Multi-classifiers of Small Treewidth

Arnoud Pastink$^{(\boxtimes)}$ and Linda C. van der Gaag

Department of Information and Computing Sciences,
Utrecht University, Utrecht, The Netherlands
{A.J.Pastink,L.C.vanderGaag}@uu.nl

Abstract. Multi-dimensional Bayesian network classifiers are becoming quite popular for multi-label classification. These models have the advantage of a high expressive power, but may induce a prohibitively high runtime of classification. We argue that the high runtime burden originates from their large treewidth. Thus motivated, we present an algorithm for learning multi-classifiers of small treewidth. Experimental results show that these models have a small runtime of classification, without loosing accuracy compared to unconstrained multi-classifiers.

1 Introduction

Multi-dimensional Bayesian network classifiers [9], or multi-classifiers for short, constitute an increasingly popular approach to multi-label classification. While these models have the advantage of a high expressive power, they may come associated with a high runtime of classification. Especially for large sets of instances to be classified and in applications in which instances are to be classified instantaneously, can this high runtime burden prove prohibitive. Although various researchers addressed the classification time of multi-classifiers and designed learning algorithms giving reasonable runtime properties in general [4,7], available algorithms do not come with any actual guarantees on classification time.

In this paper, we argue that the high runtime complexity of multi-classifiers can be attributed to their tendency to have a large treewidth. Motivated by this observation, we present an algorithm for learning multi-classifiers of small treewidth. The algorithm bounds treewidth not by imposing general topological constraints, but by iteratively monitoring treewidth of partially constructed classifiers in a branch-and-bound approach. As a result, our algorithm retains much of the expressive power of the multi-classifier framework and is expected to result in good-quality models for efficient classification. Experiments on various data sets in fact show that the classifiers learned with our algorithm perform comparably, in terms of classification accuracy, to multi-classifiers of unbounded treewidth. Our results further show that by bounding treewidth a huge reduction, up to a factor 400, of the runtime complexity of classification is achieved.

The paper is organised as follows. In Sect. 2 we introduce our notational conventions and review multi-classifiers. In Sect. 3 we address the relationship between multi-classifiers and treewidth. Earlier research addressing the runtime of multi-classifiers is reviewed in Sect. 4. In Sect. 5 we present our algorithm for

© Springer International Publishing Switzerland 2015
S. Destercke and T. Denoeux (Eds.): ECSQARU 2015, LNAI 9161, pp. 199–209, 2015.
DOI: 10.1007/978-3-319-20807-7_18

learning multi-classifiers of small treewidth. In Sect. 6 we report results achieved with our algorithm on various multi-label data sets. The paper is rounded off with our conclusions and directions for future work in Sect. 7.

2 Preliminaries

We consider a finite non-empty set $\mathbf{V}$ of discrete random variables, in which each variable $V_i \in \mathbf{V}$ takes its value from a finite set of states. The joint state space for a subset $\mathbf{S} \subseteq \mathbf{V}$ is the Cartesian product of the sets of states of the separate variables in $\mathbf{S}$; we use $\kappa_{\mathbf{S}}$ to denote the size of this joint state space. Given our focus on classification, we assume that the set $\mathbf{V}$ is partitioned into a set $\mathbf{C} = \{C_1, \ldots, C_n\}$, $n \geq 1$, of class variables and a set $\mathbf{X} = \{X_1, \ldots, X_m\}$, $m \geq 1$, of feature variables, with $\mathbf{C} \cup \mathbf{X} = \mathbf{V}$ and $\mathbf{C} \cap \mathbf{X} = \varnothing$. A joint state of the feature variables $\mathbf{X}$ is referred to as a feature vector and is denoted by $\mathbf{x}$; a joint state $\mathbf{c}$ of the class variables $\mathbf{C}$ is called a class vector. A pair $(\mathbf{c}, \mathbf{x})$ is termed an instance over $\mathbf{V}$. We further assume a (multi-)set $\mathcal{D}$ of instances over $\mathbf{V}$, which is partitioned into a training set $\mathcal{D}^{tr} = \{(\mathbf{c}_1, \mathbf{x}_1), \ldots, (\mathbf{c}_k, \mathbf{x}_k)\}$, $k \geq 1$, and a set of test instances $\mathcal{D}^{te} = \{(\mathbf{c}_1, \mathbf{x}_1), \ldots, (\mathbf{c}_l, \mathbf{x}_l)\}$, $l \geq 1$.

