Learning Decision Tree Classifiers from Attribute Value Taxonomies and Partially Specified Data

Jun Zhang JZHANG@CS.IASTATE.EDU
Vasant Honavar HONAVAR@CS.IASTATE.EDU
Artificial Intelligence Research Laboratory, Department of Computer Science, Iowa State University
Ames, IA 50011 USA

Abstract

We consider the problem of learning to classify *partially specified* instances i.e., instances that are described in terms of attribute values at different levels of precision, using user-supplied attribute value taxonomies (AVT). We formalize the problem of learning from AVT and data and present an AVT-guided decision tree learning algorithm (AVT-DTL) to learn classification rules at multiple levels of abstraction. The proposed approach generalizes existing techniques for dealing with missing values to handle instances with *partially missing* values. We present experimental results that demonstrate that AVT-DTL is able to effectively learn robust high accuracy classifiers from partially specified examples. Our experiments also demonstrate that the use of AVT-DTL outperforms standard decision tree algorithm (C4.5 and its variants) when applied to data with missing attribute values; and produces substantially more compact decision trees than those obtained by standard approach.

1. Introduction

In many pattern classification tasks, it is often the case that the instances to be classified are specified at different levels of precision. That is, the value of a particular attribute, or the class label associated with an instance, or both are specified at different levels of precision in different instances, leading to *partially specified instances*. To illustrate this phenomenon, an attribute value taxonomy (AVT) for the "color" attribute in which color takes on several values – blue, red, etc. is shown in Figure 1. Now suppose that *blue* objects can be further specified in terms of the precise shade of blue such as *sky blue*, *light blue*, *dark blue* and *navy blue.* In this case, in one instance, the color of a particular object may be described as *navy blue*, whereas in another instance, it may be specified simply as *blue* without specifying the precise shade of blue.

Partially specified instances are encountered quite often in practice. For example, in a medical diagnosis task, different cases may be described in terms of symptoms or results of diagnostic tests at different levels of precision e.g., a patient may be described as having cardiac arrhythmia without specifying the precise type of arrhythmia. In an intrusion detection task, the activity to be classified may be specified in terms of events described at different levels of precision**.** This problem can be exacerbated when data are gathered by multiple, autonomous entities (e.g., physicians, hospitals) each with its own local AVT. Hence, algorithms for learning from AVT and data are of significant practical interest.

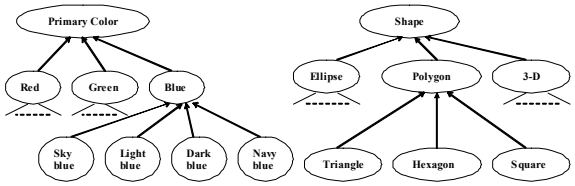

Figure 1. Simple Illustrative Taxonomies on Color and Shape

A second motivation for considering algorithms for learning from AVT and data arises from the preference for comprehensible and simple, yet accurate and robust classifiers in many practical applications of data mining. The availability of AVT presents an opportunity to learn classification rules that are expressed in terms of *abstract* attribute values (e.g., color=*blue* instead of color=*navy blue*) leading to simpler, easier-to-comprehend rules.

A third motivation for considering algorithms for learning from AVT and data arises from the need to learn from relatively small data sets where there is a greater chance of generating classifiers that overfit the training data. A common approach used by statisticians when estimating from small samples involves *shrinkage* (Duda et al, 2000) or using *abstract attribute values* which correspond to sets of the original attribute values grouped according to an AVT (or *abstract class labels* from a class taxonomy) when there are too few instances that match any specific attribute value or class label to estimate the relevant statistics with adequate confidence. Learning algorithms that exploit AVT can automatically perform *shrinkage*

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

thereby yielding robust classifiers. Hence, such algorithms can perform a simple form of regularization or pruning so as to minimize over-fitting.

A fourth motivation for exploring algorithms that can exploit user-supplied AVT to analyze a data stems from the significance of interaction between data and knowledge (Zhang et al., 2002) – in particular, knowledge that is in the form of an ontology available to the learner. An ontology consists of a set of *concepts* and *relations* among concepts. In many data-driven knowledge acquisition tasks, there is a need to explore data from *multiple points of view* that reflect *different ontological commitments* on the part of the learner. This is particularly important in scientific applications in which specific ontological and representational commitments often reflect prior knowledge and working assumptions of scientists. In such a setting, there is no single universal ontology that can serve all users in every context. Hence, methods for learning from ontologies and data are needed to support knowledge discovery. AVTs constitute an especially interesting and commonly encountered type of ontologies. Therefore, the exercise of designing algorithms for learning from AVT can offer useful insights into the more general problem of learning from ontologies and data.

