Sample Compression Bounds for Decision Trees

Mohak Shah MOHAK.SHAH@CRCHUL.ULAVAL.CA
CHUL Research Centre, Faculty of Medicine, Laval University, 2705 Laurier Blvd, Quebec, QC G1V 4G2 Canada

Abstract

We propose a formulation of the Decision Tree learning algorithm in the Compression settings and derive tight generalization error bounds. In particular, we propose Sample Compression and Occam's Razor bounds. We show how such bounds, unlike the VC dimension or Rademacher complexities based bounds, are more general and can also perform a margin-sparsity trade-off to obtain better classifiers. Potentially, these risk bounds can also guide the model selection process and replace traditional pruning strategies.

1. Introduction

Decision Trees are an important class of learning algorithms. One of the main advantages of these classifiers obviously is the ease of understandability and interpretability. Moreover, they have shown performance comparable to the state-of-the-art algorithms such as the Support Vector Machines on some domains (see, for example, (Shah et al., 2006)). However, the problem of over-fitting (and under-fitting) has been indeed central to the study and further investigations of these learning algorithms. Decision trees can be quite effective when a suitable model selection strategy is applied. This corresponds to selecting the best tree from the space of all possible decision trees. The more the number of nodes, the more specific the tree becomes and hence has a higher chance of over-fitting. On the other hand, a very restrictive tree in terms of the number of decision nodes might result in under-fitting. In order to deal with this problem, a general approach has been to construct a full tree and then prune the nodes so as to find optimal size. Various pruning strategies have been suggested to alleviate the problem of selecting the best tree size.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

On the learning theoretic front, a common approach has been to minimize a generalization error bound that expresses a trade-off between the training error, the size of the decision tree and some measure of the complexity of the hypothesis class. Various approaches aimed at proposing tight risk bounds for decision trees have been adopted. For instance, Vapnik (1982) and Anthony and Bartlett (1999) proposed bounds that depend on the margin of the linear threshold functions. This margin is basically a distance measure between the positive and negative example subsets in the training set. Golea et al. (1998) proposed a bound on decision trees in terms of the VC dimension of the class of node functions and the *effective number of leaves* in the tree.

Kearns and Mansour (1998) proposed a decision tree pruning algorithm based on the bound for subtrees of a given tree. Mansour and McAllester (2000) improved on this by proposing a *compositional* algorithm for constructing decision trees. Other approaches have also been proposed that takes into account, in addition to the factors mentioned above, the learning algorithm itself that is used to generate the tree. Examples of such approaches include self-bounding algorithms (Freund, 1998) and microchoice bounds (Langford & Blum, 1999). Bartlett and Mendelson (2002, section 4.1) used an alternate measure of the complexity of hypothesis class in the form of Rademacher complexities for deriving the bounds on decision trees.

In this paper, we visualize the decision trees with a Sample Compression viewpoint. We propose a sample compression based formulation of the learning algorithm and derive generalization error bounds that exploit the compression that the algorithm achieves over a given training set.

Pure Sample Compression bounds, in a sense, focus on obtaining as sparse decision trees[1] as possible. We, then, relax this bound to allow for a possible trade-off between sparsity and margin (the inverse of the message length) so that solutions can be obtained that do

[1] *Sparse* decision tree is one with very few nodes.

not result in under-fitting. Such bounds can then possibly lead to a forward learning algorithm for decision trees in conjunction with the risk bound to perform the model selection. Note that by forward learning algorithm, we mean that the algorithm can make a decision over the optimal size while learning itself. This can be done in a greedy manner by computing the bound at each stage and choosing the node that minimizes it and selecting a stopping criterion such that the algorithm stops growing the tree when the bound deteriorates. This is in contrast with the traditional tree-pruning strategies. Moreover, these bounds are independent of the (measures of) complexity of the hypothesis class unlike the traditional bounds based on VC dimension or Rademacher complexities.

The rest of the paper is organized as follows: Section 2 present a sample compression formulation of the decision trees. This (informal) formulation of the decision tree algorithm in the sample compression settings is then formalized in the following sections. Sections 3 and 4 give the generalization risk bounds for the decision trees in these compression settings. We state the precise details about these results and put them in context in the next section. Our main results appear in Corollary 2 (in conjunction with Equations 4 and 5), Theorem 3, and Theorem 5 (in conjunction with Equation 8). Finally, we conclude in Section 5.

2. Sample Compressed Decision Trees

We work in the sample compression settings. An algorithm, in this setting, can be called a sample compression algorithm iff it satisfies two requirements. First, there exists a Compression Function that identifies, from among the examples in the training set, a (preferably) small subset of examples known as the *Compression Set* that are used to represent the hypothesis along with some additional information. Second, there should exist a Reconstruction Function that can reconstruct the hypothesis making use of only the examples in the compression set and the additional information.

In this section we propose, informally, a representation of decision tree in the sample compression setting that allows us to reconstruct it from a subset of training examples and the corresponding additional information. This will subsequently enable us to derive a generalization error bound for a decision tree classifier in the compression setting. We will formalize these notions in the next section.

