Trading off speed and accuracy in multilabel classification

Giorgio Corani[a], Alessandro Antonucci[a], Denis Mauá[b], and Sandra Gabaglio[c]

(a) Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Lugano (Switzerland)
{giorgio,alessandro}@idsia.ch

(b) Universidade de São Paulo
São Paulo (Brazil)
denis.maua@usp.br

(c) Scuola Universitaria Professionale della Svizzera Italiana
Lugano (Switzerland)
sandra.gabaglio@supsi.ch

Abstract. In previous work, we devised an approach for multilabel classification based on an ensemble of Bayesian networks. It was characterized by an efficient structural learning and by high accuracy. Its shortcoming was the high computational complexity of the MAP inference, necessary to identify the most probable joint configuration of all classes. In this work, we switch from the ensemble approach to the single model approach. This allows important computational savings. The reduction of inference times is exponential in the difference between the treewidth of the single model and the number of classes. We adopt moreover a more sophisticated approach for the structural learning of the class subgraph. The proposed single models outperforms alternative approaches for multilabel classification such as binary relevance and ensemble of classifier chains.

1 Introduction

In traditional classification each instance is assigned to a single class. *Multilabel classification* generalizes this idea by allowing each instance to be assigned to multiple *relevant classes*. Multilabel classification allows to deal with complex problems such as tagging news articles or videos.

A simple approach to deal with multilabel classification is *binary relevance* (BR), which decomposes the problem into a set of traditional (i.e., single label) classification problems. Given a problem with n classes, binary relevance trains n independent single-label classifiers. Each classifier predicts whether a specific class is relevant or not for the given instance. Binary relevance is attractive because of its simplicity. Yet, it ignores dependencies among the different class variables. This might result in sub-optimal accuracy, since the class variables are often correlated [8]. According to the global accuracy metric, a classification is accurate only if the relevance of *every* class is correctly predicted. A sound model of the joint probability of classes given the observed features is thus necessary (see [13] and the references therein). This requires identifying the *maximum a posteriori* (MAP) configuration of the class relevances.

The classifier chain [12] is a state-of-the-art approach to model dependencies among classes. It achieves good accuracy; however, it has no probabilistic interpretation.

Bayesian networks (BNs) are an appealing tool for probabilistic multilabel classification, as they compactly represent the joint distribution of class and feature variables. When dealing with multilabel classification they pose two main challenges: structural learning and predictive inference. As for structural learning, the graph is typically partitioned into three pieces [14, 3]: the *class subgraph*, namely the structure over the class variables; the *feature subgraph*, namely the structure over the features variables; the *bridge subgraph*, namely the structure linking the feature to the class variables [14, 3, 4].

In a previous work [1] we introduced the *multilabel naive* assumption, which provides an interesting trade-off between computational speed of structural learning and effectiveness of the learned dependencies. The assumption is that the features are independent *given the classes*, thus generalizing naive Bayes to the multilabel case. As a consequence, the feature subgraph is empty. The bridge subgraph is optimally learned by independently looking for the optimal parents set of each feature. It does not require iterative adjustments. This allows for efficient structural learning. We accompany the multilabel naive assumption with a simple but effective algorithm for feature selection.

Our previous approach [1] was based on an ensemble of different Bayesian networks. Under the multilabel assumption the different BNs had an empty feature subgraph and shared the same optimal bridge subgraph. Each model had a different naive Bayes class subgraph. The ensemble approach achieved good performance. Its main shortcoming was the high complexity of the MAP inference regarding the most probable joint configuration of all classes. The high cost of MAP inference in multilabel Bayesian network classifiers has been discussed previously in the literature. In [5], the authors limit the MAP inferential complexity by constraining the underlying graph to be a collection of small disjoint graphs. However, this severely limits the expressivity of the models.

In this work, we aim at largely decreasing the computational times of our previous approach while keeping accuracy as high as possible. To this end, we move from the ensemble to a single model. Single models are known to be less accurate than ensembles. To compensate this effect, we introduce a more sophisticated structural learning procedure for the class sub-graph. We allow the class subgraph to be either a naive Bayes or a forest-augmented naive Bayes (FAN). This yields two different multilabel classifiers, called in the following mNB and mFAN. Both models optimize in two steps the class subgraph. In the first step each class is considered as a possible root of mNB (or mFAN). For each possible root, we identify the optimal naive (or FAN) structure. We thus identify

n different naive (or FAN) structures. In the second step we select the highest scoring naive (or FAN) structure.

