Bayes Optimal Classification for Decision Trees

Siegfried Nijssen SIEGFRIED.NIJSSEN@CS.KULEUVEN.BE
K.U. Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

We present an algorithm for exact Bayes optimal classification from a hypothesis space of decision trees satisfying leaf constraints. Our contribution is that we reduce this classification problem to the problem of finding a rule-based classifier with appropriate weights. We show that these rules and weights can be computed in linear time from the output of a modified frequent itemset mining algorithm, which means that we can compute the classifier in practice, despite the exponential worst-case complexity. In experiments we compare the Bayes optimal predictions with those of the maximum a posteriori hypothesis.

1. Introduction

We study the problem of Bayes optimal classification for density estimation trees. A density estimation tree in this context is a decision tree which has a probability density for a class attribute in each of its leaves. One can distinguish two Bayesian approaches to density estimation using a space of such trees.

In the first approach a single *maximum a posteriori* (MAP) density estimation tree is identified first:

$$T = \arg\max_T P(T|\mathbf{X}, \vec{y}),$$

where $\mathbf{X}$ and $\vec{y}$ together constitute the training data. The posterior probability $P(T|\mathbf{X}, \vec{y})$ of a hypothesis T is usually the product of a prior and a likelihood. The MAP hypothesis can then be used to classify a test example x' using the densities in the leaves.

The second approach is to marginalize over all possible trees, instead of preferring a single one:

$$\arg\max_c P(c|x', \mathbf{X}, \vec{y}) = \arg\max_c \sum_T P(c|x', T)P(T|\mathbf{X}, \vec{y}).$$

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

Predictions that are performed using this second approach are called *Bayes optimal predictions.* It has been claimed that "no single tree classifier using the same prior knowledge as an optimal Bayesian classifier can obtain better performance on average" (Mitchell, 1997). The Bayesian point of view is that Bayesian averaging cancels out the effects of overfitted models (Buntine, 1990), and "solves" overfitting problems.

This claim was challenged by Domingos (2000). Domingos demonstrated experimentally that an ensemble of decision trees that are weighted according to posterior probabilities performs worse than uniformly weighted hypotheses. It was found that one overfitting tree usually dominates an ensemble.

However, these results were obtained by sampling from the hypothesis space. Even though Domingos argued that similar issues should also occur in the truly optimal approach, this claim could not be checked in practice as the exact computation of Bayes optimal predictions was considered to be impractical. Indeed, in (Chipman et al., 1998) it was already claimed that "exhaustive evaluation ... over all trees will not be feasible, except in trivially small problems, because of the sheer number of trees". Similar claims were made in other papers studying Bayesian tree induction (Buntine, 1992; Chipman et al., 1998; Angelopoulos & Cussens, 2005; Oliver & Hand, 1995), and have led to the use of sampling techniques such as Markov Chain Monte Carlo sampling.

In this paper we present an algorithm that can be used to evaluate Domingos' claim in a reasonable number of non-trivial settings. Our algorithm allows us to exactly compute the Bayes optimal predictions given priors that assign non-zero probability to trees that satisfy certain constraints. An example of a constraint is that every leaf covers a significant number of examples; this constraint has been used very often in the literature (Buntine, 1992; Quinlan, 1993; Chipman et al., 1998; Angelopoulos & Cussens, 2005; Oliver & Hand, 1995).

Our algorithm is an extension of our earlier work, in which we developed the DL8 algorithm for determining one tree that maximizes accuracy (Nijssen & Fromont, 2007). DL8 is based on dynamic programming on a pre-computed lattice of itemsets, and scans these itemsets decreasing in size. Its time complexity is linear in the size of the lattice. In this paper we extend this algorithm to a Bayesian setting. From a technical point of view, the main contribution is that we prove that a different pass over the lattice allows us to perform Bayes optimal predictions without increasing the asymptotic complexity of building the lattice.

The task that our algorithm addresses is similar to the task addressed in (Cleary & Trigg, 1998). Compared to this earlier work, we study the more common Dirichlet priors also considered in (Chipman et al., 1998; Angelopoulos & Cussens, 2005); furthermore, by exploiting the link to itemset mining, our algorithm is more efficient, and its results are more interpretable.

The paper is organized as follows. Notation and concepts are introduced in Section 2. Bayes optimal classification is formalized in Section 3. We show how to map this problem to the problem of finding itemsets and building a classifier with weighted rules in Section 4. Experiments are performed in Section 5.

2. Preliminaries

Before we are ready to formalize our problem and our proposed solution, we require some notation. We restrict ourselves to binary data; we assume that data is converted in this form in a preprocessing step. The data is stored in binary matrix $\mathbf{X}$, of which each row $\vec{x}_k$ corresponds to one example. Every example $\vec{x}_k$ has a class label y_k out of a total number of C class labels. Class labels are collected in a vector $\vec{y}$.