We depict each node of the decision tree in terms of a training example. That is, we use a training example to store the value of the threshold for the predicate of each node in the tree. Hence, learning algorithm for the decision tree can basically act as the Compression Function. In order to have a reconstruction function, however, we also need our algorithm to identify some additional information that can enable us to reconstruct the classifier. This can be done as follows.

In addition to the compression set[2], we use two strings to specify the additional information for the reconstruction of the hypothesis. The first string contains a (prefix-free) code to specify the parent of the current node. This will include k bits (for a tree with k nodes) with all the bits set to zero except the one corresponding to the parent.[3] [4] That is, the string will have all zeros except the ith bit if the example i in the compression set is the parent node. In order to specify the root node, we will have all the bits as zeros in this string. Moreover, we will need one more string with one bit which will be 0 if the current node is a left child of the parent node and 1 otherwise.

In Section 3 we derive a pure sample compression bound for the decision tree classifier in this setting. This bound is minimized for sparse classifiers with low training error. In Section 4 we derive compression bounds using the Occam's Razor principle. First, we show how we can avoid using the compression set altogether to represent the decision tree (a pure Occam's Razor appraoch). There, we represent the classifier solely in terms of the additional information (message strings as we explain later) to obtain Occam's Razor bounds. We propose bounds for the case when the attribute values are discrete in Section 4.1. In particular, this bound is useful when the attributes take on binary values as we will see later. The continuous case is dealt with in Section 4.2 which then leads to a new learning algorithm for decision trees using a coding scheme for the thresholds. There, we derive a bound that can effectively perform a margin-sparsity trade-off to obtain better generalization.

3. A Data-Compression Risk Bound

We consider binary classification problems where the input space $\mathcal{X}$ consists of an arbitrary subset of $\mathbb{R}^n$ and the output space $\mathcal{Y} = \{-1, +1\}$. An example $\mathbf{z} \stackrel{\text{def}}{=} (\mathbf{x}, y)$ is an input-output pair where $\mathbf{x} \in \mathcal{X}$

[2] It consists of attributes to specify the threshold for each node; we use a training example for each threshold value.

[3] Note that this can also be done using a more efficient scheme of using $log_2(k)$ bits but would have a less direct reconstruction scheme.

[4] The compression set needs to be ordered in order to specify such a string.

and $y \in \mathcal{Y}$. We are interested in learning algorithms that have the following property. Given a training set $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ of m examples, the classifier $A(S)$ returned by algorithm A is described entirely by two *complementary sources of information*: a subset $\mathbf{z_i}$ of S, called the *compression set*, and a *message string* σ which represents the additional information needed to obtain a classifier from the compression set $\mathbf{z_i}$.

Given a training set S, the compression set $\mathbf{z_i}$ is defined by a vector $\mathbf{i}$ of indices $\mathbf{i} \stackrel{\text{def}}{=} (i_1, i_2, \ldots, i_{|\mathbf{i}|})$ with $i_j \in \{1, \ldots, m\}\ \forall j$ and $i_1 < i_2 < \ldots < i_{|\mathbf{i}|}$ and where $|\mathbf{i}|$ denotes the number of indices present in $\mathbf{i}$. Hence, $\mathbf{z}_i$ denotes the ith example of S whereas $\mathbf{z_i}$ denotes the subset of examples of S that are pointed to by the vector of indices $\mathbf{i}$ defined above. We will use $\bar{\mathbf{i}}$ to denote the set of indices not present in $\mathbf{i}$. Hence, we have $S = \mathbf{z_i} \cup \mathbf{z_{\bar{i}}}$ for any vector $\mathbf{i} \in \mathcal{I}$ where $\mathcal{I}$ denotes the set of the 2^m possible realizations of $\mathbf{i}$.

The fact that any classifier returned by algorithm A is described by a compression set and a message string essentially implies that there exists a *Reconstruction Function* $\mathcal{R}$, associated with A, that outputs a classifier $\mathcal{R}(\sigma, \mathbf{z_i})$ when given an arbitrary compression set $\mathbf{z_i} \subseteq S$ and message string σ chosen from the set $\mathcal{M}(\mathbf{z_i})$ of all distinct messages that can be supplied to $\mathcal{R}$ with the compression set $\mathbf{z_i}$. It is only when such a $\mathcal{R}$ exists that the classifier returned by $A(S)$ is *always* identified by a compression set $\mathbf{z_i}$ and a message string σ.

