Entropy **2015**, *17*, 5085-5100; doi:10.3390/e17075085

OPEN ACCESS
entropy
ISSN 1099-4300
www.mdpi.com/journal/entropy

Article

Averaged Extended Tree Augmented Naive Classifier

Aaron Meehan and Cassio P. de Campos *

EEECS, Queen's University Belfast, University Road, Belfast BT7 1NN, UK;
E-Mail: ameehan05@qub.ac.uk

* Author to whom correspondence should be addressed; E-Mail: c.decampos@qub.ac.uk;
Tel.: +44-28-9097-6795.

Academic Editors: Carlos Alberto De Bragança Pereira and Adriano Polpo

Received: 8 June 2015 / Accepted: 17 June 2015 / Published: 21 July 2015

Abstract: This work presents a new general purpose classifier named Averaged Extended Tree Augmented Naive Bayes (AETAN), which is based on combining the advantageous characteristics of Extended Tree Augmented Naive Bayes (ETAN) and Averaged One-Dependence Estimator (AODE) classifiers. We describe the main properties of the approach and algorithms for learning it, along with an analysis of its computational time complexity. Empirical results with numerous data sets indicate that the new approach is superior to ETAN and AODE in terms of both zero-one classification accuracy and log loss. It also compares favourably against weighted AODE and hidden Naive Bayes. The learning phase of the new approach is slower than that of its competitors, while the time complexity for the testing phase is similar. Such characteristics suggest that the new classifier is ideal in scenarios where online learning is not required.

Keywords: classification; tree augmented Naive Bayes; model averaging

1. Introduction

Bayesian network classifiers are based on learning the relations of (conditional) independence among variables (also called attributes) in a domain in order to predict the label (or state) of a targeted variable (also called class). They have been shown to perform well with respect to other general purpose classifiers [1,2]. Some well-known examples are the Naive Bayes and the Tree Augmented Naive Bayes (TAN) classifiers. The Naive Bayes has a very straightforward approach of assuming that all attributes are parented by the class and are conditionally independent of each other given the class. TAN on the other hand weakens this assumption of independence by using a tree structure wherein each attribute directly depends on the class and one other attribute. Both of these classifiers are based on the Bayesian networks [3,4]. Because Naive Bayes uses a much simpler model, it generally fits data less well than TAN and so is less likely to overfit compared with TAN, which forces attributes to be dependent on each other. Irrespective of their structural constraints, Naive Bayes and TAN have been shown to perform extremely well when compared to unrestricted Bayesian network classifiers [2] (that is, Bayesian networks without any constraint on their graph structure). Unrestricted Bayesian networks have also a disadvantage in terms of computational complexity: Learning their structure (or essentially the Markov blanket of the class node) is an NP-hard task [5,6]. In this work we focus on Bayesian network classifiers that can be learned exactly and efficiently from data, that is, we aim at learning both the structure and the parameters of such networks with a tractable exact approach. Learning can be performed in a generative manner (aiming at the joint distribution) or in a discriminative way (with respect to the targeted class variable). Because we are interested in exact and efficient learning, we choose to perform generative learning. Such decision precludes us from using discriminative learning of Bayesian network classifiers [7–10], which have been shown to perform well even though learning is conducted using approximate methods. On the other hand, our focus on exact and efficient learning allow us to conclude that the differences in the results are actually differences in the *quality* of the models and not due to some possibly sub-optimal solution during learning, while we acknowledge that other classifiers with greater accuracy (but approximate learning) might exist if we expanded our scope.

Very recently, the Extended TAN (ETAN) classifier [11,12] has been created to fill the gap in model complexity between Naive Bayes and TAN, in that it can detect the best model of dependencies among attributes without forcing them to be either conditionally independent or directly dependent. This approach allows ETAN to adapt to the characteristics of the data and usually produces models that fit better than both TAN and Naive Bayes while avoiding increased complexity. It is empirically shown that ETAN achieves classification accuracy comparable to the Averaged One-Dependence Estimator (AODE) classifier, one of the state-of-the-art general purpose classification methods [13]. However, ETAN is still usually outperformed by AODE.

Herein we present two simple improvements to ETAN that we can show to be effective. The first improvement builds on the structure of dependencies of ETAN by extending the search space of models that is captured by ETAN to include even further models of (conditional) independence among the variables in the domain. This idea can fit the data better than ETAN can. The second and possibly more significant extension made, is to build upon that by using a concept of model averaging: instead of picking the most probable model of independences, we average over a set of possible models. This is a promising approach as it goes in the direction of a full Bayesian estimation for the model. In fact, AODE—a high performing classifier—uses a similar technique. Our approach is different from AODE in that by building upon ETAN we have a wider range of possible models when compared with AODE (which averages only Naive Bayes classifiers), so in theory our classifier should build better models. We will empirically demonstrate this assertion. Some classifiers in the literature are related to this work and deserve a mention. Keogh and Pazzani [14] propose an extension of Naive Bayes that goes in the same direction of AODE and AETAN, however their approach is not globally optimal. Qiu *et al.* [15]

build a similar graph structure as AODE, but use a discriminative learning approach. Other attempts to improve on Naive Bayes and TAN have been proposed [16,17], but they usually do not focus on exact and efficient learning.

This document is divided as follows. Section 2 describes the problem of classification and the most common Bayesian network classifiers. These are general purpose classifiers, in the sense that no specific domain knowledge or other information is exploited but the data set. Section 3 describes the accuracy measures that are used for comparing classifiers in this work, and then Section 4 presents the new classifier and the details on how to implement it. Section 5 describes our experimental setting and results, and finally Section 6 concludes the work and points to future developments.

2. Background

We assume that a data set $\mathbf{D} = \{D_1, D_2, \ldots, D_m\}$ without missing values is available, where each D_i is a vector observation for the $n+1$ variables $\mathbf{X}$, composed of a target (or class) variable C and n covariates (or attributes) $\mathbf{Y}$. By using these data, we want to build a probabilistic model $\mathcal{M}$ with which we can evaluate the posterior probability of C given a vector $\mathbf{y}$ of observations for $\mathbf{Y}$ to make a prediction about the unknown class label. This problem is usually called *supervised classification*. That is, given a class $C \in \mathbf{X}$ and some attribute observations $\mathbf{y}$, where $\mathbf{Y} = \mathbf{X} \backslash \{C\}$, we use the classifier $\mathcal{M}$, which was previously built from $\mathbf{D}$, to calculate the posterior probability function $p_{\mathcal{M}}(C|\mathbf{y})$, which can then be used for instance to guess the most probable label of C. This probability function $p_{\mathcal{M}}$ is obtained from the distribution that we have learned from $\mathbf{D}$, which might underlying have a complicated set of conditional independences among variables in $\mathbf{X}$. This set of conditional independences is encoded by a directed acyclic graph (DAG) where each variable is associated to a node (by a one-to-one correspondence, so we might refer to either nodes or variables entailing the same meaning) in the graph and the arcs are used to define relationships between nodes. An arc from X_i to X_j denotes that X_i is a parent of X_j (and so X_j is a child of X_i). The DAG represents a Markov condition: every variable is conditionally independent of its non-descendent variables given its parents. By using such a representation, we can write

$$p_{\mathcal{M}}(C|\mathbf{y}) \propto p_{\mathcal{M}}(C|\pi_C) \prod_{i=1}^{n} p_{\mathcal{M}}(y_i|\pi_{Y_i})$$

where π_{Y_i} and π_C are observations for the parents of Y_i and C, respectively, such that they are consistent with the vector $\mathbf{y}$. Hence, the challenge is to build this model $\mathcal{M}$, composed of a DAG $\mathcal{G}$ and probability functions $p_{\mathcal{M}}(X|\Pi_X)$, for the variables $X \in \mathbf{X}$.