Both mNB and mFAN are less accurate than the original ensemble. Yet, the accuracy gap is not huge. Moreover, mNB and mFAN outperform both binary relevance and the ensemble of classifier chains (implemented using naive Bayes as base classifier). The single model approach allows large computational savings compared to the ensemble. The saving is exponential in the difference between the number of classes and the treewidth of the single model which has been learned.

We then analyze the learned class sub-graphs. Define the relevance of a class as the percentage of instances for which it is relevant. We found a positive correlation between the relevance of the root class, the score of the class sub-graph and the number of non-root classes being to the root class. The explanation is as follows. Most classes have low relevance. A class which is labeled more often as relevant allows for better estimating the correlations with the remaining classes. This yields more non-root classes being connected to the root and also a higher score of the resulting graph. This might explain why our previous ensemble is only slightly more accurate than the single model. Many models of the ensemble had as a root of the class subgraph a class with low relevance. Such models were unlikely to convey helpful information when classifying the instances.

As a final contribution we discuss the need for preventing the *empty* prediction. A prediction is empty if all classes are predicted to be *not* relevant.

2 Probabilistic Multilabel Classification

We denote the array of class *relevances* as $\mathbf{C} := (C_1, \ldots, C_n)$; this is an array of Boolean variables, with variable C_i, $i = 1, \ldots, n$, expressing the relevance of the i-th class for the given instance. Thus, $\mathbf{C}$ takes its values in $\{0,1\}^n$. We denote the set of features as $\mathbf{F} := (F_1, \ldots, F_m)$. We assume the availability of a set of complete training instances $\mathcal{D} = \{(\mathbf{c}, \mathbf{f})\}$, where $\mathbf{c} = (c_1, \ldots, c_n)$ and $\mathbf{f} = (f_1, \ldots, f_m)$ represent an *instantiation* of class relevances and features, respectively.

A *probabilistic multilabel classifier* estimates a joint probability distribution over the class relevances conditional on the features, $P(\mathbf{C}|\mathbf{F})$. Such model can predict the class relevances on new instances. We denote as $\mathbf{c}$ and $\hat{\mathbf{c}}$ respectively the set of actual and predicted class relevances. The actual and the predicted relevance of class C_i are denoted respectively by c_i and $\hat{c}_i$.

A common metric to evaluate multilabel classifiers on a given instance is *global accuracy* (also called *exact match*):

$$\text{acc} := I(\mathbf{c} = \hat{\mathbf{c}})\,, \tag{1}$$

where I is the indicator function.

Another measure of performance is *Hamming accuracy* (also called mean label accuracy):

$$\mathrm{H}_{acc} := \frac{1}{n}\sum_{i=1}^{n} I(\hat{c}_i = c_i)\,. \tag{2}$$

Commonly Hamming loss (1-H_{acc}) rather than Hamming accuracy is reported. We report Hamming accuracy to simplify results readability: both *acc* and H_{acc} are better when they are higher. Global accuracy is often zero on data sets with many classes. On such data sets Hamming accuracy is thus more meaningful than global accuracy.

When classifying an instance with features $\mathbf{f}$, two different inferences are performed depending on whether the objective is to maximize global accuracy or Hamming accuracy.

To maximize global accuracy we search for the most probable joint configuration of the class relevances (*joint query*):

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}\in\{0,1\}^n} P(\mathbf{c}|\mathbf{f}) = \arg\max_{\mathbf{c}\in\{0,1\}^n} P(\mathbf{c},\mathbf{f})\,. \tag{3}$$

In the context of Bayesian networks, the above problem is known as Maximum A Posteriori (MAP) or Most Probable Explanation (MPE) inference. Unlike in standard MAP inference problems, the prediction of all classes being non-relevant (*empty prediction*) is considered invalid. Each instance should be assigned to at least one class. If the most probable joint configuration is the empty prediction, we ignore it and return the second most probable configuration, which is necessarily non-empty. To our knowledge this issue has not yet been pointed out. For instance the algorithms implemented by MEKA[1] do not prevent the empty prediction. We show empirically in Section 5 that preventing the empty prediction can increase accuracy in some domains.

To maximize Hamming accuracy we select the most probable configuration (relevant or non-relevant) of each class C_i (*marginal query*):

$$\hat{c}_i = \arg\max_{c_i\in\{0,1\}} P(c_i|\mathbf{f}) = \arg\max_{c_i\in\{0,1\}} P(c_i,\mathbf{f})\,, \tag{4}$$

where $P(c_i|\mathbf{f}) = \sum_{\mathbf{C}\setminus\{C_i\}} P(\mathbf{c}|\mathbf{f})$. If all classes are predicted to be non-relevant, the empty prediction is avoided by predicting as relevant only the class C_i with the highest posterior probability of being relevant.

