is independently learned for each class variable, and the results are combined to determine the predicted class set. The main advantages of this approach are its low computational complexity, and that existing classification techniques can be directly applied. However, it is unable to capture the interactions between classes and, in general, the most likely class of each classifier will not match the most likely set of classes due to possible interactions among them.

The label power-set approach [Tsoumakas and Katakis, 2007] 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. In this case the interactions between the different classes are implicitly considered. Thus, it can be effective for domains with a few class variables. But its main drawback is its computational complexity, as the size of the compound class variable increases exponentially with the number of classes.

To overcome the limitations of the previous methods, there are two main strategies: (i) to incorporate class interactions in binary relevance methods, in what are known as *chain classifiers* [Read *et al.*, 2009; Dembczynski *et al.*, 2010], and (ii) to explicitly represent the dependence structure between the classes, avoiding the combinatorial explosion of the label power-set approach, via *multidimensional Bayesian network classifiers* [van der Gaag and de Waal, 2006; Bielza *et al.*, 2011].

Bayesian Chain Classifiers (see Fig. 1) combine the previous strategies, taking advantage of their strengths and at the same time avoiding their main limitations. The method for learning these classifiers consists of two main phases: (i) to obtain a dependency structure for the class variables, and (ii) based on the dependency structure, build a classifier chain. In the first phase, a Bayesian network (BN) that represents the dependency relations between the class variables is learned from data. This class structure serves as a guide for the second phase, as it restricts the possible variable orderings in the chain. In the second phase, a chain classifier is built, such that the order of the class variables in the chain is consistent with the graph of the previously learned BN (class BN). Although there are still many possible orderings, the number is reduced significantly with respect to all possible random orders. In this chain, previous classes are also incorporated as

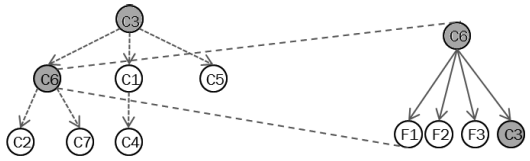

Figure 1: An example of a Bayesian Chain Classifier where each intermediate node on the chain is a naïve Bayesian classifier which has as attributes only its parent classes (C_3) and its corresponding features (F_1, F_2, F_3).

features along the chain, but only the *parents* variables in the class BN, as in a BN every variable is independent of its non-descendants given its parents. Thus, the number of additional features is restricted even for domains with a large number of classes. Finally, as for chain classifiers, the predicted class set is obtained by combining the outputs of all the classifiers in the chain.

In this paper we present the simplest version of a Bayesian Chain Classifier, as the dependency structure between class variables is restricted to a directed tree. Thus, each class variable in the tree has at most one parent, so only one additional feature is incorporated to each base classifier in the chain. Additionally, the base classifiers are naïve Bayes. As in [Read *et al.*, 2009], we combine several chain classifiers in a classifier ensemble, by changing the root node in the tree. This basic Bayesian Chain Classifier is highly efficient in terms of learning and classification times, and it outperforms other more complex MDCs as demonstrated in the experiments.

2 Multidimensional Classifiers

As previously introduced, in this contribution we will present an approach to classification problems with d class variables, $C_1, \ldots, C_d$. In this framework, the *multi-dimensional classification* problem corresponds to searching for a function h that assigns to each instance represented by a vector of m features $\mathbf{x} = (x_1, \ldots, x_m)$ a vector of d class values $\mathbf{c} = (c_1, \ldots, c_d)$:

$$h : \Omega_{X_1} \times \cdots \times \Omega_{X_m} \rightarrow \Omega_{C_1} \times \cdots \times \Omega_{C_d}$$

$$(x_1, \ldots, x_m) \mapsto (c_1, \ldots, c_d)$$

We assume that C_i and X_j for all $i = 1, \ldots, d$ and all $j = 1, \ldots, m$ are discrete, and that Ω_{C_i} and Ω_{X_j} respectively represent their sample spaces.

