Multi-label classification with Bayesian network-based chain classifiers

L. Enrique Sucar, Concha Bielza, Eduardo F. Morales, Pablo Hernandez-Leal, Julio H. Zaragoza, Pedro Larrañaga

[a]I

In multi-label 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 labels (label power-set methods) or by building independent classifiers for each class (binary relevance methods). The first approach suffers from high computationally complexity, while the second approach ignores possible dependencies among classes. Chain classifiers have been recently proposed to address these problems, where each classifier in the chain learns and predicts the label of one class given the attributes and all the predictions of the previous classifiers in the chain. In this paper we introduce a method for chaining Bayesian classifiers that combines the strengths of classifier chains and Bayesian networks for multi-label classification. A Bayesian network is induced from data to: (i) represent the probabilistic dependency relationships between classes, (ii) constrain the number of class variables used in the chain classifier by considering conditional independence conditions, and (iii) reduce the number of possible chain orders. The effects in the Bayesian chain classifier performance of considering different chain orders, training strategies, number of class variables added in the base classifiers, and different base classifiers, are experimentally assessed. In particular, it is shown that a random chain order considering the constraints imposed by a Bayesian network with a simple tree-based structure can have very competitive results in terms of predictive performance and time complexity against related state-of-the-art approaches.

1. Introduction

In contrast with traditional (one-dimensional) classifiers, multi-label classifiers assign each instance to a set of d classes. Multi-label classification has received increasing attention in recent years as several important problems need to predict a set of multiple labels (Zhang et al., 2013; Vens et al., 2008; Zhang and Zhou, 2007), such as text classification (assigning a document to several topics), HIV drug selection (determining the optimal set of drugs), and scene classification, among others.

Two main types of approaches have been proposed for multi-label classification: *binary relevance* and *label power-set*. In the binary relevance approach (Zhang and Zhou, 2007), the multi-label classification 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 and the results are combined to determine the predicted class vector. 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.

In the label power-set approach (Tsoumakas and Katakis, 2007), the multi-label classification problem is transformed into a single-class scenario by defining a new compound class variable whose possible values are all the possible combinations of values of the original classes. In this case, the interactions between classes are implicitly considered and can be an effective approach for domains with a few class variables. Its main drawback, however, is its computational complexity, as the size of the compound class variable increases exponentially with the number of classes.

To overcome the limitations of previous methods, two main strategies have been proposed within the field of probabilistic

graphical models: (i) to incorporate class interactions in binary relevance methods, in what are known as *chain classifiers* (Dembczynski et al., 2010; Read et al., 2009), and (ii) to explicitly represent the dependence structure between the classes, avoiding the combinatorial explosion of the label power-set approach, via *multi-dimensional Bayesian network classifiers* (Bielza et al., 2011; van der Gaag and de Waal, 2006; Zaragoza et al., 2011a).

Chain classifiers (Dembczynski et al., 2010, 2012; Read et al., 2009, 2011) consist of d base classifiers which are linked in a chain, such that each classifier incorporates the classes predicted by the previous classifiers as additional attributes. In this way the class interactions are incorporated while maintaining an efficiency close to the binary relevance method. The order of the classes considered in the chain can affect the final results and usually an ensemble of random orders is used, which is computationally expensive. Another potential drawback of this technique is that the number of attributes increases with the number of classes, and it can become problematic for certain domains.

A multi-dimensional Bayesian network classifier (Bielza et al., 2011; de Waal and van der Gaag, 2007; van der Gaag and de Waal, 2006; Zaragoza et al., 2011a) is a Bayesian network (BN) of restricted topology designed to solve multi-dimensional (and also multi-label) classification problems. It consists of three subgraphs, one for the class variables, one for the feature variables, and a *bridge* structure that interconnects the class and feature subgraphs allowing only arcs from classes to features. Although several alternatives have been proposed to learn these substructures (Bielza et al., 2011; Rodríguez and Lozano, 2008), it suffers from the high computational cost of determining the optimal network structure, and computing the most probable explanation for any instance with unknown values for the classes.

Bayesian Chain Classifiers (BCC) (Zaragoza et al., 2011b) 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) obtain a dependency structure for the class variables, and (ii) based on the dependency structure, build a chain classifier. In the first phase, a BN that represents the probabilistic dependency relations between the class variables is learned from data. This class structure serves as a guide for the second phase, as it constrains the possible variable orderings in the chain and reduces the number of classes considered in the chain classifier, by considering the independence conditions of the Bayesian network. Finally, as for chain classifiers, the predicted class vector is obtained by concatenating the outputs of all the classifiers in the chain.

In Zaragoza et al. (2011b) it was shown that a simple BCC with a tree structure in the first phase, and a random class order consistent with the tree, using naïve Bayes as base classifier in the chain, was able to outperform several state-of-the-art multi-dimensional Bayesian network classifiers on several testbed problems with a lower time complexity. This paper extends the approach in Zaragoza et al. (2011b) presenting a deeper analysis to get insights into the BCC behavior. We perform an extensive empirical evaluation on alternative strategies for building BCCs varying several aspects:

1. Different training schemes.
2. Several heuristics to define the order of the chain.
3. Different number of classes incorporated in each classifier in the chain.
4. Alternative base classifiers.
5. Single chain vs. ensembles.

We also compare experimentally BCCs with binary relevance and standard chain classifiers (Read et al., 2009; Read et al., 2011). All the experiments are carried out over 9 benchmark multi-label data sets using four different performance metrics.

The main contribution of this paper is the proposal and analysis of several extensions to the *basic BCC* classifier introduced in Zaragoza et al. (2011b). This work opens a new research avenue for multi-label classification research as considering dependencies among classes is clearly beneficial.

The paper is organized as follows. Section 2 describes the multi-label classification problem. Section 3 reviews related work. Bayesian chain classifiers are introduced in Section 4. In Section 5, we analyze alternative configurations for Bayesian chain classifiers. Section 6 describes the experiments and discusses the results. Section 7 summarizes the main ideas of the paper and provides future research directions.