We model the joint distribution $P(\mathbf{C},\mathbf{F})$ as a Bayesian network, and obtain classifications by running standard algorithms for either MAP inference or marginal inference in the network, according to the chosen performance measure. Once a structure (i.e., a directed acyclic graph over $\mathbf{C},\mathbf{F}$) for the corresponding Bayesian network has been defined, the model parameters are efficiently computed using Bayesian estimation. Learning a good structure is a challenging problem, which we tackle by making a number of assumptions on the structure that enable fast and exact learning. We detail the assumptions in the next section.

[1] http://meka.sourceforge.net

3 Structural learning

We address structural learning assuming the data set to be complete. The problem of how to efficiently learn the structure of a probabilistic multilabel classifier has been studied in the past [14, 3]. The graph is typically partitioned into three pieces: the *class subgraph*, namely the structure over the class variables modeling class-to-class (in)dependences; the *feature subgraph*, namely the structure over the features variables modeling feature-to-feature (in)dependences; the *bridge subgraph*, namely the structure linking the feature to the class variables and modeling the (in)dependences between features and classes.

The approach of [14] constrains both the class subgraph and feature subgraph to be a tree-augmented naive Bayes (TAN). A bridge subgraph is proposed, and the TANs of the two subgraphs are correspondingly learned. The bridge subgraph is iteratively updated following a *wrapper* approach [11]. Every time the bridge subgraph is updated, the two TANs are re-learned. Also [3] adopts a similar approach, considering a wider variety of topologies for the class and the feature subgraphs. Such approaches can result in high computational times because the bridge subgraph is incrementally improved at each iteration, requiring to correspondingly update also the other subgraphs. Conversely, [4] keeps empty both the class subgraph and the feature subgraph. This approach is fast but cannot properly model correlated class variables.

Instead, we assume the features to be independent *given the classes* as in a previous work of ours [1]. Since the feature nodes cannot have children, the feature subgraph is empty. This allows us to optimally learn the bridge subgraph by independently looking for the optimal parents set of each feature.

Our procedure is based on maximizing the BDeu score, which decomposes, in the case of complete data, as the sum of the BDeu scores of each node:

$$\mathrm{BDeu}(\mathcal{G}) := \sum_{X_i \in \{\mathbf{C},\mathbf{F}\}} \mathrm{BDeu}(X_i, \mathrm{Pa}(X_i)) \,,$$

where $\mathcal{G}$ denotes the entire graph (i.e., the union of class, feature and bridge subgraphs), X_i a generic node and $\mathrm{Pa}(X_i)$ its parents set in $\mathcal{G}$. The number of joint configurations of the parents of X_i is denoted by q_i. The score function for a single node is:

$$\mathrm{BDeu}(X_i, \mathrm{Pa}(X_i)) := \sum_{j=1}^{q_i} \left[\log \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} + \sum_{k=1}^{|X_i|} \log \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})} \right] \,, \tag{5}$$

where n_{ijk} is the number of records such that X_i is in its k-th state and its parents in their j-th configuration, while $n_{ij} = \sum_k n_{ijk}$. Finally, α_{ijk} is equal to the equivalent sample size α divided by the number of states of X_i and by the number of (joint) states of the parents, while $\alpha_{ij} = \sum_k \alpha_{ijk}$.

Class and feature nodes have only class nodes as parents (i.e., $\mathrm{Pa}(X_i) \subseteq \mathbf{C}$ for every node/variable X_i). Hence, the BDeu decomposes in two terms, referring respectively to the class and the bridge subgraphs:

$$\mathrm{BDeu}(\mathcal{G}) = \sum_{i=1}^{n} \mathrm{BDeu}(C_i, \mathrm{Pa}(C_i)) + \sum_{j=1}^{m} \mathrm{BDeu}(F_j, \mathrm{Pa}(F_j))\,.$$

Moreover, the two terms can be optimized separately, as the combined directed graph is necessarily acyclic. The optimizations of each term require different approaches:

Bridge subgraph We optimize the BDeu score of the bridge subgraph by independently searching for the optimal parents set of each feature. This strategy is optimal since: (i) the feature subgraph is empty, thus preventing the introduction of directed cycles; (ii) the BDeu scores decompose over the different feature variables.

Any subset of $\mathbf{C}$ is a candidate for the parents set of a feature, this reducing the problem to m independent local optimizations. The optimal parents set of F_j is found as follows:

$$\mathbf{C}_{F_j} := \arg \max_{\mathrm{Pa}(F_j)\subseteq \mathbf{C}} \mathrm{BDeu}(F_j, \mathrm{Pa}(F_j))\,, \tag{6}$$

for each $j = 1, \ldots, m$. The optimization in Equation (6) becomes more efficient by considering the pruning techniques proposed in [6].