To fulfill the goal of building $\mathcal{M}$, we take the (training) data set $\mathbf{D}$ and find $\mathcal{G}$ that maximizes its posterior probability, that is, $\mathcal{G}^* = \text{argmax}_{\mathcal{G} \in \boldsymbol{\mathcal{G}}} p(\mathcal{G}|\mathbf{D})$, with $\boldsymbol{\mathcal{G}}$ the set of all DAGs over node set $\mathbf{X}$ and

$$p(\mathcal{G}|\mathbf{D}) \propto p(\mathcal{G}) \cdot \int p(\mathbf{D}|\mathcal{G}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}|\mathcal{G}) d\boldsymbol{\theta} \tag{1}$$

where $p(\boldsymbol{\theta}|\mathcal{G})$ is the prior of $\boldsymbol{\theta}$ for a given graph $\mathcal{G}$, $\boldsymbol{\theta} = \{\theta_{ijk}\}_{\forall ijk}$ is the entire vector of model parameters such that θ_{ijk} is associated with $p(X_i = k|\Pi_i = j)$, where $k \in \{1, \ldots r_i\}$, $j \in \{1, \ldots, r_{\Pi_i}\}$ and $i \in \{0, \ldots, n\}$ ($r_{\mathbf{Z}} = \prod_{X \in \mathbf{Z}} r_X$ denotes the number of states of the joint variable $\mathbf{Z}$ and r_X the number of states of a single variable X). The prior is assumed to be a symmetric Dirichlet with positive hyper-parameter α^*:

$$p(\boldsymbol{\theta}|\mathcal{G}) = \prod_{i=0}^{n} \prod_{j=1}^{r_{\Pi_i}} \Gamma\left(\frac{\alpha^*}{r_{\Pi_i}}\right) \prod_{k=1}^{r_i} \frac{\theta_{ijk}^{\frac{\alpha^*}{r_{X_i} r_{\Pi_i}}-1}}{\Gamma(\frac{\alpha^*}{r_{X_i} r_{\Pi_i}})} \tag{2}$$

α^* is usually referred to as the Equivalent Sample Size (ESS). The value in Equation (1) is also known as the Bayesian Dirichlet Equivalent Uniform (BDeu) score [18,19], where we assume parameter independence and modularity [20]. We further assume that there is no preference for any graph and set $p(\mathcal{G})$ as uniform, so it can be rewritten as $s_{\mathbf{D}}(\mathcal{G}) = p(\mathbf{D}|\mathcal{G})$. Under these assumptions, it has been shown [19] that the graph maximizing its posterior probability is the argument of

$$\max_{\mathcal{G}} \sum_{i=0}^{n} \sum_{j=1}^{r_{\Pi_i}} \frac{l\Gamma(\frac{\alpha^*}{r_{\Pi_i}})}{l\Gamma(\frac{\alpha^*}{r_{\Pi_i}} + N_{ij})} \sum_{k=1}^{r_i} \frac{l\Gamma(\frac{\alpha^*}{r_{X_i} r_{\Pi_i}} + N_{ijk})}{l\Gamma(\frac{\alpha^*}{r_{X_i} r_{\Pi_i}})} \tag{3}$$

where $l\Gamma$ is the log-gamma function, N_{ijk} indicates how many elements in $\mathbf{D}$ contain both $X_i = k$ and $\Pi_i = j$. The values $\{N_{ijk}\}_{ijk}$ depend on the graph $\mathcal{G}$ (more specifically, they depend on the parent set Π_i of each X_i), so a more precise notation would be to use $N_{ijk}^{\Pi_i}$ instead of N_{ijk}, which we avoid for ease of exposition.

In order to find the graph representing the best set of conditional independences over the space of all possible DAGs $\boldsymbol{\mathcal{G}}$, multiple approaches have been proposed in the literature, including score properties to reduce the search space (De Campos and Ji [21]), dynamic programming (Silander and Myllymaki [22]), branch-and-bound [23], A* and path finding methods [24], mixed integer linear programming [25], among others. These methods do not scale with the number of variables, which is nothing but expected, as the problem is known to be NP-hard in general and even if we limit each node to have at most two parents [6,26]. Therefore, some simplification assumptions must be made in order to achieve practical running times and tractable computation.

There are two main possibilities for building classifiers on top of Bayesian networks that are tractable: the Naive Bayes classifier and the Tree Augmented Naive Bayes (TAN) classifier. The Naive Bayes classifier (equivalently a log-linear model) is the simplest of the Bayesian network classifiers, by assuming that $p_{\mathcal{M}}(Y_i|C, Y_j) = p_{\mathcal{M}}(Y_i|C)$ for any $i \neq j$. The graphical representation of this is that of a directed tree with the class, C as the sole root and with all attributes being leaves with the class as their only parent, the example in Figure 1b illustrates it. As such, a Naive Bayes classifier has a constant structural shape irrespective of the training data. This restriction will often lead to underfitting on the part of a Naive Bayes classifier, however it still performs quite well (when scored on zero-one accuracy; see Section 3 for a description of accuracy measures) [27,28]. The question of why the naive Bayes does so well despite its' deficiencies has been attempted to be answered by Friedman [29], among others. Friedman also suggested that misclassification error be decomposed into *bias* and *variance* errors; in doing so we can represent both the error from erroneous assumptions made during learning in bias error and the error from the sensitivity of the parameters of the classifier to the training data in variance error. In these terms, the Naive Bayes classifier has low variance—as it does not have a large number of parameters to estimate; but high bias due to its strict assumption of independence.