We seek a tight risk bound for arbitrary reconstruction functions that holds uniformly for all compression sets and message strings. For this, we adopt the PAC setting where each example $\mathbf{z}$ is drawn according to a fixed, but unknown, probability distribution D on $\mathcal{X} \times \mathcal{Y}$. The true risk $R(f)$ of any classifier f is defined as the probability that it misclassifies an example drawn according to D:

$$R(f) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D}\left(f(\mathbf{x}) \neq y\right) = \mathbf{E}_{(\mathbf{x},y)\sim D} I(f(\mathbf{x}) \neq y)$$

where $I(a) = 1$ if predicate a is true and 0 otherwise. Given a training set $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ of m examples, the *empirical risk* $R_S(f)$ on S, of any classifier f, is defined according to:

$$R_S(f) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(f(\mathbf{x}_i) \neq y_i) \stackrel{\text{def}}{=} \mathbf{E}_{(\mathbf{x},y)\sim S} I(f(\mathbf{x}) \neq y)$$

Let $\mathbf{Z}^m$ denote the collection of m random variables whose instantiation gives a training sample $S = \mathbf{z}^m = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$. We denote $\Pr_{\mathbf{Z}^m\sim D^m}(\cdot)$ by $\mathbf{P}_{\mathbf{Z}^m}(\cdot)$. To obtain the tightest possible risk bound, we fully exploit the fact that the distribution of classification errors is a binomial.

Our starting point is the data-compression risk bound of Shah (2006, Chap. 8). Their bound utilizes the binomial tail inversion(Langford (2005), Blum and Langford (2003)) defined as follows: The *binomial tail inversion* $\overline{\text{Bin}}\left(\frac{\lambda}{m}, \delta\right)$ is defined as the largest risk value that a classifier can have while still having a probability of at least δ of observing at most λ errors out of m examples:

$$\overline{\text{Bin}}\left(\frac{\lambda}{m}, \delta\right) \stackrel{\text{def}}{=} \sup\left\{r : \text{Bin}\left(\frac{\lambda}{m}, r\right) \geq \delta\right\}$$

We now recall the compression bound of Shah (2006):

Theorem 1 *For any reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and message strings to classifiers, for any prior distribution $P_{\mathcal{I}}$ of vectors of indices, for any compression set-dependent distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}$, and for any $\delta \in (0, 1]$, we have:*

$$\mathbf{P}_{\mathbf{Z}^m}\Bigg\{\forall \mathbf{i} \in \mathcal{I}, \forall \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \overline{\text{Bin}}\Big(R_{\mathbf{Z_{\bar{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i})), P_{\mathcal{I}}(\mathbf{i}) P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta\Big)\Bigg\} \geq 1 - \delta$$

where, for any training set $\mathbf{z}^m$, $R_{\mathbf{z_{\bar{i}}}}(f)$ denotes the empirical risk of classifier f on the examples of $\mathbf{z}^m$ that do not belong to the compression set $\mathbf{z_i}$.

Theorem 1 applies to any compression set-dependent distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}$ satisfying:

$$\sum_{\sigma \in \mathcal{M}(\mathbf{z_i})} P_{\mathcal{M}(\mathbf{z_i})}(\sigma) \leq 1 \quad \forall \mathbf{z_i} \tag{1}$$

and any prior distribution $P_{\mathcal{I}}$ of vectors of indices satisfying $\sum_{\mathbf{i}\in\mathcal{I}} P_{\mathcal{I}}(\mathbf{i}) \leq 1$

The risk bound of Theorem 1 appears to be the tightest bound obtainable under this setting. Importantly, note that, once $P_{\mathcal{I}}$ and $P_{\mathcal{M}(\mathbf{z_i})}$ are specified, the risk bound of Theorem 1 for classifier $\mathcal{R}(\sigma, \mathbf{z_i})$ depends on its empirical risk *and* on the product $P_{\mathcal{I}}(\mathbf{i}) P_{\mathcal{M}(\mathbf{z_i})}(\sigma)$. However, $\ln\left(\frac{1}{P_{\mathcal{I}}(\mathbf{i}) P_{\mathcal{M}(\mathbf{z_i})}(\sigma)}\right)$ is just the amount of information needed to specify a classifier $\mathcal{R}(\sigma, \mathbf{z_i})$ once we are given a training set and the priors $P_{\mathcal{I}}$ and $P_{\mathcal{M}(\mathbf{z_i})}$. The $\ln(1/P_{\mathcal{I}}(\mathbf{i}))$ term is the information content of the vector of indices $\mathbf{i}$ that specifies the compression set and the $\ln(1/P_{\mathcal{M}(\mathbf{z_i})}(\sigma))$ term is the information content of the message string σ. A closer look reveals that these are effectively the measures of sparsity and margin respectively. Consequently the bound of Theorem 1 specifies quantitatively how much training errors learning algorithms should trade-off with the

amount of information needed to specify a classifier by $\mathbf{i}$ and σ.

Any bound expressed in terms of the binomial tail inversion can be turned into a more conventional and looser bound by inverting a standard approximation of the binomial tail such as those obtained from the inequalities of Chernoff and Hoeffding. Shah (2006) proposed the following approximation of the above bound:

Corollary 2 *For any reconstruction function $\mathcal{R}$ that maps arbitrary subsets of a training set and message strings to classifiers, for any prior distribution $P_{\mathcal{I}}$ of vectors of indices, for any compression set-dependent distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}$, and for any $\delta \in (0,1]$, we have:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall \mathbf{i} \in \mathcal{I}, \forall \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq 1 - \exp\left(\frac{-1}{m-d-\lambda}\left[\ln\binom{m-d}{\lambda} + \ln\left(\frac{1}{P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta}\right)\right]\right)\right\} \geq 1-\delta \quad (2)$$

and, consequently:

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall \mathbf{i} \in \mathcal{I}, \forall \sigma \in \mathcal{M}(\mathbf{Z_i})\colon R(\mathcal{R}(\sigma, \mathbf{Z_i})) \leq \frac{1}{m-d-\lambda}\left[\ln\binom{m-d}{\lambda} + \ln\left(\frac{1}{P_{\mathcal{I}}(\mathbf{i})P_{\mathcal{M}(\mathbf{z_i})}(\sigma)\delta}\right)\right]\right\} \geq 1-\delta \quad (3)$$

where $d \stackrel{\text{def}}{=} |\mathbf{i}|$ is the sample compression set size of classifier $\mathcal{R}(\sigma, \mathbf{Z_i})$ and $\lambda \stackrel{\text{def}}{=} |\bar{\mathbf{i}}|R_{\mathbf{z}_{\bar{\mathbf{i}}}}(\mathcal{R}(\sigma, \mathbf{Z_i}))$ is the number of training errors that this classifier makes on the examples that are not in the compression set.

Corollary 2 suggests that the risk bound of classifier $\mathcal{R}(\sigma, \mathbf{Z_i})$ is small when its compression set size d and its number of training errors λ are both much smaller than the number of training examples m.

Let us now identify the distributions for the possible compression sets and the associated messages for the thresholded decision trees. Recall that in order to specify a decision tree classifier under the compression scheme, the compression set consists of one example per predicate. For each node we have one attribute and a corresponding threshold value determined by the numerical value that this attribute takes on the training example. Note that more than one attribute can be specified by one training example without affecting the size of compression set.

In addition to the compression set, we use two strings to specify the additional information for the reconstruction of the hypothesis. The first string contains a (prefix-free) code to specify the parent of the current node. This string consists of k bits (for a tree with k nodes) with all the bits zeros except the one corresponding to the parent. Note that for the root node, the string has all the bits as 0's. Moreover, we use one more bit which is 0 if the current node is a left child of the parent node and 1 otherwise. As noted before, a more efficient scheme of coding with $log_2(k)$ bits is possible but lacks straightforward reconstruction.

For the case of thresholded decision trees, let the subset of attributes that specifies our compression set $\mathbf{i}$ be $\mathbf{k}$. Note that $|\mathbf{i}|$ is not necessarily equal to $|\mathbf{k}|$ even though we use one example per threshold. In general, $|\mathbf{i}| \leq |\mathbf{k}|$ with equality holding when all the examples denoting the thresholds in the set $\mathbf{k}$ are different. However, the examples in $\mathbf{i}$ might need to be indexed in accordance with the coding scheme of $\mathbf{k}$. We, hence, use the following distribution of messages:

$$P_{\mathcal{M}(\mathbf{z_i})}(\sigma) = \frac{1}{n^{|\mathbf{k}|}} \cdot \frac{1}{(|\mathbf{k}|+1)^{|\mathbf{k}|}} \cdot \frac{1}{2^{|\mathbf{k}|}} \quad \forall \sigma \quad (4)$$

The rationale for this distribution of messages is the following. We assign equal probability to each of the possible $|\mathbf{k}|$ attributes (and hence thresholds) that can be selected from n attributes. This yields the first factor in Equation 4. In order to associate each node with its parent node, we need one of the $|\mathbf{k}|$ strings with the bit corresponding to the parent equalling 1 and all other bits being 0. In addition, we also need to consider the string for the root node with all zeros. We assign equal probabilities for each node to be able to take any of these $|\mathbf{k}|$ strings[5]. The second factor in the above equation denotes this probability. Finally, each selected node, can be either a left or a right child of its parent (except for the root node). We assign equal probability over this hence obtaining the third factor of Equation 4.

As for the prior $P_{\mathcal{I}}(\mathbf{i})$ over the compression sets, there are few obvious choices. One of these, that we use in our calculations further too, consist in:

$$P_{\mathcal{I}}(\mathbf{i}) = \frac{1}{\binom{m}{|\mathbf{i}|}} p(|\mathbf{i}|) \quad (5)$$

This prior signifies that the final classifier, constructed

[5]Note that this also bounds the number of possible binary trees of size $|\mathbf{k}|$.

from the group of examples specified by $\mathbf{i}$, should depend only on the number $|\mathbf{i}|$ of examples in this group. Moreover, we choose a p that decreases as we increase $|\mathbf{i}|$ if we have reasons to believe that the number of nodes of the final classifier does not grow linearly with m. One possibility, that we use in this case, exists in using $p(|\mathbf{k}|) = \frac{6}{\pi^2}(|\mathbf{k}|+1)^{-2}$.

Equations 4 and 5 along with the Corollary 2 give a sample compression bound for the decision tree classifier. Note that, being a pure sample compression bound, Corollary 2 in conjunction with Equation 4 is minimized when a classifier with a sparse solution giving a low empirical risk is found.