Feature selection Naive Bayes is surprisingly effective in traditional classification despite its simplicity. Moreover, careful feature selection allowed naive Bayes even to win data mining competitions [9].

We thus perform feature selection before learning the bridge graph. We rely on the correlation-based feature selection (CFS) [15, Chap. 7.1], which has been developed for traditional classification. We perform CFS n times, once for each different class variable. Eventually, we retain the *union* of the features selected in the different runs. This is a useful pre-processing step which reduces the number of features, removing the non-relevant ones (Table 2). It also helps from the computational viewpoint. Feature selection for multilabel classification is however an open problem, and more sophisticated approaches can be designed to this end.

Class subgraph Unlike the feature subgraph, the optimizations of the class variables cannot be carried out independently, as this might introduce directed cycles. Instead, we enable efficient structure learning by restricting the class of allowed structures. We allow the class subgraph to be either a naive Bayes or a forest-augmented naive Bayes (FAN). This yields two different multilabel classifiers, called in the following mNB and mFAN. The leading 'm' shows that such classifiers are designed for multilabel classification.

Let us assume that the class which serves as root node (C^{root}) of naive Bayes or FAN is given. We then search for the class subgraph which maximizes the

BDeu score. As for mNB, the highest scoring naive Bayes is obtained by computing for each non-root class C_i the two scores BDeu$(C_i, \emptyset)$ and BDeu(C_i, C^{root}). Class C_i is linked to the root class if

$$\text{BDeu}(C_i, C^{\text{root}}) > \text{BDeu}(C_i, \emptyset) \tag{7}$$

and unlinked otherwise. The above procedure is repeated n times. Every time a different class $C_1, \ldots, C_n$ is taken as root of the naive Bayes. This yields n different naive Bayes structures. The structure with maximum score among them is taken as class subgraph for the multilabel classifier. Summing up, we perform a two-steps optimization: first we compute the optimal naive structure for each possible root. Then we select the highest scoring structure among those identified in the first step.

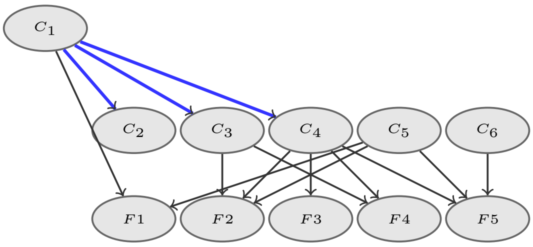

Fig. 1. Example of a mNB model. The class subgraph is a naive Bayes, with C_5 and C_6 unlinked from the root class. The arcs of the class subgraph are thicker and shown in blue.

An analogous two-step optimization is adopted when a FAN structure is looked for. The difference between FAN and TAN is that the former augments naive Bayes by a forest, while the latter augments naive Bayes by a tree (i.e., the underlying graph of a TAN is necessary connected). TAN is thus a special case of FAN, and therefore the latter can achieve a higher BDeu score than the former, as its structure has more degrees of freedom. To identify the optimal FAN we do not independently optimize the parents set of each non-root node, as this approach could introduce cycles. We instead solve an optimization problem characterized by the following constraints: cycles are not allowed; the root node has no parents; each non-root node C_i has three feasible configuration for its parents set: the empty set; the root class; the root class and another non-root class. Strategies for efficiently learning the optimal FAN structure are discussed for instance in [6, 7].

Figure 1 depicts an example of mNB, with the multilabel naive assumption for the bridge subgraph and a naive Bayes as class subgraph.

An ensemble of Bayesian networks The previously described procedure yields a single Bayesian network model. Its bridge graph is optimally learned under the multilabel naive assumption. Its class graph is either an optimal FAN or an optimal naive Bayes.

In a previous work [1], we considered instead an ensemble approach. The ensemble was constituted by n different Bayesian networks (BNs). Each BN was based on naive multilabel assumption. We recall that, under the multilabel naive assumption, the optimal bridge subgraph is independent from the class subgraph. Thus, the BNs shared the same bridge subgraph.

The class subgraph was constituted by a different naive Bayes for each member of the ensemble. Each member of the ensemble used a different root for naive Bayes. The structure of naive Bayes was not optimized. The inferences produced by the different member of the ensemble were combined through logarithmic opinion pooling. The ensemble approach showed good performance, but also high computational times, especially for MAP inference (see the discussion at the end of Section 4).

In this paper we move from the ensemble to a single model. In this way, we largely reduce the complexity of the inferences. To compensate for the loss of accuracy due to switching to a single model, we introduce the two-layer optimization for the class subgraph described in the previous section.