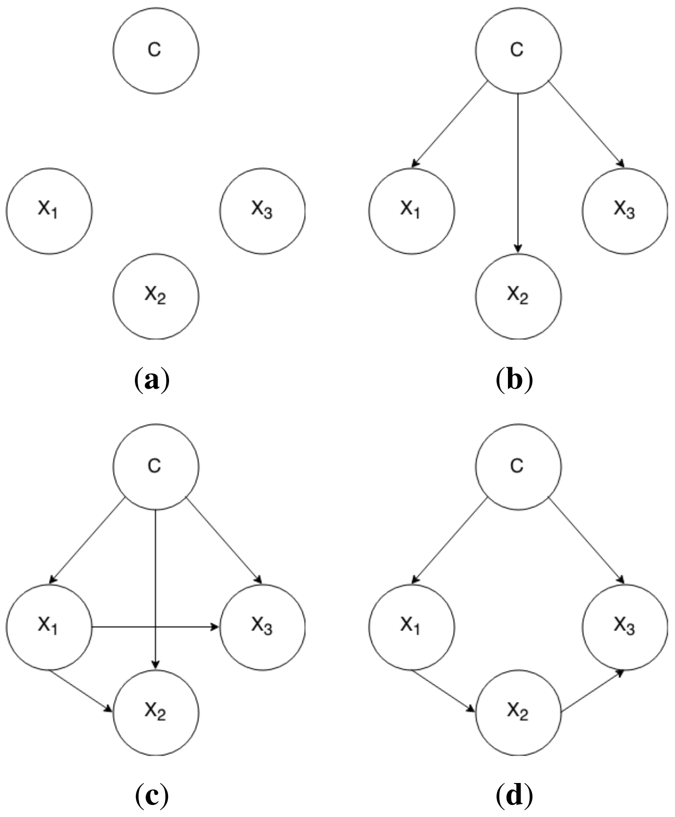

Figure 1. Some examples of the different structures possible within the classifiers we have written about here. (**a**) Possible with ETAN; (**b**) Possible with naive or ETAN; (**c**) Possible with TAN or ETAN; (**d**) Possible with ETAN.

The tree augmented Naive Bayes (TAN) classifier, presented in [29], was an attempt to reduce this high bias by relaxing the assumption of independence between attributes. It does this by allowing a more complex graph model in which each attribute has the class and one other attribute as its parents—except a single attribute which has only the class as a parent. This approach can be seen as a middle ground between general Bayesian networks which can learn without constraints and the rigid structure of naive Bayes, in which each attribute has only the class as a parent. As we can see in [2,30–32], TAN is shown to not only outperform general Bayesian networks, but Naive Bayes as well. Figure 1c shows a possible graph model with TAN from which we can see that compared with the naive model in Figure 1b, a more complex and possibly more fitting model can be learned. Under TAN we now see that two of the attributes in our example are dependent upon the third. Also of note is that if one were to remove the class (including its arcs) from the TAN classifier model, so that the graph was made up only of the attributes, then we would now be left with a directed tree; this is the origin motivation of its name.

By using the BDeu scoring function, one can devise an efficient algorithm for TAN, because BDeu is decomposable, that is, $p(\mathcal{G}|\mathbf{D})$ can be written as a sum $\sum_{i=0}^{n} s_{\mathbf{D}}(X_i, \Pi_i)$ of local computations involving a node and its parents in the graph $\mathcal{G}$. By also using the *likelihood equivalence* property of BDeu, that is, a score function is likelihood equivalent if given two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ over $\mathbf{X}$ where the set of conditional independences encoded in $\mathcal{G}_1$ equals that of $\mathcal{G}_2$, then we have $p(\mathcal{G}_1|\mathbf{D}) = p(\mathcal{G}_2|\mathbf{D})$, then one can use a simple minimum spanning tree algorithm to find the best possible graph $\mathcal{G}^*$ to build the TAN classifier [33] in quadratic time in n: $\mathcal{G}^*_{\text{TAN}} = \text{argmax}_{\mathcal{G} \in \boldsymbol{\mathcal{G}}_{\text{TAN}}} p(\mathcal{G}|\mathbf{D})$,wherein $\boldsymbol{\mathcal{G}}_{\text{TAN}}$ is the set of all the TAN structures with nodes $\mathbf{X}$.

Note that TAN forces a tree to *cover* all the attributes, as well as ensures that they are all children of the class. This might be undesirable, if the data are not sufficient to lead to such a *complex* structure of dependencies. With that in mind, very recently we have proposed the Extended TAN (or ETAN) classifier [11], intended to allow a trade-off in the model complexity between the very simple Naive and the more sophisticated TAN models. ETAN has the additional ability of being able to learn from data in which circumstances an attribute Y_i should be a child of C, as well as to identify whether attributes should be directly dependent on each other or not without having to force a tree to cover them all. More formally, we can define the ETAN graph (representing its conditional independences) as a DAG such that, for each attribute Y_i, $|\Pi_i| \leq 1$, or $|\Pi_i| = 2$ and $\Pi_i \supseteq \{C\}$. The class C still has no parents. So, we search for $\mathcal{G}^*_{\text{ETAN}} = \text{argmax}_{\mathcal{G} \in \boldsymbol{\mathcal{G}}_{\text{ETAN}}} p(\mathcal{G}|\mathbf{D})$, where $\boldsymbol{\mathcal{G}}_{\text{ETAN}}$ are all DAGs satisfying the aforementioned restrictions. In Figure 1a,d we can see some structures that are possible within the rules of ETAN. We point out that the ETAN graphs encompass both the Naive Bayes and TAN graphs. In this case, the implementation relies on the well-known Edmonds' algorithm [34] (also attributed to [35]) as means for finding the ETAN graph of maximum posterior probability. While much more complicated in its implementation, ETAN can still run in quadratic time in n [12].

3. Accuracy Measures

Clearly we must have some way of measuring the accuracy of a classifier in order to understand which is the best for our purposes. The first measure of quality of a given classifier is the value of the posterior probability of its graph $\mathcal{G}$ (or equivalently under the given assumptions the probability $p(\mathbf{D}|\mathcal{G})$ of the training data under such model), also called *Bayesian Dirichlet equivalent uniform* (BDeu) score [18]. This is a measure of fitness and not of classification accuracy, but it is still very relevant to understand the difference between the approaches.

When we compare the effectiveness of the classifiers in our testing data (which is not used for inferring the classifiers), two main measures can be used [4]. The zero-one accuracy is a measure of correctly predicted class labels. So if we let $\mathbf{T} = (\mathbf{y}_1, c_1), \ldots, (\mathbf{y}_N, c_N)$ be the testing instances $\mathbf{y}_i$ and their corresponding class labels c_i, where $\mathbf{y}_i$ is a vector observation for the attributes $\mathbf{Y}$ as defined previously, then the zero-one accuracy is such that

$$\text{zo}(\mathcal{M}, \mathbf{T}) = \frac{1}{N} \sum_{i=1}^{N} \text{I}\left(c_i = \underset{j}{\text{argmax}}\, p_{\mathcal{M}}(C = j|\mathbf{y}_i) \right),$$

where I is the indicator function, that is, the percentage of successfully classified labels. Another important measure is the log-loss function. Simply put, the objective of the log-loss function is to approximately compare the posterior probability generated by the classifier $\mathcal{M}$ with the true posterior probability of the class using the Kullback-Leibler distance [36]. Note that this is an approximation, as only with the number of testing samples $N \to \infty$ we would obtain the true difference between them. We want the estimated posterior probability of the class to be as close as possible to the true one, therefore the most accurate classifier in this case is that which has the lowest log loss:

$$\text{logl}(\mathcal{M}, \mathbf{T}) = -\sum_{i=1}^{N} \log(p_{\mathcal{M}}(C = c_i|\mathbf{y}_i)).$$

These measures are going to be used in Section 5 to evaluate the quality of the classifiers.

