Learning Extended Tree Augmented Naive Structures[☆]

Cassio P. de Campos[a,*], Giorgio Corani[b], Mauro Scanagatta[b], Marco Cuccu[c], Marco Zaffalon[b]

[a] *Queen's University Belfast, UK*
[b] *Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Switzerland*
[c] *University of Lugano, Switzerland*

Abstract

This work proposes an extended version of the well-known tree-augmented naive Bayes (TAN) classifier where the structure learning step is performed without requiring features to be connected to the class. Based on a modification of Edmonds' algorithm, our structure learning procedure explores a superset of the structures that are considered by TAN, yet achieves global optimality of the learning score function in a very efficient way (quadratic in the number of features, the same complexity as learning TANs). We enhance our procedure with a new score function that only takes into account arcs that are relevant to predict the class, as well as an optimization over the equivalent sample size during learning. These ideas may be useful for structure learning of Bayesian networks in general. A range of experiments show that we obtain models with better prediction accuracy than Naive Bayes and TAN, and comparable to the accuracy of the state-of-the-art classifier averaged one-dependence estimator (AODE). We release our implementation of ETAN so that it can be easily installed and run within Weka.

Keywords: Bayesian networks; Structure Learning; Classification; Tree Augmented Naive Bayes; Edmonds' Algorithm.

1. Introduction

Classification is the problem of predicting the *class* of a given object on the basis of some attributes (*features*) of it. A classical example is the iris problem by Fisher: the goal is to correctly predict the *class*, that is, the species of iris on the basis of four features (sepal and petal length and width). In the Bayesian framework, classification is accomplished by updating a prior density (representing the beliefs before analyzing the data) with the likelihood (modeling the evidence coming from the data), in order to compute a posterior density, which is then used to select the most probable class.

The naive Bayes classifier [2] is based on the assumption of stochastic independence of the features given the class; since the real data generation mechanism usually does not satisfy such a condition, this introduces a bias in the estimated probabilities. Yet, at least under the zero-one accuracy, the naive Bayes classifier performs surprisingly well [2, 3]. Reasons for this phenomenon have been provided, among others, by Friedman [4], who proposed an approach to decompose the misclassification error into bias error and variance error; the bias error represents how closely the classifier approximates the target function, while the variance error reflects the sensitivity of the parameters of the classifier to the training sample. Low bias and low variance are two conflicting objectives; for instance, the naive Bayes classifier has high bias (because of the unrealistic independence assumption) but low variance, since it requires to estimate only a few parameters. A way to reduce the naive Bayes bias is to relax the independence assumption using a more

[☆]This paper is an extended and revised version of material originally presented in [1].
[*]Corresponding author
Email addresses: c.decampos@qub.ac.uk (Cassio P. de Campos), giorgio@idsia.ch (Giorgio Corani), mauro@idsia.ch (Mauro Scanagatta), marco.cuccu@usi.ch (Marco Cuccu), zaffalon@idsia.ch (Marco Zaffalon)

complex graph, like a tree-augmented naive Bayes (TAN) [5]. In particular, TAN can be seen as a Bayesian network where each feature has the class as parent, and possibly also a feature as second parent. In fact, TAN is a compromise between general Bayesian networks, whose structure is learned without constraints, and the naive Bayes, whose structure is determined in advance to be naive (that is, each feature has the class as the only parent). TAN has been shown to outperform the naive Bayes classifier in a range of experiments [5, 6, 7].

In this paper we develop an extension of TAN that allows it to have (i) features without the class as parent, (ii) multiple features with only the class as parent (that is, building a forest), (iii) features completely disconnected (that is, automatic feature selection). While the most common usage of this model is traditional classification, it represents a novel way to learn Bayesian network structures that extend current polynomial-time state-of-the-art methods. In this respect, learning TANs can be seen as the best low-complexity algorithm for exact learning of Bayesian networks. In spite of that, our extension of TAN can also be used as a component of a graphical model suitable for multi-label classification [8].

Extended TAN (or simply ETAN) is learned in quadratic time in the number of features, which is essentially the same computational complexity as that of TAN (our actual ETAN implementation has a $\log^*$ term, which can be neglected for any reasonable number of features). The goodness of each (E)TAN structure is assessed through a decomposable and likelihood equivalent score, such as the Bayesian Dirichlet likelihood equivalent uniform (BDeu) [9, 10, 11, 12]. Because ETAN's search space of structures includes that of TAN, the score of the best ETAN is always equal or superior to that of the best TAN. ETAN thus provides a better fit. However, it is well known that this fit does not necessarily imply higher classification accuracy [13]. To improve on ETAN as a classifier, we propose a new score function that takes into account only features that are not (conditionally) independent of the class (given the other features). ETAN under this new scoring idea is empirically shown to produce higher accuracy than Naive Bayes, TAN, and itself (using the standard BDeu).

We perform extensive experiments with these classifiers. We empirically show that ETAN yields in general better zero-one accuracy and log loss than TAN and naive Bayes (where log loss is computed from the posterior distribution of the class given features). Log loss is relevant in cases of cost-sensitive classification [14, 15]. We also study the possibility of optimizing the equivalent sample size of ETAN [16], which makes it perform similar to the averaged one-dependence estimator (AODE).

This paper is divided as follows. Section 2 introduces notation and defines the problem of learning Bayesian networks and the classification problem. Section 3 presents our new classifier and an efficient algorithm to learn it from data. Section 4 describes our experimental setting and discusses on empirical results. Finally, Section 5 concludes the paper and suggests possible future work.

2. Learning TANs and Classification

The classifiers that we discuss in this paper are all subcases of a Bayesian network. A Bayesian network represents a joint probability distribution over a collection of categorical random variables. It can be defined as a triple $(\mathcal{G}, \mathcal{X}, \mathcal{P})$, where $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ is a directed acyclic graph (DAG) with $V_\mathcal{G}$ a collection of nodes associated to random variables $\mathcal{X}$ (a node per variable), and $E_\mathcal{G}$ a collection of arcs; $\mathcal{P}$ is a collection of conditional mass functions $p(X_i|\Pi_i)$ (one for each instantiation of Π_i), where Π_i denotes the parents of X_i in the graph (Π_i may be empty), respecting the relations of $E_\mathcal{G}$. In a Bayesian network every variable is conditionally independent of its non-descendant non-parents given its parents (Markov condition). Because of the Markov condition, the Bayesian network represents a joint probability distribution by the expression $p(\mathbf{x}) = p(x_0, \ldots, x_n) = \prod_i p(x_i|\boldsymbol{\pi}_i)$, for every $\mathbf{x} \in \Omega_\mathcal{X}$ (space of joint configurations of variables), where every x_i and $\boldsymbol{\pi}_i$ are consistent with $\mathbf{x}$.