4. Occam's Razor Bound for Decision Trees

4.1. The Discrete Case

Let us note how the bound in Corollary 2 can yield a pure Occam's razor bound when the compression set vanishes, in the following manner. More precisely, we restrict ourselves to the case when $P_{\mathcal{I}}(\mathbf{i}) = 1$ when $|\mathbf{i}| = d = 0$. That is, all the information now is held in the distribution of messages $P_{\mathcal{M}(\mathbf{z_i})}$. Hence, the classifier is solely represented now with the help of messages and does not depend on the size of compression set (which is zero by definition of $P_{\mathcal{I}}(\mathbf{i})$). Therefore, in this case the only requirement of our reconstruction function is the information from the messages. We denote this new classifier with the above mentioned restriction by $\mathcal{R}(\sigma)$.

Let us now see the case where we can incorporate the threshold information for each node in the message distribution. First, let us see the case when every node of the decision tree can take one of the values from a set of discrete values $\mathcal{T}$. Hence, each node can have one of the possible $|\mathcal{T}|$ values for the threshold and let us assume an equal probability for each of the possible values for each node in the decision tree. Let us denote this set of threshold values by $\mathbf{k}$, so that the number of nodes in the decision tree equals $|\mathbf{k}|$. Hence Corollary 2 yields:

Theorem 3 *For any reconstruction function $\mathcal{R}$ that maps subsets of message strings to classifiers, for any distribution of messages $P_{\mathcal{M}}$, and for any $\delta \in (0,1]$, we have:*

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall\sigma\in\mathcal{M}\colon R(\mathcal{R}(\sigma)) \le 1-\exp\left(\frac{-1}{m-\lambda}\left[\ln\binom{m}{\lambda} + \ln\left(\frac{1}{P_{\mathcal{M}}(\sigma)\delta}\right)\right]\right)\right\} \ge 1-\delta \quad (6)$$

and, consequently:

$$\mathbf{P}_{\mathbf{Z}^m}\left\{\forall\sigma\in\mathcal{M}\colon R(\mathcal{R}(\sigma)) \le \frac{1}{m-\lambda}\left[\ln\binom{m}{\lambda} + \ln\left(\frac{1}{P_{\mathcal{M}}(\sigma)\delta}\right)\right]\right\} \ge 1-\delta \quad (7)$$

where $\lambda \stackrel{\text{def}}{=} mR_{\mathbf{z}_m}(\mathcal{R}(\sigma))$ is the number of training errors that this classifier makes, and where:

$$P_{\mathcal{M}}(\sigma) = \frac{1}{n^{|\mathbf{k}|}}\cdot\frac{1}{(|\mathbf{k}|+1)^{|\mathbf{k}|}}\cdot\frac{1}{2^{|\mathbf{k}|}}\cdot\frac{1}{|\mathcal{T}|^{|\mathbf{k}|}} \quad \forall\sigma$$

Let us now consider the case when the attributes in the data take on binary values. That is, each $x_i \in \{0,1\}$. In this case, since $\mathcal{T} = \{0,1\}$, the message distribution above becomes:

$$P_{\mathcal{M}}(\sigma) = \frac{1}{n^{|\mathbf{k}|}}\cdot\frac{1}{(|\mathbf{k}|+1)^{|\mathbf{k}|}}\cdot\frac{1}{2^{2|\mathbf{k}|}} \quad \forall\sigma$$

Hence, it can be easily seen that the bound of Theorem 3 is especially tight when the data is binary valued.

Before going further, we take a look at how the bounds of Corollary 2 and Theorem 3 translate in practice. In Table 1, we present the empirical result of decision tree algorithm for several UCI datasets and the associated risk bounds. The "Ex" and the "Att" columns refer to the number of examples in the dataset and the number of attributes respectively. "Size" refer to the number of nodes in the final Decision tree. The "Err" column refers to the *stratified* ten-fold cross validation risk of the decision tree learning algorithm.[6] We use the Weka implementation of decision tree algorithm (Witten & Frank, 2005). The "Bound" column gives the generalization error bound obtained by computing the r.h.s. of Corollary 2 (with $\delta = 0.05$) averaged over the folds. For the breastw (discrete valued), Vote and Chess[7] datasets (both binary valued), the quantity in parentheses gives the bound obtained by Theorem 3. As can be seen the bound is esp. tight in about five of the nine cases. Moreover, it is non-trivial in other cases as well. Also, we see that the Occam's Razor bound is significantly tighter in case of binary valued data as seen in the case of Vote and Chess datasets.

Inspired by this, we explore a version of Occam's razor bound that can perform a non-trivial margin-sparsity

[6] We use default parameters since our main aim is to illustrate the bounds' practicality here.

[7] Two non-binary attributes were removed so as to apply the Occam razor bound too.