4. Averaged ETAN

In [12] it has been shown that the ETAN classifier performs quite well; in time complexity it is not worse than TAN, and it generates models that fit the data better than both TAN and Naive Bayes. Although TAN is shown to have a better zero-one accuracy in most cases, the log-loss accuracy of ETAN was shown to be significantly better than TAN—making it a good choice in situations where an accurate model of the data is as important as an accurate classification result. It was also found that the performance of ETAN increases proportionally to the number of attributes in the domain. However, ETAN was only slightly competitive with the Averaged One-Dependence Estimators (AODE) classifier, one of the state-of-the-art general purpose classification methods [13]. The idea behind AODE is to choose an attribute and elect it to be a parent of the class, while keeping the remaining graph constraints unchanged. Because AODE is based on the Naive Bayes classifier, the result is a graph that has the class plus a *super attribute* linked together, and then all other attributes as children of them both. We extend such an idea with the already good properties of ETAN to create the so-called *Averaged ETAN* (or AETAN).

First, we widen the range of possible models of ETAN by adding the possibility of a *super attribute* on top of the existing ETAN algorithm. The super attribute is an attribute S chosen to act as a parent of the class C. We iterate over all possible attributes one at a time (and also without any super attribute, in order to ensure that our approach generalizes ETAN). For each candidate super attribute, we find the best ETAN graph over the remaining attributes $\mathbf{Y} \setminus \{S\}$ such that S is a parent of C. An example is shown in Figure 2.

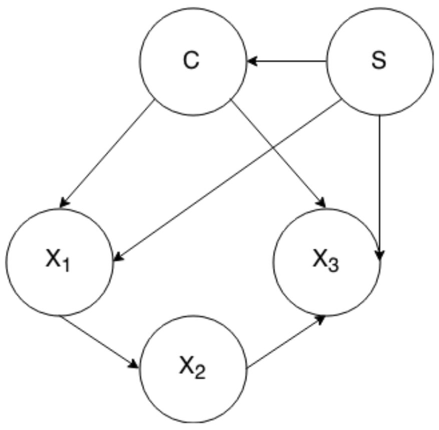

Figure 2. An example of a new structure possible with Extended Tree Augmented Naive Bayes (ETAN)pp.

All of these discovered models $\mathcal{M}_S$ and their graphs $\mathcal{G}_S$ are stored into an averaged model $\mathcal{M}$, together with their probabilities $p(\mathcal{G}_S, \mathbf{D})$. Then the class prediction for a new observation vector $\mathbf{y}$ under averaging model $\mathcal{M}$ is computed by the class posterior probability function as:

$$p_{\mathcal{M}}(C|\mathbf{y}, \mathbf{D}) \propto \sum_{\mathcal{G}_S} p(\mathcal{G}_S, \mathbf{D}) \cdot p_{\mathcal{M}_S}(C, \mathbf{y}|\mathcal{G}_S, \mathbf{D}). \tag{4}$$

Such averaging is supposed to improve classification accuracy, as it makes the inference process closer to a full Bayesian approach (even if the averaging is performed only over a very small set of graphs). In spite of that, this averaging usually has the effect of increasing the accuracy of classification while reducing the impact that any single graph has on the overall result. The non-obvious part is how to learn an ETAN model with an extra super attribute S.

The implementation of AETAN is obviously based on that of ETAN, but some non-trivial improvements are needed. Algorithm 1 contains the pseudo-code for AETAN. We give as input the variables $\mathbf{X}$ and a score function $s_{\mathbf{D}}$, as the method works with any score function which is decomposable. In each main loop, the algorithm simply adds the resultant graph into a list of all inferred graphs list; we loop through adding the best graph for each class and super attribute combination to the list. Once finished, we return the list of graphs ready to be averaged at the time of testing (the probability of each of them, which is proportional to their score, is also returned).

This implementation first initializes a temporary graph through the call to ArcsCreation and then makes use of Edmonds' algorithm for finding the minimum directed weighted spanning forest. The overall time complexity is $O(n^3m)$, which is n times slower than ETAN or TAN. This comes from a simple analysis of ArcsCreation, which is $O(n^2m)$ and Edmonds' algorithm, whose *contract* step is $O(n^2)$ while its *expand* step is $O(n)$. After the list of models is inferred, the testing procedure as defined in Equation (4) takes time $O(n^2)$ per evaluation, which is asymptotically the same time complexity as that of AODE (but much slower than TAN and ETAN).

Algorithm 1 AETAN($\mathbf{X}$, $s_{\mathbf{D}}$)

```
1:  list ← ∅
2:  for all $S \in \mathbf{X} \setminus \{C\}$ do
3:      $(arcs, otherParents) \leftarrow$ ArcsCreation($\mathbf{X}, S, s_{\mathbf{D}}$)
4:      $\mathcal{G}' \leftarrow (\mathbf{X}, arcs)$
5:      EdmondsContract($\mathcal{G}'$)
6:      $\mathcal{G}^* \leftarrow null$
7:      for all $root \in \mathbf{X} \setminus \{C, S\}$ do
8:          $in \leftarrow$ EdmondsExpand($root$)
9:          $\mathcal{G} \leftarrow$ buildGraph($\mathbf{X}, root, in, otherParents$)
10:         if $s_{\mathbf{D}}(\mathcal{G}) > s_{\mathbf{D}}(\mathcal{G}^*)$ then
11:             $\mathcal{G}^* \leftarrow \mathcal{G}$
12:     list ← (list, $(\mathcal{G}^*, s_{\mathbf{D}}(\mathcal{G}^*))$)
    return list
```

For each possible S, we first create the arcs that will be given to Edmonds using the function *ArcsCreation* (presented in Algorithm 2), by taking the highest scoring edge for each pair while testing the worth of having the super attribute and the class as possible additional parents. The logic is straightforward, and the important characteristic is that those decisions can be taken locally as they never yield directed cycles in the resulting graph. The variable *otherParents* is used to keep track of which of the four combinations of parents was used in the creation of the edge, for use later. We do not include these parents in the graph until after Edmonds' algorithm has finished (so we store them for later inclusion). With the arcs in hands, Edmonds' creates the directed minimum spanning tree. Finally, Algorithm 3 returns a graph using the result of Edmonds and the stored parent sets. More details on Edmonds' can be found elsewhere [34,37,38].