2. Multi-label classifiers

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 $\mathbf{c} = (c_1, \ldots, c_d)$ of the d dimensional class variable $(C_1, \ldots, C_d)$:

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

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

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:

$$\arg\max_{c_1,\ldots,c_d} P(C_1 = c_1, \ldots, C_d = c_d | \mathbf{x}) \tag{1}$$

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).

In this work, we consider the *multi-label classification* problem, which can be seen as a particular case of a multi-dimensional classification, where all class variables are binary, that is $|\Omega_{C_i}| = 2$ for $i = 1, \ldots, d$.

3. Related work

In this section we briefly review the main approaches that have been proposed for multi-label classification. The review is organized into three subsections, discussing research in multi-label classification, multi-dimensional Bayesian network classifiers, and chain classifiers, respectively.

3.1. Multi-label classification

As mentioned before, there are two basic approaches for multi-label classification: *binary relevance* and *label power-set* (Tsoumakas and Katakis, 2007). Binary relevance approaches transform the multi-label classification problem into d independent binary classification problems, one for each class variable, $C_1, \ldots, C_d$. A classifier is independently learned for each class and the results are concatenated to determine the predicted class vector; the dependencies between classes are not considered. The label power-set approach transforms the multi-label classification problem into a single-class scenario by defining a new compound class variable whose possible values are all the possible combinations of values of the original classes. In this case the interactions between classes are implicitly considered and can be an effective approach for domains with a few class variables; however for many classes this approach is impractical.

An overview of multi-label classification is presented in Tsoumakas and Katakis (2007), where two main methods 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.

3.2. Multi-dimensional Bayesian network classifiers

A multi-dimensional Bayesian network classifier (MBC) over a set $\mathcal{V} = \{Z_1, \ldots, Z_n\}, n \geqslant 1$, 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 $\mathbf{Pa}(Z)$ of parents variables of Z in $\mathcal{G}$. B defines a joint probability distribution P_B over $\mathcal{V}$ given by:

$$P_B(z_1, \ldots, z_n) = \prod_{i=1}^{n} P_B(z_i|\mathbf{pa}(z_i)) \tag{2}$$

The set $\mathcal{V}$ of vertexes is partitioned into two sets $\mathcal{V}_\mathcal{C} = \{C_1, \ldots, C_d\}, d \geqslant 1$, of class variables and $\mathcal{V}_\mathcal{X} = \{X_1, \ldots, X_m\}$, $m \geqslant 1$, 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 (see Fig. 1).

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. In de Waal and van der Gaag (2007) they analyze the conditions for the optimal recovery of *poly-tree* structures in both subgraphs.

Rodríguez and Lozano (2008) extend poly-trees to k-DB structures for class and features subgraphs.

Bielza et al. (2011) describe a general model in which any Bayesian network structure is allowed in the three subgraphs. Learning from data algorithms cover many possibilities: wrapper, filter and hybrid scores with different search strategies. Direct algorithms for learning these simpler MBCs, both from a wrapper point of view (Borchani et al., 2010) and from a filter Markov blanket-based perspective (Borchani et al., 2012) have been recently proposed. In Zaragoza et al. (2011a), the authors introduce a two-step method for learning multi-dimensional Bayesian network classifiers based on mutual information or dependency between the classes and the features variables.

3.3. Chain classifiers

Read et al. (2009) introduce chain classifiers as an alternative method for multi-label classification that incorporates class dependencies, while keeping the computational efficiency of the binary relevance approach. A chain classifier consists of d base binary classifiers which are linked in a chain, such that each classifier incorporates the classes predicted by the previous classifiers as additional attributes. Thus, the feature vector for each binary classifier is extended with the class values (labels) of all previous classifiers in the chain. Each classifier in the chain is trained to learn the association of label C_i given the features augmented with all previous class labels in the chain, $c_1, c_2, \ldots, c_{i-1}$. At classification time, the process starts at C_1, and propagates the predicted classes along the chain such that for c_i it computes $\arg\max_{c_i} P(c_i|\mathbf{x}, c_1, c_2, \ldots, c_{i-1})$. As in the binary relevance approach, the class vector is determined by concatenating the outputs of all the binary classifiers in the chain.

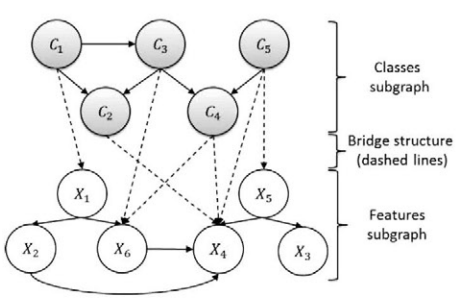

Fig. 1. A multi-dimensional Bayesian network classifier structure, showing the three subgraphs: classes, features and bridge.

In Read et al. (2009), the authors use 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 c_i receives a number of votes from the m chain classifiers, and a threshold on this number is used to determine the final predicted multi-label set. They used support vector machines as base binary classifier, and evaluate experimentally their method with 12 multi-label data sets, comparing it with binary relevance and other ensemble algorithms. The classifier chains outperformed binary relevance in terms of accuracy for most data sets, with some increase in training time. However the results were not always the best in terms of accuracy of their ensemble against other ensemble methods, with some advantage in training and classifications times, as expected.

Recently, Read et al. (2011) have presented several extensions of their previous work. In particular, they propose some improvements to make the ensemble learning process more efficient, such as taking random subsets of attributes and instances. The experimental results show a significant reduction on running time with almost the same accuracy. They also present additional experiments and compare chain classifiers and ensembles of chain classifiers with alternative techniques.

Dembczynski et al. (2010) introduce *probabilistic chain classifiers* (PCCs), by basically putting chain classifiers under a probabilistic framework. Using the chain rule of probability theory, 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:

$$P(\mathbf{c}|\mathbf{x}) = P(c_1|\mathbf{x}) \prod_{i=2}^{d} P(c_i|c_1, \ldots, c_{i-1}, \mathbf{x}). \tag{3}$$

A 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 class variables.

