VC-Dimension of Univariate Decision Trees

Olcay Taner Yıldız

***Abstract*—In this paper, we give and prove lower bounds of the VC-dimension of the univariate decision tree hypothesis class. The VC-dimension of the univariate decision tree depends on the VC-dimension values of its subtrees and the number of inputs. Via a search algorithm that calculates the VC-dimension of univariate decision trees exhaustively, we show that our VC-dimension bounds are tight for simple trees. To verify that the VC-dimension bounds are useful, we also use them to get VC-generalization bounds for complexity control using structural risk minimization in decision trees, i.e., pruning. Our simulation results show that SRM-pruning using the VC-dimension bounds finds trees that are more accurate as those pruned using cross-validation.**

***Index Terms*—Learning, Machine Learning, Supervised Learning, Computation Theory, VC-Dimension, Decision Trees**

I. Introduction

In pattern recognition the knowledge is extracted as patterns from a training sample for future prediction. Most pattern recognition algorithms such as neural networks [1] or support vector machines [2] make accurate predictions but are not interpretable, on the other hand decision trees are simple, can learn disjunctive expressions and therefore are easily comprehensible. Whatever the learning algorithm is, the main goal of the learner is to extract the optimal model (the model with least generalization error) from a training set. In the penalization approaches, the usual idea is to derive an estimate of the generalization error in terms of the training error and the complexity of the model.

In the statistical learning theory [2], Vapnik-Chervonenkis (VC) dimension is a measure of complexity defined for any type of classification algorithm. Suppose that we have a class of functions $\{f(x,\alpha)\}$ indexed by the parameter vector α. VC dimension of $\{f(x,\alpha)\}$ is defined to be the largest number of points that can be shattered by members of $\{f(x,\alpha)\}$. A set of data points is *shattered* by $\{f(x,\alpha)\}$ if for all assignments of class labels to those points, one can find a member of $\{f(x,\alpha)\}$ which makes no errors when evaluating that set of data points. For example, in two dimensions, we can separate three points with a line, but we can not separate four points (if the assignments of class labels are done like in the famous XOR problem). Therefore, the VC dimension of the linear estimator class in two dimensions is 3. In general, the VC dimension of the linear estimator class in d dimensions is $d+1$ which is also the number of free parameters.

Structural risk minimization (SRM) [2] uses the VC dimension of the estimators to select the best model by choosing the model with the smallest upper bound for the generalization error. In SRM, the possible models are ordered according to their complexity

$$M_0 \subset M_1 \subset M_2 \subset \dots \tag{1}$$

For example, if the problem is selecting the best degree of a polynomial function, M_0 will be the polynomial with degree 0, M_1 will be the polynomial with degree 1, etc. For each model, the upper bound for its generalization error is calculated.

- For binary classification, the upper bound for the generalization error is

$$E_g \leq E_t + \frac{\epsilon}{2}\left(1+\sqrt{1+\frac{4E_t}{\epsilon}}\right) \tag{2}$$

- For regression, the upper bound for the generalization error is

$$E_g \leq \frac{E_t}{1-c\sqrt{\epsilon}} \tag{3}$$

and ϵ is given by the formula

$$\epsilon = a_1 \frac{V[\log(a_2 S/V)+1]-\log(\nu)}{S} \tag{4}$$

where V represents the VC dimension of the model, ν represents the confidence level, S represents the sample size, and E_t represents the training error. These bounds hold simultaneously for all members $\{f(x,\alpha)\}$, and are taken from [3] (pages 116-118). They recommend to use $\nu = \frac{1}{\sqrt{S}}$ for large sample sizes. For regression it is recommended to use $a_1 = 1$ and $a_2 = 1$, and for classification $a_1 = 4$ and $a_2 = 2$ corresponds to the worst-case scenarios.

Obtaining the VC-dimension of a classifier is necessary for complexity control in SRM. Unfortunately, it is not possible to obtain an accurate estimate of the VC-dimension in most cases. To avoid this problem, a set of experiments on artificial sets are done and based on the frequency of the errors on these sets, a best fit for theoretical formula is calculated. Shao *et al.* [4] used optimized experimental design to improve the VC-dimension stimates. The algorithm starts with the uniform experiment design defined in Vapnik *et al.* [5] and by making pairwise exchanges between design points, the optimized design is obtained. The mean square error is used as a criterion to identify good and bad design points.

In this work, we use decision trees as our hypothesis class. Decision trees are tree-based structures where (i) each internal node implements a decision function, $f_m(\boldsymbol{x})$, (ii) each branch of an internal node corresponds to one outcome of the decision, and (iii) each leaf corresponds to a class. In a univariate decision tree [6], the decision at internal node m uses only one attribute, i.e., one dimension of $\boldsymbol{x}$, x_j. If that attribute is discrete, there will be L children (branches) of each internal node corresponding to the L different outcomes of the decision. ID3 is one of the best known univariate decision tree algorithm with discrete features [7]. Survey of work on

Olcay Taner Yıldız is with the Department of Computer Engineering, Işık University, 34398, İstanbul Turkey.

constructing and simplifying decision trees can be found in [8], [9]. There are also surveys comparing different decision tree methods with other classification algorithms such as [10], [11].

Determining the optimal complexity of a decision tree is important. With complex decision trees, we do not learn a general rule but memorize the particular training data which gives large error on unseen test data. With too simple trees, even the training data may not be learned well and the error will be large on both training and test data.

As far as our knowledge, there is no explicit formula for the VC-dimension of a decision tree. Although there are certain results for the VC-dimension of decision trees such as:

- It is known that the VC dimension of a binary decision tree with N nodes and dimension d is between $\Omega(N)$ and $\mathcal{O}(N \log d)$ [12].
- It is shown that the VC dimension of the set of all boolean functions on d boolean variables defined by decision trees of rank at most r is $\sum_{i=0}^{r} \binom{d}{i}$ [13].
- Maimon and Rokach [14] give explicit lower and upper bounds of VC-dimension of oblivious decision trees. Oblivious decision trees are decision trees, in which all nodes at the same level test the same attribute.

These bounds are either structure independent, that is, they give the same bound for all decision trees with N nodes; or the bounds are for particular type of univariate trees.

In this work, we first focus on the easiest case of univariate trees with binary features and we prove that the VC-dimension of a univariate decision tree with binary features depends on the number of binary features and the tree structure. Note that we are discussing the VC dimension of hypotheses classes defined as families of decision trees that share a tree structure, differ only in the variables being tested in the internal nodes and class labels assigned to the leaves. Our approach is the following: First, for three basic tree structures, we give and prove a lower bound of the VC-dimension. Second, we give and prove a general lower bound of the VC-dimension of the binary decision tree. Third, based on those theorems, we give an algorithm to find a structure dependent lower bound of the VC-dimension of a binary decision tree with binary features. Fourth, we use the exhaustive search algorithm to calculate the exact VC-dimension of simple trees and compare our bounds with the exact VC-dimension values.

As a next step, we generalize our work to the discrete univariate decision tree hypothesis class, where a decision node can have L children depending on the number of values of the selected discrete feature. We show that the VC-dimension of L-ary decision tree is greater than or equal to the VC-dimension of its subtrees. Based on this result, we give an algorithm to find a lower bound of the VC-dimension of a L-ary decision tree.

As a last step, we generalize our work to include continuous data, that is continuous univariate decision tree hypothesis class, where a decision node always has two children. We again give an algorithm to find a lower bound of the VC-dimension of a univariate decision tree for continuous data sets. We use these VC-dimension bounds in pruning via SRM and when compared with cross-validation pruning, we see that pruning based on SRM using our VC-dimension bounds work well and find trees that are as accurate as cross-validation pruning.

