

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

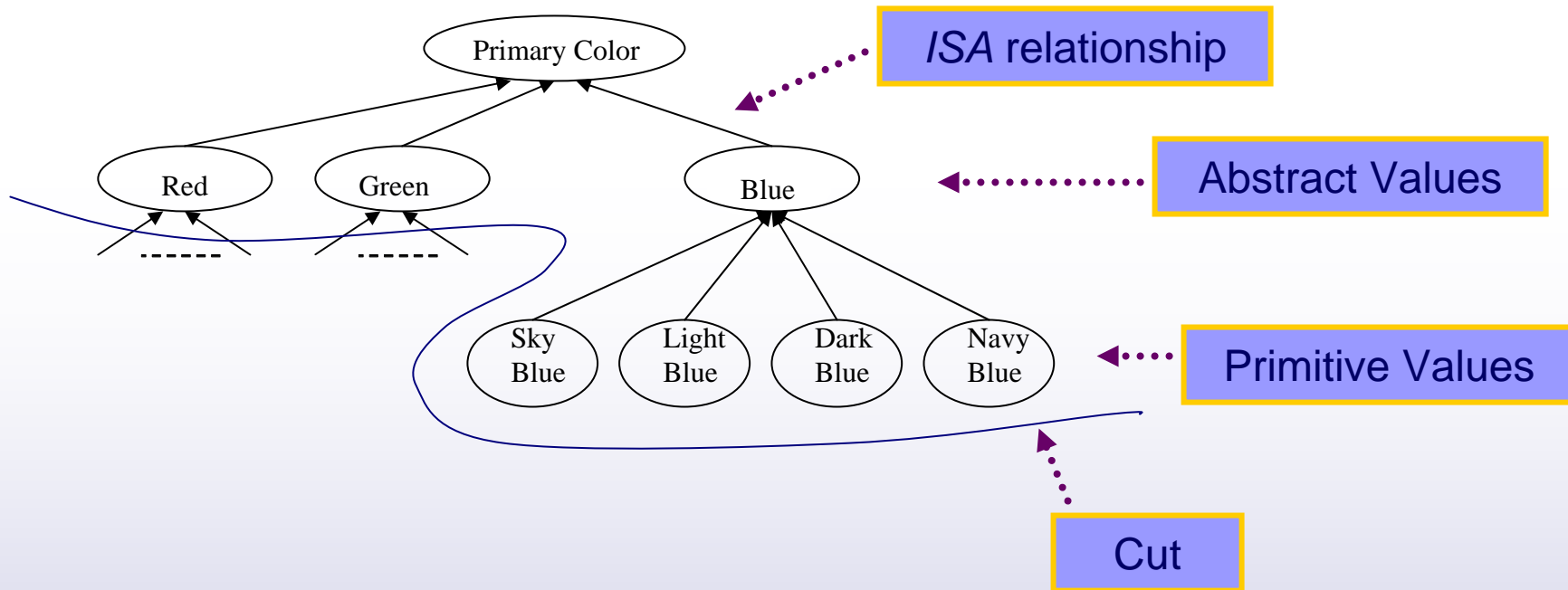
Jun Zhang and Vasant Honavar
Artificial Intelligence Research Laboratory
Iowa State University, USA

This research is sponsored in part by grants from the National Science Foundation (IIS 0219699) and National Institutes of Health (GM066387)

Overview

- Background and Motivation
- AVT-DTL Algorithm
- Experimental Results
- Summary

Attribute value taxonomies (AVT)