Algorithm 2 ArcsCreation($\mathbf{X}, S, s_{\mathbf{D}}$)

```
 1: for all $X_i \in \mathbf{X} \setminus \{C, S\}$ do
 2:     $otherParents[X_i] \leftarrow \max(s_{\mathbf{D}}(X_i, \{C, S\}), s_{\mathbf{D}}(X_i, \{C\}), s_{\mathbf{D}}(X_i, \{S\}), s_{\mathbf{D}}(X_i, \emptyset))$
 3: $arcs \leftarrow \emptyset$
 4: for all $X_i \in \mathbf{X} \setminus \{C\}$ do
 5:     for all $X_j \in \mathbf{X} \setminus \{C\}$ do
 6:         $3Parents \leftarrow s_{\mathbf{D}}(X_i, \{C, S, X_j\})$
 7:         $2ParentsClass \leftarrow s_{\mathbf{D}}(X_i, \{C, X_j\})$
 8:         $2ParentsSA \leftarrow s_{\mathbf{D}}(X_i, \{C, X_j\})$
 9:         $onlyAttrib \leftarrow s_{\mathbf{D}}(X_i, \{X_j\})$
10:         $w' \leftarrow \max(s_{\mathbf{D}}(X_i, \emptyset), s_{\mathbf{D}}(X_i, \{C\}, s_{\mathbf{D}}(X_i, \{S\}), s_{\mathbf{D}}(X_i, \{C, S\})))$
11:         $w \leftarrow \max(3Parents, 2ParentsClass, 2ParentsSA, onlyAttrib) - w'$
12:         if $w > 0$ then
13:             Add $X_j \rightarrow X_i$ with weight $w$ into $arcs$
14:             $otherParents[X_j \rightarrow X_i] \leftarrow \max(3Parents, 2ParentsClass, 2ParentsSA, onlyAttrib)$
15:         else
16:             $otherParents[X_j \rightarrow X_i] \leftarrow otherParents[X_i]$
    return $(arcs, otherParents)$
```

Algorithm 3 buildGraph($\mathbf{X}, S, in, otherParents$)

```
1: $\mathcal{G} \leftarrow (\mathbf{X}, \emptyset)$
2: for all $node \in \mathbf{X} \setminus \{C, S\}$ do
3:     $\Pi_{\text{node}} \leftarrow \emptyset$
4:     if $in[node] \neq null$ then
5:         $\Pi_{\text{node}} \leftarrow \Pi_{\text{node}} \cup \{in[node]\} \cup otherParents[in[node] \rightarrow node]$
6:     else
7:         $\Pi_{\text{node}} \leftarrow \Pi_{\text{node}} \cup otherParents[node]$
   return $\mathcal{G}$
```

The overhead of learning AETAN (the creation of multiple ETAN with super attributes) makes it slower than ETAN, as we will see in the experiments.

5. Experiments

We run experiments with the following Bayesian network classifiers: Naive Bayes, TAN, ETAN, AETAN (the new approach), AODE, Weighted AODE (or WAODE) [39] and Hidden Naive Bayes (or HNB) [40]. All experiments are executed within Weka (Waikato Environment for Knowledge Analysis [41,42]), for which we developed an extension with the ETAN and AETAN methods. The experiment results we show herein were generated by learning our classifiers using 20 runs of 5-fold cross-validation, and so each classifier's learning procedure has been called 100 times per data set, and tested over the extra fold also 100 times. We use the Bayesian Dirichlet equivalent uniform (BDeu) score with hyper-parameter $\alpha^* = 5$ unless specified (it is not well understood how this affects the accuracy of classification and we will explore the results of a few different values α^* as suggested in the literature [43–45], but a more detailed study on α^* is beyond the scope of this work).

As for the data sets, we have chosen 51 data sets from the UCI machine learning repository [46]. These data sets were selected on the basis of number of attributes and number of data samples (enough to provide for useful analysis) having up to 100 attributes.

In Table 1 we see some interesting statistics about the overall performance of AETAN. It defeats every other classifier in percentage of correct classifications (zero-one accuracy) and log loss values by a good margin (except for HNB, which is comparable in zero-one accuracy and root mean squared error, but is inferior in log loss compared to AETAN). We point out that our analysis and the presented p-values have not taken into account multiple test correction, but as can be seen from the values in the table, the conclusions are the very same even if that is considered.

Table 1. Table showing Averaged Extended Tree Augmented Naive Bayes (AETAN) *versus* other classifiers. W means AETAN wins (is superior to the other classifier), T is a tie, and L represents losses. Pvalues correspond to Wilcoxon signed-rank test. RMSE is the root mean squared error as obtained by Weka [42]. Methods were used from Weka and its plugins.

Classifier	**Zero-one Accuracy**		**Log Loss**		**RMSE**	
	W/T/L	**Pvalue**	**W/T/L**	**Pvalue**	**W/T/L**	**Pvalue**
Naive	35/0/16	6×10^{-4}	46/0/5	5×10^{-9}	41/0/10	3×10^{-7}
ETAN	42/0/9	6×10^{-7}	49/0/2	1×10^{-9}	49/0/2	7×10^{-10}
TAN	32/0/19	0.04	43/0/8	2×10^{-7}	38/0/13	9×10^{-5}
AODE	35/0/16	0.03	37/0/14	2×10^{-4}	38/0/13	6×10^{-4}
WAODE	33/0/18	0.07	36/0/15	1×10^{-3}	36/0/15	2×10^{-3}
HNB	26/0/25	0.42	30/0/21	7×10^{-3}	25/0/26	0.85
Naive ($\alpha^* = 2$)	34/0/17	1×10^{-3}	46/0/5	2×10^{-9}	41/0/10	3×10^{-7}
ETAN ($\alpha^* = 2$)	40/0/11	5×10^{-7}	48/0/3	4×10^{-9}	48/0/3	2×10^{-8}
AETAN ($\alpha^* = 2$)	38/1/12	7×10^{-5}	44/0/7	6×10^{-7}	42/0/9	4×10^{-7}
TAN ($\alpha^* = 2$)	34/0/17	0.01	45/0/6	1×10^{-8}	40/0/11	4×10^{-6}
AODE ($\alpha^* = 2$)	30/0/21	0.20	38/0/13	1×10^{-4}	37/0/14	2×10^{-3}
WAODE ($\alpha^* = 2$)	30/0/21	0.23	36/0/15	8×10^{-4}	35/0/16	4×10^{-3}

The boxplots shown in Figure 3 show the log loss information in more detail. Each of boxplot relates the results obtained from the 100 executions of the learning function for each data set (comparisons are always over the very same training and testing data, for the different classifiers). We see very good results for AETAN *versus* ETAN and AODE, even though the magnitude of the improvement is not so large (still statistically significant). We see that AETAN is on average at least as good as and often superior to both ETAN and AODE in terms of log loss, with some outliers. Very similar results (in fact noticeably better, *versus* Naive and TAN) are not shown.