In the earlier version of this work [15], we proved lower bounds of the VC-dimension of univariate decision trees with binary features; this present paper revisits the binary feature case, extends the proofs to include both L-ary and continuous univariate decision trees, and makes a more thorough comparison with cross-validation pruning.

This paper is organized as follows: In Section II, we give and prove the lower bounds of the VC-dimension of the univariate decision trees with binary features. We generalize our work to L-ary decision trees in Section III, and continuous univariate trees in Section IV. We give experimental results in Section V and conclude in Section VI.

II. VC-dimension of the Univariate Decision Trees with Binary Features

We consider the well-known supervised learning setting where the decision tree algorithm uses a sample of m labeled points $S = (\mathbf{X}, \mathbf{Y}) = ((\mathbf{x^{(1)}}, y^{(1)}), \ldots, (\mathbf{x^{(m)}}, y^{(m)}) \in (\mathcal{X} \times \mathcal{Y})^m$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ the label set, which is $\{0, 1\}$. The input space $\mathcal{X}$ is a vectorial space of dimension d, the number of features, where each feature can take values from $\{0, 1\}$.

From this point on, we refer only internal nodes of the decision tree as node(s). Note again that we are searching the VC dimension of decision tree hypotheses class that share a tree structure, differ only in variables being tested in the internal nodes and class labels assigned to the leaves. A set of instances S is shattered by hypothesis space H if and only if for every dichotomy of S there exists some hypothesis in H consistent with this dichotomy. Given a sample S with m examples, there are 2^m possible dichotomies. Each example can be labeled with 0 or 1, which gives us 2^m. From these dichotomies, two of them will be all zeros or all ones, which can be classified by any decision tree. A dichotomy and its reverse dichotomy, where all class labelings are flipped, can be classified by the same decision tree hypothesis h. This is because decision trees treat class 0 and class 1 symmetrically, that is, decision tree algorithms will construct identical decision trees if the class 0 and class 1 are interchanged. Hence, the number of dichotomies that must be checked for a sample with m examples is the half of $2^m - 2$, which is $2^{m-1} - 1$.

Theorem 1: The VC-dimension of a single decision node univariate decision tree that classifies d dimensional binary data is $\lfloor \log_2(d+1) \rfloor + 1$.

Proof: To show the VC-dimension of the single decision node univariate decision tree is at least m, we need to find such a sample S of size m that, for each possible class labelings of these m points, there is an instantiation h of our single node decision tree hypothesis class H that classifies it correctly. Let $\mathbf{C_m}$ be the matrix of size $(2^{m-1} - 1) \times m$ where the rows of the matrix represent the dichotomies that must be checked for a sample of m data points (See the discussion above). For m = 4, the matrix $\mathbf{C_4}$ is

$$\mathbf{X} = \begin{array}{c} \\ \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \mathbf{x}^{(3)} \\ \mathbf{x}^{(4)} \end{array} \begin{array}{ccccccc} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

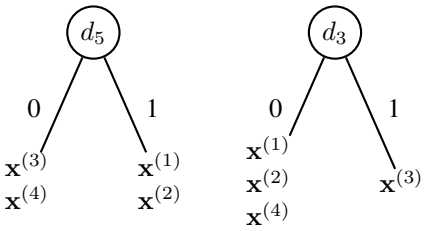


Fig. 1. Example for Theorem 1 with $d = 7$ and $m = 4$. If the class labeling of S is {1, 1, 0, 0} we select feature 5 (left decision tree). If the class labeling of S is {0, 0, 1, 0} we select feature 3 (right decision tree).

$$\mathbf{C_4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

We construct the sample S such that

$$\mathbf{X} = \mathbf{C_m^T}$$

where $\mathbf{C_m^T}$ represents transpose of the matrix $\mathbf{C_m}$. In this way, each feature d_i corresponds to a dichotomy that must be checked for a sample of m points, implying a one-to-one mapping between dichotomies and features. So for each possible dichotomy, we will choose the decision tree hypothesis h which has the corresponding feature as the split feature (See Figure 1 for an example). If we set the number of features to $2^{m-1} - 1$:

$$\begin{aligned} d &= 2^{m-1} - 1 \\ d+1 &= 2^{m-1} \\ \log_2(d+1) &= m - 1 \\ m &= \log_2(d+1) + 1 \end{aligned}$$

To show the VC-dimension of the single decision node univariate decision tree is at most $\lfloor \log_2(d+1) \rfloor + 1$, we go in reverse direction. If the VC-dimension of a single node univariate decision tree is m, for each possible dichotomy of m examples, we must be able to realize it. In a single node decision tree, we can have at most d possible orthogonal splits. The number of dichotomies that must be checked for a sample with m examples is $2^{m-1} - 1$ (See the discussion in the beginning of Section II). In order to be able to separate m instances for each possible dichotomy, the total number of splits must be at least as large as this number. So,

$$\begin{aligned} d &\geq 2^{m-1} - 1 \\ m &\leq \log_2(d+1) + 1 \end{aligned}$$

```
VCDimension LB-binary(DT, d)
1  if DT is a leaf node
2      return 1
3  if left and right subtrees of DT are leaves
4      return ⌊log2(d + 1)⌋ + 1
5  DT_L = Left subtree of DT
6  DT_R = Right subtree of DT
7  return LB-binary(DT_L, d − 1) + LB-binary(DT_R, d − 1)
```

Fig. 2. The pseudocode of the recursive algorithm for finding a lower bound of the VC-dimension of univariate decision tree with binary features: DT: Decision tree hypothesis class, d: Number of inputs

■

Theorem 2: The VC-dimension of a univariate decision tree with binary features that classifies d dimensional binary data is at least the sum of the VC-dimensions of its left and right subtrees those classifying $d - 1$ dimensional binary data.

Proof: Let the VC-dimension of two decision trees (DT_1 and DT_2) be VC_1 and VC_2 respectively. Under this assumption, those trees can classify VC_1 and VC_2 examples under all possible class labelings of those examples. Now we form the following tree: We add a new feature to the dataset and use that feature on the root node of the new decision tree, which has its left and right subtrees DT_1 and DT_2 respectively. The value of the new feature will be 0 for those instances forwarded to the left subtree (DT_1), 1 for those instances forwarded to the right subtree (DT_2). Now the new decision tree can classify at least $VC_1 + VC_2$ examples for all possible class labelings of those examples. ■

Figure 2 shows the recursive algorithm that calculates a lower bound for the VC-dimension of an arbitrary univariate decision tree using Theorems 1 and 2. There are two base cases; (i) the decision tree is a leaf node whose VC-dimension is 1, (ii) the decision tree is a single node decision tree whose VC-dimension is given in Theorem 1.

Corollary 1: A degenerate decision tree is a decision tree, where each node except the bottom one, has a single leaf. The VC-dimension of a degenerate univariate decision tree with N nodes that classifies d dimensional binary data is at least $\lfloor \log_2(d - N + 2) \rfloor + N$.

Proof: We will prove Corollary 1 by applying the algorithm LB-binary to the degenerate decision tree (See Figure 3 for an example). A degenerate decision tree with N nodes

- has $N - 1$ nodes, where each of them has one leaf and one child node. After applying LB-binary $N - 1$ times recursively, where each call contributes 1 to the VC-dimension (Line 2 of LB-binary), these nodes contribute $N - 1$ to the VC-dimension of the degenerate decision tree
- has one bottom node, which has two leaves. After applying LB-binary $N - 1$ times recursively, where each time d is decreased by one (Line 7 of LB-binary), this node contributes $\lfloor \log_2(d - (N - 1) + 1) \rfloor + 1 = \lfloor \log_2(d - N + 2) \rfloor + 1$ to the VC-dimension of the degenerate decision tree.