Attribute Value Taxonomy (AVT) for *color* attribute

Motivations for learning from AVT and data

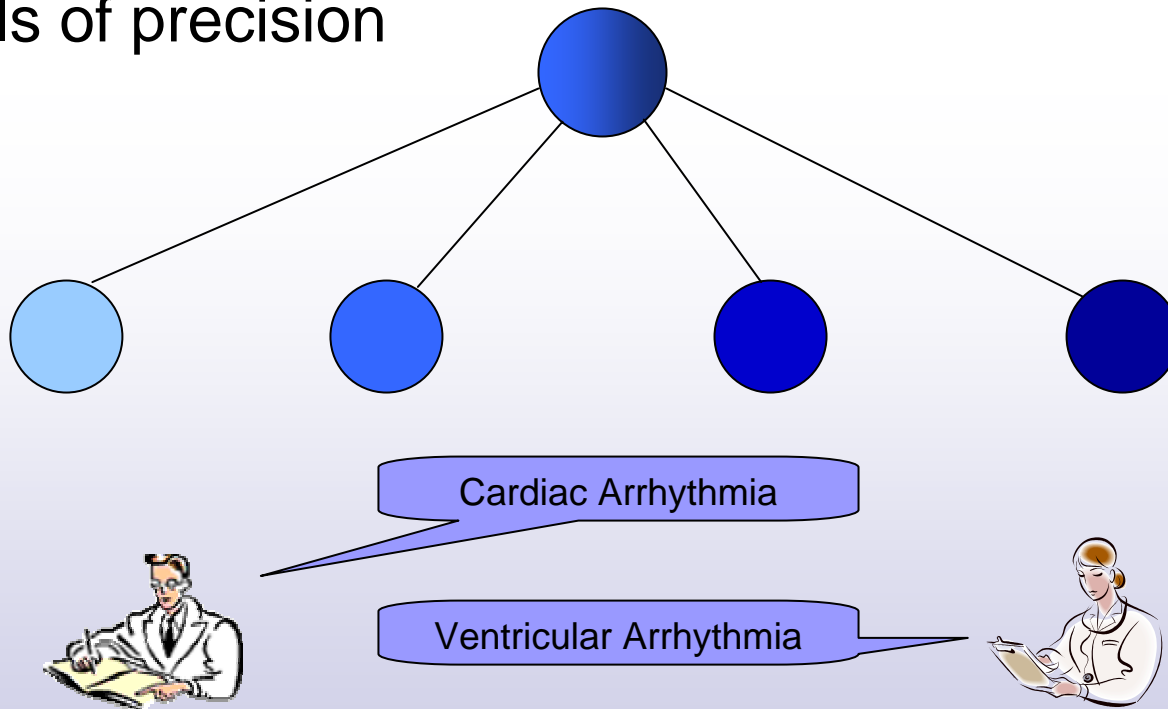
- Preference for simple, comprehensible, yet accurate and robust classifiers
- Classifiers that use *abstract* attribute values often provide simpler descriptions of the data
- When data are limited, statistics estimated from abstract values are often more reliable than statistics estimated from primitive values
- Use of AVT offers a simple way to perform to minimize over-fitting

Motivations for learning from AVT and data

- Different choices of AVT correspond to different *ontological or representational commitments*
- Appropriate choice of AVT can result in simple, robust explanations of the domain
- This requires algorithms that can exploit user-supplied AVT to guide learning

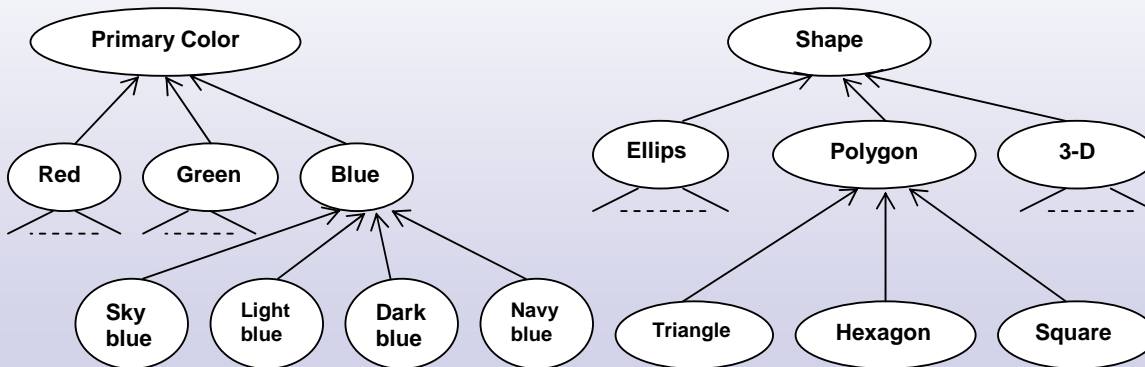
Motivations for learning from partially specified data

- Data gathered by multiple, distributed autonomous entities (e.g. hospitals) are often specified at different levels of precision



Partially Specified Instances

- Completely specified instance: all attribute values are fully specified – correspond to *primitive* attribute values
- Partially specified instance: one or more of the attribute values are partially specified – correspond to *abstract* attribute values
- Partially specified attribute values can be viewed as *partially missing*



Examples of partially specified instances:

(Blue, Polygon)