Against this background, this paper explores the problem of learning from AVT and data. We present an AVT-guided decision tree learning algorithm (AVT-DTL) to learn classification rules at multiple levels of abstraction. The proposed approach generalizes a principled approach to dealing with missing values to handle instances with partially missing values. We present experimental results that demonstrate that AVT-DTL is able to effectively learn robust high accuracy classifiers from partially specified examples. Our experiments also demonstrate that AVT-DTL outperforms standard decision tree algorithms (C4.5) when applied to data with missing attribute values; and produces substantially more compact decision trees than those obtained by C4.5 and its variants.

2. Attribute Value Taxonomies, Partially Specified Attribute Values and Partially Specified Instances

Taxonomies (also known as concept hierarchies) are among particularly common and useful class of ontologies. A taxonomy is specified by a collection of names (types or concepts) and a set of type-subtype relations. We focus on taxonomies defined over values of an attribute, namely attribute-value taxonomies (AVT). In what follows, we define AVT, introduce the notions of a partially missing value (relative to an AVT), and a partially specified instance (relative to the AVTs associated with the attributes used to describe instances).

An Attribute Value Taxonomy (AVT) associated with an attribute A, AVT(A) is a tree rooted at A. The set of leaves of the tree, *Leaves*(AVT(A)), corresponds to the set of possible *primitive values* of A. The internal nodes of the tree correspond to *abstract values* of attribute A. The *arcs* of the tree correspond to ISA *relationships* between attribute values that appear in adjacent levels in the tree. Figure 1 shows an example of an AVT for the *color* attribute. The set of abstract values at any given level in the tree form a partition of the set of values at the next level (and hence, the set of primitive values of A). More generally, it is possible to define *cuts* through the tree that correspond to a partition of *Leaves*(AVT(A)). For example, in Figure 1, the nodes at level 1 i.e., red, green, blue define a partition of attribute values that correspond to nodes at level 2 (and hence, a partition of all primitive values of the `color' attribute). Similarly, the *cut* corresponding to {red, green, sky blue, dark blue, navy blue, light blue} defines a partition of the primitive values of the `color' attribute.

When each attribute has a single AVT, we will use $T=\{T_1, T_2, ..., T_n\}$ (where $T_i =$ AVT (A_i)) to represent the set of the corresponding AVTs.

To make the notion of partially specified instances more precise, we define several operations on an AVT taxonomy T_i associated with an attribute A_i.

(1) *prim* (T_i)=*Leaves*(AVT(A_i));

(2) *depth*(T_i , $v(A_i)$) returns the length of the path from root to an attribute value $v(A_i)$ in the taxonomy;

(3) *leaf*(T_i, $v(A_i)$) return a Boolean value indicating if $v(A_i)$ is a leaf node in T_i =AVT(A_i), that is if $v(A_i) \in$ *Leaves*(T_i).

With respect to an AVT, a (completely) missing value of an attribute A corresponds to the root of AVT(A). We say that an attribute A is fully specified in an instance with respect to AVT(A) when the value of attribute A is a primitive value of A, that is, $A \in$ *Leaves*(AVT(A)). We say that the value of an attribute A is partially specified (or equivalently, partially missing) when its value is not one of the primitive values of A. Thus, we can have instances specified at different levels of precision resulting in *partially specified instances.*

An instance j is expressed as a tuple $I_j=(v_1^{(j)}, v_2^{(j)},..., v_n^{(j)})$ where each attribute A_i has a corresponding AVT T_i. I_j is:

- a completely specified instance, when $\forall i \quad v_i^{(j)} \in prim(T_i)$
- a partially specified instance when one or more of its attribute values are not primitive:

$$\exists v_i^{(j)} \in I_j, \ depth(T_i, v_i^{(j)}) \geq 0 \wedge \neg leaf(T_i, v_i^{(j)})$$

Thus, a partially specified instance is an instance in which at least one of the attributes is partially specified. For example, consider a set of objects described in terms of the attributes *color* and *shape* which have the AVTs

shown in Figure 1. The instance (Light Blue, Triangle), is fully specified with respect to the corresponding AVTs. Some examples of partially specified instances are (Blue, Polygon), (Dark Blue, Polygon), and (Blue, Square).

3. AVT-guided Decision Tree Learning

Learning from partially specified instances can be viewed as a more general case of the problem of learning from instances that have missing attribute values. Hence, we consider approaches to learning from partially specified instances based on techniques for handling missing attribute values (Quinlan, 1992) and learning from AVT and fully specified instances (Zhang et al., 2002). The proposed algorithm accepts as input, a user-supplied AVT and a data set of (possibly) partially specified instances, and produces as output, a classifier for assigning partially specified instances to one of several mutually exclusive classes.