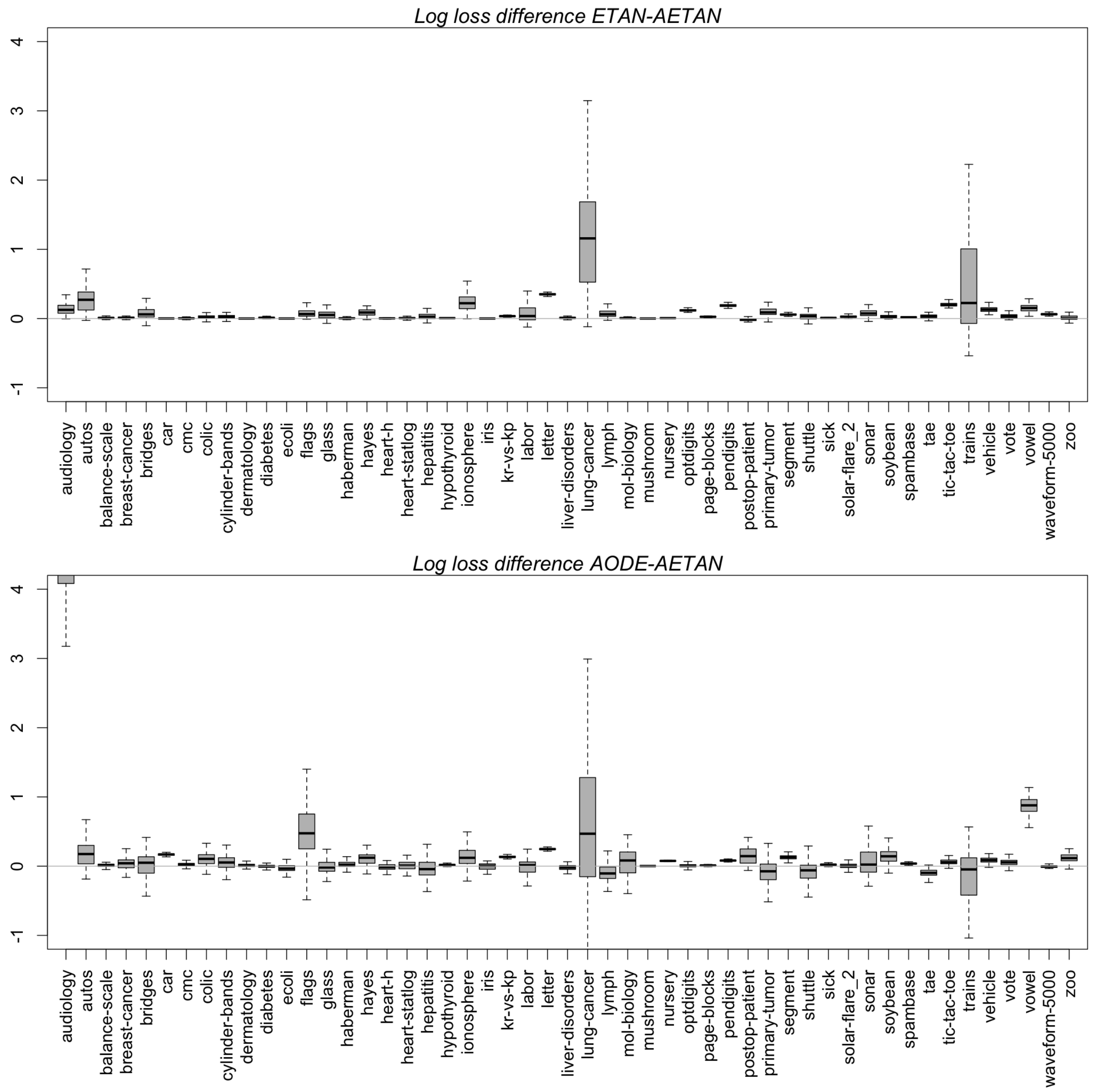

Figure 3. These graphs show the log loss difference between AETAN and ETAN (**Top**) and AETAN and Averaging One-dependent Estimator (AODE) (**Bottom**). The values are the difference between the log loss of the first classifier minus AETAN, so higher values mean AETAN is better.

In a similar way for the zero one accuracy, Figure 4 allows us to see the degree of improvement between AETAN and other classifiers (namely AODE and ETAN; results for Naive and TAN are omitted as they are always inferior to these). Again we see that it is usually slightly better than the others with some outliers, for example in the vowel data set its classification results were much better than that of AODE. There were only two losses in which the opposing classifier was much better than AETAN and these were both under AODE, when using the lung-cancer and primary-tumor data sets. Each of our boxplots relates the results obtained from the 100 executions of the learning function for each data set.