In the particular case of classification, the *class* variable X_0 has a special importance, as we are interested in its posterior probability which is used to predict unseen values; there are then several feature variables $\mathcal{Y} = \mathcal{X} \setminus \{X_0\}$. The supervised classification problem using probabilistic models is based on the computation of the posterior density, which can then be used to take decisions. The goal is to compute $p(X_0|\mathbf{y})$, that is, the posterior probability of the class variable given the values $\mathbf{y}$ of the features in a *test* instance. In this computation, p is defined by the model that has been learned from labeled data, that is, past data where class and features are all observed have been used to infer p. In order to do that, we are given a complete training data set $D = \{D_1, \ldots, D_N\}$ with N instances, where $D_u = \mathbf{x}_u \in \Omega_{\mathcal{X}}$ is an instantiation of all the variables, the first learning task is to find a DAG $\mathcal{G}$ that maximizes a given score function, that is, we look for $\mathcal{G}^* = \mathrm{argmax}_{\mathcal{G}\in\boldsymbol{\mathcal{G}}}\, s_D(\mathcal{G})$, with $\boldsymbol{\mathcal{G}}$ an arbitrary set of DAGs with nodes $\mathcal{X}$, for a given score function s_D (the dependency on data is indicated by the subscript D).[1]

For the purpose of this work, we assume that the employed score is decomposable and respects likelihood equivalence. Decomposable means it can be written in terms of the local nodes of the graph, that is, $s_D(\mathcal{G}) = \sum_{i=0}^{n} s_D(X_i, \Pi_i)$. Likelihood equivalence means that if $\mathcal{G}_1 \neq \mathcal{G}_2$ are two arbitrary graphs over $\mathcal{X}$ such that both encode the very same conditional independences among variables, then s_D is likelihood equivalent if and only if $s_D(\mathcal{G}_1) = s_D(\mathcal{G}_2)$.

The naive Bayes structure is defined as the network where the class variable X_0 has no parents and every feature (the other variables) has X_0 as sole parent. Figure 1(b) illustrates the situation. In this case, there is nothing to be learned in terms of structure, as it is fully defined by the restrictions of the naive Bayes. In spite of that, we define $\mathcal{G}^*_{\text{naive}}$ as being its (fixed) graph, for ease of exposition.

As with the naive Bayes, in a TAN structure, the class X_0 has no parents, while features must have the class as parent and are forced to have one other feature as parent too (except for one single feature, which has only the class as parent and is considered the root of the features' tree). Figure 1(c) illustrates a TAN structure, where X_1 has only X_0 as parent, while both X_2 and X_3 have X_0 and X_1 as parents. By ignoring X_0 and its connections, we have a tree structure, and that is the reason for the name TAN. Based on a decomposable and likelihood equivalent score function, an efficient algorithm for TAN can be devised. Because of the likelihood equivalence and the fact that every feature has X_0 as parent, the same score is obtained whether a feature X_i has X_0 and X_j as parent (with $i \neq j$), or X_j has X_0 and X_i (using the arc of opposite direction), that is,

$$s_D(X_i, \{X_0, X_j\}) + s_D(X_j, \{X_0\}) = s_D(X_j, \{X_0, X_i\}) + s_D(X_i, \{X_0\}) \ . \tag{1}$$

This symmetry allows for a very simple and efficient algorithm [17] that is proven to find the TAN structure which maximizes any decomposable and likelihood equivalent score, that is, to find

$$\mathcal{G}^*_{\text{TAN}} = \operatorname*{argmax}_{\mathcal{G}\in\boldsymbol{\mathcal{G}}_{\text{TAN}}} s_D(\mathcal{G}) \ , \tag{2}$$

where $\boldsymbol{\mathcal{G}}_{\text{TAN}}$ is the set of all TAN structures with nodes $\mathcal{X}$. The idea is to find the minimum spanning tree in an undirected graph defined over $\mathcal{Y}$ such that the weight of each edge (X_i, X_j) is defined by $w(X_i, X_j) = -(s_D(X_i, \{X_0, X_j\}) - s_D(X_i, \{X_0\}))$. Note that $w(X_i, X_j) = w(X_j, X_i)$. Without loss of generality, let X_1 be the only node without a feature as parent (one could rename the nodes and apply the same reasoning). Let $c_0 = s_D(X_0, \emptyset)$. Now,

$$\begin{aligned}
\max_{\mathcal{G}\in\boldsymbol{\mathcal{G}}_{\text{TAN}}} s_D(\mathcal{G}) &= c_0 + \max_{\Pi'_i:\forall i>1} \left(\sum_{i=2}^{n} s_D(X_i, \{X_0, X_{\Pi'_i}\}) + s_D(X_1, \{X_0\}) \right) \\
&= c_0 + s_D(X_1, \{X_0\}) - \min_{\Pi'_i:\forall i>1} \left(- \sum_{i=2}^{n} s_D(X_i, \{X_0, X_{\Pi'_i}\}) \right) \\
&= c_0 + \sum_{i=1}^{n} s_D(X_i, \{X_0\}) - \min_{\Pi'_i:\forall i>1} \sum_{i=2}^{n} w(X_i, X_{\Pi'_i}) \ .
\end{aligned} \tag{3}$$

This last minimization is exactly the minimum spanning tree problem, and the argument that minimizes it is the same as the argument that maximizes (2). Because this algorithm has to initialize the $\Theta(n^2)$ edges between every pair of features and then to solve the minimum spanning tree (e.g. using Prim's algorithm),

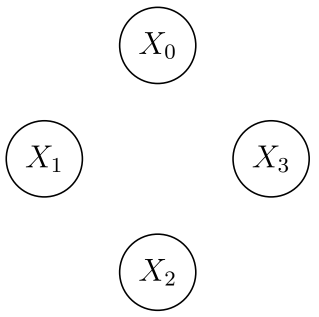
[1]In case of many optimal DAGs, then we assume to have no preference and argmax returns one of them.

its overall complexity time is $O(n^2)$, if one assumes that the score function is given as an oracle whose queries take time $O(1)$. In fact, because we only consider at most one or two parents for each node (two only if we include the class), the computation of the whole score function can be done in time $O(Nn^2)$ and stored for later use. As a comparison, naive Bayes can be implemented in time $O(Nn)$, while the averaged one-dependence estimator (AODE) [18] needs $\Theta(Nn^2)$, just as TAN does.

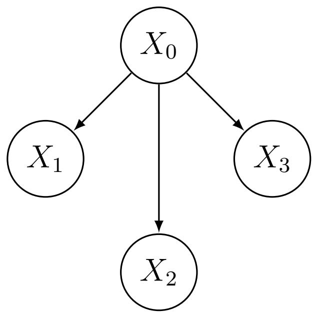
2.1. Improving Learning of TANs

A simple extension of this algorithm can already learn a forest of tree-augmented naive Bayes structures. One can simply define the edges of the graph over $\mathcal{Y}$ as in the algorithm for TAN, and then remove those edges (X_i, X_j) such that $s_D(X_i, \{X_0, X_j\}) \leq s_D(X_i, \{X_0\})$, that is, when $w(X_i, X_j) \geq 0$, and then run the minimum spanning tree algorithm over this reduced graph. The optimality of such an idea can be easily proven by the following lemma, which guarantees that we should use only X_0 as parent of X_i every time such choice is better than using $\{X_0, X_j\}$. It is a straightforward generalization of Lemma 1 in [19, 20].

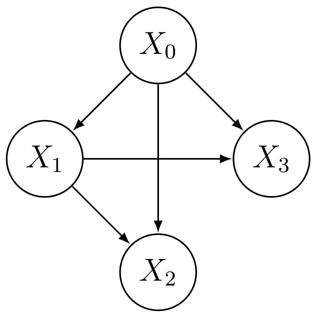
Lemma 1. *Let X_i be a node of $\mathcal{G}$, a candidate DAG where the parent set of X_i is Π'_i. Suppose $\Pi_i \subset \Pi'_i$ is such that $s_D(X_i, \Pi_i) \geq s_D(X_i, \Pi'_i)$, where s_D is a decomposable score function. If Π'_i is the parent set of X_i in an optimal DAG, then the same DAG but having Π_i as parent of X_i is also optimal.*



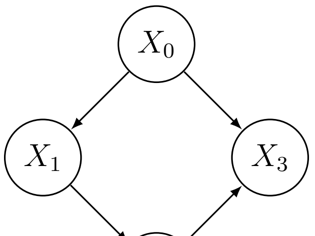
(a) Possible with ETAN.