A multi-classifier over the random variables $\mathbf{V}$ is a Bayesian network of restricted topology over $\mathbf{V}$ [9]. Its set of arcs A is partitioned into three subsets:

- $A_{\mathbf{C}} \subseteq \mathbf{C} \times \mathbf{C}$ includes the arcs among the class variables, and the subgraph induced by $A_{\mathbf{C}}$ is called the class subgraph;
- $A_{\mathbf{X}} \subseteq \mathbf{X} \times \mathbf{X}$ contains the arcs among the feature variables, and the subgraph induced by $A_{\mathbf{X}}$ is called the feature subgraph;
- $A_{\mathbf{CX}} \subseteq \mathbf{C} \times \mathbf{X}$ includes the arcs from a class variable to a feature variable, and the subgraph induced by $A_{\mathbf{CX}}$ is called the bridge subgraph.

In this paper, we focus on multi-classifiers in which each class variable has at most one class variable parent, a feature variable has at most p class parents, and the feature subgraph is either empty or a forest-structured graph.

Classification of a feature vector $\mathbf{x}$ amounts to finding a class vector $\mathbf{c}$ that maximizes the posterior probability given $\mathbf{x}$, that is, it amounts to finding

$$\underset{\mathbf{c} \in \mathbf{C}}{\operatorname{argmax}} \{\Pr(\mathbf{c} \mid \mathbf{x})\}$$

The performance of a multi-classifier is estimated from the accuracy of its classifications for a set $\mathcal{D}^{te}$ of test instances $(\mathbf{c}_i, \mathbf{x}_i)$. For each feature vector $\mathbf{x}_i$, a most likely class vector $\hat{\mathbf{c}}_i$ is computed from the classifier and compared with the true class vector $\mathbf{c}_i$ of the instance. Performance is now expressed by two metrics. The global accuracy acc_{G} of the classifier given the testset $\mathcal{D}^{te}$ is defined as:

$$acc_{\mathrm{G}}(\mathcal{D}^{te}) = \frac{1}{|\mathcal{D}^{te}|} \cdot \sum_{(\mathbf{c}_i, \mathbf{x}_i) \in \mathcal{D}^{te}} \delta(\mathbf{c}_i, \hat{\mathbf{c}}_i)$$

where $\delta(\mathbf{c}_i, \hat{\mathbf{c}}_i)$ equals 1 if $\mathbf{c}_i = \hat{\mathbf{c}}_i$ and 0 otherwise. This metric serves to measure the proportion of (complete) class vectors that are predicted correctly. The Hamming metric measures the proportion of class variables for which a correct prediction is made. The Hamming accuracy acc_{H} of the classifier is defined as:

$$acc_{\mathrm{H}}(\mathcal{D}^{te}) = \frac{1}{|\mathcal{D}^{te}|} \cdot \sum_{(\mathbf{c}_i, \mathbf{x}_i) \in \mathcal{D}^{te}} \left(\frac{1}{n} \cdot \sum_{i=1}^{n} \delta(\hat{c}_{ij}, c_{ij}) \right)$$

where c_{ij} is the state of the j-th class variable in the i-th test instance, and $\delta(\hat{c}_{ij}, c_{ij})$ equals 1 if $\hat{c}_{ij} = c_{ij}$ and 0 otherwise.