4 Computational Complexity

We distinguish between learning and classification complexity, where the latter refers to the classification of a single instantiation of the features. Both space and time required for computations are analyzed. The orders of magnitude of these descriptors are reported as a function of the (training) dataset size d, the number of classes n, the number of features m, the average number of states for the features $f = m^{-1}\sum_{j=1}^{m}|F_j|$, and the maximum in-degree of the features $g = \max_{j=1,\ldots,m}\|\mathbf{C}_{F_j}\|$ (where $\mathbf{C}_{F_j}$ is the optimal parents set of F_j computed as in Equation (6), $|\cdot|$ returns the cardinality of a variable, and $\|\cdot\|$ the number elements in a joint variable). As the class variables cannot have more than two parents in the mFAN and no more than one in the mNB, g is also the maximum in-degree of the network (provided that $g > 1$).

Regarding space, the conditional probability table (CPT) of the j-th feature, i.e., $P(F_j|\mathbf{C}_{F_j})$, needs space $O(|F_j| \cdot 2^{\|\mathbf{C}_{F_j}\|})$, while the n CPTs associated to the classes have size bounded by a small constant (8 numbers for the mFAN and 4 for the mNB). Overall, this means a space complexity $O(n + f2^g)$. These tables should be available during both learning and classification.

Regarding the time required by the learning, let us first note that, according to Equation (5), the computation of the BDeu score associated to a variable takes a time of the same order of the space required to store the corresponding CPT. The number of scores to be evaluated in order to determine the parents of F_j as in Equation (6) is of the same order of the binomial coefficient $\binom{n}{g}$, that means $O(n2^g)$. Then, we sum over the features and obtain $O(mn2^g)$ time. Regarding the class graph, for mNB we only have to evaluate the inequality in Equation (7), which only takes constant time, on the non-root classes. This means $O(n)$ time. Learning a FAN can be instead achieved in $O(n^2)$ [7]. Finally, the quantification of the network parameters requires the scan of the dataset, i.e., for the whole ensemble, $O((n+m)d)$. Such a learning procedure should be iterated over the n models of the ensemble, in order to select the one with the highest likelihood. This only affects the class subgraph, and the relative term should be therefore additionaly multiplied by n.

Concerning the classification time, both the MAP inference in Equation (3) and the computation of marginals in Equation (4) can be solved exactly by junction tree algorithms in time exponential in the *treewidth* of the network's moral graph.[2] The treewidth measures the connectivity of the network, which according to our learning procedure depends on unconditional class correlations, and the capability of the features to induce additional (conditional) correlations among classes. In the mFAN, the treewidth is at least 3, while in mNB it is at least 2. The experiments reported in the next section show that the treewidth of our models is usually much smaller than the number of classes, and often small enough to enable exact inference (by junction tree algorithms).

The situation of our previous ensemble was radically different [1]. For each pair of class variables, say C_i and C_j, there was at least a network in the ensemble such that C_i was parent of C_j (and vice-versa). Thus, when merging all the networks of the ensemble in a single Markov random field, there was a clique including all the classes, which made the treewidth equal to n, the number of classes. Such a treewidth made exact inference intractable for large n, and we were forced to resort to approximate methods such as max-product.

The ratio of the inference time of the ensemble to the inference time of the single model increases exponentially with the difference between the number of classes and the actual treewidth of the single model. Yet, there are no theoretical guarantees for the treewidth of the single model being small (i.e., bounded by a constant) [10]. In fact, as the inference problem is NP-hard even in structures as simple as the bridge graph alone, we expect the treewidth of the single models to be at least super logarithmic (i.e., greater than $\log(n)$) in the *worst case.* When this is the case, (i.e., when the treewidth is too high), approximate algorithms are used instead. In these cases the time complexity is $O(n2^g)$.

Summarizing, the maximum in-degree represents the bottleneck for space, learning time, and the classification time with approximate methods. Since, as proved by [6], $g = O(\log d)$, the overall complexity is polynomial. Regarding the classification time with exact inference, this is exponential in the treewidth.

5 Experiments

We compare mNB and mFAN against different alternative models: the ensemble of BNs we proposed in [1]; the binary relevance algorithm; the ensemble of chain classifiers (ECC). For both binary relevance and ECC we use naive Bayes as

[2] The moral graph of a Bayesian network is the undirected graph obtained by linking nodes with a common child and dropping arc directions; its treewidth is the maximum size of a clique after being triangulated.

base classifier. Binary relevance thus runs n independent naive Bayes classifiers, where n is the number of classes.