(b) Possible with naive or ETAN.



(c) Possible with TAN or ETAN.



(d) Possible only with ETAN.

Figure 1: Some examples of structures allowed by the different classifiers. The labels indicate which classifier allows them as part of their whole structure.

Using a forest as structure of the classifier is not new (see Chapter 26.3 in [21]). We want to go even further and allow situations as in Figs. 1(a) and 1(d). The former would automatically disconnect a feature X_i if that maximizes the overall score of the structure. The second case (Fig. 1(d)) allows some features to have another feature as parent without the need of having also the class. For this purpose, we define the set of structures named Extended TAN (or ETAN for short), as DAGs such that X_0 has no parents and X_i

($i \neq 0$) is allowed to have the class and at most one feature as parent (but it is not obliged to having any of them), that is, the parent set Π_i is such that $|\Pi_i| \leq 1$, or $|\Pi_i| = 2$ and $\Pi_i \supseteq \{X_0\}$.

$$\mathcal{G}^*_{\text{ETAN}} = \operatorname*{argmax}_{\mathcal{G} \in \boldsymbol{\mathcal{G}}_{\text{ETAN}}} s_D(\mathcal{G}) \ . \tag{4}$$

This is clearly a generalization of TAN, of the forest of TANs, and of naive Bayes in the sense that they are all subcases of ETAN. Note that TAN is not a generalization of naive Bayes in this same reasoning, as TAN forces arcs among features even if these arcs were not useful (thus a naive Bayes structure is not a particular case of TAN). Because of that, we have the following result.

Lemma 2. *The following relations among subsets of DAGs hold.*

$$s_D(\mathcal{G}^*_{ETAN}) \geq s_D(\mathcal{G}^*_{TAN}) \quad \textit{and} \quad s_D(\mathcal{G}^*_{ETAN}) \geq s_D(\mathcal{G}^*_{naive}) \ .$$

We point out that $\mathcal{G}^*_{\text{ETAN}}$ cannot be found anymore using the same procedure as for TAN with the minimum spanning tree algorithm, because for two nodes X_i and X_j we may have

$$s_D(X_i, \Pi_i) + s_D(X_j, \{X_0\}) \neq s_D(X_j, \Pi_j) + s_D(X_i, \{X_0\}) \ , \tag{5}$$

where $\Pi_i \supseteq \{X_j\}$ and $\Pi_j \supseteq \{X_i\}$, given that X_0 is not necessarily present in Π_i or Π_j. We will see in the next section that Edmonds' algorithm for directed spanning trees suffices.

For the purpose of classification with complete data, one could argue that the arc $X_1 \rightarrow X_2$ in Fig. 1(d) is irrelevant because X_1 d-separates X_2 and X_0 through such path, and so the local score of X_2 should not be taken into account, as the choice of parent set for X_2 is irrelevant to predicting X_0 (when testing data are incomplete, then that arc can be relevant to the prediction of X_0 as X_1 may be missing). In other words, the graph of Fig. 1(d) would issue the very same predictions about X_0 with or without the arc $X_1 \rightarrow X_2$. To address this issue and to only account for arcs that are useful for predicting a target variable X_0, we propose the following *prediction-targeted* score function to learn Bayesian networks that is particularly suitable for classification.

$$s^{\text{dsep}}_D(\mathcal{G}) = s_D(\mathcal{G}) - \mathcal{I}(\mathcal{G}), \tag{6}$$

where $\mathcal{I}(\mathcal{G})$ is ∞ if there is $\mathcal{G}' \subset \mathcal{G}$ over the very same set of nodes such that $p_{\mathcal{G}'}(X_0|\mathbf{y}) = p_{\mathcal{G}}(X_0|\mathbf{y})$ for every $\mathbf{y} \in \Omega_{\mathcal{Y}}$, and zero otherwise. At first it might seem harder to deal with this new score, because we need to figure out which arcs are irrelevant. However, identifying irrelevant arcs is trivial: they are exactly those arriving to nodes that do not have the class as parent, as every path between class and a node using these arcs will be d-separated by the other features. Hence we can compute with the following criterion.

$$\mathcal{G}^*_{\text{S-ETAN}} = \operatorname*{argmax}_{\mathcal{G}} s^{\text{dsep}}_D(\mathcal{G}) = \operatorname*{argmax}_{\substack{\mathcal{G}:\ \forall i:\ \Pi_i = \emptyset \text{ or} \\ (|\Pi_i| \leq 2 \text{ and } \Pi_i \supseteq \{X_0\})}} \left(c_0 + \sum_{i=1}^{n} s_D(X_i, \Pi_i) \right) \ . \tag{7}$$

This expression is a summation of the score for each node, with some constraints about which parent sets are allowed (we keep the score of the class inside the formula even if it is not important for the argmax – this is useful when one wants to use different prior strengths in the BDeu score, as we will discuss later). So in order to compute with (7), we simply need to appropriately restrict the parent sets that are allowed in the construction of the Edmonds' algorithm input, as we describe later on. We emphasize that learning ETANs under s^{dsep}_D is not the same as learning TANs, as it might look (it seems that all features need to have the class as parent – that is not the case). There are features which are relevant (that is, not d-separated from the class given the other features) and yet are not linked to the class. This is for example the case of X_2 in Fig. 1(d): under s^{dsep}_D, X_2 should have no parent there, but the observation in X_3 makes it nevertheless relevant to the prediction.

3. Learning Extended TANs

The goal of this section is to present an efficient algorithm to find the DAG defined in (4) and in (7). Unfortunately the undirected version of the minimum spanning tree problem is not enough, because (1) does not hold anymore. To see that, take the example in Fig. 1(d). The arc from X_1 to X_2 cannot be reversed without changing the overall score (unless we connect X_0 to X_2). In other words, every node in a TAN has the class as parent, which makes possible to use the minimum spanning tree algorithm for undirected graphs by realizing that any orientation of the arcs between features will produce the same overall score (as long as the weights of the edges are defined as in the previous section).

Edmonds' algorithm [22] (also attributed to Chu and Liu [23]) for finding minimum spanning arborescence in directed graphs comes to our rescue. Its application is however not immediate, and its implementation is not as simple as the minimum spanning tree algorithm for TAN. Our algorithm to learn ETANs is presented in Algorithm 1. It is composed of a preprocessing of the data to create the arcs of the graph that will be given to Edmonds' algorithm for directed minimum spanning tree (in fact, we assume that Edmonds' algorithm computes the directed maximum spanning tree, which can be done trivially by negating all weights). `EdmondsContract` and `EdmondsExpand` are the two main steps of that algorithm, and we refer the reader to the description in Zwick's lecture notes [24] or to the work of Tarjan [25] and Camerini et al. [26] or Gabow et al. [27] for further details on the implementation of Edmonds' idea. In fact, we have not been able to find a stable and reliable implementation of such algorithm, so our own implementation of Edmonds' algorithm has been developed based on the description in [24], even though some fixes had to be applied. Because Edmonds' algorithm finds the best spanning tree for a given "root" node (that is, a node that is constrained not to have features as parents), Algorithm 1 loops over the possible roots and extract from Edmonds' the best parent for each node given that fixed root node (line 6), and then stores the best solution over all such possible root nodes. At each loop, Algorithm 3 is called and builds a graph using the information from the result of Edmonds'. Algorithm 1 also loops over a set of score functions that are given to it. This is used later on to optimize the value of the equivalent sample size in each of the learning steps by giving a list of scores with different prior strengths to the algorithm.