■

$$\mathbf{X} = \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \mathbf{x}^{(1)} & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \mathbf{x}^{(2)} & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \mathbf{x}^{(3)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{x}^{(4)} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \mathbf{x}^{(5)} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \mathbf{x}^{(6)} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \mathbf{x}^{(7)} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

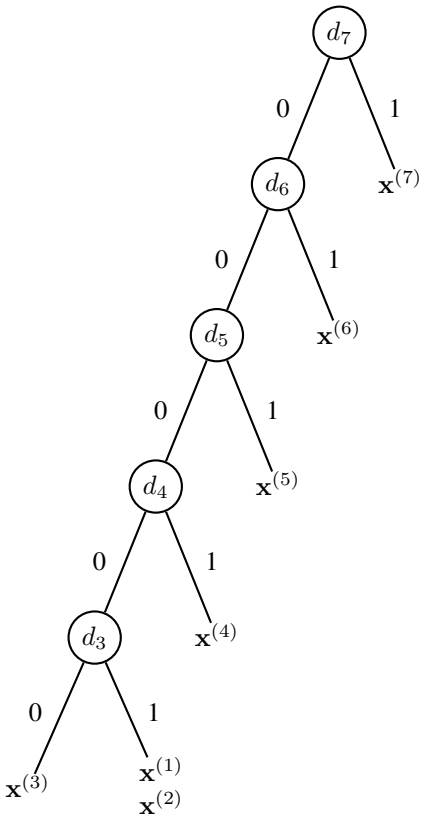


Fig. 3. Example for Corollary 1 with $d = 7$ and $N = 5$. If the class labeling of S is $\{1, 1, 0, x, x, x, x\}$ we select feature 3 in the bottom node. The labelings of the last four examples do not matter since they are alone in the leaves they reside.

Corollary 2: The VC-dimension of a full univariate decision tree of height h that classifies d dimensional binary data is at least $2^{h-1}(\lfloor \log_2(d-h+2) \rfloor + 1)$. In a full decision tree each node has two child nodes. The height of a tree is defined as the longest path taken from the root node to a leaf.

Proof: Similar to Corollary 1, we will prove Corollary 2 by applying the algorithm LB-binary to the full decision tree (See Figure 4 for an example). Similar to the bottom node in Corollary 1, each bottom node contributes $\lfloor \log_2(d-(h-1)+1) \rfloor + 1 = \lfloor \log_2(d-h+2) \rfloor + 1$ to the VC-dimension of the full decision tree. Since there are 2^{h-1} such nodes, the VC-dimension of the full decision tree is at least $2^{h-1}(\lfloor \log_2(d-h+2) \rfloor + 1)$. ■

III. Generalization to L-ary Decision Trees

Until now, we considered the VC-dimension of univariate decision trees with binary features. In this section, we generalize our idea to univariate decision trees with discrete features. In a univariate decision tree generated for such a dataset, there will be L children (branches) of each internal node corresponding to the L different outcomes of the decision. For this case, the input space X is a vectorial space of dimension d, the number of features, where each feature X_i can take values from discrete set $\{1, 2, \ldots, L_i\}$.

$$\mathbf{X} = \begin{array}{c|ccccc} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \mathbf{x}^{(1)} & 1 & 0 & 1 & 0 & 0 \\ \mathbf{x}^{(2)} & 0 & 1 & 1 & 0 & 0 \\ \mathbf{x}^{(3)} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{x}^{(4)} & 1 & 0 & 1 & 0 & 1 \\ \mathbf{x}^{(5)} & 0 & 1 & 1 & 0 & 1 \\ \mathbf{x}^{(6)} & 0 & 0 & 0 & 0 & 1 \\ \mathbf{x}^{(7)} & 1 & 0 & 1 & 1 & 0 \\ \mathbf{x}^{(8)} & 0 & 1 & 1 & 1 & 0 \\ \mathbf{x}^{(9)} & 0 & 0 & 0 & 1 & 0 \\ \mathbf{x}^{(10)} & 1 & 0 & 1 & 1 & 1 \\ \mathbf{x}^{(11)} & 0 & 1 & 1 & 1 & 1 \\ \mathbf{x}^{(12)} & 0 & 0 & 0 & 1 & 1 \end{array}$$

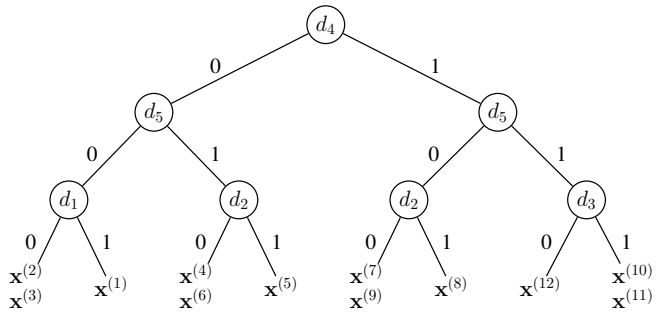


Fig. 4. Example for Corollary 2 with $d = 5$ and $h = 3$. Using feature 5 in the first level and feature 4 in the second level, one divides the class labelings into 4 subproblems of size 3. Each subproblem can then be shattered with a single node.

Theorem 3: The VC-dimension of a single node L-ary decision tree that classifies d dimensional discrete data is $\lfloor \log_2(\sum_{i=1}^{d}(2^{L_i-1}-1)+1) \rfloor + 1$.

Proof: The proof is similar to the proof of Theorem 1. We only give the reverse direction. If the VC-dimension of a single node L-ary decision tree is m, for each possible class labeling of m examples, we must be able to realize it. In a single node L-ary decision tree, we can have at most d possible splits corresponding to d features. Each split (corresponding to the feature d_i) can also divide the examples into $2^{L_i-1}-1$ distinct dichotomies. Therefore the total number of splits in a single node L-ary decision tree is $\sum_{i=1}^{d}(2^{L_i-1}-1)$. The number of dichotomies that must be checked for a sample with m examples is $2^{m-1}-1$ (See the discussion in the beginning of Section II). In order to be able to separate m instances for each possible dichotomy, the total number of splits must be at

```
VCDimension LB-L-ary(DT, d)
1  if DT is a leaf node
2      return 1
3  if all subtrees of DT are leaves
4      return ⌊log2(∑_{i=1}^{d}(2^{L_i−1} − 1) + 1)⌋ + 1
5  sum = 0
6  for i = 1 to number of subtrees
7      sum += LB-L-ary(DT_i, d − 1)
8  return sum
```

Fig. 5. The pseudocode of the recursive algorithm for finding a lower bound of the VC-dimension of L-ary decision tree: DT: Decision tree hypothesis class, d: Number of inputs

least as large as this number. So,

$$\sum_{i=1}^{d}(2^{L_i-1}-1) \geq 2^{m-1}-1$$

$$m \leq \log_2(\sum_{i=1}^{d}(2^{L_i-1}-1)+1)+1$$

■

Theorem 4: The VC-dimension of L-ary decision tree that classifies d dimensional discrete data is at least the sum of the VC-dimensions of its subtrees those classifying $d-1$ dimensional discrete data.