The proposed AVT-guided Decision Tree Learning algorithm (AVT-DTL) is a top-down multi-level AVT-guided search in decision tree hypothesis space. AVT-DTL has a bias in favor of splits based on more abstract attribute values (i.e., those that appear closer to the roots of the corresponding AVTs). Thus, AVT-DTL has to choose not just a particular attribute, but also an appropriate level of abstraction in the AVT. To facilitate description of AVT-DTL, we introduce the following notations:

- $A=\{A_1, A_2, ..., A_n\}$ is an ordered sequence of attribute names. Let $T=\{T_1, T_2, ..., T_n\}$ be the corresponding set of AVTs.
- $C=\{C_1, C_2, ..., C_m\}$ is a set of mutually disjoint class labels.
- $\psi(v, T_i)$ is the set of descendents of a node corresponding to value v in a taxonomy T_i
- *Children* (v, T_i) is the set of all children – that is, direct descendents of a node corresponding to value v in a taxonomy T_i
- $\Lambda(v, T_i)$ is a list of ancestors, including the root, for v in T_i
- $\sigma_i(v, S)$ is the frequency count of value v of attribute A_i in a training set S
- $Counts(T_i)$ is a *tree of counts* corresponding to nodes in T_i
- $P_S=\{p_1^{(S)}, p_2^{(S)}, ..., p_n^{(S)}\}$ is a pointing vector (also called AVT frontier), for a set of instances S where $p_i^{(S)}$ is a pointer to a value in T_i of attribute A_i
- $\Phi(P_S)$= *true* if and only if $\forall p_i^{(S)} \in P_S$, and $\psi(p_i^{(S)}, T_i) = \{\}$

During learning, *if the candidate split is based on a node in the AVT taxonomy that is a descendent of the value of the attribute in an instance*, the fractional counts corresponding to the different branches are computed from statistics gathered from the rest of the data set. *Otherwise*, the partially specified instance is treated as though it is a fully specified instance.

AVT-DTL works top-down starting at the root of each AVT and builds a decision tree that uses the most abstract attribute values that are sufficiently informative for classifying the training set consisting of partially specified instances. The AVT-DTL algorithm performs a AVT-guided hill-climbing search in a decision tree hypothesis space. It is straightforward to extract classification rules from the set of pointing vectors associated with the leaf nodes of the decision tree constructed by AVT-DTL.

The AVT-DTL algorithm consists of the following steps:

(a) Computation of the counts based on the given AVT (Figure 2a)

(b) Construction of the decision Tree based on the Counts (Figure 2b)

Figure 2a: Computation of counts based on AVT

1. Create a root node, set sample set to S, and set $P_S=\{A_1, A_2, ..., A_n\}$ so that elements of P_S point to the roots of the corresponding AVTs $T_1, T_2, ..., T_n$.
2. Initialize frequency counts for AVT by scanning training samples:

a) Accumulate the frequency counts associated with each value of each attribute based on the values that appear in instances in S. Thus for each $T_i \in T$, we compute $\sigma_i(v, S)$ associated with all attribute values v that correspond to nodes in T_i.

b) For each $T_i \in T$ update the counts associated with ancestors of nodes in T_i which received non-zero counts as a result of step (a) by propagating the counts up from each such node v to its ancestors. Let $Counts(T_i)$ be the resulting counts.

c) For each $T_i \in T$, and each partially specified attribute value v in each instance $I_j \in S$, calculate the fractional counts recursively for all descendents of v in T_i based on $Counts(T_i)$ and update $Counts(T_i)$. That is, for each d in $\psi(v, T_i)$, update $\sigma_i(d, S)$ as follows:

$$\sigma_i(d,S) \leftarrow \sigma_i(d,S)\left(1+\frac{1}{\sum_{d\in Children(v,T_i)}\sigma_i(d,S)}\right)$$

For example, Figure 3 shows a pointing vector that points to two high-level attribute values *blue* and *polygon* in the two taxonomies of color and shape. If this pointing vector appears in the leaf node with class label + in the decision tree, the corresponding rule will be: If (*Color=blue & Shape=polygon &…*) then *Class* =+.

Figure 2b: Constructing the decision tree

***Tree-Build*(*S*, P_S)**

1. If each instance in S has same label C_i, return(C_i) else if $\Phi(P_s)$ is true then assign majority label to generate a leaf node in decision tree (Stopping Criterion).
2. For each pointer $p_i^{(S)}$ in P_S do:
   (a) Check each partially specified instance with partially specified value v for attribute A_i with the pointer $p_i^{(S)}$. If $depth(T_i, p_i^{(S)}) < depth(T_i, v)$, then the partially specified instance is treated as though it were a fully specified instance and sent along the appropriate branch of the decision tree rooted at $p_i^{(S)}$. Otherwise (that is, if $depth(T_i, p_i^{(S)}) \geq depth(T_i, v)$), replace v probabilistically with d (an element of $Children(v, T_i)$) according to the distribution of counts associated with Children(v, T_i).