They analyze different scoring functions, and argue that for certain loss functions considering class dependencies can be important, and not for others, as confirmed by their experiments with artificial data. For independent classes, the results in terms of certain loss functions are almost the same for binary relevance, chain and probabilistic chain classifiers; while for dependent classes, binary relevance has competitive performance for certain loss functions, and it is clearly outperformed for certain loss functions by the other methods (that consider label dependencies). Their experiments with artificial and benchmark data sets show that PCC and its corresponding ensemble, EPCC, have a better performance than the chain classifier and the ensemble of chain classifiers, for some loss scoring functions.

In Zhang and Zhang (2010), $P(\mathbf{c}|\mathbf{x})$ is decomposed according to a Bayesian network: $P(\mathbf{c}|\mathbf{x}) = \prod_{i=1}^{d} P(c_i|\mathbf{pa}(c_i), \mathbf{x})$. Finding $\mathbf{pa}(c_i)$ is complex under the setting of (many) continuous features, whose effect on the labels may be nonlinear. Thus, this is simplified as follows. First, d classifiers are built for all labels independently,

from a nonlinear regression model of $\mathbf{x}$ over c_i, whose output is thresholded to yield the predicted labels. Second, from the corresponding errors for each label, a Bayesian network structure is learnt. The links found here are incorporated as $\mathbf{pa}(c_i)$ in the sought Bayesian network. Finally, a PCC is implemented according to an order implied by the network.

As shown in Dembczynski et al. (2010), a method that considers class dependencies under a probabilistic framework can have a significant impact on the performance of multi-label 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 while being very efficient at the same time. Unlike Zhang and Zhang (2010), we directly work on the original class variables (and not over the regression errors), where we find a simple tree structure. The feature variables are assumed to be discrete. We allow to have ensembles of classifiers.

4. Bayesian chain classifiers

In this section we consider the multi-dimensional classification problem under a Bayesian network framework; and in particular we analyze the assumptions implied by a Bayesian chain classifier approximation.

If we apply the chain rule of probability theory, we can rewrite Eq. (1) as:

$$\arg\max_{c_1,\dots,c_d} P(c_1|c_2,\dots,c_d,\mathbf{x})P(c_2|c_3,\dots,c_d,\mathbf{x})\cdots P(c_d|\mathbf{x}) \tag{4}$$

If we assume we can represent the joint probability distribution of the class variables given the features as a Bayesian network, then we can simplify Eq. (1) by considering the independencies implied by the Bayesian network; so that only the *parents* of each class variable are included in the chain, and all other *previous* classes according to the chain order are eliminated. So we can write Eq. (4) as:

$$\arg\max_{c_1,\dots,c_d}\prod_{i=1}^{d} P(c_i|\mathbf{pa}(C_i),\mathbf{x}) \tag{5}$$

where $\mathbf{Pa}(C_i)$ are the parents of class i in the Bayesian network.

Next we make a further simplification by assuming that the most probable joint combination of classes can be approximated by just concatenating the individual most probable classes from the base classifier. That is, we solve the following set of equations as an approximation of Eq. (1):

$$\begin{aligned}
&\arg\max_{c_1} P(c_1|\mathbf{pa}(C_1),\mathbf{x})\\
&\arg\max_{c_2} P(c_2|\mathbf{pa}(C_2),\mathbf{x})\\
&\dots\\
&\arg\max_{c_d} P(c_d|\mathbf{pa}(C_d),\mathbf{x})
\end{aligned}$$

This last approximation corresponds to a Bayesian chain classifier. Thus, a BCC makes two basic assumptions:

1. The class dependency structure given the features can be represented by a Bayesian network.
2. The most probable joint combination of class assignment (total abduction) is approximated by the concatenation of the most probable individual classes.

The first assumption is reasonable if we have enough data to obtain a good approximation of the class dependency structure, and assuming that this is obtained conditioned on the features. With respect to the second assumption, it is well known that the total abduction or most probable explanation is not always equivalent to the maximization of the individual classes. However, this assumption, also considered by chain classifiers and PCCs, is less strong than that assumed by the binary relevance approach.

In this setting, a chain classifier can be constructed by inducing first the class that does not depend on any other class and then proceed with its children. We can:

- Create an (partial) order of classes in the chain based on the dependencies between classes given the features. Assuming that these dependencies can be represented as a BN, the chain structure complies with the structure of the BN, such that we can then start building base classifiers for the classes without parents, and continue with their children classes, and so on. We can further simplify the problem by considering the marginal dependencies between classes as a first approximation (without conditioning on the features) to obtain an order for the chain classifier, and then induce base classifiers considering such an order.
- Consider conditional independencies between classes to create simpler base 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.

The general idea for building a BCC is illustrated in Fig. 2.

We first introduce the simplest option to build a BCC, and then present several possible extensions.

4.1. Tree naïve Bayesian chain classifier

The simplest Bayesian chain classifier considers only one parent per class in a chain. 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 spanning tree* (*MWST*). This algorithm builds the structure that maximizes the likelihood of the data over all possible trees. The weights are computed as the mutual information between pairs of variables.

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

For d classes we can build d trees. Then we can choose an ancestral ordering from each tree and build a chain classifier. Finally, we

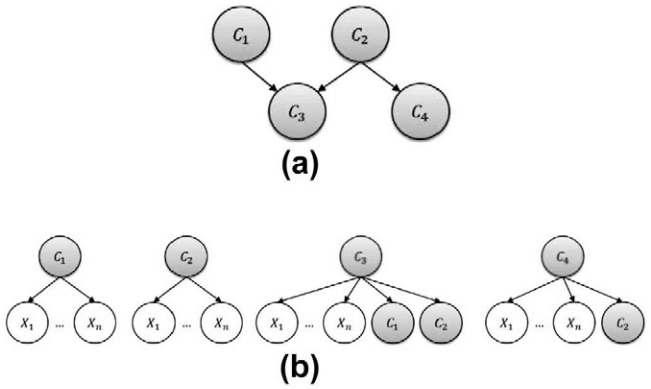

Fig. 2. An example of a BCC: (a) a BN that represents the class dependency structure; (b) set of naive Bayes classifiers, one for each class. Each base classifier defined for C_i includes the set of attributes, $X_1,\dots,X_n$, plus its parents in the BN as additional attributes.

can combine the chain classifiers in an ensemble (if d is very large we can limit the number of chains by selecting a random subset of trees).