Proof: The proof is similar to the proof of Theorem 2. Let the VC-dimension of L decision trees $(DT_1, DT_2, \ldots, DT_L)$ be $VC_1, VC_2, \ldots, VC_L$ respectively. Under this assumption, those trees can classify $VC_1, VC_2, \ldots, VC_L$ examples under all possible class labelings of those examples. Now we form the following tree: We add a new feature which can have L different values to the dataset and use that feature on the root node of the new decision tree, which has its subtrees $DT_1, DT_2, \ldots, DT_L$. The value of the new feature will be 1 for those instances forwarded to the subtree (DT_1), 2 for those instances forwarded to the subtree (DT_2), ..., L for those instances forwarded to the subtree (DT_L). Now the new decision tree can classify at least $\sum_{i=1}^{L} VC_i$ examples for all possible class labelings of those examples. ■

Figure 5 shows the recursive algorithm that calculates a lower bound for the VC-dimension of an arbitrary L-ary decision tree using Theorems 3 and 4. There are two base cases; (i) the L-ary decision tree is a leaf node whose VC-dimension is 1, (ii) the L-ary decision tree is a single node decision tree whose VC-dimension is given in Theorem 3.

IV. Generalization to Decision Trees with Continuous Features

Until now, we considered the VC-dimension of univariate decision trees with discrete features. In this section, we generalize our idea to univariate decision trees with continuous features. In a univariate decision tree generated for such a dataset, there will be always two children (branches) of each internal node. For this case, the input space X is a vectorial space of dimension d, where each feature d_i can take values from continuous space $\mathcal{R}$. We assume that, for at least one feature d_i, all instances have distinct values.

Corollary 3: The VC-dimension of a single node decision tree that classifies d dimensional continuous data is at least $\lfloor \log_2(d+1) \rfloor + 1$.

Proof: The proof directly follows the proof of Theorem 1 given a slight modification. We construct the sample S such that

$$\mathbf{X} = \mathbf{C_m^T} + \mathbf{R_m}$$

where $\mathbf{R_m}$ is a random matrix of size $m \times (2^{m-1}-1)$ containing random values from the interval (0, 1). Given such an $\mathbf{X}$, $\lfloor x_i^{(t)} \rfloor$ will correspond to a possible class labeling of $\mathbf{x}^{(t)}$, implying a one-to-one mapping between dichotomies and features. So for each possible dichotomy to be checked, we will choose the decision tree hypothesis h which has the corresponding feature as the split feature and the split is $x_i \leq 1$ (See Figure 6 for an example). ■

Figure 7 shows the recursive algorithm that calculates a lower bound for the VC-dimension of an arbitrary decision tree for continuous data using Theorem 3. There are two differences between algorithm LB-binary in Figure 2 and algorithm LB-continuous in Figure 7. For discrete data sets, a feature can only be used once through a path from the root node to a leaf node. On the other hand, for continuous data sets, one can construct all nodes of the decision tree based on a single feature. For this reason, on each recursive call, LB-binary decreases d by 1, where LB-continuous does not (Line 7).

As explained above, we spare one feature for constructing the inner splits of the decision tree (Lines 5-6). The remaining features are used to forward the instances to the nodes having two leaves as children (Lines 3-4). We set the values of the spared feature in increasing order from left to right, that is, the instances forwarded to the leftmost/rightmost node will have the smallest/largest value in that spared feature. After that, with the appropriate splits based on that single feature, the same instances are always forwarded to the same node(s) (See Figure 8 for an example). For this reason, when we encounter a node with two leaves in LB-binary, the VC-dimension is $\lfloor \log_2(d+1) \rfloor + 1$, where d represents the remaining features for that node, whereas when we encounter a node with two leaves in LB-continuous, the VC-dimension is $\lfloor \log_2(d) \rfloor + 1$, where d represents the number of all features in that data set (Line 4).

V. Experiments

A. Exhaustive Search Algorithm

To show the bounds found using Theorems 1-4 or using the algorithm in Figure 2 are tight, we search the VC-dimension of different decision tree hypothesis classes exhaustively using the algorithm in [16].

The pseudocode for finding VC-Dimension of univariate decision tree with binary features is given in Figure 9. Given a dataset with d dimensions, we generate all possible data combinations having N data points iteratively (Line 3 and Line 17). For each data combination, we generate all possible class labelings iteratively (Line 7 and Line 12). For each

$$\mathbf{X} = \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \mathbf{x}^{(1)} & 1.2 & 0.3 & 0.2 & 0.6 & 1.4 & 1.5 & 1.1 \\ \mathbf{x}^{(2)} & 0.8 & 1.4 & 0.8 & 0.2 & 1.3 & 0.3 & 0.2 \\ \mathbf{x}^{(3)} & 0.6 & 0.6 & 1.1 & 0.7 & 0.1 & 1.1 & 0.7 \\ \mathbf{x}^{(4)} & 0.4 & 0.7 & 0.7 & 1.9 & 0.2 & 0.9 & 1.7 \end{array}$$

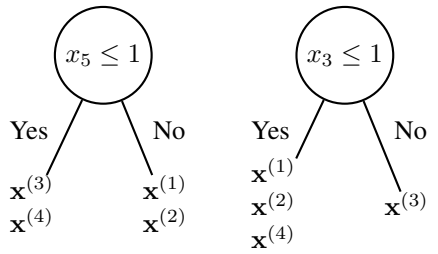


Fig. 6. Example for Corollary 3 with $d = 7$ and $m = 4$. If the class labeling of S is {1, 1, 0, 0} we select feature 5 and the split $x_5 \leq 1$ (left decision tree). If the class labeling of S is {0, 0, 1, 0} we select feature 3 and the split $x_3 \leq 1$ (right decision tree).

```
VCDimension LB-continuous(DT)
1  if DT is a leaf node
2      return 1
3  if left and right subtrees of DT are leaves
4      return ⌊log2(d)⌋ + 1
5  DT_L = Left subtree of DT
6  DT_R = Right subtree of DT
7  return LB-continuous(DT_L) + LB-continuous(DT_R)
```

Fig. 7. The pseudocode of the recursive algorithm for finding a lower bound of the VC-dimension of univariate decision tree for continuous data: DT: Decision tree hypothesis class, d: Number of inputs

possible class labeling of a data combination, we check if there is an hypothesis h from the decision tree hypothesis class H that classifies the data correctly (Line 9). If there is not such an hypothesis (Line 10, 11), we break the class labeling search and continue the search with the next data combination (Line 17). If there is such an hypothesis, we iterate to next class labeling (Line 12). If for all class labelings of a data combination we can find a decision tree hypothesis h (Lines 13, 14), we increment N (Lines 18, 19) and continue the search. If all subsets N of 2^d are iterated and no subset is classified for all class labelings, then the search is over and VC dimension is taken as $N - 1$. Since the computational complexity of the exhaustive search is exponential, we can run the exhaustive search algorithm only on cases with small d and N.