$$\Pr(d) = \frac{\sigma(d, T_i)}{\sum_{e \in Children(v, T_i)} \sigma(e, T_i)}.$$

   (b) Calculate the entropy of the set S based on the partition by $Children(p_i^{(S)}, T_i)$ in S.
3. Choose the best pointer $p_a^{(S)}$ in P_S to partition S that yields the maximum information gain.
4. Partition the sample set S into subsets $S_1, S_2, \ldots S_{|Children(p_a^{(S)}, T_i)|}$ by using attribute values in $Children(p_a^{(S)}, T_i)$.
5. Extend P_S to obtain a new pointing vector corresponding to each of the subsets $S_1, S_2, \ldots S_{|Children(p_a^{(S)}, T_i)|}$ by replacing the pointer $p_a^{(S)}$ with the value of attribute A_i in the corresponding subset S_j ($1 \leq j \leq |Children(p_a^{(S)}, T_i)|$).
6. For each $1 \leq j \leq |Children(p_a^{(S)}, T_i)|$, Do $Tree - Build(S_j, P_{S_j})$.

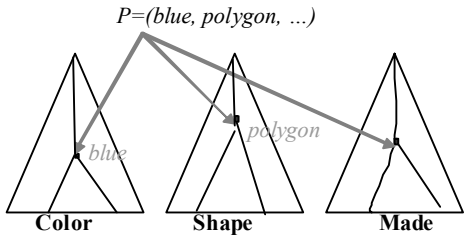

Figure 3. Illustrations of a Pointer Vector P

4. Experiments and Results

4.1 Experiments

Several experiments were performed to explore the performance of AVT-DTL on two data sets from UC Irvine Repository - the *Mushroom Toxicology* data set, and *Nursery* data set.

The *Mushroom Toxicology* dataset consists of 8124 instances, described in terms of 22 nominal attributes. Of the 8124 instances, 4208 are labeled *edible* (E) and the remaining 3916 are labeled *poisonous* (P). The twelfth attribute value is missing in 2480 of the 8124 instances. We adopted the AVT for this data set from Taylor et al. (1997). Of the 22 attributes, 17 attributes have AVTs with more than 3 levels in depth.

The *Nursery* dataset contains 12960 instances. Instances are described in term of 8 nominal attributes. Each instance belongs to one of 5 classes. Fairly obvious groupings of values yield AVT for 6 of the 8 attributes. [1]

In each case, the error rate of the resulting decision tree was estimated using 10-fold cross-validation. The reported size of the decision tree corresponds to the average size computed from the 10 experiments. The experiments compare the decision trees constructed using

a) AVT-DTL with and without pruning,

b) C4.5 with and without pruning, and

c) C4.5 with 'subsetting' (option '-s') with and without pruning.

The `subsetting' option allows C4.5 to consider splits based on subsets of attribute values (as opposed to single values) along each branch.

In the case of C4.5, an instance with a missing attribute value at a node in the tree is converted into a set of *fractional* instances corresponding to possible values of the attribute. The weight of the fractional instance assigned to a branch is set equal to the estimated probability of observing the corresponding value at that branch (Quinlan, 1993). Note that the approach used by AVT-DTL to deal with partially specified attribute values is a natural generalization of this method for dealing with missing values.

Whenever pruning was used with C4.5, it involved sub-tree raising with the default setting of the confidence level at 25%. Pruning in AVT-DTL involves pre-pruning which simply avoids splitting a node further if all the candidate splits are not significantly different from random splits at a confidence level of 95%.

In order to explore the performance of AVT-DTL on data sets with different percentage of totally missing or

[1] The AVTs used in our experiments have been submitted to the UC-Irvine repository

partially missing attribute values, data sets with a pre-specified percentage (0%, 5%, 10%, 20%, 30%, 40%, or 50%, *excluding* the missing values in the original data set) of (totally) missing attribute values were generated by assuming that the missing values are uniformly distributed on the attributes as well as instances. Datasets were generated for each choice of percentage of missing values. From a data set D_m with totally missing values, a data set D_p of partially missing values was generated as follows: Let (n_l n_{l-1} … n_0) be the path from the leaf node n_l to the root n_0 of the AVT corresponding to a (totally) missing attribute value; Select one of the nodes along this path, excluding n_l (with uniform probability); Read the corresponding attribute value from the AVT and assign it as the value of the corresponding attribute. Note that the selection of the root of the AVT would result in a totally missing attribute value. Thus, corresponding to every instance $I \in D_m$ that has a missing value for some attribute (say the jth attribute), there is a corresponding instance in D_p in which the jth attribute is partially missing.

4.2 Results

AVT-DTL yields significantly lower error rates than C4.5 on data sets with substantially large percentage of partially missing attribute values.