Once the dependencies between classes have been taken into account to generate a chain classifier, we only need to define the base classifier. Our baseline approach is to use a naïve Bayes classifier, see Fig. 3. We call this approach a *tree naïve Bayesian chain classifier* (TNBCC). Each naïve Bayes classifier for C_i has as attributes all the features and also $\mathbf{Pa}(C_i)$ according to the BN structure. Therefore for each base classifier we solve $\arg\max_{c_i} P(c_i|\mathbf{pa}(C_i), \mathbf{x})$, which shapes the chain classifier (as in Eq. (5)).

We can summarize the algorithm to build the TNBCC as follows. Given a multi-label classification problem with d classes:

1. Build an undirected tree to approximate the dependency structure among class variables.
2. Create an order for the chain classifier by randomly selecting one class as the root of the tree and assigning the rest of the links in order.
3. For each class variable (node) in the chain, build a naïve Bayes classifier for class C_i which has as attributes its parent $\mathbf{Pa}(C_i)$ and all the features $\mathbf{x}$, taking advantage of the conditional independence properties.
4. To classify a new instance concatenate the outputs of the chain.

This is a very fast and easy way to build chain classifiers, which represents the simplest alternative for a BCC. Other, more complex alternatives, are explored in the following section.

5. Alternative Bayesian chain classifiers

The basic, tree naïve Bayesian chain classifier can be extended in different ways. We consider five dimensions:

1. The training scheme.
2. The order in which the classes are considered in the chain.
3. The number of classes in each base classifier (chain complexity).
4. The base classifier.
5. Whether to use one chain or an ensemble of chains.

In the following sections we analyze each one in detail.

5.1. Training scheme

There are at least two options for training chain classifiers:

- The first approach considers the predicted output of the previous classifiers as training input for the next classifier in the chain. The rationale is to build a new classifier considering what will be the "actual" output of the first classifiers during the testing phase. The disadvantage of this approach is that a poor classifier will tend to produce erratic predictions and consequently affect the subsequent classifiers.
- The second approach constructs classifiers in the chain considering the actual class values given in the original training set. The rationale is that using the "real" value (as opposed to the predicted value by the previous classifiers) will tend to produce more accurate classifiers.

Section 6.4 describes the performance results for both approaches.

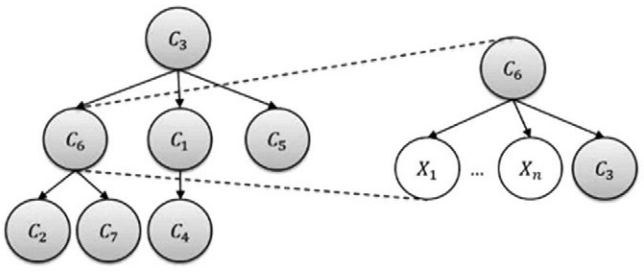

Fig. 3. An example of a Tree Naïve Bayesian Chain Classifier where each node (C_6, for instance) in the chain yields a naïve Bayesian classifier which has as attributes its parent class (C_3) and all the features ($X_1, \ldots, X_n$).

5.2. Order of classes in the chain

For a tree naïve BCC, where we have a tree-based structure for the classes, there are different ways of deciding which node to be selected as the root of the tree, from which the chain order can be directly obtained. Here we propose several alternatives:

- Random choice. A class node is randomly selected as the root of the tree. For example, in Fig. 3, C_3 is selected to produce the tree to the left.
- Select the node with the largest number of incident edges. In Fig. 3 it could be C_3 or C_6, both of which have three associated links.
- Construct a base classifier for each class independently and select the class with the best classification accuracy as the root of the tree.
- Construct a base classifier for each class independently, sort the classes according to their classification accuracy and use that order for the chain regardless of the Bayesian network.

In Section 6.4 we describe the results of tests with different options for the chain order.

5.3. Complexity of the chain

The main idea of constructing a (class) Bayesian network representing the dependencies among classes is to restrict the number of classes to consider in the base classifiers to only those related to each class and to help to choose a suitable chain order. The number of classes considered for each classifier depends on the structure of the class Bayesian network.

- The simplest approach is to consider a single parent in a tree-based structure (restricting the class Bayesian network structure to a tree), which is the approach taken by the tree naïve BCC.
- By still working with a tree structure, an alternative approach is to consider all the class ancestors in the tree as input for the next classifier, providing additional information from indirectly related classes.
- Other approach is to decide on a traversing order of the class Bayesian network or choosing a random order of classes as suggested in Read et al. (2009), and using all the previous classes in the classifier chain.
- Alternatively more complex class structures could be built, such as polytrees or multi-connected networks. In both cases, each base classifier could have several parents which would be incorporated as additional attributes in the base classifiers. An example is shown in Fig. 2.

For multi-label classification problems where there can be a large number of classes, it is important to see if considering a small subset of related classes can produce competitive results against a more expensive strategy that uses all the previous classes in the chain. We test these alternatives in Section 6.3.3.

5.4. Base classifier

There are many different choices for constructing each classifier of the chain. So far, the same classifier has been used for all the classifiers of the chain, but nothing prevents us from using different classifiers along the chain. Among the possible choices, TNBCC uses a naïve Bayes classifier, which has the advantage of being simple and easy to implement. In this paper we also use support vector machines to assess the effect on the predictive performance of a generally stronger, although computationally more expensive, classifier. A comparison of these base classifiers is described in Section 6.3.4.