Under a $0-1$ loss function, the h function should assign to each instance $\mathbf{x}$ the most likely combination of classes, that is:

$$argmax_{c_1,\ldots,c_d} p(C_1 = c_1, \ldots, C_d = c_d | \mathbf{x})$$

This assignment amounts to solving a total abduction inference problem and corresponds to the search for the most probable explanation (MPE), a problem that has been proved to be an NP-hard problem for Bayesian networks [Shimony, 1994].

3 Related Work

In this section we briefly review the main approaches for multidimensional classification. The review is organized into three subsections, discussing research in multi-label classification, multidimensional Bayesian networks classifiers, and chain classifiers, respectively.

3.1 Multi-label Classification

In multi-label classification domains each instance is associated with a subset of labels (present in the instance) from a set of d labels. Taking the notation introduced in previous sections into account, this multi-label classification problem can be seen as a particular case of a multidimensional classification problem where all class variables are binary, that is $|\Omega_{C_i}| = 2$ for $i = 1, \ldots, d$.

An overview of multi-label classification is given in [Tsoumakas and Katakis, 2007], where two main categories are distinguished: (a) problem transformation methods, and (b) algorithm adaptation methods. Methods in (a) transform the multi-label classification problem into either one or more single-label classification problems. Methods in (b) extend specific learning algorithms to handle multi-label data directly. For example, decision trees [Vens *et al.*, 2008], support vector machines [Boutell *et al.*, 2004], k-nearest neighbor [Zhang and Zhou, 2007], neural networks [Zhang and Zhou, 2006], and a hybrid of logistic regression and k-nearest neighbor [Cheng and Hullermeier, 2009] have been proposed.

3.2 Multidimensional Bayesian Network Classifiers

A multidimensional Bayesian network classifier (MBC) over a set $\mathcal{V} = \{Z_1, \ldots, Z_n\}$, $n \geq 1$, of discrete random variables is a Bayesian network $B = (\mathcal{G}, \Theta)$, where $\mathcal{G}$ is an acyclic directed graph with vertexes Z_i and Θ is a set of parameters $\theta_{z|\mathbf{pa}(z)} = p(z|\mathbf{pa}(z))$, where $\mathbf{pa}(z)$ is a value for the set $\mathbf{Pa}(Z)$, parents variables of Z in $\mathcal{G}$. B defines a joint probability distribution p_B over $\mathcal{V}$ given by:

$$p_B(z_1, \ldots, z_n) = \prod_{i=1}^{n} p(z_i|\mathbf{pa}(z_i)). \quad (1)$$

The set $\mathcal{V}$ of vertexes is partitioned into two sets $\mathcal{V}_\mathcal{C} = \{C_1, \ldots, C_d\}$, $d \geq 1$, of class variables and $\mathcal{V}_\mathcal{X} = \{X_1, \ldots, X_m\}$, $m \geq 1$, of feature variables ($d + m = n$). The set $\mathcal{A}$ of arcs is also partitioned into three sets, $\mathcal{A}_\mathcal{C}$, $\mathcal{A}_\mathcal{X}$, $\mathcal{A}_{\mathcal{C}\mathcal{X}}$, such that $\mathcal{A}_\mathcal{C} \subseteq \mathcal{V}_\mathcal{C} \times \mathcal{V}_\mathcal{C}$ is composed of the arcs between the class variables, $\mathcal{A}_\mathcal{X} \subseteq \mathcal{V}_\mathcal{X} \times \mathcal{V}_\mathcal{X}$ is composed of the arcs between the feature variables and finally, $\mathcal{A}_{\mathcal{C}\mathcal{X}} \subseteq \mathcal{V}_\mathcal{C} \times \mathcal{V}_\mathcal{X}$ is composed of the arcs from the class variables to the feature variables. The corresponding induced subgraphs are $\mathcal{G}_\mathcal{C} = (\mathcal{V}_\mathcal{C}, \mathcal{A}_\mathcal{C})$, $\mathcal{G}_\mathcal{X} = (\mathcal{V}_\mathcal{X}, \mathcal{A}_\mathcal{X})$ and $\mathcal{G}_{\mathcal{C}\mathcal{X}} = (\mathcal{V}, \mathcal{A}_{\mathcal{C}\mathcal{X}})$, called respectively class, feature and bridge subgraphs.