Table 1 shows the resulting error estimates along with the corresponding 90% confidence intervals. Note that all algorithms except for AVT-DTL treat a partially missing attribute value as a totally missing attribute value during decision tree construction because they do not utilize AVTs.

In the case of *Mushroom Toxicology* data, the error rate of AVT-DTL (5.58% without pruning and 6.51% with pruning) is substantially smaller than error rates of each of the variants of C4.5 (which range from 15.92% in the case of C4.5 with subsetting but no pruning to 24.04% in the case of C4.5 with pruning) when *half* (50%) of the attribute values (excluding the attribute values that were originally totally missing in the data set) are partially

Table 1: The error rate estimates for AVT-DTL and C4.5 (with different options) with different percentages of partially and totally missing values. We use the following abbreviations: **C4.5** – standard decision tree learning algorithm without pruning; **C4.5P** – C4.5 with pruning; **C4.5S** – C4.5 with subsetting; **C4.5SP** – C4.5 with subsetting and pruning; **AVT-DTL(T)** – AVT-DTL without applied to data sets with totally missing values; **AVT-DTL(TP)** – AVT-DTL with pruning applied to data sets with totally missing values. **AVT-DTL(Y)** – AVT-DTL without pruning applied to data sets with partially missing values; **AVT-DTL(YP)** – AVT-DTL with pruning applied to data sets with partially missing values.

Percentage of totally or partially missing values		0%	5%	10%	20%	30%	40%	50%
		% Error rates estimated using 10-fold cross validation with 90% confidence interval						
Mushroom toxicology Data	C4.5	0	0.99 ± 0.64	2.01±0.90	4.22±1.27	8.08±1.73	14.21±2.21	22.95±2.69
	C4.5P	0	1.03±0.62	2.16±0.91	5.31±1.42	12.46±2.09	16.26±2.33	24.04±2.70
	C4.5S	0	0.99±0.64	1.68±0.81	2.87±1.06	7.14±1.63	10.75±1.96	15.92±2.32
	C4.5SP	0	0.99±0.64	1.90±0.86	3.70±1.19	8.90±1.80	12.60±2.11	18.80±2.48
	AVT-DTL(T)	0	0.94±0.6	1.72±0.82	2.99±1.08	8.73±1.79	12.08±2.07	20.36±2.55
	AVT-DTL(TP)	0	0.95±0.61	1.97±0.87	3.84±1.21	9.78±1.88	13.62±2.17	22.45±2.64
	AVT-DTL(Y)	0	0.47±0.43	1.42±0.75	2.18±0.91	3.19±1.11	4.09±1.24	5.58±1.45
	AVT-DTL(YP)	0	0.52±0.45	1.72±0.82	2.59±1.00	3.94±1.23	4.87±1.36	6.51±1.56
Nursery Data	C4.5	3.34±0.90	10.03±1.51	14.77±1.78	22.11±2.08	30.01±2.30	36.32±2.41	41.79±2.47
	C4.5P	5.51±1.14	11.12±1.58	16.27±1.85	24.61±2.16	32.64±2.35	38.49±2.44	43.72±2.49
	C4.5S	0.96±0.49	5.75±1.17	11.08±1.57	20.67±2.03	28.59±2.27	34.87±2.39	41.12±2.47
	C4.5SP	1.84±0.68	7.93±1.35	13.65±1.72	22.96±2.11	30.61±2.31	36.92±2.42	43.37±2.49
	AVT-DTL(T)	1.21±0.55	5.86±1.18	11.85±1.62	21.34±2.05	29.14±2.28	34.69±2.39	42.26±2.48
	AVT-DTL(TP)	2.89±0.84	7.94±1.35	13.35±1.70	23.01±2.11	30.17±2.30	36.18±2.41	43.32±2.49
	AVT-DTL(Y)	1.21±0.55	3.10±0.87	6.32±1.22	10.89±1.56	20.17±2.01	27.11±2.23	32.75±2.35
	AVT-DTL(YP)	2.89±0.84	3.62±0.93	7.38±1.31	12.70±1.67	21.93±2.08	27.69±2.25	33.17±2.36

missing. Qualitatively similar results are obtained in the case of *Nursery* data as well. This is consistent with the fact that AVT-DTL constructs decision trees that favor tests on abstract attributes that appear close to the roots of the AVTs. Consequently, the partially missing attribute values which correspond to nodes in the AVT that are further away from the root than the attribute tests chosen in the decision tree have no adverse impact on error rate.

AVT-DTL (either with or without pruning) slightly outperforms C4.5 with or without pruning (but no subsetting), with respect to estimated error rates over a broad range of percentage of totally missing attribute values. However, C4.5 with subsetting sometimes yields slightly lower error rates than AVT-DTL. This may be explained by the fact that C4.5 with subsetting is less constrained than AVT-DTL in its choice of tests.