Multi-classifiers are usually learned from a (multi-)set of instances. The objective then is to construct a classifier from the training instances that performs well on the test data and allows good classification of yet unseen instances. In this paper, we take a score-based approach to learning. Each possible graphical structure is assigned a numerical score which describes how well the structure fits the training data. To this end, we employ the BDeu score [5], which conveniently decomposes as a sum of BDeu scores per variable:

$$\mathrm{BDeu}(G) = \sum_{V_i \in \mathbf{C} \cup \mathbf{X}} \mathrm{BDeu}(V_i, \mathrm{pa}(V_i))$$

where G is the graphical structure under consideration, V_i is a variable in G and $\mathrm{pa}(V_i)$ are its parents. For a variable V_i, the BDeu score given its parents equals:

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

where n_{ijk} is the number of training instances in which the variable V_i is in state k and its parents are in their j-th joint state, and $n_{ij} = \sum_k n_{ijk}$; q_i is the number of joint states of the parents of V_i. With an equivalent sample size of α, we get that $\alpha_{ij} = \frac{\alpha}{q_i}$ and $\alpha_{ijk} = \frac{\alpha_{ij}}{|V_i|}$, where $|V_i|$ is the number of states of V_i. Upon learning, the goal now is to maximise the BDeu score with respect to the training data, subject to the structural constraints imposed.

The class subgraph of a multi-classifier can be learned independently from its feature and bridge subgraphs [9]. If the feature subgraph is known to be empty, then the bridge subgraph can be learned optimally by selecting the best scoring parent set per feature variable [7]. If the feature subgraph is non-empty however, the bridge subgraph cannot be learned independently from the feature subgraph. To arrive at optimality with respect to the BDeu score, the bridge and feature subgraphs should then be learned simultaneously. We note that simply selecting the best scoring parent set per feature variable would now not necessarily result in a valid multi-classifier as acyclicity would not be guaranteed.

In this paper, we adapt an existing approach to learning extended tree augmented naive Bayesian classifiers (ETANs) in general [6], to learning the combined bridge and feature subgraph of a multi-classifier. First a feature subgraph is learned on a copy $\mathbf{X}'$ of the set of feature variables $\mathbf{X}$. The BDeu scores for the variables in $\mathbf{X}'$ are established from those for $\mathbf{X}$ by collapsing the scores of the parent sets that include class variables. More specifically, for each variable $X'_i \in \mathbf{X}'$ the following BDeu scores are established:

$$\mathrm{BDeu}(X'_i, \varnothing) = \max_{\mathbf{S} \subseteq \mathbf{C}} \mathrm{BDeu}(X_i, \mathbf{S})$$
$$\mathrm{BDeu}(X'_i, X'_j) = \max_{\mathbf{S} \subseteq \mathbf{C}} \mathrm{BDeu}(X_i, \mathbf{S} \cup \{X_j\})$$

Using a standard Bayesian network learner [1] or a modified version of the ETAN-algorithm [6], we now learn feature subgraph on $\mathbf{X}'$, where each feature variable X'_i is allowed at most one parent. For each variable X'_i, the chosen parent X'_j (or the empty parent set) is then expanded to the parent set of the original score. The result is a combined bridge and feature subgraph with maximal BDeu score given the structural constraints of the multi-classifier.

3 Multi-classifiers and Treewidth

Computing a class vector of highest posterior probability from a multi-classifier given a specific feature vector, is equal to solving the most probable explanation (MPE) problem. This MPE problem is known to be NP-hard in general [12], and remains NP-hard even for binary networks in which both the indegree and the outdegree of all variables is at most two [12]. The MPE problem can be solved in polynomial time however, for networks of bounded treewidth.

We briefly revisit the importance of the concept of treewidth in Bayesian networks in general. Current algorithms for probabilistic inference build upon a junction-tree representation of a network. To construct such a representation, the network's graphical structure G is first moralised by adding edges between all pairs of parents of a variable and subsequently dropping directions. The moralised graph is then triangulated by adding edges to make it chordal, that is, to render a graph in which any cycle of four or more variables has a shortcut. A tree-decomposition of the triangulated graph G_T now is a tree $\mathcal{T}_G$ such that:

– each node $Cl_i \in \mathcal{T}_G$ corresponds with a maximal clique in G_T, and vice versa;
– for every i, j, k, if node Cl_j lies on the path from Cl_i to Cl_k in $\mathcal{T}_G$, then $Cl_i \cap Cl_k \subseteq Cl_j$.