5.5. One chain vs. an ensemble of chains

Lastly, there is the choice of whether to use a single chain or an ensemble of chain classifiers. Ensembles have proved to be an effective mechanism to improve the performance metrics of simple classifiers at the expense of using more computational resources. Due to the nature of several multi-label problems (large number of data, attributes and classes), it is relevant to determine whether finding a good chain order could produce equivalent performance results to an ensemble of random chain orders, with significant savings in computational resources. This is analyzed in Section 6.4.

6. Empirical evaluation

In this section we empirically evaluate different choices for building BCCs and compare them against other state-of-the-art multi-label classifiers.

6.1. Data sets

Different Bayesian chain classifiers were tested on 9 benchmark multi-label data sets[1]; each of them with different dimensions ranging from 6 to 983 labels, and from about 600 examples to more than 43,000. All class variables of the data sets are binary, however, in some of the data sets 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 data sets are summarized in Table 1.

6.2. Evaluation metrics

Several metrics have been recently proposed to evaluate the performance of multi-label classifiers. They basically range from considering the performance of the multi-label classifier over each class independently of the rest, to considering the performance of all the classes at the same time. For the purpose of comparison we used four different multi-label evaluation measures (Bielza et al., 2011; Read et al., 2009):

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

$$\mathrm{M}-\mathrm{Acc}=\frac{1}{d}\sum_{j=1}^{d}Acc_j=\frac{1}{d}\sum_{j=1}^{d}\frac{1}{N}\sum_{i=1}^{N}\delta(c'_{ij},c_{ij}) \tag{6}$$

   where $\delta(c'_{ij},c_{ij})=1$ if $c'_{ij}=c_{ij}$ and 0 otherwise, and c'_{ij} denotes the C_j class value outputted by the model for instance i and c_{ij} is its true value.
2. *Global accuracy* over the d-dimensional class variable (accuracy per example, also called subset zero-one loss):

$$\mathrm{G}-\mathrm{Acc}=\frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{c}'_i,\mathbf{c}_i) \tag{7}$$

   where $\delta(\mathbf{c}'_i,\mathbf{c}_i)=1$ if $\mathbf{c}'_i=\mathbf{c}_i$ and 0 otherwise. Therefore, we call for a total coincidence on all the components of the vector of predicted classes $\mathbf{c}'_i$ and the vector of real classes $\mathbf{c}_i$.
3. *Multi-label accuracy*, also called Jaccard measure, as defined in Tsoumakas and Katakis (2007):

$$\mathrm{ML}-\mathrm{Acc}=\frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{c}_i\wedge\mathbf{c}'_i|}{|\mathbf{c}_i\vee\mathbf{c}'_i|} \tag{8}$$

   where in the numerator we count the number of coincidences of the two vectors (real and predicted), and in the denominator we count the number of labels covered by some of both vectors.
4. *F-measure* is the harmonic mean between precision and recall:

$$\mathrm{F}-\mathrm{measure}=\frac{1}{d}\sum_{j=1}^{d}\frac{2p_jr_j}{(p_j+r_j)} \tag{9}$$

where p_j and r_j are the precision and recall for C_j. Here, the F-measure is calculated per label and then averaged.

Table 1
Multi-label data sets used in the experiments and associated statistics. N is the size of the data set, d is the number of binary classes or labels, m is the number of features. + indicates that there are numeric attributes.

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

6.3. Experiments and results

In Zaragoza et al. (2011b) it was shown that the simplest BCC, a TNBCC, was able to outperform nine state-of-the-art multi-dimensional Bayesian network classifiers (including: tree–tree, polytree–polytree, pure-wrapper, pure-filter and hybrid, among others) on several testbed problems with a significantly lower time complexity. In this paper, we perform several experiments to evaluate different variants of BCCs, and compared them against other multi-label classifiers, such as binary relevance and chain classifiers:

1. TNBCC against the binary relevance method (Section 6.3.1).
2. Different chain orders (Section 6.3.2) and chain complexities (Section 6.3.3).
3. One vs. all previous classes in the tree incorporated in each base classifier (Section 6.3.3).
4. Different base classifiers (Section 6.3.4).
5. Different heuristics for selecting the root node (Section 6.4).
6. Different training techniques for chain classifiers (Section 6.4).
7. A single chain versus an ensemble of chains (Section 6.4).

We used 10-fold cross-validation for the five smaller data sets and 3-fold cross-validation for the larger data sets. We repeated this process 10 times for the smaller data sets and 5 times for the larger ones and reported the average results. We performed statistical significance tests using a t-test for the five smaller data sets. For the larger data sets, however, we used Wilcoxon rank sum test, since we performed only a small number of runs. In both cases

[1] The data sets can be found at http://mulan.sourceforge.net/datasets.html, <http://mlkd.csd.auth.gr/multilabel.html> and <http://www.cs.waikato.ac.nz/ml/weka/index.html>.

we used $\alpha = 0.05$. If the differences in the results are statistically significant an "$*$" symbol is shown in the tables. In Tables 2–6 the best results for each database and for each evaluation metric are shown in bold.

We used the naïve Bayes and SVM implementations of Weka (Hall et al., 2009).

6.3.1. TNBCC against binary relevance

We start by comparing the TNBCC approach with a baseline algorithm that constructs independent classifiers for each class – binary relevance method (BRM). The results are presented in Table 2 for the four evaluation metrics.

From the table, it can be seen that in average, the TNBCC approach obtained better results than the baseline algorithm. Moreover, for most of the data sets the results are statistically significant. However, for the `Medical` and `TMC2007` data sets the independent approach obtained better results than TNBCC for some metrics. We believe that in these data sets the classes are fairly independent between each other. To corroborate this hypothesis, we constructed a (Pearson) correlation matrix between classes for all the data sets. From this analysis, we found that for the data sets `Scene`, `Medical` and `TMC2007` there is almost no correlation between classes.