Different graphical structures for the class and feature subgraphs may lead to different families of MBCs. [van der Gaag and de Waal, 2006] learn *trees* for both subgraphs by searching for the maximum weighted undirected spanning tree and transforming it into a directed tree using Chow and Liu's algorithm [1968]. The bridge subgraph is greedily learnt in a

wrapper way, trying to improve the percentage of correctly classified instances. [de Waal and van der Gaag, 2007] is a theoretical work for finding the conditions for the optimal recovery of *polytree* structures in both subgraphs.

[Rodríguez and Lozano, 2008] extend polytrees to k-DB structures for class and features subgraphs. Learning these structures is carried out using a multi-objective genetic algorithm where the individuals are permitted structures coded with three substrings, one per subgraph. Simpler models are used in an application for heart wall motion prediction [Qazi *et al.*, 2007]: a directed acyclic graph for the class subgraph, an empty graph for the features, and a bridge subgraph where features receive arcs from some class variables, without sharing any of them.

Finally, [Bielza *et al.*, 2011] present the most general models since any Bayesian network structure is allowed in the three subgraphs. Learning from data algorithms cover all the possibilities: wrapper, filter and hybrid score+search strategies. Moreover, since the computation of the MPE involves a high computational cost, several contributions are designed to alleviate it.

3.3 Chain Classifiers

Read and others [2009] introduce chain classifiers as an alternative method for multi-label classification that incorporates class dependencies, while it tries to keep the computational efficiency of the binary relevance approach. Chain classifiers consist of d binary classifiers which are linked in a chain, such that each classifier incorporates the class predicted by the previous classifiers as additional attributes. Thus, the feature vector for each binary classifier, L_i, is extended with the labels $(0/1)$ of all previous classifiers in the chain. Each classifier in the chain is trained to learn the association of label l_i given the features augmented with all previous binary predictions in the chain, $l_1, l_2, \ldots, l_{i-1}$. For classification, it starts at L_1, and propagates along the chain such that for $i \in \mathcal{L}$ (where $\mathcal{L} = \{l_1, l_2, \ldots, l_d\}$) it predicts $p(l_i \mid \mathbf{x}, l_1, l_2, \ldots, l_{i-1})$. As in the binary relevance approach, the class vector is determined by combining the outputs of all the binary classifiers in the chain.

[Read *et al.*, 2009] combine several chain classifiers by changing the order for the labels, building an ensemble of chain classifiers. Thus, m chain classifiers are trained, by varying the training data and the order of the classes in the chain (both are set randomly). The final label vector is obtained using a voting scheme; each label l_i receives a number of votes from the m chain classifiers, and a threshold is used to determine the final predicted multi-label set.

Recently, Dembczynski et al. [2010] present *probabilistic chain classifiers* (PCCs), by basically putting chain classifiers under a probabilistic framework. Using the chain rule, the probability of the vector of d class values $\mathbf{C} = (C_1, \ldots, C_d)$ given the feature vector $\mathbf{x}$ can be written as:

$$p(\mathbf{C}) = p(C_1 \mid \mathbf{x}) \prod_{i=2}^{d} p(C_i \mid C_1, C_2, \ldots, C_{i-1}, \mathbf{x}). \quad (2)$$

Given a function f_i that provides an approximation of the probability of $C_i = 1$, they define a probabilistic chain classifier as:

$$p(\mathbf{C}) = f_1(\mathbf{x}) \prod_{i=2}^{d} f_i(\mathbf{x}, C_1, C_2, \ldots, C_{i-1}). \quad (3)$$

PCC estimates the joint probability of the classes, providing better estimates than the chain classifiers, but with a much higher computational complexity. In fact, the experiments reported by [Dembczynski *et al.*, 2010] are limited to 10 classes.

As shown by [Dembczynski *et al.*, 2010], a method that considers class dependencies under a probabilistic framework can have a significant impact on the performance of multidimensional classifiers. However, both MBCs and PCCs have a high computational complexity, which limits their applicability to high dimensional problems. In the following section we describe an alternative probabilistic method which also incorporates class dependencies but at the same time is very efficient.