Table 1. Compression Bounds on UCI datasets

Dataset	Ex	Att	Size	Err	Bound
Vote	435	17	11	0.04	0.36(0.29)
Pima	768	9	39	0.26	0.78
Breastw	699	10	27	0.05	0.49(0.61)
Ionosphere	351	35	35	0.09	0.81
Mushroom	8124	23	30	0	0.05
Credit-a	690	15	42	0.14	0.72
Chess	3196	34	59	0.01	0.25(0.18)
Heart-s	270	14	35	0.23	0.9
Thyroid	3772	29	61	0.01	0.25

trade-off and hence in effect gives *a new algorithm for learning decision trees based on a coding scheme for the continuous valued thresholds.*

4.2. The Continuous Case

In order to take into account the continuous valued attributes, we make the following assumption about the input space. We still consider the input space $\mathcal{X}$ consisting of all the n-dimensional vectors $\mathbf{x} = (x_1, \ldots, x_n)$ but now each real valued component $x_i \in [A_i, B_i]$ for $i = 1, \ldots, n$.

We start with the Occam's razor bound of Langford (2005) which is a tighter version of the bound proposed by Blumer et al. (1987). It is also more general in the sense that it applies to any prior distribution P over any countable class of classifiers.

Theorem 4 (Langford (2005)) *For any prior distribution P over any countable class $\mathcal{F}$ of classifiers, and for any $\delta \in (0, 1]$, we have:*

$$\Pr_{S\sim D^m}\left\{\forall f \in \mathcal{F}\colon R(f) \leq \overline{\mathrm{Bin}}\big(R_S(f), P(f)\delta\big)\right\} \geq 1-\delta$$

The proof directly follows from a straightforward union bound argument and from the fact that $\sum_{f\in\mathcal{F}} P(f) \leq 1$.

In order now to obtain a bound for decision trees, let us start by choosing a suitable prior P for this class.

As we saw before, a decision tree classifier can be represented by the following quantities. First a set of indices that forms the nodes of the tree, represented by $\mathbf{k}$. Second, for each node, we need one bit to signify whether a node is a left or right child of the parent node. Let us denote by $\mathbf{d}$ the vector containing this information for all the nodes. Next, with each node is associated a string of size $|\mathbf{k}|$ to locate the parent node. This coding scheme has a string of all zeros denoting the root node of the tree. Finally, let us consider the continuous parameter, i.e. the threshold of each node, the vector of which is denoted by $\mathbf{t}$.

Let us denote by $P(\mathbf{k}, \mathbf{d}, \sigma)$ the prior probability assigned to the decision tree $D^{\mathbf{k}}_{\sigma\mathbf{d}}$ described by $(\mathbf{k}, \mathbf{d}, \sigma)$. We choose a prior of the following form:

$$P(\mathbf{k}, \mathbf{d}, \sigma) = \frac{1}{n^{|\mathbf{k}|}} \frac{1}{(|\mathbf{k}|+1)^{|\mathbf{k}|}} \frac{1}{2^{|\mathbf{k}|}} g_{\mathbf{k},\mathbf{d}}(\sigma)$$

where $g_{\mathbf{k},\mathbf{d}}(\sigma)$ is the prior probability assigned to string σ given that we have chosen $\mathbf{k}$ and $\mathbf{d}$. Let $\mathcal{M}(\mathbf{k}, \mathbf{d})$ be the set of all message strings that we can use given that we have chosen $\mathbf{k}$ and $\mathbf{d}$. If $\mathcal{I}$ denotes the set of all 2^n possible attribute index vectors and $\mathcal{D}_{\mathbf{k}}$ denotes the set of all $2^{|\mathbf{k}|}$ binary vectors $\mathbf{d}$ of dimension $|\mathbf{k}|$, we have that $\sum_{\mathbf{k}\in\mathcal{I}} \sum_{\mathbf{d}\in\mathcal{D}_{\mathbf{k}}} \sum_{\sigma\in\mathcal{M}(\mathbf{k},\mathbf{d})} P(\mathbf{k}, \mathbf{d}, \sigma) \leq 1$ whenever $\sum_{\sigma\in\mathcal{M}(\mathbf{k},\mathbf{d})} g_{\mathbf{k},\mathbf{d}}(\sigma) \leq 1 \ \forall \mathbf{k}, \mathbf{d}$.

The reasons motivating this choice for the prior are the following. The first factor basically gives a uniform distribution over choosing each of the $|\mathbf{k}|$ nodes from n attributes. The second factor of $P(\mathbf{k}, \mathbf{d}, \sigma)$ gives equal prior probabilities over associating each string from the set of $|\mathbf{k}|$ strings identifying the parent to each node of the tree. The third factor gives equal prior probabilities for each of the two possible values for the bit $d_j \in \mathbf{d}$ denoting whether the current node is a left or a right child.

To specify the distribution of strings $g_{\mathbf{k},\mathbf{d}}(\sigma)$, we work on the lines of Shah (2006). Let $[A, B]$ is some predefined interval in which we are permitted to choose a threshold t. Moreover we need to choose t from an interval $[a, b] \subset [A, B]$, which is an interval of "equally good" threshold values.[8] We consider the problem of coding this threshold value. The following diadic coding scheme can be used in order to identify the thresholds that belong to this interval:

$$\Lambda_l \stackrel{\text{def}}{=} \left\{\left[1 - \frac{2j-1}{2^{l+1}}\right] A + \frac{2j-1}{2^{l+1}} B\right\}_{j=1}^{2^l}$$

where Λ_l is the set of threshold values and l is the maximum number of bits used for the code.