The width of the tree-decomposition $\mathcal{T}_G$ of G is equal to $\max_i\{\, |Cl_i| - 1 \mid Cl_i \in \mathcal{T}_G\}$, where $|Cl_i|$ is the number of variables in the i-th clique. The treewidth of the graphical structure G of a network now is equal to the minimum width over all tree-decompositions of its moralised graph, and is denoted by $\tau(G)$.

Current inference algorithms for Bayesian networks in general pass messages through a junction-tree representation of a network [10], which embeds for its graphical structure a tree-decomposition of the network's original graph. The processing time of a single clique in the junction tree is proportional to the size of the clique's state space. Only if this size is bounded by a constant for all

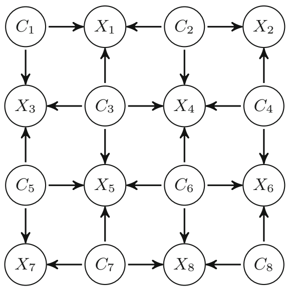

Fig. 1. A 4×4 grid multi-classifier

cliques, can inference be performed efficiently. We note that if the treewidth is bounded, a bounded state space per variable suffices for feasible inference.

The treewidth of a multi-classifier in general is not bounded by a constant, not even if the indegree and outdegree of all variables are small. As an example, we consider a multi-classifier constructed from a generalised $n \times n$ chessboard; the classifier for $n = 4$ is depicted in Fig. 1. Each black tile of the chessboard is represented by a class variable, and each white tile is captured by a feature variable; for each adjacent pair of tiles, an arc is added from the associated class variable to the feature variable. The resulting multi-classifier has empty class and feature subgraphs; the number of parents per feature variable is at most four. The treewidth of this classifier is n [2], which shows that even simple multi-classifiers can have a prohibitively large treewidth and, hence, a high classification runtime.

4 Related Work

The runtime of classification with a multi-classifier having been addressed before, we briefly review earlier work on reducing the computational burden involved.

The property of class-bridge decomposability for multi-classifiers was introduced to allow a divide-and-conquer strategy for classification. If a classifier is class-bridge decomposable, its graphical structure decomposes, given a feature vector, into multiple components defined by the bridge subgraph; classification then is performed in each component separately. The treewidth of such a classifier is equal to the maximum treewidth of its individual components. Heuristic algorithms have been designed for learning class-bridge decomposable multi-classifiers [4]. Since these algorithms do not address treewidth explicitly, the treewidth of a class-bridge decomposable multi-classifier may still be prohibitively large. The algorithm in fact does not give any guarantee on the runtime complexity of classification with the learned classifier.

Corani *et al.* [7] learn sparse multi-classifiers. The class subgraphs of their classifiers are forests, and the bridge subgraphs are learned optimally by taking the best scoring parent set per feature variable, given an empty feature subgraph. Despite their sparsity, the treewidth of the resulting classifiers may be quite large and bounded only by the total number of class variables.

```
Learn a multi-classifier of treewidth at most $k$:
1. Learn an optimal forest-structured class subgraph;
2. Search the space of all possible bridge subgraphs, maintaining a branch-and-bound
   tree of partial multi-classifiers. While the branch-and-bound tree has unvisited
   nodes, perform the following steps:
   2.1 Take the next partial multi-classifier $P$ from the branch-and-bound tree;
   2.2 Add a new feature variable with its best-scoring parent set to $P$;
   2.3 if BDeu($P$) < lowerbound then
         – Stop expanding $P$, and go to Step 2.1;
   2.4 if $\tau(P) > k$ then
         – Replace the current parent set by the next one, and go to Step 2.3
   2.5 else
         – Add each remaining feature variable to $P$, and insert into the search tree;
         – Update the lower bound if applicable;
3. Optionally, learn a forest-structured feature subgraph.
```

Fig. 2. An algorithm for learning multi-classifiers of treewidth at most k

5 Learning Multi-classifiers of Small Treewidth

Our algorithm for learning multi-classifiers of small treewidth takes a branch - and-bound approach to systematically searching the space of all graphical structures for one with highest BDeu score given a treewidth of at most k. Since a large treewidth can be induced by just the bridge subgraph of such a classifier, the algorithm focuses on this subgraph. The algorithm is summarised in Fig. 2.