4 Bayesian Chain Classifiers

Given a multidimensional classification problem with d classes, a Bayesian Chain Classifier (BCC) uses d classifiers, one per class, linked in a chain. The objective of this problem can be posed as finding a joint distribution of the classes $\mathbf{C} = (C_1, C_2, \ldots, C_d)$ given the attributes $\mathbf{x} = (x_1, x_2, \ldots, x_l)$:

$$p(\mathbf{C}|\mathbf{x}) = \prod_{i=1}^{d} p(C_i|\mathbf{pa}(C_i), \mathbf{x})$$

where $\mathbf{pa}(C_i)$ represents the parents of class C_i. In this setting, a chain classifier can be constructed by inducing first the classifiers that do not depend on any other class and then proceed with their sons. A Bayesian framework allows us to:

- Create a (partial) order of classes in the chain classifier based on the dependencies between classes given the features. Assuming that these dependencies can be represented as a Bayesian network (directed acyclic graph), the chain structure is defined by the structure of the BN, such that we can then start building classifiers for the classes without parents, and continue with their children classes, and so on.
- Consider conditional independencies between classes to create simpler classifiers. In this case, construct d classifiers considering only the parent classes of each class. For a large number of classes this can be a huge reduction as normally we can expect to have a limited number of parents per class.

In general, when we induce a Bayesian network to represent the above joint distribution, it is not always possible to find directions for all the links. In that case we can have different orders depending on the chosen directions. In the worst case the number of possible directions grows exponentially with the number of links (2^k for k undirected links). In practice we expect to have only a limited number of undirected links. In that case we can obtain (a subset of) the

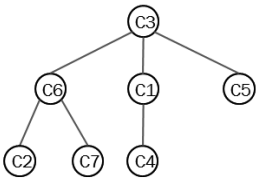

Figure 2: Example of a Maximum Weight Spanning Tree of Classes.

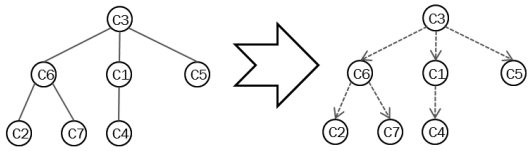

Figure 3: Using the Maximum Weighted Spanning Tree of Classes with node 3 as root for determining the chaining order.

possible orders and build an ensemble of chain classifiers with different orders. Given a new instance, we determine $p(C_i|\mathbf{pa}(C_i), \mathbf{x})\forall i \in D$ with each chain classifier, and use a voting scheme to output a set of classes.

We can simplify the problem by considering the marginal dependencies between classes (as a first approximation) to obtain an order in the chain classifier and then induce classifiers considering such order. Additionally, we can simplify even further the problem by considering only one parent per class. This can be solved by obtaining the skeleton of a tree-structured BN for the classes using Chow and Liu's algorithm (1968), that is, a *maximum weight undirected spanning tree* ($MWST$) (see Fig. 2).

Chow and Liu's algorithm does not give us the directions of the links, however, we can build d directed trees by taking each class (node) as root of a tree and assigning directions to the arcs starting from this root node to build a directed tree (Fig. 3). The chaining order of the classifiers is given by traversing the tree following an ancestral ordering.

For d classes we build d classifiers in the order given by the different trees and then combine them in an ensemble (if d is very large we can limit the number of chains by selecting a random subset of trees).

There are many different choices for representing each classifier, one of the simplest and fastest to build is the naïve Bayes classifier (NBCs); although other classifiers could be used as well. With naïve Bayes classifiers in the chain, we need to consider only the class parent, $\mathbf{pa}(C_i)$, and the feature vector, $\mathbf{x}$, as attributes for each NBC, see Fig. 1.

We can summarize our algorithm as follows. Given a multidimensional classification problem with d classes:

1. Build an undirected tree to approximate the dependency structure among class variables.
2. Create d orders for the chain classifiers by taking each class as the root of the tree and assigning the rest of the links in order.
3. For each classifier in each chain, build an NBC with the class C_i as root and only the parents $\mathbf{pa}(C_i)$ and all the attributes $\mathbf{x}$ as children, taking advantage of conditional independence properties.