AVT-DTL produces compact, easy-to-comprehend classifiers

Table 2 summarizes the results of experiments comparing AVT-DTL with several variants of C4.5 on the original *MushroomToxicology* and *Nursery* data (without any partially missing values). AVT-DTL *even without pruning* yields substantially smaller trees compared to standard C4.5 (with or without pruning) and C4.5 with subsetting but no pruning. Furthermore, the smaller tree size is achieved by AVT-DTL without any deterioration in predictive accuracy. C4.5 (with or without pruning) yields a tree with 31 nodes which correspond to 26 rules each of which includes tests on attribute values corresponding to the lowest levels of the AVTs. In contrast, AVT-DTL produces a decision tree with 16 nodes which corresponds to 10 rules in which tests correspond to attribute values at the higher levels of the AVTs. C4.5 with 'subsetting' but without pruning produces trees that are substantially larger than those obtained by AVT-DTL. It is only when C4.5 uses both subsetting and pruning that the resulting trees are comparable in size (as measured by the number of nodes or the number of leaves) or smaller than those obtained by AVT-DTL.

Table 2. Comparison of tree size and number of leaves in decision trees built by variants of C4.5 and AVT-DTL (with pruning) in original data set. **C4.5** – standard decision tree learning algorithm without pruning; **C4.5P** – C4.5 with pruning; **C4.5S** – C4.5 with subsetting; **C4.5SP** – C4.5 with subsetting and pruning;. **AVT-DTL(Y)** – AVT-DTL without pruning.

	MUSHROOM		NURSERY	
	TREE SIZE	NUMBER OF LEAVES	TREE SIZE	NUMBER OF LEAVES
C4.5	31	26	944	680
C4.5P	31	26	511	359
C4.5S	20	12	455	272
C4.5SP	15	9	327	168
AVT-DTL(Y)	16	10	298	172

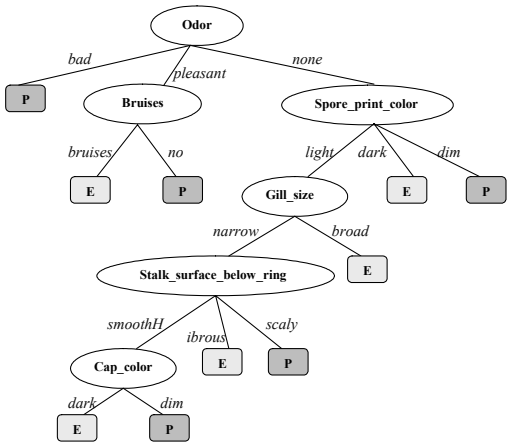

Figure 4. A decision tree learned by AVT-DTL for the *Mushroom Toxicology* data.

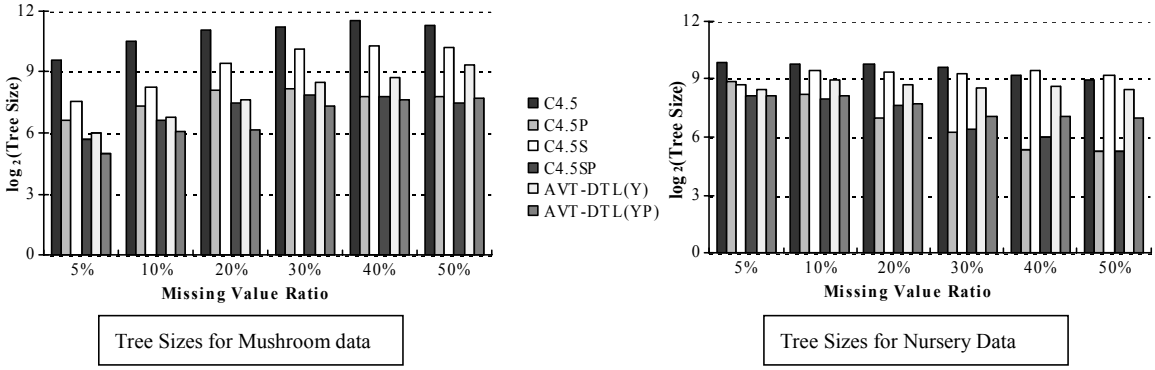

Figure 5. Comparison of the sizes of the induced decision trees with different percentages of partially missing values. Note that the Y axis shows the logarithm of the tree size to base 2. We use the following abbreviations: **C4.5** – standard decision tree learning algorithm without pruning; **C4.5P** – C4.5 with pruning; **C4.5S** – C4.5 with subsetting; **C4.5SP** – C4.5 with subsetting and pruning;. **AVT-DTL(Y)** and **AVT-DTL(YP)** – AVT-DTL without and with pruning respectively.