Step 1: The algorithm starts by learning an optimal forest-structured graph over the class variables; to this end any standard algorithm can be used [8]. The learned subgraph is then fixed for the remainder of the learning process.

Step 2: Given the learned class subgraph, the algorithm builds a bridge subgraph by iteratively adding feature variables to the partially constructed multi-classifier. The algorithm computes to this end, for each such variable, the BDeu scores of all its possible parent sets. After adding a new feature variable and its current best-scoring parent set, the algorithm compares the BDeu score of the partial multi-classifier so far against a lower bound.

An initial lower bound on the BDeu score of the classifier to be learned is established by finding the best single class parent for each feature variable, and taking the score of the resulting forest of naive Bayesian classifiers. When joined with the class subgraph from Step 1, the resulting classifier has treewidth one, and hence constitutes a feasible solution. To compare the BDeu score of a partially constructed multi-classifier against the lower bound, this partial classifier needs to be extended to a multi-classifier with all feature variables involved. To this end, the algorithm adds the best parent set for each yet remaining feature variable; we note that this completed classifier may be infeasible as its treewidth may be larger than k. The BDeu score of the completed classifier thus is an upper bound on the score which can be attained by the current partial multi-classifier.

If the score of the partial multi-classifier so far is smaller than the lower bound, the current branch of the branch-and-bound tree is abandoned. If the BDeu score is larger than the lower bound, the algorithm verifies that the treewidth of the classifier does not yet exceed k. Computing the treewidth of a partial multi-classifier being its most intensive task, the algorithm minimises the number of computations involved. Since the feature subgraph of a partial multi-classifier is empty, each feature variable X_i is connected with class parents only. After moralisation therefore, X_i and its parents constitute a (maximal) clique. From this property, we have that [3]:

$$\tau(P) = \max\{d, \tau(P \setminus \{X_i\})\}$$

where $\tau(P)$ is the treewidth of the multi-classifier so far, and d is the indegree of X_i; $P \backslash \{X_i\}$ is the graphical structure obtained by moralising P and subsequently removing X_i. From this property we find that the treewidth of the partial classifier P can be computed from the moralised subgraph of class variables only. If the treewidth of the partial multi-classifier exceeds k, the current branch of the branch-and-bound tree is abandoned. Otherwise, the associated node in the tree is expanded by adding the next feature variable; if its BDeu-score exceeds the current lower bound moreover, this bound is updated.

Step 3: In an optional post-processing step, the algorithm adds a forest-structured feature subgraph to the classifier. Given the learned class and bridge subgraphs, the algorithm greedily inserts arcs that serve to increase the overall BDeu score of the multi-classifier yet keep its treewidth smaller than k.

Although Step 1 of the learning algorithm yields an optimal class subgraph and Step 2 results in an optimal bridge subgraph given this class subgraph, the algorithm is not guaranteed to yield a multi-classifier of highest BDeu score. Since the class subgraph resulting from Step 1 may affect the treewidth of the model under construction, a multi-classifier with another, non-optimal class subgraph could have a higher BDeu score.

6 Experiments

We conducted a number of experiments with our algorithm for learning multi-classifiers of small treewidth. Before reporting the results obtained, we first discuss the general set-up of our experiments.

6.1 Data Sets and Baseline Characteristics

For our experiments, we used the data sets listed in Table 1; these sets are commonly used for multi-label classification. In a pre-processing step, numerical variables were discretised into four bins of equal size. We further performed feature selection to remove any irrelevant features. Since feature selection for multi-classifiers is an open problem, we used to this end the approach by Corani *et al.* [7], performing correlation-based feature selection per class variables and retaining the union of all selected feature variables.

Table 1. Data set properties, the treewidth (τ) of the unconstrained multi-classifier, and the baseline performance of the associated constant classifiers.