ECC stands for 'ensemble of chain classifiers'. Each chain is characterized by a different order of the labels in the chain. We set to 20 the number of chains in the ensemble. Therefore, ECC runs $20{\cdot}n$ naive Bayes, We use the implementation of these methods provided by MEKA.[3]

It has not been possible to include in our experiments other multi-label classifiers based on BNs [14, 3] because of the lack of public domain software.

Regarding the parameters of our model, we set the equivalent sample size for the structural learning to $\alpha = 5$. No other parameter needs to be specified.

We have implemented the high-level part of the algorithm in Python. For structural learning, we adopted the GOBNILP package.[4] We performed the inferences using the junction tree and belief propagation algorithms implemented in libDAI, a library for inference in probabilistic graphical models.[5]

We compare the classifiers on 8 different data sets, whose characteristics are given in Table 1. The *density* is the average number of relevant labels per instance.

Data set	Classes	Features	Instances	Density
Emotions	6	72	593	.31
Scene	6	294	2407	.18
Yeast	14	103	2417	.30
Slashdot	22	1079	3782	.05
Genbase	27	1186	662	.04
Enron	53	1001	1702	.06
Cal500	174	68	502	.15
Medical	45	1449	978	.03

Table 1. Datasets used for experiments.

We validate the classifiers by a 5-folds cross-validation. We stratify training and test sets according to the least relevant label (i.e., the label which is less often annotated as relevant, and whose distribution among folds risks to be very uneven if not stratified).

Before training any classifier, we perform two pre-processing steps. First, we discretize numerical features into four bins. The bins are given by the 25-th, the 50-th and 75-th percentile of the value of the feature. Then we perform feature selection as described in Section 3. The effectiveness of feature selection can be appreciated from the third column of Table 2.

[3] `http://meka.sourceforge.net`

[4] `http://www.cs.york.ac.uk/aig/sw/gobnilp`

[5] `http://www.libdai.org`

Data set	Treewidth (mNB/ensemble)	Features (selected/original)
Emotions	5/6	22/72
Scene	6/6	184/294
Yeast	8/14	28/103
Slashdot	22/22	465/1079
Genbase	23/27	82/1186
Enron	32/53	220/1001
Cal500	10/174	66/68
Medical	35/45	436/1449

Table 2. Treewidth and feature selection on the benchmark data sets.

The results regarding global accuracy and Hamming accuracy are provided in Table 3 and 4 respectively. The Friedman test rejects the null hypothesis of all classifiers having the same median rank. This happens both for global accuracy (p<0.01) and for Hamming accuracy (p<0.01). The following rank is consistently found under both accuracies: the first ranked classifier is the ensemble, followed by mNB, mFAN, binary relevance and ECC.

Data set	ensemble	mNB	mFAN	ECC	binary rel.
Cal500	0.00	0.00	0.00	0.00	0.00
Emotions	0.28	0.22	0.22	0.25	0.25
Enron	0.12	0.06	0.11	0.02	0.02
Genbase	0.95	0.94	0.93	0.76	0.77
Medical	0.69	0.65	0.62	0.20	0.18
Scene	0.64	0.61	0.59	0.30	0.29
Slashdot	0.49	0.42	0.42	0.44	0.41
Yeast	0.14	0.07	0.09	0.13	0.11
Average rank	**1.2**	**3.1**	**3.2**	**3.4**	**4.0**

Table 3. Global accuracy. Classifiers are sorted according to their average rank. Lower rank is better.

We exclude from the subsequent analysis the mFAN model. It has lower rank than mNB on both Hamming accuracy and global accuracy, despite higher complexity. The reason of this phenomenon is not yet clear and it is worth further investigation. However, according to Occam razor, mNB should be preferred over mFAN.

We then perform the statistical multiple comparisons among ensemble, mNB, binary relevance and ECC. We adopt the Wilcoxon signed-rank test to perform the pairwise comparisons. Before declaring significance, we adjust the p-values according to the false discovery rate (FDR) correction [2]. FDR adjusts the p-values of multiple comparisons in a more powerful (i.e., less conservative) way

Data set	ensemble	mNB	mFAN	ECC	binary rel.
Cal500	0.86	0.86	0.86	0.59	0.63
Emotions	0.79	0.77	0.77	0.76	0.75
Enron	0.95	0.94	0.94	0.77	0.87
Genbase	1.00	1.00	1.00	0.99	0.98
Medical	0.99	0.99	0.98	0.98	0.69
Scene	0.91	0.90	0.89	0.83	0.82
Slashdot	0.96	0.95	0.95	0.95	0.86
Yeast	0.78	0.77	0.76	0.76	0.75
Average rank	**1.3**	**2.2**	**2.8**	**4.0**	**4.8**

Table 4. Hamming accuracy. Classifiers are sorted according to their average rank. Lower rank is better.

than traditional methods such as Bonferroni. We report a significant difference when the *adjusted* p-value is smaller than 0.05.