Algorithm 1 ETAN($\mathcal{X}, S$): $\mathcal{X}$ are variables and S is a set of score functions

```
1: $s^* \leftarrow -\infty$
2: for all $s_D \in S$ do
3:     $(\text{arcs}, \text{classAsParent}) \leftarrow \text{ArcsCreation}(\mathcal{X}, s_D)$
4:     $\text{EdmondsContract}(\text{arcs})$
5:     for all $\text{root} \in \mathcal{X} \setminus \{X_0\}$ do
6:         $\text{in} \leftarrow \text{EdmondsExpand}(\text{root})$
7:         $\mathcal{G} \leftarrow \text{buildGraph}(\mathcal{X}, \text{root}, \text{in}, \text{classAsParent})$
8:         if $s_D(\mathcal{G}) > s^*$ then
9:             $\mathcal{G}^* \leftarrow \mathcal{G}$
10:            $s^* \leftarrow s_D(\mathcal{G})$
11: return $\mathcal{G}^*$
```

The particular differences with respect to a standard call of Edmonds' algorithm are defined by the methods `ArcsCreation` and `buildGraph`. The method `ArcsCreation` is the algorithm that creates the directed graph that is given as input to Edmonds'. The overall idea is that we must decide whether the class should be a parent of a node or not, and whether it is worth having a feature as a parent. The core argument is again given by Lemma 1. If $s_D(X_i, \{X_0\}) \leq s_D(X_i, \emptyset)$, then we know that no parent is preferable to having the class as a parent for X_i. We store this information in a matrix called `classAsParent` (line 2 of Algorithm 2). Because this information is kept for later reference, we can use from that point onward the value $\max(s_D(X_i, \emptyset), s_D(X_i, \{X_0\}))$ as the weight of having X_i with only the class as parent (having or not the class as parent cannot create a cycle in the graph, so we can safely use this max value). After that, we loop over every possible arc $X_j \to X_i$ between features, and define its weight as the maximum between having X_0 also as parent of X_i or not, minus the value that we would achieve for X_i if we did not include X_j as its parent (line 8). This is essentially the same idea as done in the algorithm of TAN, but here we must consider both $X_j \to X_i$ and $X_i \to X_j$, as they are not necessarily equivalent (this happens for instance if for one of the two features the class is included in its parent set and for the other it is not, depending on the maximization, so scores defining the weight of each arc direction might be different). After that, we also keep track of whether the class was included in the definition of the weight of the arc or not, storing the information in `classAsParent` for later recall. In case the weight is not positive (line 9), we do not even include this arc in the graph that will be given to Edmonds' (recall we are using the maximization version of Edmonds'), because at this early stage we already know that either no parents for X_i or only the class as parent of X_i (which one of the two is the best can be recalled in `classAsParent`) are better than the score obtained by including X_j as parent, and using once more the arguments of Lemma 1 and the fact that the class as parent never creates a cycle, we can safely disregard X_j as parent of X_i. All these cases can be seen in Fig. 1 by considering that the variable X_2 shown in the figure is our X_i. There are four options for X_i: no parents (a), only X_0 as parent (b), only X_j as parent (d), and both X_j and X_0 (c). The trick is that Lemma 1 allows us to reduce these four options to two: best between (a) and (b), and best between (c) and (d). After the arcs with positive weight are inserted in a list of arcs that will be given to Edmonds' and `classAsParent` is built, the algorithm ends returning both of them. In order to compute with S-ETAN, that is, ETAN using the score s_D^{dsep}, we only need to define in the line 7 of Algorithm 2 that $s_D(X_i, \{X_j\})$ equals $-\infty$ (this is because such an arc would be irrelevant to the prediction of the class unless X_0 is also a parent of X_i).

Algorithm 2 ArcsCreation($\mathcal{X}, s_D$)

```
1: for all $X_i \in \mathcal{X} \setminus \{X_0\}$ do
2:     classAsParent[$X_i$] $\leftarrow s_D(X_i, \{X_0\}) > s_D(X_i, \emptyset)$
3: arcs $\leftarrow \emptyset$
4: for all $X_i \in \mathcal{Y}$ do
5:     for all $X_j \in \mathcal{Y}$ do
6:         twoParents $\leftarrow s_D(X_i, \{X_0, X_j\})$
7:         onlyFeature $\leftarrow s_D(X_i, \{X_j\})$
8:         $w \leftarrow$ max(twoParents, onlyFeature) $-$ max($s_D(X_i, \emptyset), s_D(X_i, \{X_0\})$)
9:         if $w > 0$ then
10:            Add $X_j \to X_i$ with weight $w$ into arcs
11:            classAsParent[$X_j \to X_i$] $\leftarrow$ twoParents $>$ onlyFeature
12:        else
13:            classAsParent[$X_j \to X_i$] $\leftarrow$ classAsParent[$X_i$]
14: return (arcs, classAsParent)
```

Finally, Algorithm 3 is responsible for building back the best graph from the result obtained by Edmonds'. Inside `in` is stored the best parent for each node, and `root` indicates a node that shall have no other feature as parent. The goal is to recover whether the class shall be included as parent of each node, and for that we use the information in `classAsParent`. The algorithm is quite straightforward: for each node that is not the root and has a parent chosen by Edmonds', include it as parent each check if that arc was associated to having or not the class (if it had, include also the class); for each node that has no parent as given by Edmonds' (including the root node), simply check whether it is better to have the class as parent.

Somewhat surprisingly, learning ETANs can be accomplish in time $O(n^2)$ (assuming that the score function is given as an oracle, as discussed before), the same complexity for learning TANs. Algorithm 2 takes $O(n^2)$, because it loops over every pair of nodes and only performs constant time operations inside the loop. `EdmondsContract` can be implemented in time $O(n^2)$ and `EdmondsExpand` in time $O(n)$ [25, 26]. Finally, `buildGraph` takes time $O(n)$ because of its loop over nodes, and the comparison between scores of two ETANs as well as the copy of the structure of an ETANs takes time $O(n)$. So the overall time of the loop in Algorithm 1 takes time $O(n^2)$. Our current implementation can be found at `http://ipg.idsia.ch/software`. We

Algorithm 3 buildGraph($\mathcal{X}$, root, in, classAsParent)

```
1: $\mathcal{G} \leftarrow (\mathcal{X}, \emptyset)$
2: for all node $\in \mathcal{X} \setminus \{X_0\}$ do
3:     $\Pi_{\text{node}} \leftarrow \emptyset$
4:     if node $\neq$ root and in[node] $\neq$ null then
5:         $\Pi_{\text{node}} \leftarrow \Pi_{\text{node}} \cup \{\text{in[node]}\}$
6:         if classAsParent[in[node] $\rightarrow$ node] then
7:             $\Pi_{\text{node}} \leftarrow \Pi_{\text{node}} \cup \{X_0\}$
8:     else if classAsParent[node] then
9:         $\Pi_{\text{node}} \leftarrow \Pi_{\text{node}} \cup \{X_0\}$
10: return $\mathcal{G}$
```

have also re-implemented the TAN algorithm of Weka, as the standard version available with the data mining package had cubic time complexity in n instead of quadratic, because of a naive algorithmic choice. Hence, all experiments with TAN refer to our optimized implementation using a quadratic time (in n) algorithm for minimum spanning tree.