We assume that the reader is familiar with the concept of decision trees (see (Breiman et al., 1984; Quinlan, 1993) for details). Essential in our work is a link between decision trees and *itemsets*. Itemsets are a concept that was introduced in the data mining literature (Agrawal et al., 1996). If $\mathcal{I}$ is a domain of items, $I \subseteq \mathcal{I}$ is an itemset. In our case, we assume that we have *two* types of items: for every attribute there is a positive item i that represents a positive value, and a negative item $\neg i$ that represents a negative value. An example $\vec{x}$ can be represented as an itemset

$$\{i|x_i = 1\} \cup \{\neg i|x_i = 0\}.$$

Thus, for a data matrix with n columns, we have that $\mathcal{I} = \mathcal{I}_{pos} \cup \mathcal{I}_{neg}$, where $\mathcal{I}_{pos} = \{1, 2, \ldots n\}$ and $\mathcal{I}_{neg} = \{\neg 1, \neg 2, \ldots \neg n\}$. We overload the use of the $\subseteq$ operator: when I is an itemset, and $\vec{x}$ is an example, we use $I \subseteq \vec{x}$ to denote that I is a subset of $\vec{x}$ after translating $\vec{x}$ into an itemset.

Every sequence of test outcomes in a decision tree, starting from the root of the tree to an arbitrary node deeper down the tree, can be represented as an itemset. For instance, a decision tree with B in the root, and A in its right-hand branch can be represented by:

$$T = \{\emptyset, \{B\}, \{\neg B\}, \{\neg B, A\}, \{\neg B, \neg A\}\}.$$

Every itemset in T corresponds to one node in the tree. By $\mathcal{T}$ we denote all subsets of $2^{\mathcal{I}}$ that represent decision trees. A decision tree structure is an element $T \in \mathcal{T}$. Consequently, when T is a decision tree we can write $I \in T$ to determine if the itemset I corresponds to a path occurring in the tree.

An itemset is an unordered set: given an itemset in a tree, we cannot derive from this itemset in which order its tests appear in the tree. This order can only be determined by considering all itemsets in a tree T.

We are not always interested in all nodes of a tree. The subset of itemsets that correspond to the leaves of a tree T will be denoted by $leaves(T)$; in our example,

$$leaves(T) = \{\{B\}, \{\neg B, A\}, \{\neg B, \neg A\}\}.$$

The most common example of a decision tree is the *classification tree*, in which every leaf is labeled with a single class. In a *density estimation tree*, on the other hand, we attach a class *distribution* to each leaf, represented by a vector $\vec{\theta}_I$; for each class c this vector contains the probability θ_{Ic} that examples $\vec{x} \supseteq I$ belong to class c. All the parameters of the leaves of a tree are denoted by $\mathbf{\Theta}$. The vectors in $\mathbf{\Theta}$ are thus indexed by the itemsets representing leaves of the tree.

For the evaluation of a tree T on a binary matrix $\mathbf{X}$, it is useful to have a shorthand notation for the number of examples covered by a leaf:

$$f(I, \mathbf{X}) = |\{\vec{x}_k | \vec{x}_k \supseteq I\}|\,;$$

usually we omit the matrix $\mathbf{X}$ in our notation, as we assume the training data to be fixed. We call $f(I, \mathbf{X})$ the *frequency* of I. Class-based frequency is given by:

$$f_c(I, \mathbf{X}, \vec{y}) = |\{\vec{x}_k | \vec{x}_k \supseteq I, y_k = c\}|\,.$$

The *frequent itemset mining* problem is the problem of finding all $I \subseteq \mathcal{I}$ such that $f(I) \geq \gamma$, for a given threshold γ. Many algorithms for computing this set exist (Agrawal et al., 1996; Goethals & Zaki, 2003). They are based on the property that the frequency constraint is *anti-monotonic.* A binary constraint p on itemsets is called anti-monotonic iff $\forall I' \subseteq I : p(I) =$

true $\implies p(I') = \textit{true}$. Consequently, these algorithms do not need to search through all supersets $I' \supseteq I$ of an itemset I that is found to be infrequent.

One application of itemsets is in the construction of rule-based classifiers (CMAR (Li et al., 2001) is an example). Many rule-based classifiers traverse rules sequentially when predicting examples. Here, we study a rule-based classifier that derives a prediction from all rules through voting. Such a classifier can be seen as a subset $\mathcal{P} \subseteq 2^{\mathcal{I}}$ of itemsets, each of which has a weight vector $\vec{w}(I)$. We predict an example $\vec{x}$ by computing

$$\arg\max_c \sum_{\{I \in \mathcal{P} | I \subseteq \vec{x}\}} w_c(I),$$

where we thus pick the class that gets most votes of all rules in the ruleset; each rule votes with a certain weight on each class. The aim of this paper is to show that we can derive a set of itemsets $\mathcal{P}$ and weights $\vec{w}(I)$ for all $I \in \mathcal{P}$ such that the predictions of the rule-based classifier equal those of a Bayes optimal classifier. The rules in $\mathcal{P}$ represent all paths that can occur in trees in the hypothesis space.