The above coding scheme is obtained as follows: Let l be the number of bits used for the code. We adopt the following convention for coding the bits in the l-bit coding string. A code of $l = 0$ bits specifies the threshold value $(A + B)/2$, the mid-interval value. A code of $l = 1$ bit specifies the mid-values of the two half-intervals. That is, a code of $l = 1$ bit either specifies the value $(3A + B)/4$ or the value $(A + 3B)/4$,

[8] By a "good" threshold value, we mean a threshold value that optimize the node selection criterion such as information gain, or minimize the risk bound.

when the bit is 0 or 1 respectively. On similar lines, a code of $l = 2$ specifies one of the following values: $(7A + B)/8, (5A + 3B)/8, (3A + 5B)/8, (A + 7B)/8$. Each successive bit halves the interval. Hence, a code of l bits specifies one value among the set Λ_l of threshold values as shown in the above coding scheme.

The above scheme enables us to find all the threshold values that lie in the interval $[a, b] \subset [A, B]$ and the corresponding number of bits l that we need to use to code these threshold values. We select the smallest number l of bits such that there exists a threshold value in Λ_l that falls in the interval $[a, b]$. This yields an upper bound on the number of bits required to obtain a threshold value falling in the interval $[a, b]$. We will need at most $\lfloor \log_2((B - A)/(b - a)) \rfloor$ bits to do this.

Now, in order to specify the threshold for each node, we need to specify two quantities. First, the number l of bits and second, an l-bit string s, that together identify one of the threshold values in Λ_l. Moreover, we also need to specify an interval $[A, B]$ of permitted values for the threshold t.

We proceed along the similar lines to achieve this and see how this can eventually be done in practice. We choose the following scheme.

We need to identify an interval $[A^*, B^*]$ that can be deduced from the nature of the attribute independent of the values that this attribute in question takes on the examples in training set. One possibility typically exists in using the smallest and the largest value that the attribute in question can have. An attribute definition can easily yield this information. We then find the mid-value as $C^* = (A^* + B^*)/2$. Also we compute the smallest and the largest values that the attribute takes on the training set. We denote these values by A' and B' respectively.

Now, we can find the largest integer κ such that $2^{-\kappa}(C^* - A^*) \geq (C^* - A')$. The next step is to choose A such that $C^* - A = 2^{-\kappa}(C^* - A^*)$ for that value of κ. Similarly, we can find the largest integer κ' such that $2^{-\kappa'}(B^* - C^*) \geq (B' - C^*)$. This enables us to choose B such that $B - C^* = 2^{-\kappa'}(B^* - C^*)$ for that value of κ'. After choosing, in this way, $[A_i, B_i]$ for each attribute i a first run of the learning algorithm can be performed. Then, the algorithm can be rerun by halving again each interval $[A_i, B_i]$ and repeat until the risk bound, proposed in Theorem 5, of the classifier becomes very large.

As a result, the message string σ that we use for any choice of $\mathbf{k}$ consists of that pair of numbers κ_i and κ'_i that we have just defined and the pair (l_i, s_i) of numbers needed to identify the threshold for each attribute (and hence node) $i \in \mathbf{k}$. The risk bound does not depend on how we actually code σ but only on the a priori probabilities we assign to each possible realization of σ. We choose the following distribution:

$$g_{\mathbf{k},\mathbf{d}}(\sigma) \stackrel{\text{def}}{=} g_{\mathbf{k},\mathbf{d}}(\kappa_1, \kappa'_1, l_1, s_1, \ldots, \kappa_{|\mathbf{k}|}, \kappa'_{|\mathbf{k}|}, l_{|\mathbf{k}|}, s_{|\mathbf{k}|}) = \prod_{i \in \mathbf{k}} \zeta(\kappa_i)\zeta(\kappa'_i)\zeta(l_i) \cdot 2^{-l_i} \quad (8)$$

$$\text{where} : \zeta(a) \stackrel{\text{def}}{=} \frac{6}{\pi^2}(a+1)^{-2} \quad \forall a \in \mathbb{N}$$

The sum over all the possible realizations of σ gives 1 since $\sum_{i=1}^{\infty} i^{-2} = \pi^2/6$. Note that by giving equal a priori probability to each of the 2^{l_i} strings s_i of length l_i, we give no preference to any threshold value in Λ_{l_i} once we have chosen an interval $[A_i, B_i]$ that we believe is appropriate.

Alternatively, for homogeneous systems where each attribute has the same definition, we could use the same interval value $[A, B]$ for each attribute. In that case, $g_{\mathbf{k},\mathbf{d}}(\sigma)$ would be defined with only one κ and one κ' instead $|\mathbf{k}|$ pairs of parameters κ_i, κ'_i.

The distribution ζ that we have chosen for each string length l_i has the advantage of decreasing slowly so that the risk bound does not deteriorate too rapidly as l_i increases. Other choices are clearly possible.