This can be explained by the fact that C4.5 with subsetting is less constrained in the choice of splits at each node compared to AVT-DTL (whose splits are constrained by the AVT). The results are qualitatively similar in the case of the *Nursery* data as well.

A representative decision tree generated by AVT-DTL is shown in the Figure 4. AVT-DTL is quite effective in selecting tests based on attribute values from the higher levels of AVT (and hence correspond to more abstract attribute values) e.g., ***if*** (Odor=*none*) *&* (Spore_print_color = *dark*) ***then*** *edible*, and *dark* is an abstract attribute value corresponding to a set of colors: {*black, brown, chocolate*} (See Figure 4). Examination of decision trees generated using C4.5 shows that this rule in fact summarizes three separate rules generated using C4.5.

Figure 5 compares AVT-DTL with C4.5 variants in terms of the size of the decision trees generated from data with different percentages of partially missing attribute values. AVT-DTL without pruning generates trees that are smaller than those generated by C4.5 (with or without pruning) and C4.5 with subsetting but no pruning. Decision trees generated by AVT-DTL with pruning are comparable to those generated by C4.5 with subsetting and pruning.

5. Summary and Discussion

5.1 Summary

In this paper, we have presented AVT-DTL, an algorithm for learning decision trees using attribute value taxonomies from partially specified data in which different instances have attribute values specified at different levels of precision. Our approach extends the ontology-based decision tree algorithm proposed in (Zhang et al., 2002) to learn from, and classify partially specified instances. The technique used in AVT-DTL for handling partially specified attribute values is a generalization of an existing approach to dealing with missing attribute values in decision tree construction and classification.

Experimental results presented in the paper show that:

(1) AVT-DTL algorithm is able to learn robust high accuracy classifiers from data sets consisting of relatively high percentage of partially specified instances.

(2) AVT-DTL algorithm yields substantially more compact yet high accuracy decision trees than standard decision tree learning algorithm (C4.5) and its variants that do not use subsetting or utilize attribute value taxonomies to guide decision tree construction when applied to data sets with fully specified instances as well as data sets with relatively high percentage of missing attribute values.

5.2 Related Work

Learning from AVT and data has received some attention in the machine learning literature (Núñez, 1991; Quinlan, 1992; Dhar & Tuzilin, 1993; Almuallim, et al., 1995; Taylor et al., 1997; Han & Fu, 1996; Cheung et al., 2000; Chen et al., 2002; desJardins et al., 2000; Zhang et al., 2002). However, this work has not resulted in AVT-based algorithms for learning classifiers from partially specified data.

There has been some work on methods for handling partially specified data in the context of data integration from distributed databases (DeMichiel, 1989; Chen & Tseng, 1996; McClean et al., 2001). However, this work has not addressed the problem of learning classifiers from partially specified data.

Attribute value taxonomies allow the use of a hierarchy of *abstract* attribute values (corresponding to nodes in an AVT) in building classifiers. Each abstract value of an attribute corresponds to a *set* of primitive values of the corresponding attribute. The classification rules constructed by the RIPPER rule learning algorithm proposed by Cohen (1996) utilize tests for membership in attribute-value sets. C4.5 with subsetting option can also be seen as working with set-valued attributes. However, the attribute value sets considered by these algorithms are not constrained by any AVT (other than the default single level taxonomy with a "don't care" value as the root and primitive values at the leaves). An unconstrained search through candidate subsets of values of each attribute during the learning phase might, at first glance, result in more compact classifiers (e.g. when compactness is measured in terms of the number of nodes in a decision tree) than those produced by AVT-guided learning algorithms such as AVT-DTL. However, in the absence of the structure imposed over sets of attribute values used in constructing the classifier, specifying the outcome of each test (outgoing branch from a node in the decision tree) requires enumerating the members of the set of values corresponding to that branch. Thus, each rule extracted from a decision tree produced by C4.5 with subsetting is a conjunction of disjunctions, making the resulting classifiers difficult to interpret. This is also true of rules generated by RIPPER using set-valued attributes. In contrast, the rules that utilize abstract attribute values from an AVT are much more compact because each attribute value subset has a name that needs to be specified only once – in the AVT, and the attribute value sets considered are constrained by the tree structure of the AVT. Consequently, the rules generated from trees produced by AVT-DTL are compact, and easy to interpret because each rule is a simple conjunction of conditions of the form (*attribute=value*). Because algorithms like RIPPER and C4.5 with subsetting have to search the set of candidate value subsets for each attribute under consideration while adding conditions to a rule or a node to trees, they are computationally more demanding than AVT-DTL. Furthermore, neither C4.5 with subsetting nor

RIPPER is equipped to build classifiers from partially specified data. Lastly, many scientific applications require users to be able to explore a given data set using alternative AVTs which reflect different ontological commitments or different ways of conceptualizing a domain. Unlike C4.5 with subsetting and RIPPER, AVT-DTL can utilize a user-supplied AVT to construct classification rules that are expressed in terms of abstract attribute values that are specified by the AVT.