3. Problem Specification

In this section we formalize the problem of Bayes optimal classification for a hypotheses space of decision trees. Central in the Bayesian approach is that we first define the probability of the data given a tree structure T and parameters $\mathbf{\Theta}$:

$$P(\vec{y}|\mathbf{X}, T, \mathbf{\Theta}) = \prod_{I \in \textit{leaves}(T)} \prod_{c=1}^{C} (\theta_{Ic})^{f_c(I)}$$

In Bayes optimal classification we are interested in finding for a particular example $\vec{x}'$ the class y' which maximizes the probability

$$\begin{aligned} y' &= \arg\max_c P(c|\vec{x}', \mathbf{X}, \vec{y}) & (1) \\ &= \arg\max_c \sum_{T \in \mathcal{T}} \int_{\mathbf{\Theta}} P(c|\vec{x}', T, \mathbf{\Theta}) P(T, \mathbf{\Theta}|\mathbf{X}, \vec{y}) d\mathbf{\Theta}, \end{aligned}$$

where we sum over the space of all decision trees and integrate over all possible distributions in the leaves of each tree. Applying Bayes' rule on the second term, and observing that $\mathbf{\Theta}$ is dependent on the tree T, we can rewrite this into

$$\sum_{T \in \mathcal{T}} \int_{\mathbf{\Theta}} P(c|\vec{x}', T, \mathbf{\Theta}) P(\vec{y}|T, \mathbf{\Theta}, \mathbf{X}) P(\mathbf{\Theta}|T, \mathbf{X}) P(T|\mathbf{X}) d\mathbf{\Theta}; \quad (2)$$

in this formula $P(T|\mathbf{X})$ is the probability of a tree given that we have seen all data except the class labels. Our method is based on the idea that we can constrain the space of decision trees by manipulating this term.

A first possibility is that we set $P(T|\mathbf{X}) = 0$ if there is a leaf $I \in \textit{leaves}(T)$ such that $f(I) < \gamma$, for a frequency threshold γ. We call such leaves *small leaves*. The class estimates of a small leaf are often unreliable, and it is therefore common in many algorithms to consider only large leaves.

Additionally, we can set $P(T|\mathbf{X}) = 0$ if the depth of the decision tree exceeds a predefined threshold.

Both limitations impose *hard constraints* on the trees that are considered to be feasible estimators. We denote trees in $\mathcal{T}$ that satisfy all hard constraints by $\mathcal{L}$.

In the simplest case we can assume a uniform distribution on the trees that satisfy the hard constraints. Effectively, this would mean that we set $P(\mathbf{\Theta}|T, \mathbf{X}) = 1$ in Equation 2 for all $T \in \mathcal{L}$. However, we will study a more sophisticated prior in this paper to show the power of our method. The aim of this prior, which was proposed in (Chipman et al., 1998), is to give more weight to smaller trees; it can be seen as a *soft constraint*. This prior is defined as follows.

$$P(T|\mathbf{X}) = \prod_{I \in T} P_{node}(I, T, \mathbf{X})$$

Here, the term $P_{node}(I, T, \mathbf{X})$ is defined as follows.

$$P_{node}(I, T, \mathbf{X}) = \begin{cases} P_{leaf}(I, \mathbf{X}), & \text{if } I \text{ is a leaf in } T; \\ P_{intern}(I, \mathbf{X}), & \text{if } I \text{ is internal in } T; \end{cases}$$

where

$$P_{leaf}(I, \mathbf{X}) = \begin{cases} 0, & \text{if } f(I) < \gamma \text{ or } |I| > \delta; \\ 1, & \text{else if } |I| = \delta \text{ or } e(I) = 0; \\ 1 - \alpha(1 + |I|)^{-\beta}, & \text{otherwise;} \end{cases}$$

and

$$P_{intern}(I, \mathbf{X}) = \begin{cases} 0, & \text{if } f(I) < \gamma \text{ or } |I| \geq \delta \\ & \text{or } e(I) = 0; \\ \alpha(1 + |I|)^{-\beta}/e(I), & \text{otherwise;} \end{cases}$$

Here $e(I)$ is the size of the set $\{i \in \mathcal{I}_{pos} | f(I \cup i) \geq \gamma \wedge f(I \cup \neg i) \geq \gamma\}$, which consists of all possible tests that can still be performed to split the examples covered by itemset I.

The term $\alpha(1 + |I|)^{-\beta}$ makes it less likely that nodes at a higher depth are split. The term $e(I)$ determines how many tests are still available if a test is to be performed. We assume that tests are apriori equally likely, independent of the order in which the previous tests on the path have been performed. An alternative could be to give more likelihood to tests that are well-balanced.

Note that P_{leaf} and P_{intern} are computed for an itemset I almost independently from the tree T: we only need to know if I is a leaf or not.

As common in Bayesian approaches, we assume that the parameters in every leaf of the tree are Dirichlet distributed with the same parameter vector $\vec{\alpha}$, i.e.

$$P(\mathbf{\Theta}|T,\mathbf{X}) = P(\mathbf{\Theta}|T) = \prod_{I \in leaves(T)} Dir(\vec{\theta}_I|\vec{\alpha}),$$

where

$$Dir(\vec{\theta}_I|\vec{\alpha}) = \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \prod_c {\theta_{Ic}}^{\alpha_c - 1},$$

and Γ is the gamma function.