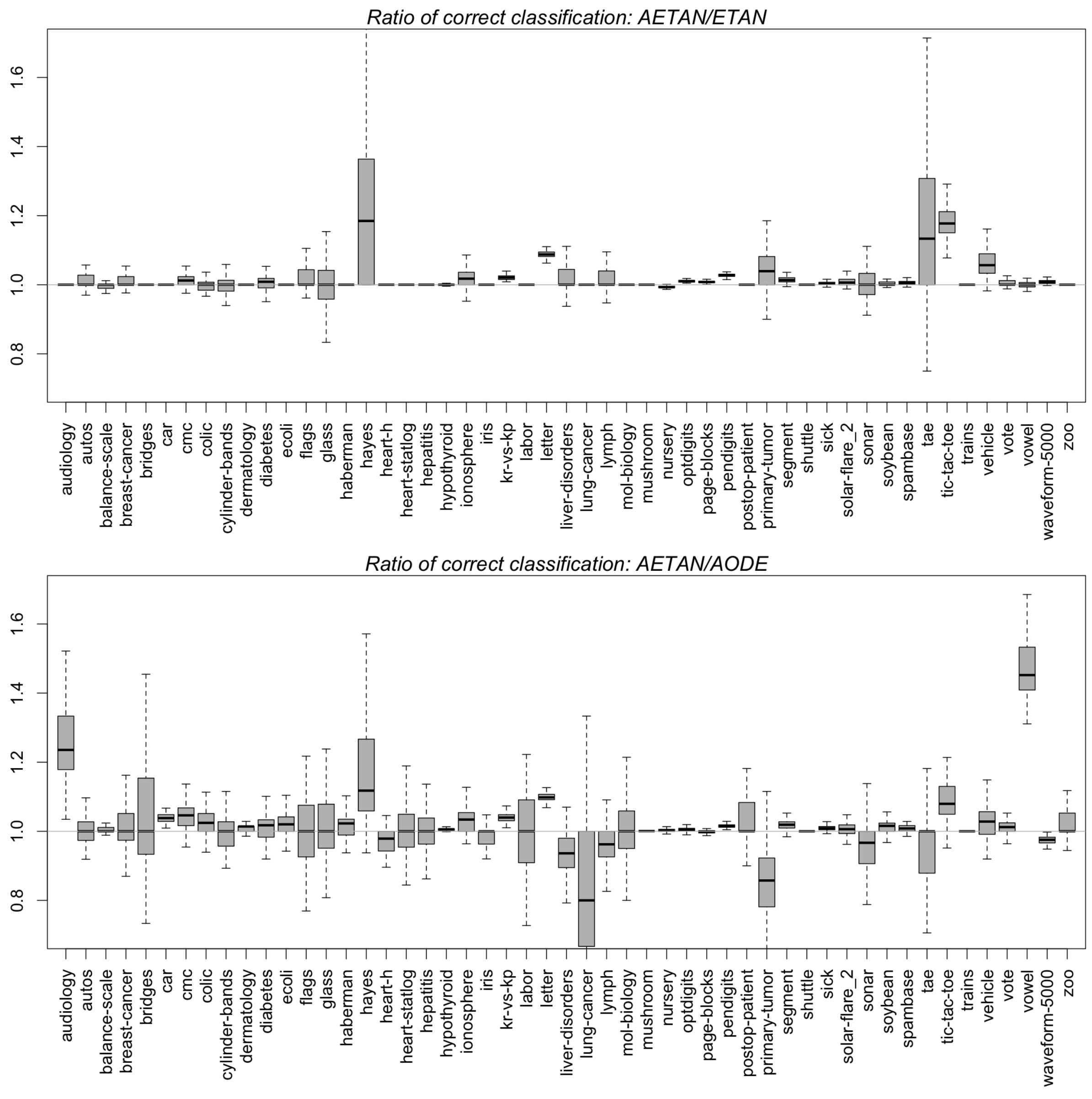

Figure 4. A comparison of correct classification ratio between AETAN and ETAN (**Top**) and AETAN and AODE (**Bottom**), higher value means AETAN is better as these results are the accuracy of AETAN divided by the others.

The great advantages of AETAN over its competitors in terms of zero one accuracy and log loss do not come for free. In Figure 5 we see that AETAN is considerably slower to be inferred than ETAN, which runs in similar time as TAN and AODE. It is worth mentioning that this difference is significant only for learning the model. During testing, the time complexity of AETAN is comparable to that of AODE, so the slower time of AETAN is not a great concern unless the targeted application requires online learning.

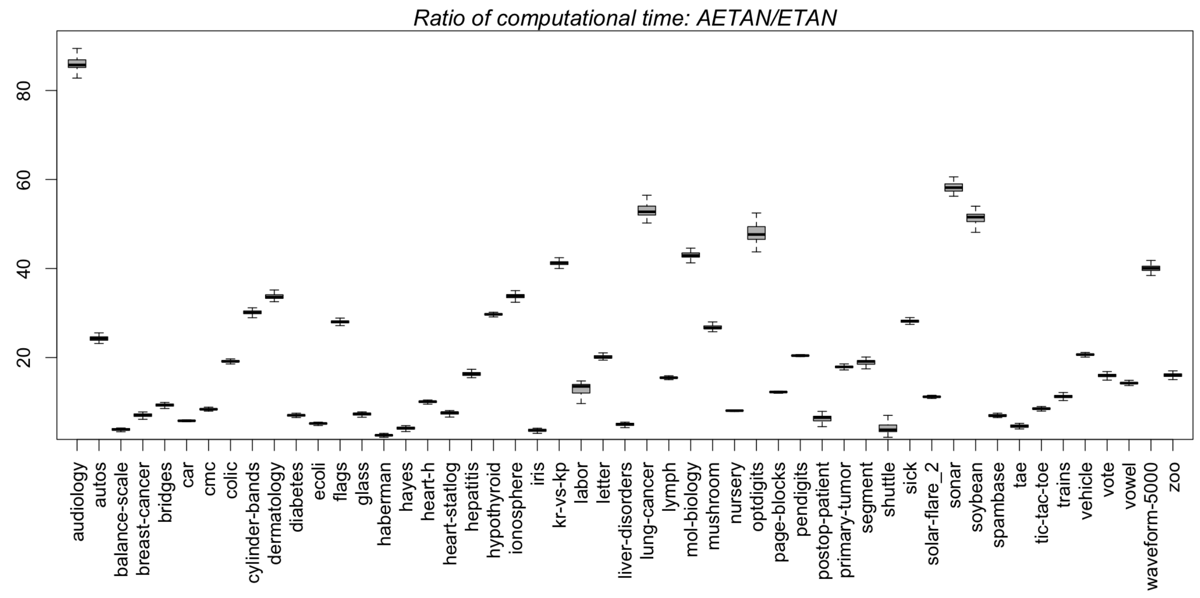

Figure 5. Comparison of computation time to learn the classifier between AETAN and ETAN, expressed as a ratio of AETAN/ETAN so higher values mean AETAN is slower.

6. Conclusions

In this work we propose a new general purpose classifier called Averaged Extended Tree Augmented Naive Bayes, or simply AETAN. Its main idea is to combine the good properties of the Averaging One-dependent Estimator (AODE) and the Extended Tree Augmented Naive Bayes (ETAN) into a single classifier which could potentially improve on both. Empirical results with numerous benchmark data sets show that AETAN indeed outperforms the others, with the drawback of larger computational time for inferring the models. As future work, we intend to explore different score functions and the effect of the equivalent sample size in the classification results for general purpose Bayesian network classifiers. We also intend to expand on AETAN towards more general Bayesian networks where exact and efficient learning is still possible.

Author Contributions

Both authors have designed the algorithms and developed their implementations, performed the experiments, analyzed the results and written the manuscript. Both authors have read and approved the final manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Cheng, J.; Greiner, R. Comparing Bayesian Network Classifiers. In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI), Stockholm, Sweden, 30 July–1 August 1999; Morgan Kaufmann: San Francisco, CA, USA, 1999; pp. 101–108.

2. Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian Network Classifiers. *Mach. Learn.* **1997**, *29*, 131–163.
3. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann: Burlington, MA, USA, 1988.
4. Koller, D.; Friedman, N. *Probabilistic Graphical Models*; MIT Press: Cambridge, MA, USA, 2009.
5. Chickering, D.M. *Learning Bayesian Networks is NP-Complete*; Lecture Notes in Statistics; Springer: New York, NY, USA, 1996; Volume 112, pp. 121–130.
6. Chickering, D.M.; Meek, C.; Heckerman, D. Large-Sample Learning of Bayesian Networks is NP-Hard. *J. Mach. Learn. Res.* **2004**, *5*, 1287–1330.
7. Pernkopf, F.; Bilmes, J. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In Proceedings of the International Conference on Machine Learning (ICML), Bonn, Germany, 7–11 August 2005; pp. 657–664.
8. Pernkopf, F.; Wohlmayr, M. Stochastic margin-based structure learning of Bayesian network classifiers. *Pattern Recognit.* **2013**, *46*, 464–471.
9. Pernkopf, F.; Bilmes, J.A. Efficient Heuristics for Discriminative Structure Learning of Bayesian Network Classifiers. *J. Mach. Learn. Res.* **2010**, *11*, 2323–2360.
10. Grossman, D.; Domingos, P. Learning Bayesian network classifiers by maximizing conditional likelihood. In Proceedings of the 21st International Conference on Machine Learning (ICML), Banff, AB, Canada, 1–4 July 2004; pp. 361–368.
11. De Campos, C.P.; Cuccu, M.; Corani, G.; Zaffalon, M. Extended Tree Augmented Naive Classifier. In Proceedings of the 7th European Workshop on Probabilistic Graphical Models (PGM), Utrecht, The Netherlands; Lecture Notes in Artificial Intelligence Volume 8754; Springer, Switzerland, 2014; pp. 176–189.
12. De Campos, C.P.; Corani, G.; Scanagatta, M.; Cuccu, M.; Zaffalon, M. Learning Extended Tree Augmented Naive Structures. *Int. J. Approx. Reason.* **2015**, in press.
13. Webb, G.I.; Boughton, J.R.; Wang, Z. Not So Naive Bayes: Aggregating One-Dependence Estimators. *Mach. Learn.* **2005**, *58*, 5–24.
14. Keogh, E.; Pazzani, M.J. Learning Augmented Bayesian Classifiers: A Comparison of Distribution-Based and Classification-Based Approaches. In Proceedings of the Uncertainty'99: The Seventh International Workshop on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 4–6 January 1999.
15. Qiu, C.; Jiang, L.; Li, C. Not always simple classification: Learning SuperParent for class probability estimation. *Expert Syst. Appl.* **2015**, *42*, 5433–5440.
16. Jiang, L.; Cai, Z.; Wang, D. Improving tree augmented naive Bayes for class probability estimation. *Knowl.-Based Syst.* **2012**, *26*, 239–245.
17. Jiang, L. Random one-dependence estimators. *Pattern Recognit. Lett.* **2011**, *32*, 532–539.
18. Buntine, W. Theory refinement on Bayesian networks. In Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence (UAI'92), Los Angeles, CA, USA, 13–15 July 1991; Morgan Kaufmann: San Francisco, CA, USA, 1991; pp. 52–60.

19. Cooper, G.F.; Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.* **1992**, *9*, 309–347.
20. Heckerman, D.; Geiger, D.; Chickering, D.M. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.* **1995**, *20*, 197–243.
21. De Campos, C.P.; Ji, Q. Properties of Bayesian Dirichlet Scores to Learn Bayesian Network Structures. In Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI), Atlanta, GA, USA, 11–15 July 2010; pp. 431–436.
22. Silander, T.; Myllymaki, P. A simple approach for finding the globally optimal Bayesian network structure. In Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI), Cambridge, MA, USA, 13–16 July 2006; pp. 445–452.
23. De Campos, C.P.; Ji, Q. Efficient Structure Learning of Bayesian Networks Using Constraints. *J. Mach. Learn. Res.* **2011**, *12*, 663–689.
24. Yuan, C.; Malone, B. Learning Optimal Bayesian Networks: A Shortest Path Perspective. *J. Artif. Intell. Res.* **2013**, *48*, 23–65.
25. Barlett, M.; Cussens, J. Advances in Bayesian Network Learning Using Integer Programming. In Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI), Bellevue, WA, USA, 11–15 July 2013; pp. 182–191.
26. Dasgupta, S. Learning polytrees. In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI), Stockholm, Sweden, 30 July–1 August 1999; Morgan Kaufmann: San Francisco, CA, USA, 1999; pp. 134–141.
27. Domingos, P.; Pazzani, M. On the optimality of the simple Bayesian classifier under zero-one loss. *Mach. Learn.* **1997**, *29*, 103–130.
28. Hand, D.J.; Yu, K. Idiot's Bayes-Not So Stupid after All? *Int. Stat. Rev.* **2001**, *69*, 385–398.
29. Friedman, J. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Min. Knowl. Discov.* **1997**, *1*, 55–77.
30. Corani, G.; de Campos, C.P.; Sun, Y. A tree augmented classifier based on Extreme Imprecise Dirichlet Model. In Proceedings of the International Symposium on Imprecise Probability: Theories and Applications (ISIPTA), Durham, UK, 14–18 July 2009; pp. 89–98.
31. Corani, G.; de Campos, C.P. A tree augmented classifier based on Extreme Imprecise Dirichlet Model. *Int. J. Approx. Reason.* **2010**, *51*, 1053–1068.
32. Madden, M.G. On the classification performance of TAN and general Bayesian networks. *Knowl.-Based Syst.* **2009**, *22*, 489–495.
33. Chow, C.K.; Liu, C.N. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory* **1968**, *14*, 462–467.
34. Edmonds, J. Optimum Branchings. *J. Res. Natl. Bureau Stand. B* **1967**, *71B*, 233–240.
35. Chu, Y.J.; Liu, T.H. On the Shortest Arborescence of a Directed Graph. *Sci. Sin.* **1965**, *14*, 1396–1400.
36. Kullback, S.; Leiber, R.A. On information and sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79–86.
37. Tarjan, R.E. Finding Optimum Branchings. *Networks* **1977**, *7*, 25–35.
38. Camerini, P.M.; Fratta, L.; Maffioli, F. A note on finding optimum branchings. *Networks* **1979**, *9*, 309–312.

39. Jiang, L.; Zhang, H.; Cai, Z.; Wang, D. Weighted average of one dependence estimators. *J. Exp. Theor. Artif. Intell.* **2012**, *24*, 219–230.
40. Jiang, L.; Zhang, H.; Cai, Z. A novel Bayes model: Hidden naive Bayes. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1361–1371.
41. University of Waikato. Weka 3: Data Mining Software in Java. Available online: http://www.cs.waikato.ac.nz/ml/weka/ (accessed on 1 June 2015).
42. Witten, I.H.; Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed.; Morgan Kaufmann: San Francisco, CA, USA, 2005.
43. De Campos, C.P.; Benavoli, A. Inference with multinomial data: Why to weaken the prior strength. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, Spain, 16–22 July 2011; pp. 2107–2112.
44. Scanagatta, M.; de Campos, C.P.; Zaffalon, M. Min-BDeu and Max-BDeu Scores for Learning Bayesian Networks. In Proceedings of the 8th European Workshop on Probabilistic Graphical Models (PGM), Utrecht, The Netherlands, 17–19 September 2014; Volume 8754, pp. 426–441.
45. Silander, T.; Kontkanen, P.; Myllymäki, P. On Sensitivity of the MAP Bayesian Network Structure to the Equivalent Sample Size Parameter. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), Vancouver, BC, Canada, 19–22 July 2007; pp. 360–367.
46. Asuncion, A.; Newman, D.J. UCI Machine Learning Repository. 2007. Available online: http://www.ics.uci.edu/~mlearn/MLRepository.html (accesssed on 1 June 2015).