The ensemble is significantly more accurate than binary relevance, ECC and mNB. This is verified both on global accuracy and Hamming accuracy. No significant difference can be detected between ECC and mNB. Moreover, both ECC and mNB have significantly higher Hamming accuracy than binary relevance. As for global accuracy, the ECC and mNB have higher rank than binary relevance, but the difference is not significant. A possible reason is that on the Cal500 data sets the global accuracy of all classifiers is zero because of the high number of classes. Thus, global accuracy provides less evidence than Hamming accuracy when analyzed by a hypothesis test.

Summing up, the ensemble is significantly more accurate than mNB, which however compares favorably to both binary relevance and ECC. However, mNB provides huge computational savings compared to the ensemble. The saving is linear in the number in classes when performing the marginal inference (Equation 4). The ensemble performs the marginal query n^2 times (n members of the ensemble multiplied by n classes). The mNB classifier performs this query n times (once for each class). As already discussed in Section 4, even larger computational savings are obtained on the query about the most probable joint configuration of the classes (Equation 3). Consider the ratio of the inference time of the ensemble to the inference time of mNB. This ratio varies between 3 and 280 depending on the data set. Figure 2 shows the relation between the logarithm of the ratio and the difference between the treewidth of the ensemble (equal to the number of classes, see Section 4) and the treewidth of mNB (see the values in Table 2). The relation is roughly linear as expected.

5.1 Insights on the mNB structure

In this section we derive some insights analyzing the structure of the class subgraphs learned for the mNB model. Learning the class subgraph according to the two steps procedure of Section 3 boils down to identify a) which non-root

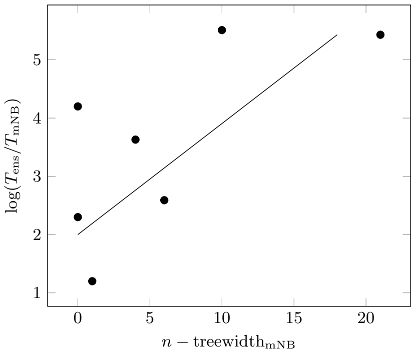

Fig. 2. Logarithm of the ratio of inference times (ensemble/mNB, in log scale) against difference between the number of classes and the treewidth of the mNB. A log ratio of 2.5 implies a 12-folds speed up. A log ratio of 5 implies a 150-folds speed up.

classes should be connected to each possible root class and b) which is the best structure, among the n characterized by different roots. Let us call *optimal root* the root of the graph which is eventually chosen.

The optimal root is usually among the most relevant classes (i.e., those which are more frequently tagged as relevant) available in the data set. This point is illustrated by the following analysis. Each class has its own relevance (% of instances in which it is relevant). Consider the relevance of the class selected as optimal root. Sort the classes according to their relevance and compute the percentile of the root class. This percentile is generally well above 0.8. Often the root class is the most relevant one (Figure 3, left).

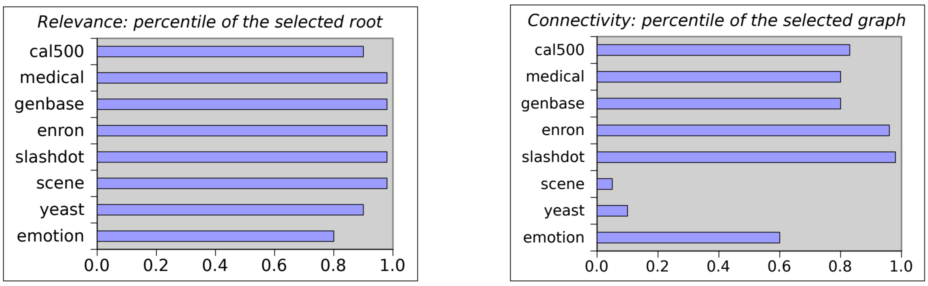

Fig. 3. Percentile of relevance of the optimal root (*left*) and connectivity of the optimal naive Bayes (*right*).

We then analyze the connectivity of the optimal naive Bayes. If n is the number of classes, naive Bayes contains at most $n - 1$ arcs between the root and the non-root classes. The *connectivity* of a naive Bayes is how many arcs are instantiated out of the $n - 1$ possible ones. Consider the connectivity of the naive Bayes selected as optimal. Sort the n different naive Bayes (each characterized by a different root) according to their connectivity. Take the percentile of the optimal naive Bayes. Such percentile is usually well above 0.6 (Figure 3, right). An exception is found on the Yeast and Scene data sets. On Scene, the optimal naive Bayes has connectivity of 4/5 while the alternative naive Bayes have connectivity 5/5. The optimal model has thus high connectivity, but this is hidden when computing the percentile. Only on yeast the optimal naive Bayes has limited connectivity (6/13).