Data set	*Training*	*Test*	*Classes*	*Features*	*Selected*	τ	acc_H	acc_G
`Emotions`	391	202	6	72	19	3	0.67	0.11
`Scene`	1211	1196	6	294	119	4	0.82	0.16
`Yeast`	1500	917	14	103	39	4	0.77	0.10
`Genbase`	465	199	27	1186	72	7	0.95	0.27
`Medical`	645	333	45	1449	342	18	0.97	0.17
`Enron`	1123	579	53	1001	243	13	0.94	0.067

Prior to the experiments with our learning algorithm, we established baseline accuracies for the various data sets. For computing a baseline Hamming accuracy, a constant classifier was constructed which returns for each class variable separately the value that appears most often in the training set; for a baseline global accuracy, the constant classifier returns the class vector that appears most often. The baseline accuracies thus obtained are reported in Table 1. We further learned multi-classifiers without any restrictions on treewidth, allowing at most three parents per feature variable for reasons of feasibility. The treewidths of the resulting classifiers are also reported in Table 1. Since the treewidths of the multi-classifiers learned from the data sets `Emotions`, `Scene` and `Yeast` proved small, we decided to exclude these data sets from our further experiments.

We then studied the performance of our learning algorithm, both with and without using the option to add a feature subgraph to the multi-classifier under construction. The global and Hamming accuracies established from the test sets for the learned multi-classifiers were compared against those obtained from the classifiers without any restrictions on treewidth. All multi-classifiers were learned under the topological constraints introduced in Sect. 2. The BDeu scores were computed with GOBNILP [1], with an equivalent sample size of $\alpha = 5$. The software for our algorithm was written in Java.

6.2 Results

Although experiments were run with various small τ values, we report the results obtained with $\tau = 3$ only; the results with other small treewidths were similar.

The accuracies established from the respective test sets for the learned multi-classifiers are summarised in Table 2; the table also reports the accuracies of the classifiers that were learned without any bounds on treewidth. For each data set we used a Wilcoxon signed-rank test with $p < 0.05$ on both Hamming and global accuracy to detect significantly better performance of either type of classifier. With the `Medical` data set, the global accuracy of the unconstrained multi-classifier proved to be significantly better than that of the classifier of small treewidth. No further significant differences in performance were found. The total state space sizes κ of the junction-tree representations of the learned classifiers are also reported in Table 2. For the `Enron` and `Medical` data sets specifically,

Table 2. Accuracies of multi-classifiers with unbounded and small treewidth (τ) respectively, and the total state space size (κ) of the associated junction trees.

	Empty feature subgraph				*Forest-structured feature subgraph*			
Data set	τ	κ	acc_H	acc_G	τ	κ	acc_H	acc_G
`Genbase`	7	3396	0.999	0.965	13	142828	0.998	0.960
	3	1492	0.999	0.970	3	1520	0.999	0.970
`Medical`	18	2204040	0.987	**0.622**	36	$6.6 \cdot 10^{11}$	0.988	0.622
	3	5700	0.987	0.586	3	6048	0.987	0.595
`Enron`	13	82292	0.945	0.138	34	$1.8 \cdot 10^{9}$	0.946	0.130
	3	3996	0.945	0.143	3	4456	0.946	0.147

the differences in state space size between the small-treewidth classifier and the classifier of unbounded treewidth are quite large. By constraining treewidth, a reduction of the state space size by a factor 20 for `Enron` and by almost a factor 400 for `Medical` is achieved, indicating a major reduction of classification time. We further used the option to learn an additional forest-structured feature subgraph for our multi-classifiers, keeping treewidth within the same $\tau = 3$ bound. We compared the accuracies found with the learned classifiers against those obtained from similar multi-classifiers of unbounded treewidth. Table 2 reports the results obtained. Again applying, per data set, a Wilcoxon signed-rank test with $p < 0.05$ on both types of accuracy, revealed no significant differences. With respect to the differences in total state space size, the table shows again that by constraining treewidth major reductions of classification time are achieved.