To classify a new instance combine the output of the d chains, using a simple voting scheme.

This is a very fast and easy to build ensemble of chain classifiers, which represents the simplest alternative for a BCC. Other, more complex alternatives can be explored by: (i) considering conditional dependencies between classes, (ii) building more complex class dependency structures, and (iii) using other base classifiers.

In the next section we present the experimental results where we compare BCCs with other state of the art multidimensional classifiers.

5 Experiments and Results

The proposed method was tested on 9 different benchmark multidimensional datasets[1]; each of them with different dimensions ranging from 6 to 983 labels, and from about 600 examples to more than $40,000$. All class variables of the datasets are binary, however, in some of the datasets the feature variables are numeric. In these cases we used a static, global, supervised and top-down discretization algorithm [Cheng-Jung *et al.*, 2008]. The details of the datasets are summarized in Table 1.

Table 1: Multidimensional datasets used in the experiments and associated statistics. N is the size of the dataset, d is the number of binary classes or labels, m is the number of features. $*$ indicates numeric attributes.

No.	*Dataset*	N	d	m	*Type*
1	*Emotions*	593	6	72$*$	Music
2	*Scene*	2407	6	294$*$	Vision
3	*Yeast*	2417	14	103$*$	Biology
4	*TMC2007*	28596	22	500	Text
5	*Medical*	978	45	1449	Text
6	*Enron*	1702	53	1001	Text
7	*MediaMill*	43907	101	120$*$	Media
8	*Bibtex*	7395	159	1836	Text
9	*Delicious*	16105	983	500	Text

First, we compared BCCs against 9 different state-of-the-art methods (shown in Table 2) using the *Emotions*, *Scene* and *Yeast* datasets. Algorithm 1 is the basic Binary Relevance method [Tsoumakas and Katakis, 2007]. Algorithms 2 to 6 are methods explicitly designed for learning MBCs. Algorithms 7 and 8 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 *et al.*, 2011] (wrapper approach). Algorithm 9 is a multi-label lazy learning approach named *ML-KNN* [Zhang and Zhou, 2006], derived from the traditional K-nearest neighbor algorithm. In this experiment for the *ML-KNN* method K was set to 3 in the *Emotions* and *Scene* data sets, and 5 in the *Yeast*

[1]The data sets can be found at mulan.sourceforge.net/datasets.html, mlkd.csd.auth.gr/multilabel.html and www.cs.waikato.ac.nz /jmr30/#—datasets.

data set. Since it is unfeasible to compute the mutual information of two features given all the class variables, as required in [de Waal and van der Gaag, 2007], the implementation of the *polytree-polytree* learning algorithm uses the marginal mutual information of pairs of features. Algorithm 10 from Table 2 is our Bayesian Chain Classifier.

Table 2: Algorithms used in the experiments.

No.	*Algorithm [Reference]*
1	binary relevance [Tsoumakas and Katakis, 2007]
2	tree-tree [van der Gaag and de Waal, 2006]
3	polytree-polytree [de Waal and van der Gaag, 2007]
4	pure-filter [Bielza *et al.*, 2011]
5	pure-wrapper [Bielza *et al.*, 2011]
6	hybrid [Bielza *et al.*, 2011]
7	K2 BN [Cooper and Herskovits, 1992]
8	wrapper BN [Bielza *et al.*, 2011]
9	ML-KNN [Zhang and Zhou, 2006]
10	Bayesian Chain Classifiers (BCC)

For the purpose of comparison we used two different multidimensional performance measures [Bielza *et al.*, 2011]:

1. *Mean accuracy* (accuracy per label or per class) over the d class variables:

$$\overline{Acc_d} = \frac{1}{d}\sum_{j=1}^{d} Acc_j = \frac{1}{d}\sum_{j=1}^{d}\frac{1}{N}\sum_{i=1}^{N}\delta(c'_{ij}, c_{ij}) \quad (4)$$

   where $\delta(c'_{ij}, c_{ij}) = 1$ 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.