Our explanation is as follows. Consider that most classes have low relevance: the average relevance of a class is around 10%, with huge differences among data sets. A class which is more often relevant allows to reliably estimate the correlations with the relevance of the remaining classes. Using a class of this type as root of the class subgraph results both in higher score and in higher number of arcs going from the root to the non-root classes.

5.2 Avoiding empty predictions

The least dense data sets are Slashdot, Genbase, Enron and Medical (see Table 1). On such data sets, *global accuracy* increases when empty predictions are prevented (Table 5). The effect is noteworthy on Slashdot. We did not find empty predictions on the other data sets.

Preventing empty prediction sometimes improves also *Hamming accuracy*, but the impact on this indicator is narrow. Preventing empty predictions should become common practice in multilabel classification because of the following reasons: it sometimes improves accuracy; it never worsens it. It is trivial to implement. Most important, it avoids returning a non-sensible prediction.

	ensemble		**mNB**	
Data set	empty prevented	empty allowed	empty prevented	empty allowed
Slashdot	.49	.40	.42	.34
Genbase	.95	.94	.94	.94
Enron	.12	.07	.06	.05
Medical	.69	.64	.65	.60

Table 5. Change in global accuracy when preventing the empty prediction.

6 Conclusions

A new approach to multilabel classification based on Bayesian networks has been proposed. Some of the ideas of our previous work [1] are kept: the naive multiclass assumption (i.e., features are conditionally independent given the joint class), a singly connected subgraph over the classes, and no feature-to-class arcs. Yet, we consider a more sophisticated learning of the structure. We show empirically that the approximation largely decreases the computational burden while incurring only a small worsening of accuracy. We also provide an original analysis of the identified structures. We plan to make our code available in the near future.

Acknowledgements

We thank Cassio P. de Campos and David Huber for valuable discussions. Work supported by the Swiss NSF grant no. 200020_134759 / 1, by the Hasler foundation grant n. 10030 and by FAPESP grant no. 2013/23197-4.

Bibliography

[1] Antonucci A, Corani G, Mauá D, Gabaglio S (2013) An ensemble of Bayesian networks for multilabel classification. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13), pp 1220–1225

[2] Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society Series B (Methodological) pp 289–300

[3] Bielza C, Li G, Larrañaga P (2011) Multi-dimensional classification with Bayesian networks. International Journal of Approximate Reasoning 52(6):705–727

[4] Bolt JH, van der Gaag LC (2012) Multi-dimensional classification with naive Bayesian network classifiers. In: BNAIC 2012 The 24th Benelux Conference on Artificial Intelligence, pp 27–34

[5] Borchani H, Bielza C, Larrañaga P (2010) Learning CB-decomposable multi-dimensional Bayesian network classifiers. In: Proceedings of the 5th European Workshop on Probabilistic Graphical Models (PGM-10), pp 25–32

[6] de Campos C, Ji Q (2011) Efficient structure learning of Bayesian networks using constraints. Journal of Machine Learning Research 12:663–689

[7] de Campos C, Cuccu M, Corani G, Zaffalon M (2014) The extended tree augmented naive classifier. Submitted to PGM '14

[8] Dembczynski K, Waegeman W, Hüllermeier E (2012) An analysis of chaining in multi-label classification. In: Proceedings of the 20th European Conference on Artificial Intelligence (ECAI), pp 294–299

[9] Elkan C (2001) Magical thinking in data mining: lessons from coil challenge 2000. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge discovery and data mining, pp 426–431

[10] Karpas E, Solomon E, Beimel A (2009) Approximate belief updating in max-2-connected Bayes networks is NP-hard. Artificial Intelligence 173(12-13):1150–1153

[11] Kohavi R, John G (1997) Wrappers for feature subset selection. Artificial intelligence 97(1):273–324

[12] Read J, Pfahringer B, Holmes G, Frank E (2011) Classifier chains for multi-label classification. Machine learning 85(3):333–359

[13] Read J, Bielza C, Larra P, et al (2013) Multi-dimensional classification with super-classes. IEEE Transactions on Knowledge and Data Engineering p 1

[14] Van Der Gaag L, De Waal P (2006) Multi-dimensional Bayesian network classifiers. In: Proceedings of the Third European Workshop on Probabilistic Graphical Models, pp 107–114

[15] Witten I, Frank E, Hall M (2011) Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques. Morgan Kaufmann