Figures 10 and 11 show our calculated lower bound and exact VC-dimension of decision trees for datasets with 3 and 4 input features. It can be seen that the VC-dimension increases as the number of nodes in the decision tree increases, but there are exceptions where the VC-dimension remains constant though the number of nodes increases, which shows that the VC-dimension of a decision tree not only depends the number of nodes, but also the structure of the tree. The results show that our bounds are tight for small d and N: the maximum difference between the calculated lower bound

$$\mathbf{X} = \begin{array}{c|cccc} & d_1 & d_2 & d_3 & d_4 \\ \mathbf{x}^{(1)} & 1.2 & 0.8 & 1.1 & 0.1 \\ \mathbf{x}^{(2)} & 0.9 & 1.3 & 1.5 & 0.2 \\ \mathbf{x}^{(3)} & 0.8 & 0.7 & 0.5 & 0.1 \\ \mathbf{x}^{(4)} & 1.1 & 0.6 & 1.3 & 0.6 \\ \mathbf{x}^{(5)} & 0.7 & 1.1 & 1.2 & 0.7 \\ \mathbf{x}^{(6)} & 0.6 & 0.9 & 0.8 & 0.8 \\ \mathbf{x}^{(7)} & 1.4 & 0.8 & 1.7 & 1.1 \\ \mathbf{x}^{(8)} & 0.8 & 1.3 & 1.2 & 1.3 \\ \mathbf{x}^{(9)} & 0.7 & 0.4 & 0.9 & 1.2 \\ \mathbf{x}^{(10)} & 1.3 & 0.8 & 1.4 & 1.6 \\ \mathbf{x}^{(11)} & 0.8 & 1.7 & 1.3 & 1.9 \\ \mathbf{x}^{(12)} & 0.9 & 0.7 & 0.6 & 1.8 \end{array}$$

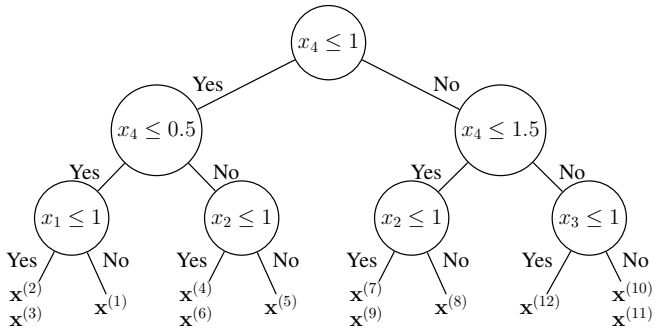


Fig. 8. Example for algorithm LB-continuous with $d = 4$ and $m = 12$. Using the spared feature 4 in all levels except the last level, one divides the class labelings into 4 subproblems of m = 3. Each subproblem can then be shattered with a single node.

```
VCDimension ExhaustiveSearch(H, d, data)
1  N = 1
2  while TRUE
3      dataComb = getDataCombination(data, N)
4      successful = FALSE
5      while dataComb != NULL
6          classifiedAllCombinations = TRUE
7          classComb = getClassCombination(N)
8          while classComb != NULL
9              if not treeClassify(H, dataComb, classComb)
10                 classifiedAllCombinations = FALSE
11                 break
12             classComb = getNextClassCombination(N)
13         if classifiedAllCombinations
14             successful = TRUE
15             break
16         else
17             dataComb = getNextDataCombination(data, N)
18     if successful
19         N = N + 1
20     else
21         break
22 return N − 1
```

Fig. 9. The pseudocode of the exhaustive search algorithm for finding VC-dimension of univariate decision tree: H: Decision tree hypothesis class, d: Number of inputs in the dataset, $data$: Universal set for d dimensional input

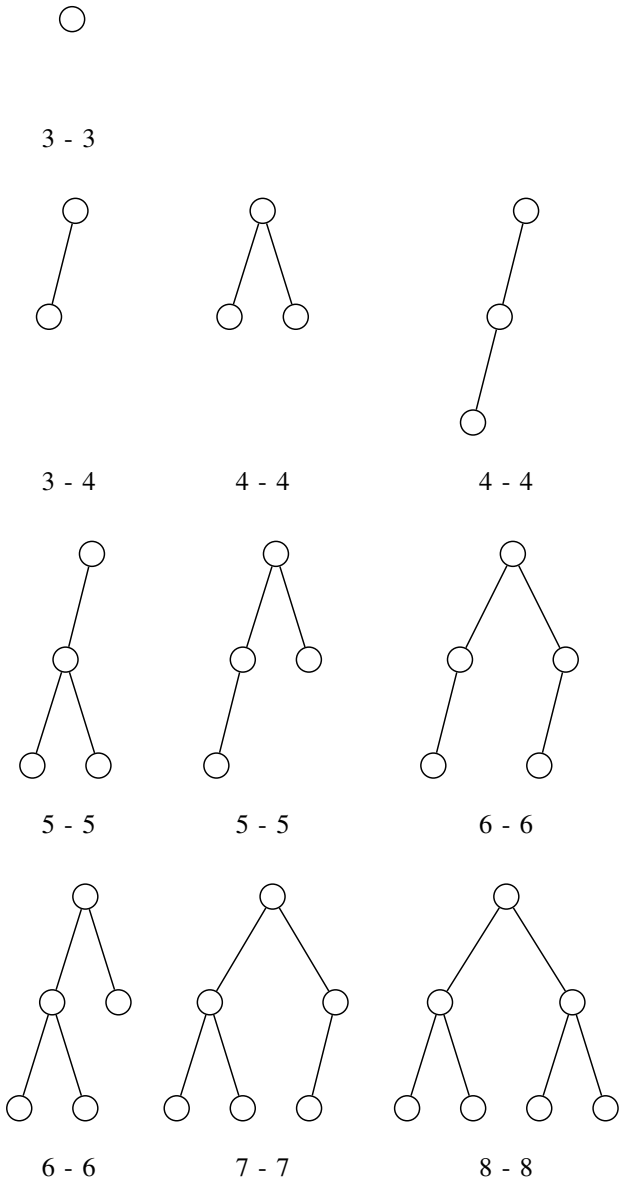


Fig. 10. Calculated lower bound and the exact VC-dimension of univariate decision trees for datasets with 3 input features. Only the internal nodes are shown.

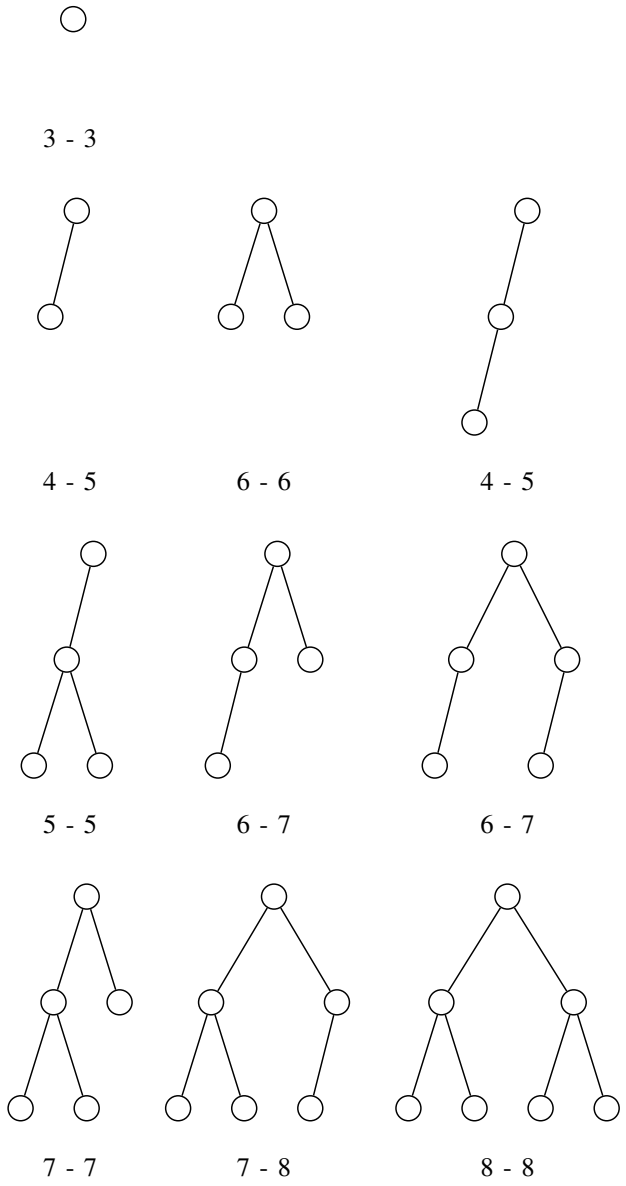


Fig. 11. Calculated lower bound and the exact VC-dimension of univariate decision trees for datasets with 4 input features. Only the internal nodes are shown.

and the exact VC-dimension is 1. Also for most of the cases, our proposed algorithm based on lower bounds finds the exact VC-dimension of the decision tree.