This choice of prior enables us to give the following bound for the decision tree learner:

Theorem 5 *Given all our previous definitions and for any $\delta \in (0, 1]$, we have:*

$$\Pr_{S \sim D^m}\left\{\forall \mathbf{k}, \mathbf{d}, \sigma\colon R(D^{\mathbf{k}}_{\sigma\mathbf{d}}) \leq \overline{\text{Bin}}\left(R_S(D^{\mathbf{k}}_{\sigma\mathbf{d}}), \frac{g_{\mathbf{k},\mathbf{d}}(\sigma)\delta}{n^{|\mathbf{k}|}(|\mathbf{k}|+1)^{|\mathbf{k}|}2^{|\mathbf{k}|}}\right)\right\} \geq 1 - \delta$$

where $g_{\mathbf{k},\mathbf{d}}(\sigma)$ is given by Equation 8.

Finally, we emphasize that the risk bound of Theorem 5, used in conjunction with the distribution of messages given by $g_{\mathbf{k},\mathbf{d}}(\sigma)$, provides a guide for choosing the appropriate tradeoff between the message length (via $g_{\mathbf{k},\mathbf{d}}(\sigma)$) and $|\mathbf{k}|$. Note that the risk bound for a decision tree with a decision surface having a small coding string (small l_is) may be smaller than the risk bound of a sparse tree having a large coding string (larger l_is).

5. Conclusion

In this paper, we presented a Sample Compression formulation of decision tree learning algorithm and de-

rived generalization risk bounds in the compression settings. In particular, we derived a pure Sample Compression bound and Occam's razor bounds for the discrete and continuous valued data. The Compression bound is minimized when a classifier with the smallest number of nodes making a small training error is found. On the other hand, the Occam's Razor bound of Theorem 3 is minimized when a classifier that utilizes the smallest message string with low training error is found. Moreover we noted that the bound of Theorem 3 is tight especially for decision tree classifiers built on binary valued data.

In contrast to the bounds of Corollary 2 in conjunction with Equation 4 and Theorem 3, the bound of Theorem 5 spreads the reconstruction information for the classifier more evenly to perform a non-trivial trade-off between the size of the tree (sparsity) and the message string (margin) to find better classifiers. In effect, this also gives *an alternate learning algorithm for decision trees.*

These bounds not only provide tight guarantees in terms of data-compression but can also lead to a forward algorithm for selecting the optimal size of the decision-tree. That is, instead of building a full tree and then pruning it, an alternate approach can be to compute the risk bound at every step of adding an additional node to the tree (traditional greedy approach combined with model selection using the risk bound) and adopting a stopping criterion when the risk bound starts deteriorating. This is a step forward in obtaining theoretically motivated and justifiable algorithms that can learn by bound minimization. Some successful examples of such algorithms have recently appeared based on Sample Compression and related approaches (see e.g. (Shah, 2006)). Moreover, the bounds are independent of the measures of hypothesis class complexity unlike VC dimension bounds or those based on Rademacher complexities as discussed in Section 1.

Acknowledgements

The author would like to thank the anonymous reviewers and the ICML senior program committee for their comments and suggestions that helped improve the paper significantly.

References

Anthony, M., & Bartlett, P. (1999). *Neural network learning: Theoretical foundations.* Cambridge: Cambridge University Press.

Bartlett, P., & Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, *3*, 463–482.

Blum, A., & Langford, J. (2003). PAC-MDL bounds. *Proceedings of 16th Annual Conference on Learning Theory, COLT 2003,* Washington, DC, August 2003 (pp. 344–357). Springer, Berlin.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1987). Occam's razor. *Information Processing Letters*, *24*, 377–380.

Freund, Y. (1998). Self bounding learning algorithms. *COLT: Proceedings of the Workshop on Computational Learning Theory* (pp. 247–258). Morgan Kaufmann Publishers.

Golea, M., Bartlett, P., Lee, W. S., & Mason, L. (1998). Generalization in decision trees and DNF: Does size matter? *Advances in Neural Information Processing Systems.* The MIT Press.

Kearns, M., & Mansour, Y. (1998). A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. *Proc. 15th International Conf. on Machine Learning* (pp. 269–277). Morgan Kaufmann, San Francisco, CA.

Langford, J. (2005). Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, *3*, 273–306.

Langford, J., & Blum, A. (1999). Microchoice bounds and self bounding learning algorithms. *Computational Learing Theory* (pp. 209–214).

Mansour, Y., & McAllester, D. (2000). Generalization bounds for decision trees. *Proc. 13th Annu. Conference on Comput. Learning Theory* (pp. 69–80). Morgan Kaufmann, San Francisco.

Shah, M. (2006). *Sample compression, margins and generalization: Extensions to the set covering machine.* Doctoral dissertation, SITE, University of Ottawa, Ottawa, Canada.

Shah, M., Sokolova, M., & Szpakowicz, S. (2006). Process-specific information for learning e-negotiation outcomes. *Fundamenta Informaticae*, *74*, 351–373.

Vapnik, V. (1982). *Estimation of dependences based on empirical data.* New York: Springer-Verlag.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques, 2nd ed.* San Francisco: Morgan Kaufmann.