Finally, it can be seen that

$$P(c|\vec{x}', T, \mathbf{\Theta}) = \theta_{I(T,\vec{x}')c},$$

where $I(T,\vec{x}')$ is the leaf of T for which $I \subseteq \vec{x}'$.

We now have formalized all terms of Equation 2.

4. Solution Strategy

An essential step in our solution strategy is the construction of the set

$$\mathcal{P} = \{I | T \in \mathcal{L}, I \in T\},$$

which consists of all itemsets in trees that satisfy the hard constraints. Only these paths are needed when we wish to compute the posterior distribution over class labels, and are used as rules in our rule-based classifier. The weights of these rules are obtained by rewriting the Bayesian optimization criterion for a test example $\vec{x}'$ (Equation 1) as

$$\arg\max_c \sum_{\{I \in \mathcal{P} | I \subseteq \vec{x}'\}} w_c(I)$$

where

$$w_c(I) = \sum_{T \in \mathcal{L},\text{ has leaf } I} \left(\prod_{I \in T} P_{node}(I,T,\mathbf{X}) \right)$$
$$\left(\int_{\mathbf{\Theta}} \theta_{Ic} \prod_{I' \in leaves(T)} Dir(\vec{\theta}'_I|\vec{\alpha}) \prod_c \theta_{I'c}^{f_c(I')} d\mathbf{\Theta} \right). \quad (3)$$

The idea behind this rewrite is that the set of all trees in $\mathcal{L}$ can be partitioned by considering in which leaf a test example ends up. An example ends in exactly one leaf in every tree, and thus every tree belongs to one partition as determined by that leaf. We sum first over all possible leaves that can contain the example, and then over all trees having that leaf. The weights of the rules in our classifier consist of the terms $w_c(I)$, and will be computed from the training data in the training phase; the sum of the weights $w_c(I)$ is computed for a test example in the classification phase.

This rewrite shows that in the training phase we need to compute weights for all itemsets that are in $\mathcal{P}$. We will discuss now how to compute these.

In the formulation above we multiply over all leaves, including the leaf that we assumed the example ended up in. Taking this special leaf apart we obtain:

$$w_c(I) = W_c(I) \sum_{T \in \mathcal{L},\text{ has leaf } I} \prod_{I' \in T, I' \neq I} V(I',T); \quad (4)$$

where

$$W_c(I) = P_{leaf}(I,\mathbf{X}) \int_{\vec{\theta}_I} \theta_{Ic} Dir(\vec{\theta}_I|\vec{\alpha}) \prod_c \theta_{Ic}^{f_c(I)} d\vec{\theta}_I$$

and

$$V(I,T) = \begin{cases} P_{intern}(I,\mathbf{X}) & \text{if } I \text{ is internal in } T; \\ P_{leaf}(I,\mathbf{X}) \int_{\vec{\theta}_I} Dir(\vec{\theta}_I|\vec{\alpha}) \prod_c \theta_{Ic}^{f_c(I)} d\vec{\theta}_I, & \\ & \text{otherwise.} \end{cases}$$

This rewrite is correct due to the fact that we can move the integral of Equation 3 within the product over the leaves: the parameters of the leaves are independent from each other.

Let us write the integrals in closed form. First consider $W_c(I)$. As the Dirichlet distribution is the conjugate prior of the binomial distribution, we have

$$W_c(I) =$$
$$P_{leaf}(I,\mathbf{X}) \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \int_{\vec{\theta}_I} \theta_{Ic} \prod_c \theta_{Ic}^{\alpha_c - 1 + f_c(I)} d\vec{\theta}_I =$$
$$P_{leaf}(I,\mathbf{X}) \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \frac{\prod_c \Gamma(\alpha_c + f'_c(I))}{\Gamma(\sum_c \alpha_c + f'_c(I))}$$

Here $f'_{c'}(I) = f_{c'}(I)$ if $c \neq c'$, else $f'_{c'}(I) = f_{c'}(I) + 1$.

Similarly, we can compute $V(I,T)$ as follows.

$$V(I,T) = \begin{cases} V_{intern}(I) = P_{intern}(I,\mathbf{X}), \\ \qquad \text{if } I \text{ is internal in } T; \\ V_{leaf}(I) = \\ \quad P_{leaf}(I,\mathbf{X}) \frac{\Gamma(\sum_c \alpha_c) \prod_c \Gamma(\alpha_c + f_c(I))}{\prod_c \Gamma(\alpha_c) \Gamma(\sum_c \alpha_c + f_c(I))}, \\ \qquad \text{otherwise.} \end{cases}$$

The remaining question is now how to avoid summing all trees of Equation 4 explicitly. In the following, we will derive a dynamic programming algorithm to implicitly compute this sum. We use a variable that is defined as follows.

$$u(I) = \sum_{T \in \mathcal{L}(I)} \prod_{I' \in T} V(I', T); \qquad (5)$$

Here we define $\mathcal{L}(I)$ as follows:

$$\mathcal{L}(I) = \{\{I' \in T | I' \supseteq I\} | \text{ all } T \in \mathcal{L} \text{ for which } I \in T\};$$

thus, $\mathcal{L}(I)$ consists of all subtrees that can be put below an itemset I while satisfying the hard constraints. As usual, we represent a subtree by listing all its paths.