B. Complexity Control Using VC-Dimension Bounds

In this section, we use our VC-dimension bounds for complexity control in decision trees. Controlling complexity in decision trees could be done in two ways.

- We can control the complexities of the decision nodes by selecting the appropriate model for a node. For example, an omnivariate decision tree can have univariate, linear multivariate or nonlinear multivariate nodes [17], [18].
- We can control the overall complexity of the decision tree via pruning.

Since this paper covers only univariate trees, we take the second approach and use the VC-dimension bounds found in the previous sections for pruning.

In postpruning (CvPrune), for each subtree T,

- We calculate the validation error of the tree
- We replace T with a leaf and calculate the validation error of the pruned tree.

If there is overfitting, we expect the more complex subtree T to learn the noise and perform worse than the simple leaf. Here the validation set determines the number of nodes after pruning and it is not possible to determine the number of nodes beforehand.

When we try to prune a subtree T using SRM, we have two choices, namely SrmLocal and SrmGlobal. In our previous work [15], we proposed SrmLocal, where

TABLE I
DETAILS OF THE DATASETS. d: NUMBER OF ATTRIBUTES, S: SAMPLE SIZE

Set	d	S	Type
Acceptors	88	3889	Discrete
Arabidopsis	1558	3279	Continuous
Artificial	10	320	Discrete
Breast	9	699	Continuous
Bupa	6	345	Continuous
Dlbcl	5439	77	Continuous
Donors	13	6246	Discrete
German	24	1000	Continuous
Hepatitis	19	155	Continuous
Haberman	3	306	Continuous
Heart	13	270	Continuous
Ironosphere	34	351	Continuous
Krvskp	36	3196	Discrete
Magic	10	19020	Continuous
Monks	6	432	Discrete
Mushroom	22	8124	Discrete
Musk2	166	6598	Continuous
Parkinsons	22	195	Continuous
Pima	8	768	Continuous
Polyadenylation	169	6371	Continuous
Promoters	57	106	Discrete
Prostatetumor	10509	102	Continuous
Ringnorm	20	7400	Continuous
Satellite47	36	2134	Continuous
Spambase	57	4601	Continuous
Spect	22	267	Discrete
Tictactoe	9	958	Discrete
Titanic	3	2201	Discrete
Transfusion	4	748	Continuous
Twonorm	20	7400	Continuous
Vote	16	435	Discrete

- We find the upper bound of the generalization error of T using Equation 2 where V is the VC-dimension and E_t is the training error of the subtree T to be pruned. The training error of a subtree T is calculated over the instances arriving into T.
- We find the upper bound of the generalization error of the leaf replacing T using Equation 2 where the VC-dimension of a leaf is 1 and E_t is the training error of the leaf. The training error of a leaf is calculated over the instances arriving into it.

If the upper bound of the generalization error of the leaf is smaller than the upper bound of the generalization error of the subtree T, we prune the subtree T, otherwise we keep it.

In this paper, we propose SrmGlobal, where

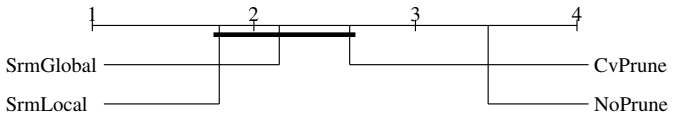
- We find the upper bound of the generalization error using Equation 2 where V is the VC-dimension and E_t is the training error of the whole tree without pruning.
- We prune subtree T by replacing it with a leaf and find the upper bound of the generalization error using Equation 2 where V is the VC-dimension and E_t is the training error of the pruned tree.

If the upper bound of the generalization error of the pruned tree is smaller than the upper bound of the generalization error of the unpruned tree, we prune the subtree T, otherwise we keep it.

CvPrune corresponds to ID3 [7] for discrete datasets and C4.5 [6] for continuous datasets where pruning is done via cross-validation. SrmLocal and SrmGlobal correspond to ID3 for discrete datasets and C4.5 for continuous datasets where pruning is done via SRM locally and globally respectively.

TABLE II
THE AVERAGE AND STANDARD DEVIATIONS OF ERROR RATES OF DECISION TREES GENERATED USING NOPRUNE, CVPRUNE, SRMLOCAL, AND SRMGLOBAL. THE FIGURE BELOW SHOWS THE RESULT OF THE POST-HOC NEMENYI'S TEST.

Dataset	NoPrune	CvPrune	SrmLocal	SrmGlobal
artificial	0.7 ± 1.6	1.1 ± 1.8	0.7 ± 1.6	0.7 ± 1.6
mushroom	0.0 ± 0.1	0.0 ± 0.1	0.0 ± 0.1	0.0 ± 0.1
tictactoe	22.1 ± 3.5	23.8 ± 2.2	22.1 ± 3.5	22.1 ± 3.5
titanic	21.4 ± 0.4	21.8 ± 0.5	21.4 ± 0.4	21.4 ± 0.4
acceptors	18.6 ± 1.0	16.1 ± 2.0	16.5 ± 0.8	17.3 ± 0.7
arab.	3.7 ± 0.4	3.4 ± 0.6	3.4 ± 0.4	3.5 ± 0.6
bupa	39.8 ± 4.2	38.6 ± 4.1	37.2 ± 4.5	39.1 ± 4.9
donors	9.4 ± 0.4	7.7 ± 0.4	8.1 ± 0.4	7.5 ± 0.5
german	32.7 ± 1.9	29.9 ± 0.0	28.9 ± 1.9	29.9 ± 0.0
haberman	35.3 ± 3.9	26.6 ± 0.3	26.5 ± 0.0	26.3 ± 0.6
heart	30.9 ± 3.5	28.3 ± 4.7	26.9 ± 2.3	29.8 ± 3.6
hepatitis	26.5 ± 4.1	22.1 ± 4.4	21.2 ± 0.0	21.2 ± 0.0
ironosphere	14.9 ± 2.7	13.1 ± 1.9	14.2 ± 3.2	13.1 ± 2.9
magic	18.9 ± 0.7	17.5 ± 0.6	16.7 ± 0.4	16.6 ± 0.4
pima	31.9 ± 1.4	27.9 ± 3.4	26.7 ± 2.0	30.4 ± 1.9
poly.	32.8 ± 1.4	30.5 ± 1.3	29.1 ± 1.3	30.2 ± 1.1
promoters	33.3 ± 10.1	26.1 ± 9.9	30.0 ± 11.4	27.5 ± 10.9
ringnorm	12.7 ± 1.2	12.2 ± 1.1	12.6 ± 1.1	12.3 ± 1.2
satellite47	17.5 ± 0.9	15.4 ± 1.5	15.3 ± 1.3	16.3 ± 1.7
spect	23.3 ± 4.7	19.1 ± 2.8	20.8 ± 2.2	21.1 ± 0.0
transfusion	30.1 ± 2.5	24.0 ± 0.0	23.8 ± 1.1	24.0 ± 0.0
vote	6.9 ± 1.7	5.2 ± 0.7	4.8 ± 0.0	4.8 ± 0.0
breast	6.7 ± 0.9	6.7 ± 1.1	6.0 ± 1.2	5.8 ± 1.1
dlbcl	20.4 ± 6.1	23.7 ± 4.7	20.4 ± 6.1	20.4 ± 6.1
krvskp	1.0 ± 0.3	1.2 ± 0.4	1.0 ± 0.4	1.0 ± 0.4
parkinsons	13.5 ± 3.1	13.8 ± 2.3	12.9 ± 3.3	13.5 ± 3.1
prostate.	22.9 ± 6.6	23.1 ± 7.1	22.3 ± 6.8	22.9 ± 7.1
spambase	9.7 ± 0.6	9.9 ± 0.7	9.0 ± 0.5	9.0 ± 0.4
twonorm	16.0 ± 0.5	17.0 ± 0.7	15.9 ± 0.5	15.9 ± 0.6
monks	13.3 ± 6.8	12.8 ± 7.8	13.3 ± 6.8	13.3 ± 6.8
musk2	5.7 ± 0.9	5.5 ± 0.6	5.6 ± 0.9	5.7 ± 0.8