5.3 Future Work

Some directions for future work include:

(1) Development AVT-based variants of other machine learning including Bayesian networks and Support Vector Machines, and multi-relational data mining algorithms for construction of classifiers from partially specified data.

(2) Development of algorithms that exploit other types of ontologies over attribute values and class labels including ontologies that capture *part-of* relations, combinations of *ISA* and *part-of* relations between attribute values, and multiple competing attribute value taxonomies defined for each attribute.

(3) Further experimental evaluation of AVT-DTL and related learning algorithms on a broad range of data sets in scientific knowledge discovery applications e.g., computational biology.

(4) Integration of AVT-DTL with approaches to automated construction of AVT e.g., (Pereira et al., 1993; Yamazaki et al., 1995).

Acknowledgements

This research was supported in part by a grant from the National Science Foundation (IIS 0219699) and a BISTI award from the National Institutes of Health. The authors wish to thank members of the Iowa State University Artificial Intelligence Laboratory and anonymous referees for their helpful comments on earlier drafts of this paper.

References

Almuallim H., Akiba, Y. & Kaneda, S. (1995). On Handling Tree-Structured Attributes. In: Proceedings of the Twelfth International Conference on Machine Learning. pp. 12-20. Morgan Kaufmann.

Chen, A.L.P. & Tseng, F.S.C. (1996). Evaluating aggregate operations over imprecise data. IEEE Transactions on Knowledge and Data Engineering, 8, 273-284.

Chen, J., Shapcott, M., McClean, S.I. & Adamson, K. (2002). Learning with Concept Hierarchies in Probabilistic Relational Data Mining. In: Proceedings of the Third International Conference on Web Age Information Management. Lecture Notes in Artificial Intelligence 2419: 104-115. Springer-Verlag.

Cheung D., Hwang, H.Y., Fu, A. & Han, J. (2000). An Efficient Rule-based Attribute-Oriented Induction for Data Mining. J. Intelligent Information Systems 15: 175-200

Cohen, W. (1996). Learning Trees and Rules with Set-valued Features. In. Proceedings of the Thirteenth National Conference on Artificial Intelligence. pp. 709-716. AAAI Press.

DeMichiel, L.G. (1989). Resolving database incompatibility: an approach to performing relational operations over mismatched domains. IEEE Transactions on Knowledge and Data Engineering, 4, 485-493.

desJardins, M., Getoor, L. & Koller, D. (2000). Using Feature Hierarchies in Bayesian Network Learning. In: Proceedings of the Symposium on Abstraction, Reformulation, Approximation. Lecture Notes in Artificial Intelligence 1864: 260-270. Springer-Verlag

Dhar, V. & Tuzhilin, A. (1993). Abstract-Driven Pattern Discovery in Databases. IEEE Transactions on Knowledge and Data Engineering 5: 926-938.

Duda, R., Hart, P. & Stork, D. (2000). Pattern Classification New York: Wiley.

Han, J. & Fu, Y. (1996). Exploration of the Power of Attribute-Oriented Induction in Data Mining. U.M. Fayyad, et. al. (eds.), Advances in Knowledge Discovery and Data Mining. MIT Press.

McClean, S.I., Scotney, B.W. & Shapcott, M. (2001). Aggregation of Imprecise and Uncertain Information in Databases. IEEE Transcactions on Knowledge and Data Engineering 13: 902-912

Núñez, M. (1991). The Use of Background Knowledge in Decision Tree Induction. Machine Learning 6:231-250.

Pereira, F., Tishby, N. & Lee, L. (1993). Distributional clustering of English words. In: Proceedings of the Thirty-first Annual Meeting of the Association for Computational Linguistics, pp. 183-190.

Quinlan, J. R. (1992). *C4.5: Programs for machine learning.* San Mateo, CA: Morgan Kaufmann (1992)

Taylor, M., Stoffel, K. & Hendler, J. (1997). Ontology-based Induction of High Level Classification Rules. In: SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery.

Yamazaki, T., Pazzani, M. & Merz, C. (1995). Learning Hierarchies from Ambiguous Natural Language Data. In: Proceedings of the Twelfth International Conference on Machine Learning. pp. 329-342. Morgan-Kaufmann.

Zhang, J., Silvescu, A. & Honavar, V. (2002). Ontology-Driven Induction of Decision Trees at Multiple Levels of Abstraction. In: Proceedings of the Symposium on Abstraction, Reformulation, and Approximation. Lecture Notes in Artificial Intelligence 2371: 316-323 Springer-Verlag.