For this variable we will first prove the following.

Theorem 1. *The following recursive relation holds for $u(I)$:*

$$u(I) = V_{leaf}(I) + \sum_{i \in \mathcal{I}_{pos} \text{ s.t. } I \cup i, I \cup \neg i \in \mathcal{P}} V_{intern}(I) u(I \cup i) u(I \cup \neg i).$$

Proof. We prove this by induction. Assume that for all itemsets $|I| > k$ our definition holds. Let us fill in our definition in the recursive formula, then we get:

$$u(I) = V_{leaf}(I) + \sum_{i \in \mathcal{I}_{pos}, \text{ s.t. } I \cup i, I \cup \neg i \in \mathcal{P}} \sum_{T \in \mathcal{L}(I \cup i)} \sum_{T' \in \mathcal{L}(I \cup \neg i)} V_{intern}(I) \prod_{I' \in T} V(I', T) \prod_{I' \in T'} V(I', T');$$

This can be written as Equation 5 to prove our claim: the term for V_{leaf} corresponds to the possibility that I is a leaf, the first sum passes over all possible tests if the node is internal, the second and third sum traverse all possible left-hand and right-hand subtrees; the product within the three sums is over all nodes in each resulting tree. $\square$

We can use this formula to write $w_c(I)$ as follows.

Theorem 2. *The formula $w_c(I)$ can be written as:*

$$w_c(I) = W_c(I) \sum_{\pi \in \Pi(I)} \prod_{i=1}^{|I|} V_{intern}(\{\pi_1, \ldots, \pi_{i-1}\}) u(\{\pi_1, \ldots, \pi_{i-1}, \neg\pi_i\}).$$

Here, $\Pi(I)$ contains all permutations $(\pi_1, \ldots, \pi_n)$ of the items in I for which it holds that $\forall 1 \leq i \leq n : \{\pi_1, \ldots, \pi_i\}, \{\pi_1, \ldots, \pi_{i-1}, \neg\pi_i\} \in \mathcal{P}$.

Proof. The set of permutations $\Pi(I)$ consists of all (ordered) paths that can be constructed from the items in I and that fulfill the constraints on size and frequency. Each tree $T \in \mathcal{L}$ with $I \in T$ must have exactly one of these paths. Given one such path, Equation 4 requires us to sum over all trees that contain this path. Each tree in this sum consists of a particular choice of subtrees for each sidebranch of the path. Every node in a tree $T \in \mathcal{L}$ with $I \in T$ is either (1) part of the path to node I or (2) part of a sidebranch; this means that we can decompose the product $\prod_{I' \in T, I' \neq I} V(I', T)$, which is part of Equation 4, into a product for nodes in sidebranches, and a product for nodes on the path to I. The term for nodes on the path is computed by

$$W_c(I) \prod_{i=1}^{|I|} V_{intern}(\{\pi_1, \ldots, \pi_{i-1}\});$$

considering the side branches, $u(I)$ sums over all possible subtrees below sidebranches of the path $\{\pi_1, \ldots, \pi_n\}$; using the product-of-sums rule that $\prod_{i=1}^{n} \sum_{j=1}^{m_i} \alpha_{ij} = \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} x_{1\alpha_1} \cdots \alpha_{ni_n}$, where $\sum_{j=1}^{m_i} \alpha_{ij}$ corresponds to a u-value of a sidebranch, we can deduce that the product $\prod_{i=1}^{|I_k|} u(\{\pi_1, \ldots, \pi_{i-1}, \neg\pi_i\})$ sums over all possible combinations of side branches. $\square$

Given their potentially exponential number it is undesirable to enumerate all permutations of item orders for every itemset. To avoid this let us define

$$v(I) = \sum_{\pi \in \Pi(I)} \prod_{k=1}^{|I|} V_{intern}(\{\pi_1, \ldots, \pi_{k-1}\}) u(\{\pi_1, \ldots, \pi_{k-1}, \neg\pi_k\}),$$

such that $w_c(I) = W_c(I) v(I)$.