6.3.2. TNBCC against a random tree and a random chain

The next experiment compares the performance of a TNBCC against a random tree that is built without considering the dependencies between classes, and also a random chain of classifiers as in Read et al. (2009). Table 3 shows that for the four different measures.

The results with the TNBCC strategy are, as expected, better than the ones using a random tree. When compared to a random chain as in Read et al. (2009) we can see that TNBCC achieves better performance results in most measures and data sets: (a) in *M-Acc*, TNBCC significatively outperforms a random chain in two data sets and is significatively beaten by a random chain in two other data sets (i.e. two wins and two losses); (b) in *G-Acc*, there are three wins vs. one loss; (c) in *ML-Acc*, we obtain three wins and two losses, and (d) in *F-measure*, four wins and no losses are found. On average, TNBCC is superior to a random tree and a random chain for three of the four measures.

Table 2
Tree naïve Bayesian chain classifiers (TNBCC) against binary relevance (BRM).

Data set	TNBCC	BRM	TNBCC	BRM
	M-Acc		*G-Acc*	
`Emotions`	**0.8478**	0.8423	**0.3922**	0.3887
`Scene`	**0.9497**$*$	0.9392	**0.7312**$*$	0.7187
`Yeast`	**0.8707**$*$	0.8646	**0.2688**	0.2593
`Medical`	**0.9756**	0.9746	0.2302	**0.2464**$*$
`Enron`	**0.7984**	0.7829	**0.0010**	0.0007
`TMC2007`	**0.8911**$*$	0.8891	0.1361	**0.1384**$*$
`Bibtex`	**0.9285**$*$	0.9126	**0.0675**$*$	0.0603
`MediaMill`	**0.6927**$*$	0.6532	0.0004	0.0000
`Delicious`	**0.8912**$*$	0.8847	0.0000	0.0000
Average	**0.8718**	0.8604	**0.2030**	0.2014
	ML-Acc		*F-measure*	
`Emotions`	**0.6677**	0.6609	**0.7661**	0.7653
`Scene`	**0.8310**	0.8276	**0.8700**$*$	0.8603
`Yeast`	**0.6620**	0.6580	**0.5718**	0.5693
`Medical`	0.3161	**0.3349**$*$	0.0719	**0.0865**$*$
`Enron`	**0.2083**	0.1957	**0.1516**	0.1470
`TMC2007`	0.4834	**0.4836**$*$	**0.4798**$*$	0.4685
`Bibtex`	**0.2004**$*$	0.1874	**0.1885**$*$	0.1838
`MediaMill`	0.0373	**0.0825**	**0.0955**$*$	0.0883
`Delicious`	**0.1433**	0.1454	**0.0880**	0.0812
Average	0.3944	**0.3973**	**0.3648**	0.3611

The results show that using information about the dependencies of the classes, even by considering a single parent, clearly benefits the performance of the classifier.

6.3.3. Number of parents for each base classifier

The next experiment compares TNBCC against a more elaborated strategy. Given that TNBCC only uses the parent node as a new attribute for the chain, in this experiment we incorporate all previous classes in the path towards the root of the tree as additional attributes. The idea is to incorporate more contextual information to each classifier, considering not only the directly relevant class but also all the indirectly related classes. We call this scheme a *path*-BCC. Table 4 presents the results of comparing the TNBCC approach and the path-BCC version.

The results show that using all the precedent classes as attributes contributes to statistically significant better results (two wins in *M-Acc*, one win in *G-Acc*, four wins in *ML-Acc* and five wins in *F-measure*), although in some data sets the best significant results are obtained when using a single parent (one win in *M-Acc*, *G-Acc* and *F-measure*, and two wins in *G-Acc*). The path-BCC is still computationally more efficient than considering all the previous classes as in a random chain classifier.

6.3.4. Different base classifiers

So far, we have used in all the experiments naïve Bayes as base classifier. In this experiment we want to evaluate the relevance of the base classifier, so we compare the results obtained by tree naïve BCC with a tree BCC that uses kernel support vector machines (TSVMBCC). We also present a comparison with a chain classifiers, including a chain with NB as base classifier (NB-CC) and a chain with SVMs (SVM-CC) as in Read et al. (2009). The results are presented in Tables 5 and 6.

From these results we can notice that using a more elaborate classifier yields better performance on average; however, as it happens with other classification problems, a simpler classifier sometimes obtains the best results. This applies to both, the tree BCCs and the chain classifiers. It should be noticed that TSVMBCC and SVM-CC tend to produce better results on the larger data sets.

An important result is that TSVMBCC is in most cases superior for all measures and data sets to the SVM-CC approach, which is basically the same as the chain classifier in Read et al. (2009) and Read et al. (2011).

6.4. Other tests

We performed other experiments, but due to space limitations, we only describe here our main results. In particular, during the training phase of a chain classifier, the learning method can use the predicted values of the previous classes in the chain or the original values as previously discussed in Section 5.1. In our experiments, in most cases training with the original data produces better results, although for the F-measure the results are very similar.

In all the previous experiments, once a tree-based structure was built with Chow and Liu's algorithm, the root node was randomly selected. We tested different strategies for selecting the root node and found that using an ordered derived from the dependencies of the classes is relevant; however, once the tree-structure is obtained, there is no significant difference between which node to select as root.

We also compared the results of tree naïve BCC and an ensemble of ten TNBCCs, denominated ETNBCC, with different roots in the tree selected randomly and found that ETNBCCs performs better than the single TNBCC specially in the larger data sets. The difference, however, is not very large considering that it is, in this case, ten times slower.

Table 3
Experimental comparison between TNBCC, a random tree (Random Tree), and a random chain (Random Chain). "∗" means significant difference between TNBCC and Random Tree and "†" means significant difference between TNBCC and Random Chain. (Results are not reported for *Delicious* because the Random Chain did not finish after one week.)