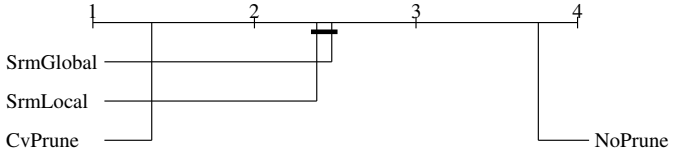
We compare SRM based prunings with CvPrune. For the sake of generality, we also include the results of trees before any pruning is applied (NoPrune). We use Friedman's test for the equality of the results of the algorithms and *Nemenyi's test* as the post-hoc test to compare neighboring algorithms for significant difference in rank [19].

We did the experiments on a total of 31 data sets where 22 of them are from UCI repository [20] and 9 are (*acceptors*, *arabidopsis*, *dlbcl*, *donors*, *musk2*, *parkinsons*, *polyadenylation*, *prostatetumor*, and *transfusion*) bioinformatics datasets (see Table I). We first separate one third of the data set as the test set over which we evaluate and report the final performance. With the remaining two thirds, we apply 5×2-fold cross validation, which gives a total of ten folds for each data set. For CvPrune, validation folds are used as a pruning set. For both SrmLocal and SrmGlobal, we did a grid-search on a_1 with values from $\{0.1, 0.2, \ldots, 3.9, 4\}$ and a_2 with values from $\{0.1, 0.2, \ldots, 1.9, 2\}$ using also validation folds.

Tables II and III show the average and standard deviations of error rates and tree complexities of decision trees generated

TABLE III
THE AVERAGE AND STANDARD DEVIATIONS OF NUMBER OF DECISION NODES OF DECISION TREES GENERATED USING NOPRUNE, CVPRUNE, SRMLOCAL, AND SRMGLOBAL. THE FIGURE BELOW SHOWS THE RESULT OF THE POST-HOC NEMENYI'S TEST.

Dataset	NoPrune	CvPrune	SrmLocal	SrmGlobal
artificial	4.7 ± 0.7	4.4 ± 1.0	4.7 ± 0.7	4.7 ± 0.7
mush.	4.9 ± 0.3	4.9 ± 0.3	4.9 ± 0.3	4.9 ± 0.3
tictactoe	54.3 ± 4.9	22.2 ± 6.8	54.3 ± 4.9	54.3 ± 4.9
titanic	8.3 ± 0.8	4.2 ± 0.6	8.3 ± 0.8	8.3 ± 0.8
acceptors	79.0 ± 4.0	7.1 ± 6.9	54.9 ± 1.8	61.6 ± 6.1
arab.	62.7 ± 19.0	12.5 ± 5.3	25.7 ± 9.0	31.8 ± 14.0
bupa	25.4 ± 3.8	5.4 ± 3.7	18.5 ± 5.0	18.2 ± 2.3
donors	124.8 ± 1.9	21.0 ± 3.7	72.1 ± 7.1	58.3 ± 7.3
german	68.5 ± 5.6	0.0 ± 0.0	40.4 ± 8.0	0.0 ± 0.0
haberman	29.7 ± 3.2	1.0 ± 3.2	0.0 ± 0.0	1.6 ± 1.6
heart	14.9 ± 1.9	3.5 ± 2.7	6.9 ± 2.5	12.2 ± 3.1
hepatitis	6.1 ± 1.3	0.7 ± 0.9	0.0 ± 0.0	0.0 ± 0.0
irono.	7.8 ± 0.9	3.8 ± 1.9	4.4 ± 1.3	4.0 ± 1.1
magic	627.0 ± 10.9	30.1 ± 16.5	308.7 ± 29.9	76.6 ± 23.2
pima	41.7 ± 3.9	3.8 ± 2.6	5.9 ± 4.6	28.5 ± 3.6
poly.	231.1 ± 7.0	22.5 ± 18.4	44.1 ± 15.6	89.2 ± 18.8
promoters	4.7 ± 0.9	2.0 ± 1.3	2.8 ± 1.0	1.0 ± 0.0
ringnorm	123.9 ± 10.3	45.7 ± 5.2	110.1 ± 9.3	94.2 ± 6.6
satellite47	57.2 ± 3.8	11.9 ± 4.5	21.2 ± 4.6	30.5 ± 9.9
spect	21.1 ± 2.3	5.2 ± 5.8	6.8 ± 7.3	0.0 ± 0.0
trans.	67.8 ± 4.2	0.0 ± 0.0	8.6 ± 14.0	0.0 ± 0.0
vote	8.9 ± 2.5	2.9 ± 1.7	1.0 ± 0.0	1.0 ± 0.0
breast	9.9 ± 1.9	4.1 ± 2.2	4.0 ± 2.0	4.9 ± 1.8
dlbcl	1.4 ± 0.5	0.5 ± 0.7	1.4 ± 0.5	1.4 ± 0.5
krvskp	35.7 ± 3.2	23.7 ± 4.2	31.2 ± 4.4	30.3 ± 4.5
park.	5.6 ± 0.8	3.3 ± 1.7	4.5 ± 1.4	5.6 ± 0.8
prostate.	2.2 ± 0.4	1.3 ± 0.5	1.2 ± 0.4	1.0 ± 0.0
spambase	101.8 ± 5.5	20.3 ± 8.4	65.6 ± 7.0	57.0 ± 13.9
twonorm	152.0 ± 6.5	79.9 ± 8.0	139.6 ± 7.2	133.9 ± 6.9
monks	24.0 ± 6.1	11.2 ± 2.4	24.0 ± 6.1	24.0 ± 6.1
musk2	68.6 ± 4.2	28.5 ± 7.0	60.8 ± 7.8	62.7 ± 4.8



using NoPrune, CvPrune, SrmLocal, and SrmGlobal respectively. Friedman's test rejects the equality of error rates. Post-hoc Nemenyi's test's results on error rates show that pruning works, that is, all three pruning strategies form a clique and they are significantly better than NoPrune.

Friedman's test also rejects the equality of tree complexities. According to the post-hoc Nemenyi's test's results on tree complexity, there are three groups: CvPrune generates significantly smaller trees than (SrmLocal, SrmGlobal) group, which also generate significantly smaller trees than NoPrune.

On four discrete datasets (first group) there is no need to prune, i.e., pruning decreases performance and in this cases, CvPrune prunes trees aggressively by sacrificing accuracy, whereas both SrmLocal and SrmGlobal do not prune and gets the best performance with NoPrune.

On eighteen datasets (second group) pruning helps, i.e., pruning reduces both the error rate and the tree complexity as needed.

On seven datasets (third group) CvPrune prunes trees aggressively by sacrificing accuracy, whereas both SrmLocal and SrmGlobal prune well and gets smaller and at least as accurate trees as NoPrune.