4. Experiments

This section presents results with TAN, ETAN and AODE using 60 data sets from the UCI machine learning repository [28]. We release our implementation of ETAN so that it can be easily installed and run within Weka. Data sets with many different characteristics have been used. Data sets containing continuous variables have been discretized in two bins, using the median as cut-off (we do not further investigate the impact of this decision, but there is no reason to believe that it yields an advantage to any of the methods). Missing values have been imputed by mean or mode (according to the variable type). The results are obtained out of 10 runs of 10-fold cross-validation (each run splits the data into folds randomly and in a stratified way), so the learning procedure of each classifier is called 100 times per data set. For learning the classifiers we use the Bayesian Dirichlet equivalent uniform (BDeu) and assume parameter independence and modularity [11]. The BDeu score computes a function based on the posterior probability of the structure $p(\mathcal{G}|D)$:

$$s_D(\mathcal{G}) = \log\left(p(\mathcal{G}) \cdot \int p(D|\mathcal{G}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}|\mathcal{G}) d\boldsymbol{\theta}\right) ,$$

where the logarithm is used to simplify computations, $p(\boldsymbol{\theta}|\mathcal{G})$ is the prior of $\boldsymbol{\theta}$ (vector of parameters of the Bayesian network) for a given graph $\mathcal{G}$, assumed to be a symmetric Dirichlet. BDeu respects likelihood equivalence and its function is decomposable. The only free parameter is the prior strength α (assuming $p(\mathcal{G})$ is uniform), also known as the equivalent sample size (ESS). We make comparisons using α equal to two (arguably one of the most used values in the literature). As described before, we implemented ETAN with the *prediction-targeted* score, which we call S-ETAN. Afterwards, it is also evaluated in a way that α is optimized over the set $\{1, 2, 5, 10, 20, 30, 50, 70\}$ and chosen according to the value that achieves the highest BDeu score, that is, the learner attempts multiple BDeu score functions with different values of α. This version will be called SA-ETAN.

Table 1: Number of wins, ties and losses of ETAN over competitors in terms of BDeu score, and p-values using the Wilcoxon signed rank test on 60 data sets (one-sided in the direction of the median difference).

	BDeu (ETAN vs. competitor)	
Learner	W/T/L	p-value
Naive	60/0/0	8e-12
TAN	58/2/0	1e-11

As previously demonstrated, ETAN always obtains better BDeu score than its competitors. TAN is usually better than naive Bayes, but there is no theoretical guarantee it will always be the case. Table 1 shows the comparisons of BDeu scores achieved by different classifiers. It presents the number of wins, ties and losses of ETAN against the competitors when comparing the BDeu scores for each data set (internally averaged over the cross validation runs), and finally the p-value from the Wilcoxon signed rank test for the 60 data points (one-sided in the direction of the median value). The statistical test indicates that the score achieved by ETAN is significantly superior than scores of the other methods. We note that ETAN may lose against others only if the values of α are different, otherwise ETAN is provenly equal to or better than Naive and TAN.

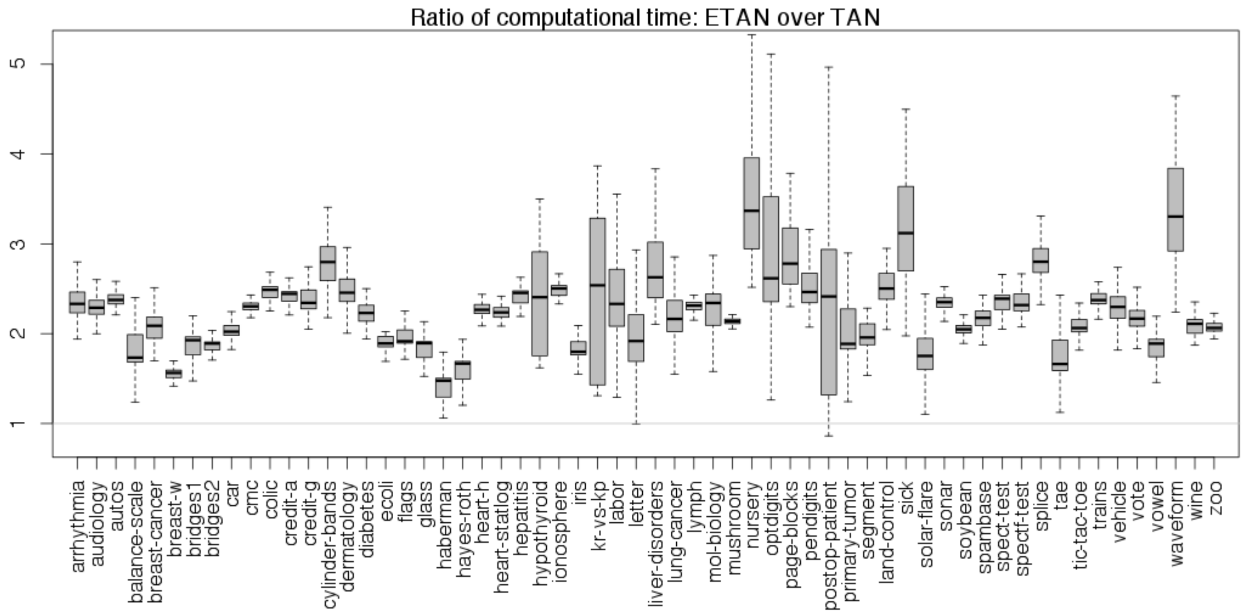


Figure 2: Computational time to learn the classifier as a ratio ETAN time divided by TAN time, so higher values mean ETAN is slower.

Figure 2 shows the computational time cost to run the learning in the 100 executions per data set for ETAN and TAN. All experiments were run in a couple of hours in a modern computer. We can see in the graph that learning ETAN has been less than five times slower than learning TAN in all situations (and almost always less than four times slower). S-ETAN runs in the same (or faster) time as ETAN, while SA-ETAN is around eight times slower because it needs to optimize over α. We recall that all these classifiers can be run in quadratic time in the number of features and linear in the sample size, which is asymptotically as efficient as other state-of-the-art classifiers, such as the averaged one-dependence estimator (AODE) [18]. In fact, TAN and all versions of ETAN learn in quadratic time in the number of features (we are ignoring a $\log^*$ term in our implementation, which in fact could be further improved, but it is nevertheless smaller than 5 for any reasonable number of features), but they evaluate new instances in linear time (and ETAN is as fast as TAN for that), while AODE is quadratic also there. Hence, ETAN is equal to TAN and faster than AODE if there are many more instances to evaluate than to train the model.

In terms of accuracy, we compare classifiers by measuring their zero-one loss and log loss. Zero-one loss is the number of incorrectly classified instances divided by the total number of instances, while log loss equals minus the sum (over the testing instances) of the log-probability of the class (according to the classifier) given the instance's features.