The experimental results summarised in Table 2 suggest that adding a forest-structured feature subgraph to a learned multi-classifier does not significantly improve its performance. This finding may be explained by the fundamental property of Bayesian networks that direct probabilistic influences dominate over induced ones [11]. We consider as an example the simple network structure in Fig. 3 on the left, and study the influence of the feature variable X_q on the class variable C_j. Now, if an arc is added from X_p to X_q as shown in the figure on the right, entering a feature vector will induce an intercausal influence between X_p and C_j. This influence is known to be dominated by the direct influence from X_q on C_j. As it further tends to be weak, the induced influence in fact is not likely to cause large shifts in the probability distribution over the class variables.

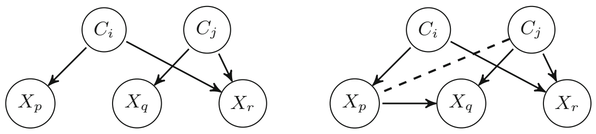

Fig. 3. The effect of an induced intercausal influence

7 Conclusions

Attributing their often high classification time to their large treewidth, we designed a branch-and-bound algorithm for learning multi-classifiers of small treewidth. For various well-known multi-label data sets, we showed that the performance of the resulting treewidth-constrained classifiers does not differ significantly from that of multi-classifiers without any bounds on treewidth. Our experimental results further showed that by constraining treewidth, major reductions of the runtime of classification are achieved. In future research, we will conduct a deeper study of the performance of our treewidth-constrained classifiers compared to unbounded multi-classifiers, especially in view of more informative feature subgraphs. We will further investigate whether our learning algorithm can be improved from a computational point of view to bring real-world application of multi-classifiers within closer reach.

References

1. Bartlett, M., Cussens, J.: Advances in Bayesian network learning using integer programming. In: Nicholson, A., Smyth, P. (eds.) Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, pp. 182–191. AUAI Press (2013)
2. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoret. Comput. Sci. **209**(1), 1–45 (1998)
3. Bodlaender, H.L., Koster, A.M., Van den Eijkhof, F., Van der Gaag, L.C.: Preprocessing for triangulation of probabilistic networks. In: Breese, J., Koller, D. (eds.) Proceedings of 17th Conference on Uncertainty in Artificial Intelligence, pp. 32–39. Morgan Kaufmann (2001)
4. Borchani, H., Bielza, C., Larrañaga, P.: Learning CB-decomposable multi-dimensional Bayesian network classifiers. In: Myllymaki, P., Roos, T., Jaakkola, T. (eds.) Proceedings of the 5th European Workshop on Probabilistic Graphical Models. pp. 25–32 (2010)
5. Buntine, W.: Theory refinement on Bayesian networks. In: Bonissone, P., D'Ambrosio, B., Smets, P. (eds.) Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence, pp. 52–60. Morgan Kaufmann (1991)
6. de Campos, C.P., Cuccu, M., Corani, G., Zaffalon, M.: Extended tree augmented naive classifier. In: van der Gaag, L.C., Feelders, A.J. (eds.) PGM 2014. LNCS, vol. 8754, pp. 176–189. Springer, Heidelberg (2014)
7. Corani, G., Antonucci, A., Mauá, D.D., Gabaglio, S.: Trading off speed and accuracy in multilabel classification. In: van der Gaag, L.C., Feelders, A.J. (eds.) PGM 2014. LNCS, vol. 8754, pp. 145–159. Springer, Heidelberg (2014)
8. Edmonds, J.: Optimum branchings. J. Res. Natl. Bur. Stan. B **71**(4), 233–240 (1967)
9. Van der Gaag, L.C., De Waal, P.R.: Multi-dimensional Bayesian network classifiers. In: Studeny, M., Vomlel, J. (eds.) Proceedings of the 3rd European Workshop on Probabilistic Graphical Models, pp. 107–114 (2006)
10. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. J. Roy. Stat. Soc. **50**(2), 157–224 (1988)

11. Renooij, S., Van der Gaag, L.C., Parsons, S.: Propagation of multiple observations in QPNs revisited. In: Van Harmelen, F. (ed.) Proceedings of the 15th European Conference on Artificial Intelligence. pp. 665–669 (2002)
12. Shimony, S.E.: Finding MAPs for belief networks is NP-hard. Artif. Intell. **68**(2), 399–410 (1994)