Data set	TNBCC	Random Tree	Random Chain	TNBCC	Random Tree	Random Chain
	M-Acc			*G-Acc*		
Emotions	**0.848**	0.845	0.843	**0.392**†	0.388	0.376
Scene	0.950	0.949	**0.953**	0.731	0.725	**0.746**†
Yeast	**0.871**∗†	0.862	0.856	**0.269**	0.263	0.242
Medical	**0.976**	0.974	0.975	**0.230**†	0.228	0.182
Enron	0.798	0.796	**0.812**†	0.001	0.001	0.001
TMC2007	**0.891**∗†	0.890	0.883	0.136†	**0.144**∗	0.122
Bibtex	0.929∗	0.927	**0.938**	**0.067**	**0.067**	0.065
MediaMill	0.692	0.699	**0.714**†	0.000	0.000	0.000
Average	0.869	0.868	**0.872**	**0.228**	0.227	0.217
	ML-Acc			*F-measure*		
Emotions	**0.668**	0.665	0.655	**0.766**	0.762	0.756
Scene	**0.831**	0.828	0.824	0.870	0.868	**0.871**
Yeast	**0.662**∗†	0.647	0.631	**0.572**∗†	0.550	0.521
Medical	**0.316**†	0.314	0.253	**0.072**†	0.070	0.053
Enron	0.208	0.205	**0.218**†	**0.152**	0.150	0.147
TMC2007	0.483†	**0.485**	0.461	**0.480**∗†	0.474	0.458
Bibtex	**0.200**∗†	0.198	0.186	**0.189**∗†	0.181	0.161
MediaMill	0.030	0.010	**0.095**†	**0.092**	0.086	0.089
Average	**0.425**	0.419	0.415	**0.399**	0.393	0.382

Table 4
Experimental comparison of TNBCC and path-BCC.

Data set	TNBCC	Path-BCC	TNBCC	Path-BCC
	M-Acc		*G-Acc*	
Emotions	0.848	**0.851**	0.392	**0.399**
Scene	0.950	**0.951**	0.731	**0.738**
Yeast	0.871	**0.885**∗	0.269	**0.273**
Medical	**0.976**	0.974	0.230	**0.271**∗
Enron	**0.798**	0.796	0.001	0.001
TMC2007	**0.891**∗	0.887	**0.136**∗	0.128
Bibtex	0.928	**0.931**	**0.068**∗	0.067
MediaMill	0.692	**0.848**∗	0.000	0.000
Delicious	0.891	**0.906**	0.000	0.000
Average	0.872	**0.892**	0.203	**0.209**
	ML-Acc		*F-measure*	
Emotions	0.668	**0.679**	0.766	**0.775**
Scene	0.831	**0.845**∗	0.870	**0.875**
Yeast	0.662	**0.697**∗	0.572	**0.652**∗
Medical	0.316	**0.371**∗	0.072	**0.088**∗
Enron	0.208	0.208	0.152	**0.162**∗
TMC2007	**0.483**∗	0.470	**0.480**∗	0.474
Bibtex	**0.200**	0.198	0.189	**0.212**∗
MediaMill	0.030	**0.083**∗	0.092	**0.226**∗
Delicious	**0.143**	0.138	**0.088**	0.084
Average	0.394	**0.410**	0.364	**0.394**

Table 5
Mean accuracy and global accuracy of TNBCC, TSVMBCC (BCC with support vector machine as base classifier), NB-CC and SVM-CC (chain classifier as in Read et al. (2009) with naive Bayes and support vector machine as base classifiers). "∗" means significant difference between TNBCC and SVM-CC, "†" means significant difference between TSVMBCC and SVM-CC, "§" means significant difference between TNBCC and NB-CC and "‡" means significant difference between TSVMBCC and NB-CC. (Results are not reported for Mediamill and Delicious because the SVMs did not finish after one week.)

Data set	TNBCC	TSVMBCC	NB-CC	SVM-CC
M-Acc				
Emotions	**0.848**∗	0.843	0.843	0.826
Scene	0.950	0.929	**0.953**‡	0.944
Yeast	**0.871**∗§	0.856†	0.856	0.807
Medical	0.976	**0.991**	0.975	0.989
Enron	0.798	**0.942**‡	0.812§	0.940∗
TMC2007	0.891§	**0.944**‡	0.883	0.943∗
Bibtex	0.928	**0.985**	0.938	**0.985**∗
Mediamill	0.692	–	**0.714**§	–
Delicious	0.891	–	–	–
Average	0.895	**0.927**	0.872	0.919
G-Acc				
Emotions	**0.392**∗§	0.373	0.376	0.348
Scene	0.731	0.665	**0.746**§‡	0.724†
Yeast	**0.269**∗	0.178	0.242‡	0.149
Medical	0.230§	**0.681**‡	0.182	0.651∗
Enron	0.001	**0.124**‡	0.001	0.108∗
TMC2007	0.136	**0.313**‡	0.122	0.311∗
Bibtex	0.068	**0.151**‡	0.065	0.150∗
Mediamill	0.000	–	0.000	–
Delicious	0.000	–	–	–
Average	0.261	**0.355**	0.217	0.349

6.5. Discussion

From the experiments we can draw the following conclusions:

- Finding dependencies among classes guides the chaining process and achieves better evaluation performance, even with a simple tree-based structure.
- The basic TNBCC is competitive with state of the art chain classifiers and at the same time very efficient.
- Once the dependency structure among classes is defined, choosing a particular root node is apparently not relevant.
- Incorporating information of indirectly relevant classes does make a difference. Although it requires more computational resources it is still less expensive than a chain classifier that considers all the previous classes in the chain.
- It is not always necessary to build a complete chain classifier as for some domains the classes are independent between each other.
- Stronger base classifiers produce stronger BCCs.
- A tree BCC with SVMs as base classifiers has in average superior performance than the standard chain classifier that includes all previous classes in the chain as in Read et al. (2009).
- Ensembles of tree naïve BCCs (ETNBCCs) appear to perform better than single TNBCCs.