We start by comparing ETAN with TAN (we omit further comparisons with Naive Bayes because all other classifiers are superior to it in our experiments). In zero-one loss, ETAN against TAN achieves

Wins/Ties/Losses (W/T/L – where *win* means better result, not necessarily smaller or larger value) of 23/5/32 with median 0.1178 and p-value 0.047 (all tests are Wilcoxon signed rank on 60 datasets), while in log loss it gets W/T/L of 33/3/24, median -0.0067 and p-value 0.006. That is, ETAN is superior in log loss but inferior in zero-one loss to TAN. Figures 3 and 4 show boxplots, each regarding one data set and considering 100 points defined by the runs of cross-validation.

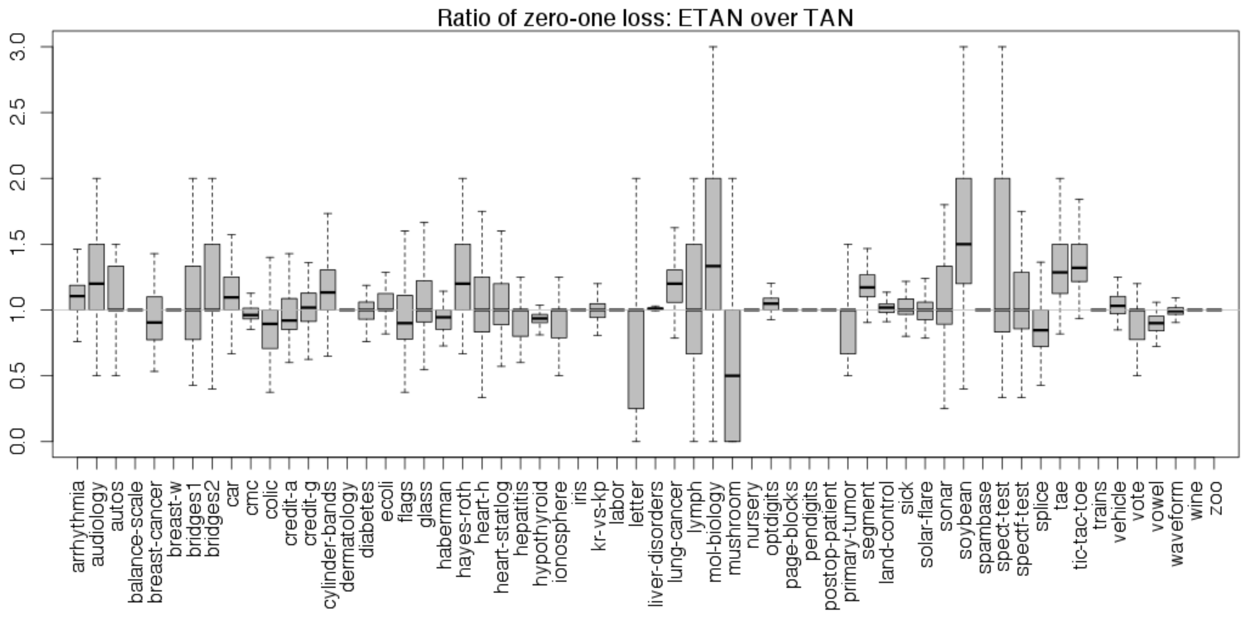


Figure 3: Comparison of zero-one loss between ETAN and TAN. Values are ratios of the loss of ETAN divided by TAN's loss, so smaller values mean ETAN is better.

Given the results of ETAN and TAN, we decide to verify the gains that S-ETAN can produce when compared to ETAN, that is, we want to check whether S-ETAN yields both zero-one loss and log loss improvements. Comparing scores of S-ETAN with the others is not appropriate, as they are not directly compatible. The comparison of losses is shown in Table 2. We see that S-ETAN has significant better log loss than TAN and significant better zero-one loss than ETAN. It is however significantly less accurate than AODE in log loss and marginally better in zero-one loss.

Table 2: Median value of the difference S-ETAN minus competitor (negative means S-ETAN is better), followed by number of wins, ties and losses of S-ETAN over competitors, and p-values using the Wilcoxon signed rank test on 60 data sets (one sided in the direction of the median difference).

S-ETAN vs.	Zero-one loss			Log loss		
competitor	Median	W/T/L	p-value	Median	W/T/L	p-value
Naive	-0.8679	40/1/19	0.002	-0.1043	47/0/13	2e-6
TAN	0.0000	26/8/26	0.415	-0.0002	33/5/22	0.015
ETAN	0.0000	29/12/19	0.013	0.0000	26/11/23	0.259
AODE	-0.0425	32/1/27	0.662	0.0279	22/0/38	0.013

In order to understand the effect of α, we perform our final comparison by including in S-ETAN the optimization over the equivalent sample size α. The results for SA-ETAN are shown in Table 3. SA-ETAN drastically improves over S-ETAN in log loss while keeping zero-one loss slightly inferior (not significant). We see that SA-ETAN produces a compromise between the loss functions. It is superior to ETAN in both

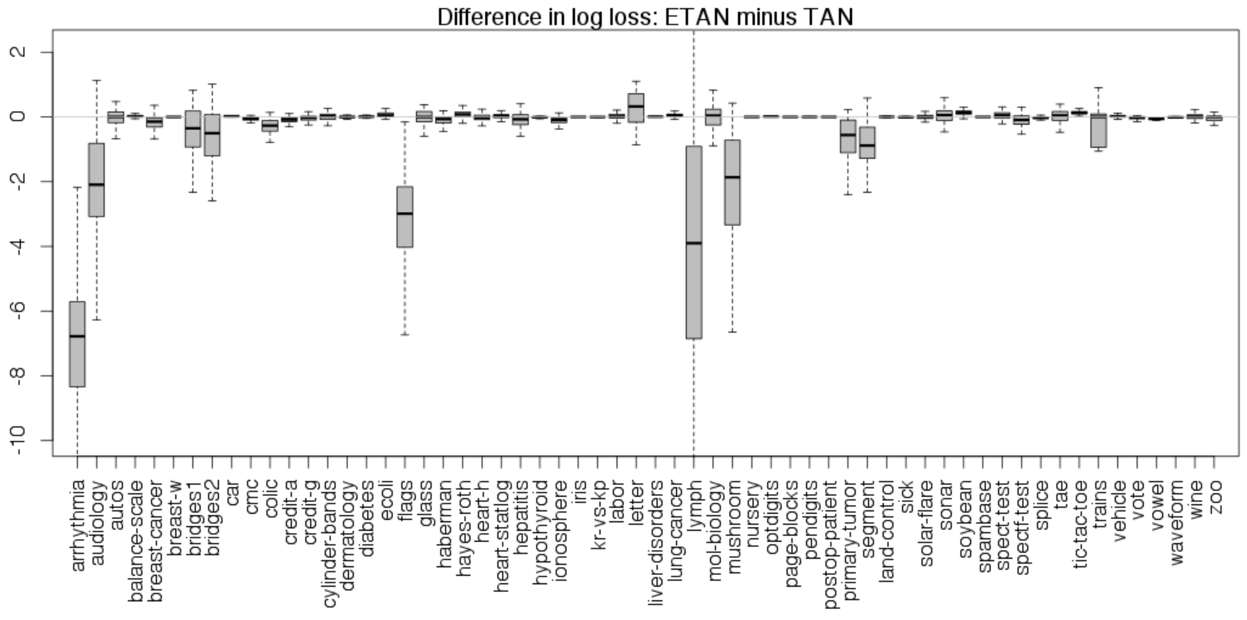


Figure 4: Comparison of log loss between ETAN and TAN. Values are loss of ETAN minus TAN's loss, so smaller values mean ETAN is better.

criteria, and greatly superior to TAN in log loss, although similar in zero-one loss. Finally, even if SA-ETAN achieves worse accuracy than AODE, the latter is only marginally better (not significant). In fact the tables suggest that the improvement of optimizing over α is not so large, but just about enough to make the classifier closer to the log loss measure achieved by AODE. Figures 5 and 6 show the comparison of SA-ETAN with AODE. Hence the choice between SA-ETAN and S-ETAN (or in other words between optimizing α or not) seems to related to one's goal between improving log loss or zero-one loss (with a trade-off between the two of them).