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

$$Acc = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{c'_i}, \mathbf{c_i}) \quad (5)$$

   where $\mathbf{c_i}$ is the d-dimensional vector of class values and $\delta(\mathbf{c'_i}, \mathbf{c_i}) = 1$ if $\mathbf{c'_i} = \mathbf{c_i}$ 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.

The estimation method for performance evaluation is 10-fold cross-validation. Results for accuracy and a comparison of the different methods are shown in Table 3. Table 4 shows the average rankings of the 9 algorithms used for comparison and that of our method. As can be seen from this table, in general, the performance of our method is better that the other methods used in these experiments.

Secondly, we performed experiments with the *TMC2007*, *Medical*, *Enron*, *MediaMill*, *Bibtex* and *Delicious* datasets. Given the complexity of these datasets, in particular in the number of classes, they can not be tested with the other methods; so in this case we compared Bayesian Chain Classifiers and Ensembles of BCCs (EBCCs). In Table 5 we show the Mean and Global accuracies per dataset for BCCs and EBCCs, respectively. We observe that for most of the datasets there is a significant improvement in mean and global accuracy with the ensemble. The number of iterations on the ensembles were set to 10. A class is determined whether positive or negative by taking the value with the higher number of votes in the ensemble.

Table 3: Performance metrics (mean ± std. deviation) and rank (in brackets) of the algorithms using 10-fold cross-validation.

Dataset	*Mean Accuracy*	*Global Accuracy*
Emotions		
binary relevance	0.7762 ±0.1667(7)	0.2860 ±0.0452(8)
tree-tree	0.8300 ±0.0151(3)	**0.3844 ±0.0398(1)**
polytree-polytree	0.8209 ±0.0243(5)	0.3776 ±0.0622(3)
pure filter	0.7548 ±0.0280(9)	0.2866 ±0.0495(7)
pure wrapper	0.8333 ±0.0123(2)	0.3708 ±0.0435(4)
hybrid	0.8210 ±0.0170(4)	0.3557 ±0.0435(5)
K2 BN	0.7751 ±0.0261(8)	0.2812 ±0.0799(9)
wrapper BN	0.7985 ±0.0200(6)	0.3033 ±0.0752(6)
ML-KNN	0.6133 ±0.0169(10)	0.0254 ±0.0120(10)
BCC	**0.8417 ±0.0231(1)**	0.3822 ±0.0631(2)
Scene		
binary relevance	0.8236 ±0.0250(2)	0.2898 ±0.0149(3)
tree-tree	0.7324 ±0.0359(9)	0.1857 ±0.0977(8)
polytree-polytree	0.7602 ±0.0663(8)	0.2643 ±0.1915(6)
pure filter	0.7726 ±0.0700(6)	**0.3067 ±0.1991(1)**
pure wrapper	0.7765 ±0.0580(4)	0.2688 ±0.1642(5)
hybrid	0.7229 ±0.0442(10)	0.1570 ±0.1018(9)
K2 BN	0.7689 ±0.0692(7)	0.2883 ±0.1995(4)
wrapper BN	0.7739 ±0.0492(5)	0.2277 ±0.1372(7)
ML-KNN	0.8196 ±0.0092(3)	0.0311 ±0.0147(10)
BCC	**0.8260 ±0.0373(1)**	0.2920 ±0.1218(2)
Yeast		
binary relevance	0.7297 ±0.2380(9)	0.0890 ±0.0242(8)
tree-tree	0.7728 ±0.0071(4)	**0.1953 ±0.0208(1)**
polytree-polytree	0.7336 ±0.0182(8)	0.1431 ±0.0258(3)
pure filter	0.7480 ±0.0119(6)	0.0989 ±0.0342(7)
pure wrapper	**0.7845 ±0.0131(1)**	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(9)
ML-KNN	0.6364 ±0.0196(10)	0.0062 ±0.0029(10)
BCC	0.7771 ±0.0147(2)	0.1616 ±0.0875(2)

Table 4: Average rank values for each algorithm from the results in Table 3.