Theorem 3. *The following recursive relation holds.*

$$v(I) = \begin{cases} 1, & \text{if } I = \emptyset, \\ \sum_{i \in I \text{ s.t. } I - i \cup \neg i \in \mathcal{P}} V_{intern}(I - i) \\ \quad u(I - i \cup \neg i) v(I - i), & \text{otherwise.} \end{cases}$$

Proof. This can be shown by induction: if we fill in our definition of $v(I)$ in the recursive formula we get

$$\sum_{i \in I \text{ s.t. } I - i \cup \neg i \in \mathcal{P}} V_{intern}(I - i) u(I - i \cup \neg i)$$

$$\sum_{\pi \in \Pi(I - i)} \prod_{k=1}^{|I|-1} V_{intern}(\{\pi_1, \ldots, \pi_{k-1}\}) u(\{\pi_1, \ldots, \neg\pi_k\})$$

Both sums together sum exactly over all possible permutations of the items; the product is exactly over all terms of every permutation. $\square$

Algorithm 1 Compute Bayes Optimal Weights

```
input The set of itemsets $\mathcal{P}$ and for all $I \in \mathcal{P} : \vec{f}(I)$
output The weight vectors $\vec{w}(I)$ for all $I \in \mathcal{P}$
 1: % Bottom-up Phase
 2: Let $n$ be the size of the largest itemset in $\mathcal{P}$
 3: for $k := n$ downto $0$ do
 4:    for all $I \in \mathcal{P}$ s.t. $|I| = k$ do
 5:       $u[I] := V_{leaf}(I)$
 6:       for all $i \in \mathcal{I}_{pos}$ s.t. $I \cup i, I \cup \neg i \in \mathcal{P}$ do
 7:          $u[I] := u[I] + V_{intern}(I)u[I \cup i]u[I \cup \neg i]$
 8:       end for
 9:    end for
10: end for
11: % Top-down Phase
12: $v[\emptyset] := 1$
13: for $k := 1$ to $n$ do
14:    for all $I \in \mathcal{P}$ s.t. $|I| = k$ do
15:       $v[I] := 0$
16:       for all $i \in I$ s.t. $I - i \cup \neg i \in \mathcal{P}$ do
17:          $v[I] := v[I] + V_{intern}(I - i)u[I - i \cup \neg i]v[I - i]$
18:       end for
19:       for $c := 1$ to $C$ do
20:          $w_c[I] := W_c(I)v[I]$
21:       end for
22:    end for
23: end for
```

A summary of our algorithm is given in Algorithm 1. The main idea is to apply the recursive formulas for $u(I)$ and $v(I)$ to perform dynamic programming in two phases: one bottom-up phase to compute the $u(I)$ values, and one top-down phase to compute the $v(I)$ values. Given appropriate data structures to perform the look-up of sub- and supersets of itemsets I, this procedure has complexity $\mathcal{O}(|\mathcal{P}|\delta C)$. As $|\mathcal{P}| = \mathcal{O}(n2^m)$, where n is the number of examples in the training data and m the number of attributes, this algorithm is exponential in the number of attributes.

After the run of this algorithm, for a test example we can compute $q_c(\vec{x}') = \sum_{I \subseteq \vec{x}'} w_c(I)$ for every c. We can easily compute the exact class probability estimates from this: $P(y' = c|\vec{x}', \mathbf{X}, \vec{y}) = \frac{q_c(\vec{x}')}{\sum_{c'} q_{c'}(\vec{x}')}$.

To compute the set $\mathcal{P}$ of paths in feasible trees, we can modify a frequent itemset miner (Goethals & Zaki, 2003), as indicated in our earlier work (Nijssen & Fromont, 2007). We replace the itemset lattice post-processing method of (Nijssen & Fromont, 2007) by the algorithm for computing Bayes optimal weights.

Compared to the OB1 algorithm of Cleary & Trigg (1998), the main advantage of our method is its clear link to frequent itemset mining. OB1 is based on the use of option trees, which have a worst case complexity of $O(nm!)$ instead of $O(n2^m)$. Cleary et al. suggest that sharing subtrees in option trees could improve performance; this exactly what our approach achieves in a fundamental way. The link between weighted rule-based and Bayes optimal classification was also not made by Cleary et al., making the classification phase either more time or space complex. We can interpret predictions by our approach by listing the (maximal) itemsets that contribute most weight to a prediction.

5. Experiments

We do not perform a feasibility study here, as we did such a study in earlier work (Nijssen & Fromont, 2007).

We performed several experiments to determine the importance of the α and β parameters of the size prior. We found that the differences between values $\alpha, \beta \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$ were often not significant and choose $\alpha = 0.80$ and $\beta = 0.80$ as defaults. We also experimented with a uniform prior. We choose $\vec{\alpha} = (1.0, \ldots, 1.0)$ as default for the Dirichlet prior. This setting is common in the literature.

All comparisons were tested using a corrected, two-tailed, paired $t-$test with a 95% confidence interval.

Artificial Data In our first set of experiments we use generated data. We use this data to confirm the influence of priors and the ability of the Bayes optimal classifier to recognize that data can best be represented by an ensemble of multiple trees.