Table 3: Median value of the difference SA-ETAN minus competitor (negative means SA-ETAN is better), followed by number of wins, ties and losses of SA-ETAN over competitors, and p-values using the Wilcoxon signed rank test on 60 data sets (one sided in the direction of the median difference).

SA-ETAN vs.	Zero-one loss			Log loss		
competitor	Median	W/T/L	p-value	Median	W/T/L	p-value
TAN	0.0009	28/1/31	0.556	-0.0178	46/0/14	7e-7
ETAN	-0.0593	33/3/24	0.040	-0.0043	35/1/24	0.138
S-ETAN	0.0006	23/7/30	0.532	-0.0101	43/5/12	4e-5
AODE	0.0299	28/1/31	0.274	0.0080	23/0/37	0.122

Our final experimental setup considers missing data in the testing set. We assume the training data are complete, as the goal is not to harden learning but to verify the conjecture that the new prediction targeted score is not suitable for this case, as with missing data in the testing set one cannot tell which arcs are important for the prediction of the class before hand (missing values may make every arc relevant). We perform the same experiments as before with 10-fold cross validation, but prior to running the classifier on the testing fold we uniformly generate missing data in there using a desired percentage of missing values (our procedure is such that the very same data with the same missing values is used across competing classifiers). Results are shown in Table 4 and confirm the expected outcome: A-ETAN is superior to SA-ETAN, that is, using the usual BDeu score is better than using the prediction targeted score in the presence of missing

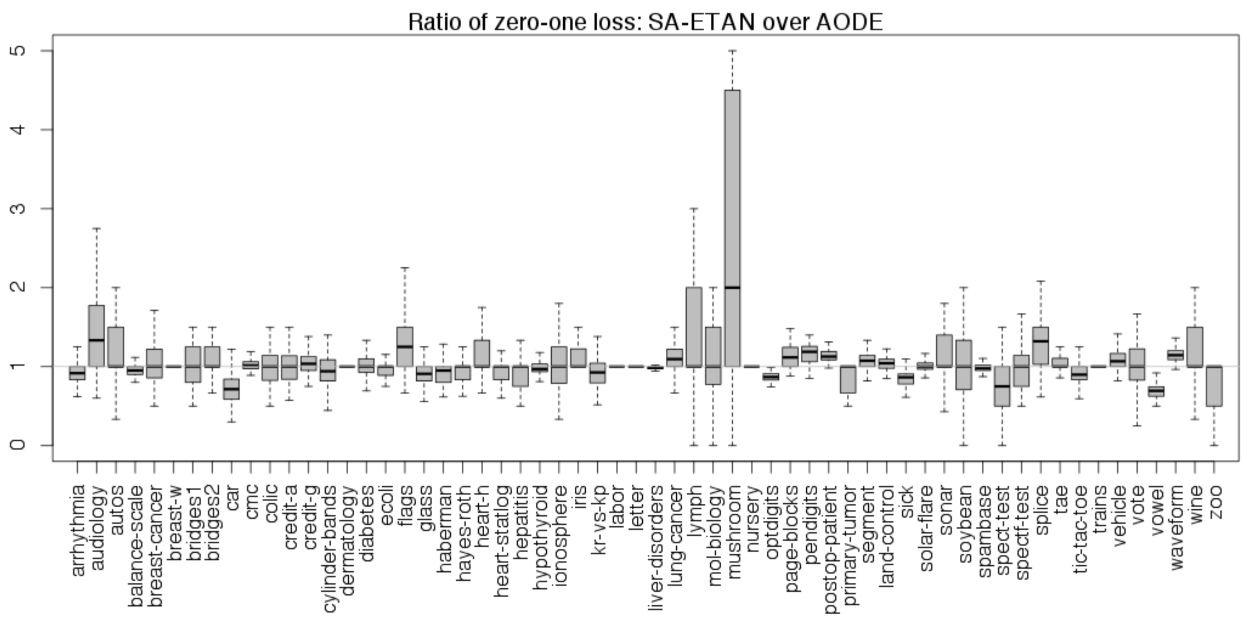


Figure 5: Comparison of zero-one loss between SA-ETAN and AODE. Values are ratios of the loss of SA-ETAN divided by AODE's loss, so smaller values mean SA-ETAN is better.

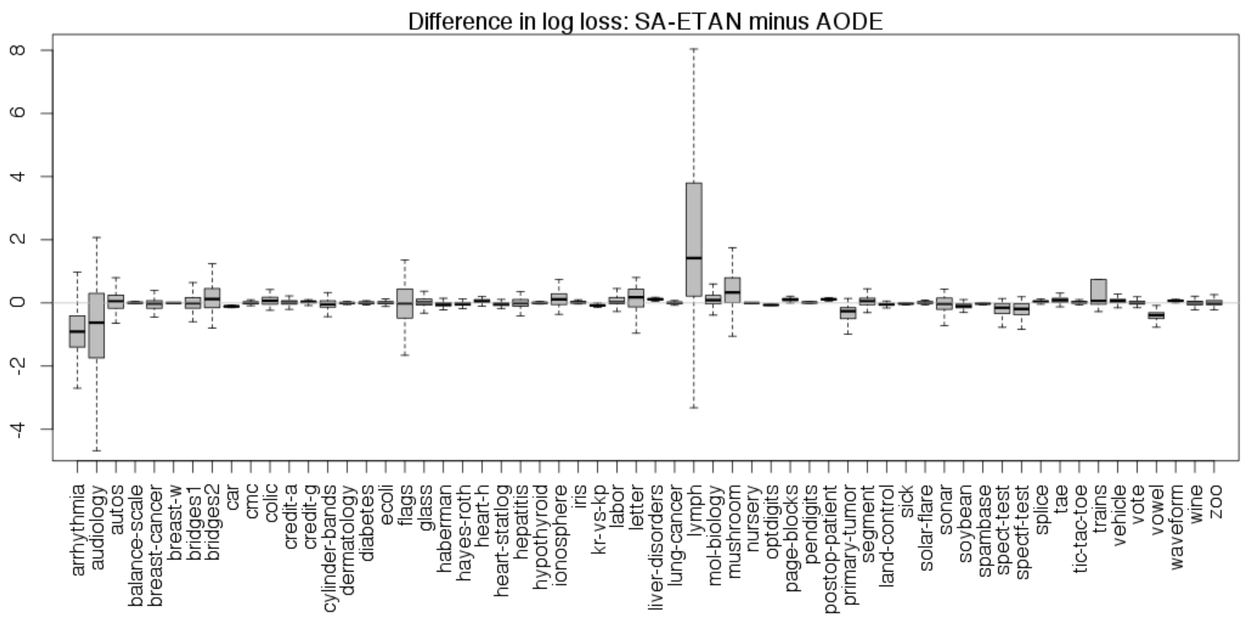


Figure 6: Comparison of log loss between SA-ETAN and AODE. Values are loss of SA-ETAN minus AODE's loss, so smaller values mean SA-ETAN is better.

data, even if this superiority is not so prominent.