Table 6
Multilabel accuracy and F-measure of TNBCC, TSVMBCC (BCC with support vector machine as base classifier), NB-CC and SVM-CC (chain classifier as in Read et al. (2009) with naive Bayes and support vector machine as base classifiers). "∗" means significant difference between TNBCC and SVM-CC, "†" means significant difference between TSVMBCC and SVM-CC, "§" means significant difference between TNBCC and NB-CC and "‡" means significant difference between TSVMBCC and NB-CC. (Results are not reported for `Mediamill` and `Delicious` because the SVMs did not finish after one week.)

Data set	TNBCC	TSVMBCC	NB-CC	SVM-CC
ML-Acc				
`Emotions`	**0.668**∗	0.630	0.655	0.594
`Scene`	**0.831**∗	0.721	0.824‡	0.759
`Yeast`	**0.662**∗§	0.616	0.631	0.534
`Medical`	0.316§	**0.771**†‡	0.253	0.734
`Enron`	0.208	**0.424**‡	0.218§	0.389
`TMC2007`	0.483	**0.605**‡	0.461	0.598
`Bibtex`	0.200§	**0.339**‡	0.186	0.316
`Mediamill`	0.030	–	0.095	–
`Delicious`	0.143	–	–	–
Average	0.481	**0.586**	0.415	0.561
F-measure				
`Emotions`	**0.766**∗	0.739	0.756	0.706
`Scene`	0.870	0.794	**0.871**‡	0.833
`Yeast`	**0.572**∗§	**0.542**†	0.521	0.470
`Medical`	0.072§	**0.406**‡	0.053	**0.377**∗
`Enron`	0.152	**0.246**‡	0.147	0.201
`TMC2007`	0.480§	**0.612**‡	0.458	**0.576**∗
`Bibtex`	0.189§	**0.354**†‡	0.161	0.300
`Mediamill`	0.092	–	0.089	–
`Delicious`	0.088	–	–	–
Average	0.443	**0.528**	0.382	0.495

7. Conclusions and future work

In this paper we have introduced Bayesian chain classifiers for multi-label classification problems. 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. The proposed approach is simple and easy to implement, and yet is highly competitive against multidimensional Bayesian network classifiers (as shown in Zaragoza et al. (2011b)) and also against chain classifiers. In this paper, we extend our previous work with a more thorough analysis of BCCs and considering several alternative strategies for building them.

It is shown that inducing an undirected tree and randomly picking a class as root of this tree is enough to produce competitive results, both in terms of accuracy and time complexity, against other state-of-the-art algorithms. We proposed and analyzed experimentally several alternatives to the basic BCC, showing that some extensions do make a difference in performance with a small increment in complexity; while other do not have a significant impact.

BCC opens a new research avenue for multi-label classification research as considering dependencies among classes is clearly beneficial, as shown in this paper.

As future work we will explore alternative models considering more complex dependency structures, identifying independent classes to simplify the chaining process, and more alternatives on how to incorporate other related classes to improve the performance results.

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 Economy and Competitiveness, projects TIN2010-20900-C04-04, Consolider Ingenio 2010-CSD2007-00018 and Cajal Blue Brain.

References

Bielza, C., Li, G., Larrañaga, P., 2011. Multi-dimensional classification with Bayesian networks. International Journal of Approximate Reasoning 52, 705–727.

Borchani, H., Bielza, C., Larrañaga, P., 2010. Learning CB-decomposable multi-dimensional Bayesian network classifiers. In: Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM'10), pp. 25–32.

Borchani, H., Bielza, C., Martínez-Martín, P., Larrañaga, P., 2012. Markov blanket-based approach for learning multi-dimensional Bayesian network classifiers: an application to predict the European quality of life-5dimensions (EQ-5D) from the 39-item Parkinson's disease questionnaire (PDQ-39). Journal of Biomedical Informatics 45, 1175–1184.

Cheng-Jung, T., Chien-I, L., Wei-Pang, Y., 2008. A discretization algorithm based on class-attribute contingency coefficient. Information Sciences 178 (3), 714–731.

Dembczynski, K., Cheng, W., Hüllermeier, E., 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). Omnipress, pp. 279–286.

Dembczynski, K., Waegeman, W., Hüllermeier, E., 2012. An analysis of chaining in multi-label classification. In: European Conference on Artificial Intelligence, pp. 294–299.

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

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I., 2009. The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter 11 (1), 10–18.

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2009. Classifier chains for multi-label classification. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD). Lecture Notes in Computer Science, vol. 5782. Springer, pp. 254–269.

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2011. Classifier chains for multi-label classification. Machine Learning 85, 333–359.

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

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

Tsoumakas, G., Katakis, I., 2007. Multi-label classification: an overview. International Journal of Data Warehousing and Mining 3 (3), 1–13.

van der Gaag, L.C., de Waal, P.R., 2006. Multi-dimensional Bayesian network classifiers. In: Third European Conference on Probabilistic Graphical Models, pp. 107–114.

Vens, C., Struyf, J., Schietgat, L., Dz-eroski, S., Blockeel, H., 2008. Decision trees for hierarchical multi-label classification. Machine Learning 73 (2), 185–214.

Zaragoza, J.C., Sucar, L.E., Morales, E.F., 2011a. A two-step method to learn multidimensional Bayesian network classifiers based on mutual information measures. In: Proceedings of the 24th International Florida Artificial Intelligence Research Society Conference (FLAIRS). AAAI Press, pp. 644–649.

Zaragoza, J.H., Sucar, L.E., Morales, E.F., Bielza, C., Larrañaga, P., 2011. Bayesian chain classifiers for multidimensional classification. In: International Joint Conference on Artificial Intelligence, pp. 2192–2197.

Zhang, M.-L., Zhang, K., 2010. Multi-label learning by exploiting label dependency. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 999–1008.

Zhang, M.L., Zhou, Z.H., 2007. ML-KNN: a lazy learning approach to multi-label learning. Pattern Recognition 40 (7), 2038–2048.

Zhang, M.-L., Zhou, Z.-H., 2013, in press. A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering 99. http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.39.