A common approach is to generate data from a model and to compute how well a learning algorithm recovers this original model. In our setting this approach is however far from trivial, as it is hard to generate a realistic lattice of itemsets: Calders (2007) showed that it is NP-hard to decide if a set of itemset frequencies can occur at all in data. Hence we used an alternative approach. The main idea is that we wish to generate data such that different trees perform best on different parts of the data. We proceed as follows: we first generate n tree structures (in our experiments, all trees are complete trees of depth 7; the trees do not yet have class labels in their leaves); from these n trees we randomly generate a database of given size (4000 examples with 15 binary attributes in our experiments, without class labels). We make sure that every leaf in every tree has at least γ examples (3% of the training data in our experiments). Next, we iterate in a fixed order over these trees to assign classes to the examples in one leaf of each tree; in each tree we pick the leaf which has the largest number of examples without class, and assign a class to these examples, taking care that two adjacent leaves get different majority classes. We aim for pure leaves, but these are less likely for higher numbers of generating trees.

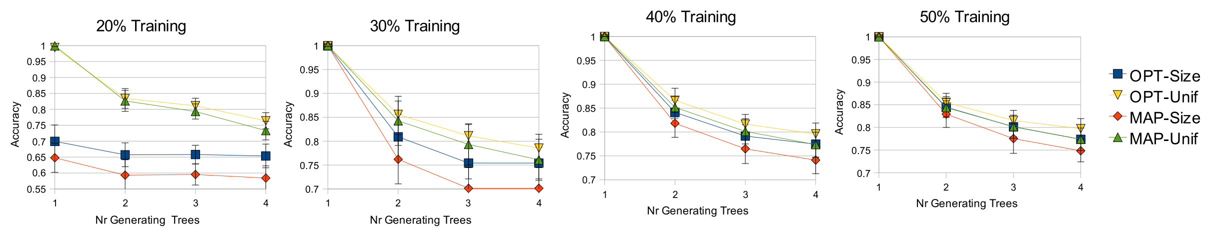

Figure 1. Results on artificial data.

The results of our experiments are reported in Figure 1. The accuracies in these experiments are computed for 20 randomly generated datasets. Each figure represents a different fraction of examples used as training data; remaining examples were as test data. The learners were run using the same depth and support constraints as used to generate the data.

We can learn the following from these experiments.

As all our datasets were created from trees with maximal height, the prior which prefers small trees performs worse than the one which assigns equal weight to all trees. If the amount of training data is small, the size prior forces the learner to prefer trees which are not 100% accurate for data created from one tree.

In all cases, the Bayes optimal approach is significantly more accurate than the corresponding MAP approach, except if the data was created using a single tree; in this case we observe that a single (correct) tree is dominating the trees in the ensembles.

The more training data we provide, the smaller the differences between the approaches are. For the correct prior the optimal approach has a better learning curve.

Additional experiments (not reported here) for other tree depths, dataset sizes and less pure leaves confirm the results above, although sometimes less pronounced.

UCI Data In our next set of experiments we determine the performance of our algorithm on common benchmark data, using ten-fold cross validation.

The frequency and depth constraints in our prior influence the efficiency of the search; too low frequency or too high depth constraints can make the search infeasible. Default values for δ that we considered were 4, 6 and ∞; for γ we considered 2, 15 and 50. We relaxed the constraints as much as was computationally possible; experiments (not reported here) show that this usually does not worsen accuracy.

As our algorithm requires binary data, numeric attributes were discretized in equifrequency bins. Only a relatively small number of 4 bins was feasible in all experiments; we used this value in all datasets to avoid drawing conclusions after parameter overfitting. Where feasible within the range of parameters used, we added results for other numbers of bins to investigate the influence of discretization.

The experiments reported in Figure 1 help to provide more insight in the following questions:

(Q1) Is a single tree dominating a Bayes optimal classifier in practice?

(Q2) Are there significant differences between a uniform and a size-based prior in practice?

(Q3) Is the optimal approach overfitting more in practice than the traditional approach, in this case Weka's implementation of C4.5?

(Q4) What is the influence of the 4-bin discretization?

To get an indication about (Q1) we compare the optimal and MAP predictions. We underlined those cases where there is a significant difference between optimal and MAP predictions. We found that in many cases there is indeed no significant difference between these two settings; in particular when hard constraints impose a high bias, such as in the Segment and Vote data, most predictions turn out to be equal. If there is a significant difference, the optimal approach is always the most accurate.

To answer (Q2) we highlighted in bold for each dataset the system that performs significantly better than all other systems. In many cases, the differences between the most accurate settings are not significant; however, our results indicate that a uniform prior performs slightly better than a size prior in the Bayes optimal case; the situation is less clear in the MAP setting.

Answering (Q3), we found not many significant differences between J48's and Bayes optimal predictions in those cases where we did not have to enforce very hard constraints to turn the search feasible. This supports the claim of Domingos (2000) that Bayes optimal predictions are not really much better. However, our results also indicate that there is no higher risk of overfitting either. The optimal learner does not perform as well as J48 in those cases where the search is only feasible for high frequency or low depth constraints, and