Algorithm	*Mean Acc.*	*Global Acc.*	*Global Rank*
binary relevance	6.0000	6.3334	6.1667(7)
tree-tree	5.3334	3.3334	4.3334(3)
polytree-polytree	7.0000	4.0000	5.5000(4)
pure filter	7.0000	5.0000	6.0000(5)
pure wrapper	2.3334	4.3334	3.3334(2)
hybrid	7.0000	6.6667	6.8333(9)
K2 BN	6.6667	6.0000	6.3333(8)
wrapper BN	4.6667	7.3334	6.0000(5)
ML-KNN	7.6667	10.000	8.8333(10)
BCC	**1.3333**	**2.0000**	**1.6667(1)**

Table 5: Global and Mean accuracy results for the Bayesian Chain Classifiers (BCC) and for the Ensembles of Bayesian Chain Classifiers (EBCC).

	Mean Accuracy		*Global Accuracy*	
Dataset	*BCC*	*EBCC*	*BCC*	*EBCC*
TMC2007	0.8211	0.8666	0.2576	0.3784
Medical	0.9999	0.9999	0.9989	0.9999
Enron	0.7439	0.7931	0.0011	0.0341
MediaMill	0.6980	0.7376	0.0489	0.1382
Bibtex	0.8319	0.8518	0.0304	0.0993
Delicious	0.5197	0.5950	0.0009	0.0286

In terms of computational resources, the training and classification times for the *small* data sets (*Emotions*, *Scene* and *Yeast*) are less than one minute; and for the more complex datasets are in the order of hours[2].

6 Conclusions and Future Work

In this paper we have introduced Bayesian Chain Classifiers for multidimensional classification. The proposed approach is simple and easy to implement, and yet is highly competitive against other Bayesian multidimensional classifiers. We experimented with the simplest model for a BCC, considering a tree structure for the class dependencies and a simple naïve Bayes classifier as base classifier. In the future we will explore alternative models considering more complex dependency structures and other more powerful base classifiers. We also plan to compare our approach with other classifier chains using different metrics and datasets.

7 Acknowledgments

The authors wish to acknowledge **FONCICYT** for the support provided through Project No. 95185 (DyNaMo).

Also, this research has been partially supported by the Spanish Ministry of Science and Innovation, projects TIN2010-20900-C04-04, Consolider Ingenio 2010-CSD2007-00018 and Cajal Blue Brain.

References

[Bielza *et al.*, 2011] C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 2011.

[Boutell *et al.*, 2004] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[Cheng and Hullermeier, 2009] Weiwei Cheng and Eyke Hullermeier. Combining instance-based learning and logistic regression for multi-label classification. *Machine Learning*, 76(23):211–225, 2009.

[Cheng-Jung *et al.*, 2008] Tsai Cheng-Jung, Lee Chien-I, and Yang Wei-Pang. A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences*, (178):714–731, 2008.

[2]In a Celeron Dual-Core at 1.8 GHz with 4 GB of RAM.

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

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

[de Waal and van der Gaag, 2007] Peter R. de Waal and Linda C. van der Gaag. 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, pages 501–511, 2007.

[Dembczynski *et al.*, 2010] K. Dembczynski, W. Cheng, and E. Hullermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings ICML*, 2010.

[Qazi *et al.*, 2007] Maleeha Qazi, Glenn Fung, Sriram Krishnan, Romer Rosales, Harald Steck, R. Bharat Rao, Don Poldermans, and Dhanalakshmi Chandrasekaran. Automated heart wall motion abnormality detection from ultrasound images using bayesian networks. *International Joint Conference on Articial Intelligence*, pages 519–525, 2007.

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

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

[Shimony, 1994] S. E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410, 1994.

[Tsoumakas and Katakis, 2007] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

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

[Vens *et al.*, 2008] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classication. *Machine Learning*, 73(2):185–214, 2008.

[Zhang and Zhou, 2006] Min Ling Zhang and Zhi Hua Zhou. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[Zhang and Zhou, 2007] Min Ling Zhang and Zhi Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.