We end the section pointing out that if the objective is only in classification, the ETAN classifier with $\alpha = 10$ or 20 achieves better performance than AODE. For instance, ETAN($\alpha = 10$) against AODE has

Table 4: Number of wins, ties and losses of SA-ETAN over A-ETAN on data with missing values during testing, and p-values using the Wilcoxon signed rank test on 60 data sets (one sided in the direction of the difference).

Missing	Zero-one loss		Log loss	
percentage	W/T/L	p-value	W/T/L	p-value
10	20/14/26	0.308	19/12/29	0.026
20	22/15/23	0.411	18/12/30	0.024
30	18/14/28	0.235	22/12/26	0.071

W/T/L of 33/2/25 in zero-one loss and 34/0/26 for log loss. We have not tried to fine tune the designed classifiers in order to improve classification results, which is something that would need further empirical analyses.

5. Conclusions

We presented ideas for structure learning that are used to build an extended version of the well-known tree-augmented naive Bayes (TAN) classifier, namely the extended TAN (or ETAN). ETAN does not demand features to be connected to the class, so it has properties of feature selection (when a feature ends up disconnected) and allows features that are important to other features but are not directly linked to the class. We also extend ETAN to account only for arcs that are relevant to the prediction by devising a new score function that discards structures containing arcs that could be removed without changing the classification results in favor of arcs that directly affect the class prediction. Finally, the learning procedure is also implemented to optimize the equivalent sample size. We describe a globally optimal algorithm to learn ETANs that is quadratic in the number of variables, that is, it is asymptotically as efficient as the algorithm for TANs and the algorithm of averaged one-dependence estimator (AODE). The class of structures defined by ETANs can be seen as the (currently) most sophisticated Bayesian networks for which there is a polynomial-time algorithm for structure learning, and this will hardly expand much further as it has been proven that learning with two parents per node (besides the class) is an NP-hard task [29].

Experiments demonstrate that the time complexity of our implementation of ETAN is only a few times slower than that of TAN (and hence negligible for small to moderate number of features) , and show that ETAN always provides equal or better fit (in terms of the score function) than TAN. In spite of that, ETAN achieves worse zero-one loss than TAN, but better log loss. The picture changes to the better when we employ the new score function that is targeted to the prediction of the class. ETAN with such specialized score surpassed both TAN and ETAN (with usual score) according to both losses, but it is still significantly inferior to AODE in log loss (while slightly superior in zero-one loss). Finally, the optimization of the equivalent sample size produces an ETAN classifier that is not dominated by AODE in either loss function (even though slightly inferior). Nevertheless, the new polynomial-time algorithm and ideas to learn specialized Bayesian network constitute a new avenue for building classifiers. For instance, further tuning and possibly averaging ideas would probably make ETAN even more competitive. Future work will investigate such possibilities, as well as the relation between BDeu and classification accuracy, and other scenarios with missing data where ETAN might be preferable. It may also include additional ideas beyond ETAN that could be useful in building Bayesian networks in polynomial time.

Acknowledgments

Work partially supported by the Swiss NSF grant n. 200021_146606 / 1 and by the Swiss CTI project with Hoosh Technology. The first author has done part of the work while he was with the Dalle Molle Institute for Artificial Intelligence.

References

[1] C. P. de Campos, M. Cuccu, G. Corani, M. Zaffalon, Extended tree augmented naive classifier, in: Proceedings of the 7th European Workshop on Probabilistic Graphical Models (PGM), Vol. LNAI 8754, 2014, pp. 176–189.

[2] P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Machine Learning 29 (2/3) (1997) 103–130.

[3] D. Hand, K. Yu, Idiot's Bayes-Not So Stupid After All?, International Statistical Review 69 (3) (2001) 385–398.

[4] J. Friedman, On bias, variance, 0/1 - loss, and the curse-of-dimensionality, Data Mining and Knowledge Discovery 1 (1997) 55–77.

[5] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian Network Classifiers, Machine Learning 29 (2) (1997) 131–163.

[6] G. Corani, C. P. de Campos, A tree augmented classifier based on extreme imprecise Dirichlet model, International Journal of Approximate Reasoning 51 (2010) 1053–1068.

[7] M. Madden, On the classification performance of TAN and general Bayesian networks, Knowledge-Based Systems 22(7) (2009) 489–495.

[8] G. Corani, A. Antonucci, D. Maua, S. Gabaglio, Trading off speed and accuracy in multilabel classification, in: Proceedings of the 7th European Workshop on Probabilistic Graphical Models (PGM), Vol. LNAI 8754, 2014, pp. 145–159.

[9] W. Buntine, Theory refinement on Bayesian networks, in: Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence, UAI'92, Morgan Kaufmann, San Francisco, CA, 1991, pp. 52–60.

[10] G. F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Machine Learning 9 (1992) 309–347.

[11] D. Heckerman, D. Geiger, D. M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Machine Learning 20 (1995) 197–243.

[12] C. P. de Campos, Q. Ji, Properties of Bayesian Dirichlet scores to learn Bayesian network structures, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI Press, 2010, pp. 431–436.

[13] P. Kontkanen, P. Myllymäki, T. Silander, H. Tirri, On supervised selection of Bayesian networks, in: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1999, pp. 334–342.

[14] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI'01, Morgan Kaufmann, 2001, pp. 973–978.

[15] P. D. Turney, Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm, Journal of Artificial Intelligence Research 2 (1995) 369–409.

[16] T. Silander, P. Kontkanen, P. Myllymaki, On sensitivity of the map bayesian network structure to the equivalent sample size parameter, in: Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, UAI'07, AUAI Press, 2007, pp. 360–367.

[17] C. K. Chow, C. N. Liu, Approximating discrete probability distributions with dependence trees, IEEE Transactions on Information Theory IT-14 (3) (1968) 462–467.

[18] G. I. Webb, J. R. Boughton, Z. Wang, Not so naive Bayes: Aggregating one-dependence estimators, Machine Learning 58 (1) (2005) 5–24.

[19] C. P. de Campos, Z. Zeng, Q. Ji, Structure learning of Bayesian networks using constraints, in: Proceedings of the 26th International Conference on Machine Learning, ICML'09, Omnipress, Montreal, 2009, pp. 113–120.

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

[21] K. P. Murphy, Machine Learning: a Probabilistic Perspective, MIT Press, 2012.

[22] J. Edmonds, Optimum branchings, Journal of Research of the National Bureau of Standards B 71B(4) (1967) 233–240.

[23] Y. J. Chu, T. H. Liu, On the shortest arborescence of a directed graph, Science Sinica 14 (1965) 1396–1400.

[24] U. Zwick, Lecture notes on Analysis of Algorithms: Directed Minimum Spanning Trees (April 22, 2013).

[25] R. E. Tarjan, Finding optimum branchings, Networks 7 (1977) 25–35.

[26] P. M. Camerini, L. Fratta, F. Maffioli, A note on finding optimum branchings, Networks 9 (1979) 309–312.

[27] H. N. Gabow, Z. Galil, T. Spencer, R. E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, Combinatorica 6 (2) (1986) 109–122.

[28] M. Lichman, UCI machine learning repository (2013).
URL `http://archive.ics.uci.edu/ml`

[29] S. Dasgupta, Learning polytrees, in: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, 1999, pp. 134–141.