Dataset	γ	δ	Bins	Accuracy				
				Opt - Size	MAP - Size	Opt - Unif	MAP - Unif	J48
Anneal	2	6	4	0.81±0.02	0.80±0.01	0.82±0.03	0.81±0.03	0.82±0.04
Anneal	15	6	10	0.86±0.04	0.86±0.04	0.86±0.04	0.85±0.04	**0.89**±0.03
Anneal	2	4	10	0.81±0.02	0.81±0.01	0.81±0.01	0.81±0.01	**0.89**±0.03
Balance	2	∞	4	$\underline{0.81}$±0.04	0.76±0.06	$\underline{\mathbf{0.84}}$±0.03	0.83±0.03	0.76±0.06
Balance	2	∞	10	$\underline{0.80}$±0.03	0.74±0.06	$\underline{\mathbf{0.85}}$±0.03	0.79±0.03	0.78±0.03
Heart	2	6	4	$\underline{0.82}$±0.07	0.79±0.05	$\underline{\mathbf{0.84}}$±0.05	0.73±0.08	0.78±0.06
Heart	2	4	10	0.81±0.06	0.79±0.05	$\underline{0.81}$±0.06	0.78±0.04	0.79±0.05
Vote	15	4	–	0.95±0.03	0.96±0.02	0.95±0.03	0.94±0.03	0.96±0.02
Segment	15	4	4	0.78±0.02	0.78±0.02	0.78±0.02	0.78±0.02	**0.95**±0.02
P-Tumor	2	∞	–	$\underline{0.40}$±0.05	0.37±0.05	$\underline{0.43}$±0.05	0.37±0.05	0.40±0.05
Yeast	2	6	4	0.52±0.03	0.52±0.03	$\underline{0.53}$±0.03	0.52±0.03	0.54±0.05
Yeast	2	4	10	0.51±0.03	0.50±0.03	0.49±0.03	0.49±0.03	**0.58**±0.03
Diabetes	2	6	4	0.75±0.06	0.74±0.06	$\underline{0.75}$±0.05	0.71±0.05	0.74±0.06
Diabetes	2	4	10	0.76±0.05	0.75±0.04	0.77±0.05	0.75±0.05	0.74±0.06
Ionosphere	15	4	4	0.87±0.06	0.87±0.06	0.87±0.06	0.87±0.05	0.86±0.07
Ionosphere	15	4	10	0.91±0.04	0.91±0.04	0.90±0.03	0.88±0.03	0.92±0.03
Vowel	50	6	4	0.42±0.04	0.40±0.04	0.41±0.07	0.38±0.05	**0.78**±0.04
Vehicle	50	6	4	$\underline{0.67}$±0.03	0.66±0.03	0.66±0.03	0.65±0.03	0.70±0.04

Table 1. Experimental results on UCI data. A result is highlighted if it is the best in its row; significant winners of comparisons between MAP and Opt settings are underlined. Bins are not indicated for datasets without numeric attributes.

thus quite unrealistic priors; in (Nijssen & Fromont, 2007) we found that under the same hard constraints J48 is not able to find accurate trees either, and often finds even worse trees in terms of accuracy.

To provide more insight in (Q4), we have added results for different discretizations. In the datasets where we used harder constraints to make the search feasible, a negative effect on accuracy is observed compared to J48. Where the same hard constraints can be used we observe similar accuracies as in J48. The experiments do not indicate that a higher number of bins leads to increased risks of overfitting.

6. Conclusions

Our results indicate that instead of constructing the optimal MAP hypothesis, it is always preferable to use the Bayes optimal setting; even though we found many cases in which the claim of Domingos (2000) is confirmed and a single tree performs equally well, in those cases where there is a significant difference, the comparison is always in favor of the optimal setting. The computation of both kinds of hypothesis remains challenging if no hard constraints are applied, while incorrect constraints can have a negative impact.

Acknowledgments S. Nijssen was supported by the EU FET IST project "IQ", contract number FP6-516169.

References

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). Fast discovery of association rules. In *Advances in knowledge discovery and data mining*, 307–328.

Angelopoulos, N., & Cussens, J. (2005). Exploiting informative priors for Bayesian classification and regression trees. *Proceedings of IJCAI* (pp. 641–646).

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Statistics/Probability Series. Belmont, California, U.S.A.

Buntine, W. (1990). *A theory of learning classification rules*. Doctoral dissertation, Sydney.

Buntine, W. (1992). Learning classification trees. *Statistics and Computing 2* (pp. 63–73).

Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, *93*, 935–947.

Cleary, J. G., & Trigg, L. E. (1998). *Experiences with OB1, an optimal Bayes decision tree learner* (Technical Report). University of Waikato.

Domingos, P. (2000). Bayesian averaging of classifiers and the overfitting problem. *Proceedings of ICML* (pp. 223–230).

Goethals, B., & Zaki, M. J. (Eds.). (2003). *Proceedings of the ICDM 2003 FIMI workshop*, vol. 90 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. *Proceedings of ICDM* (pp. 369–376).

Nijssen, S., & Fromont, E. (2007). Mining optimal decision trees from itemset lattices. *Proceedings of KDD* (pp. 530–539).

Mitchell, T. (1997). *Machine learning*. New York: McGraw-Hill.

Oliver, J. J., & Hand, D. J. (1995). On pruning and averaging decision trees. *Proceedings of ICML* (pp. 430–437).

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.