TABLE IV
THE AVERAGE AND STANDARD DEVIATIONS OF ERROR BOUNDS OF DECISION TREES GENERATED USING SRMLOCAL, AND SRMGLOBAL.

Dataset	SrmLocal	SrmGlobal
acceptors	19.3 ± 0.8	17.8 ± 0.6
arabidopsis	8.9 ± 2.7	4.5 ± 0.4
artificial	47.5 ± 0.2	47.5 ± 0.2
breast	13.1 ± 1.7	12.8 ± 1.3
bupa	69.1 ± 4.9	64.9 ± 29.7
dlbcl	146.6 ± 6.4	146.6 ± 6.4
donors	10.4 ± 0.3	13.3 ± 0.3
german	30.0 ± 1.3	61.6 ± 0.0
haberman	93.7 ± 0.0	31.3 ± 5.6
heart	71.8 ± 3.1	34.7 ± 2.0
hepatitis	127.8 ± 0.5	127.8 ± 0.5
ironosphere	16.3 ± 2.5	6.8 ± 30.7
krvskp	3.1 ± 0.2	3.1 ± 0.2
magic	17.5 ± 0.5	23.1 ± 0.6
monks	15.6 ± 3.2	15.6 ± 3.2
mushroom	7.1 ± 0.1	2.6 ± 0.4
musk2	1.8 ± 1.1	12.2 ± 0.4
parkinsons	45.4 ± 5.6	80.5 ± 2.8
pima	36.7 ± 2.8	36.5 ± 2.2
polyadenylation	37.2 ± 2.0	38.1 ± 2.1
promoters	46.5 ± 5.3	21.9 ± 4.2
prostatetumor	76.4 ± 4.5	117.4 ± 60.8
ringnorm	5.4 ± 0.4	11.6 ± 0.6
satellite47	22.7 ± 1.3	21.6 ± 1.4
spambase	9.8 ± 0.7	12.1 ± 0.4
spect	34.1 ± 2.1	91.1 ± 0.5
tictactoe	7.5 ± 2.0	7.5 ± 2.0
titanic	22.9 ± 3.2	36.2 ± 1.6
transfusion	32.8 ± 1.2	59.1 ± 0.0
twonorm	10.4 ± 0.6	14.4 ± 0.4
vote	36.6 ± 1.4	36.6 ± 1.4

Table IV shows the average and standard deviations of error bounds of decision trees generated using SrmLocal and SrmGlobal. It is well known that the generalization bounds given by the VC-dimension are not necessarily tight, that is, the upper bound for generalization error given by the equation 2 can be very loose [21]. For example, in our experiments, in datasets such as *artificial*, *bupa*, *dlbcl*, *hepatitis*, *prostatetumor* the bounds are extremely large. On the other hand, there are also cases in the literature where SRM works well and the bounds are not necessarily loose [3], [22], [23]. Cherkassy and Ma [24] show that SRM consistently outperforms AIC (Akaike Information Criterion) in all datasets they covered and SRM and BIC (Bayesian Information Criterion) methods attain similar predictive performance. Also in our experiments, in many cases the error bounds are useful, i.e., the difference between the test error and the error bound is quite small. In general, the size of the dataset inversely effects the error bound. For small datasets, the differences between the test error and error bound are large, for large datasets it is reverse.

VI. CONCLUSION

This paper tries to fill the gap in the statistical learning theory, where there is no explicit formula for the VC-dimension of a decision tree. In this work, we first focused on the easiest case of univariate trees with binary features. Starting from basic decision tree with a single decision node, we give and prove lower bounds of the VC-dimension of different decision tree structures. We also show that our approach can

be generalized to decision trees having more than two feature values and continuous univariate trees. In general, we prove that the VC-dimension of a univariate decision tree depends on the number of features and the VC-dimension of the left and right subtrees of it (tree structure).

To show our bounds are tight, we use the exhaustive search algorithm given in [16] to calculate the exact VC-dimension of simple trees and compare our bounds with the exact VC-dimension values. These VC-dimension bounds are then used in pruning decision trees and when compared with cross-validation pruning, we see that SRM-pruning using our VC-dimension values work well and find trees that are as accurate as CV pruning.

REFERENCES

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995.

[2] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[3] V. Cherkassky and F. Mulier, *Learning From Data: Concepts, Theory, and Methods*, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., 2007.

[4] X. Shao, V. Cherkassky, and W. Li, "Measuring the VC-dimension using optimized experimental design," *Neural Computation*, vol. 12, no. 8, pp. 1969–1986, Aug. 2000.

[5] V. Vapnik, E. Levin, and Y. L. Cun, "Measuring the VC-dimension of a learning machine," *Neural Computation*, vol. 6, no. 5, pp. 851–876, Sep. 1994.

[6] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[7] ——, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[8] S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 345–389, Dec. 1998.

[9] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers - a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 35, no. 4, pp. 476–487, Nov. 2005.

[10] T. S. Lim, W. Y. Loh, and Y. S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine Learning*, vol. 40, no. 3, pp. 203–228, Sep. 2000.

[11] M. Fernndez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014.

[12] Y. Mansour, "Pessimistic decision tree pruning based on tree size," in *Proceedings of the 14th international conference on Machine learning*, Nashville, TN, USA, Jul. 1997, pp. 195–201.

[13] H. U. Simon, "The Vapnik-Chervonenkis dimension of decision trees with bounded rank," *Information Processing Letters*, vol. 39, no. 3, pp. 137–141, 1991.

[14] O. Maimon and L. Rokach, "Improving supervised learning by feature decomposition," in *Proceedings of the Second International Symposium on Foundations of Information and Knowledge Systems*. Salzau Castle, Germany: Springer Verlag, Feb. 2002, pp. 178–196.

[15] O. T. Yıldız, "On the VC-dimension of univariate decision trees," in *1st International Conference on Pattern Recognition and Methods*, Vilamoura, Algarve, Portugal, Feb. 2012, pp. 205–210.

[16] O. Aslan, O. T. Yıldız, and E. Alpaydın, "Calculating the VC-dimension of decision trees," in *Proceedings of the 24th International Symposium on Computer and Information Sciences*, North Cyprus, Sep. 2009, pp. 193–198.

[17] O. T. Yıldız and E. Alpaydın, "Omnivariate decision trees," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1539–1546, Nov. 2001.

[18] O. T. Yıldız, "Model selection in omnivariate decision trees using structural risk minimization," *Information Sciences*, vol. 181, no. 23, pp. 5214–5226, Dec. 2011.

[19] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, Dec. 2006.

[20] C. Blake and C. Merz, "UCI repository of machine learning databases," 2000. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[21] D. Cohn and G. Tesauro, "How tight are the Vapnik-Chervonenkis bounds," *Neural Computation*, vol. 4, pp. 249–269, Mar. 1992.

[22] V. Cherkassky, X. Shao, F. M. Mulier, and V. Vapnik, "Model complexity control for regression using VC generalization bounds," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1075–1089, Dec. 1999.

[23] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed. New York, NY, USA: Springer-Verlag New York, Inc., 2009.

[24] V. Cherkassky and Y. Ma, "Comparison of model selection for regression," *Neural Computation*, vol. 15, no. 7, pp. 1691–1714, Jul. 2003.

Olcay Taner YILDIZ received his BSc, MSc, and PhD degrees in computer science from Boğaziçi University in 1997, 2000, and 2005. He did postdoctoral work at the University of Minnesota in 2005. He is associate professor of Computer Engineering at Işık University. He worked on machine learning, specifically model selection and decision trees. His current research is on software engineering, natural language processing, and bioinformatics.