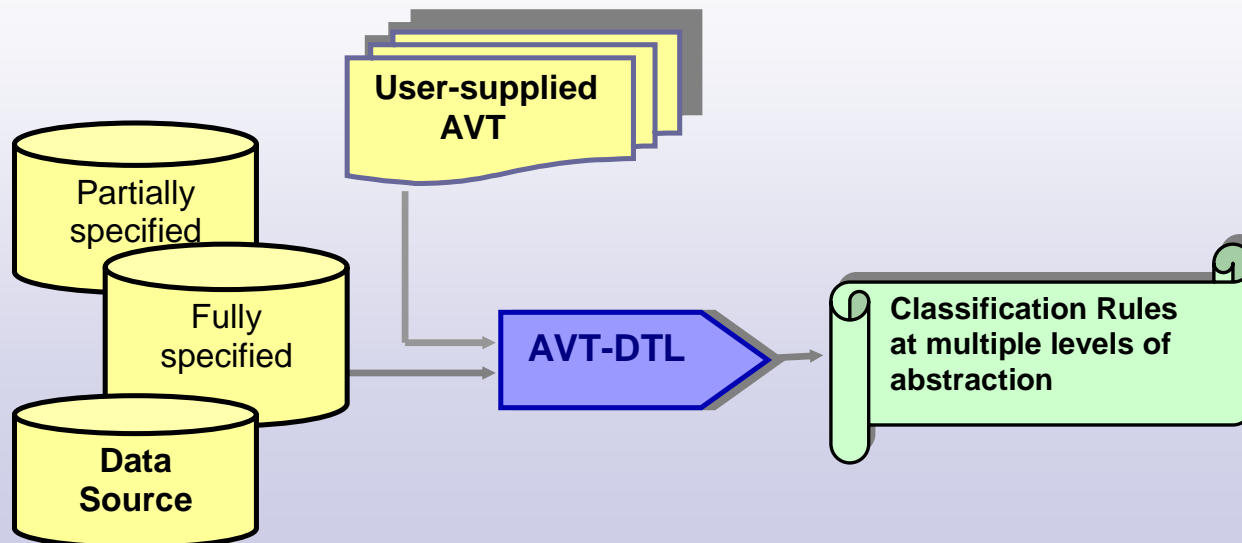
(Dark Blue, Polygon)

(Blue, Square)

Learning from Partially Specified Instances

■ Our Approach:

- Adapt methods for learning from fully specified instances into methods for learning from AVT and fully specified instances
- Generalize techniques for handling *missing* attribute values to work with *partially missing* attribute values



Background

Using set valued features

Quinlan (1992), Cohen (1996)

Using attribute value taxonomies in learning

Núñez, M. (1991), Dhar & Tuzhilin (1993), Han & Fu (1996), Taylor, Stoffel, & Hendler (1997), desJardins, Getoor, & Koller, (2000), Zhang, Silvescu, & Honavar (2002)

Learning attribute value taxonomies

Pereira, Tishby & Lee (1993), Yamazaki, Pazzani, Merz, (1995) ...

Gathering statistics from data with missing attribute values

Quinlan (1992), McClean, Scotney & Shapcott (2001)

Our focus – Learning from attribute value taxonomies and partially specified data

Overview

- Background and Motivation
- AVT-DTL Algorithm
- Experimental Results
- Summary

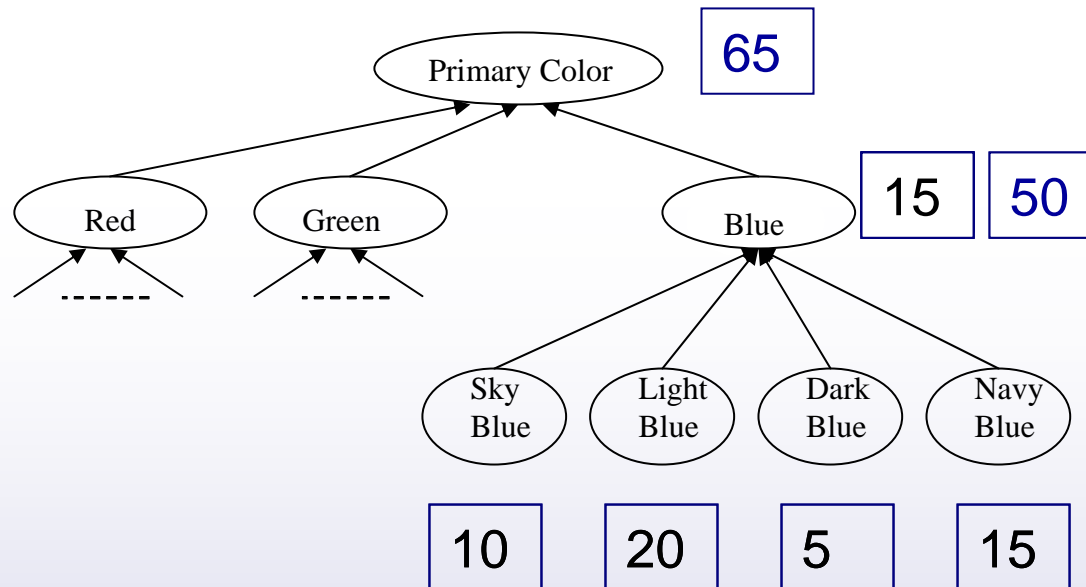
AVT-guided Decision Tree Learning

- AVT-DTL is a top-down multi-level AVT-guided search in decision tree hypothesis space
- AVT-DTL incorporates a *bias* in favor of splits based on more abstract attribute values that are *sufficiently informative* for classifying data
- AVT-DTL uses *fractional counts* of instances that match certain constraints on attribute values to deal with *partially missing* instances

AVT-DTL algorithm

- Dealing with Partially missing attribute values - Computation of AVT based counts
 - Accumulate frequency counts for each value in AVT based on instance values (possibly partially specified)
 - Aggregate the (non-zero) counts upward from each such node to its ancestors
 - Propagate and Update the fractional counts downward for partially specified attribute values

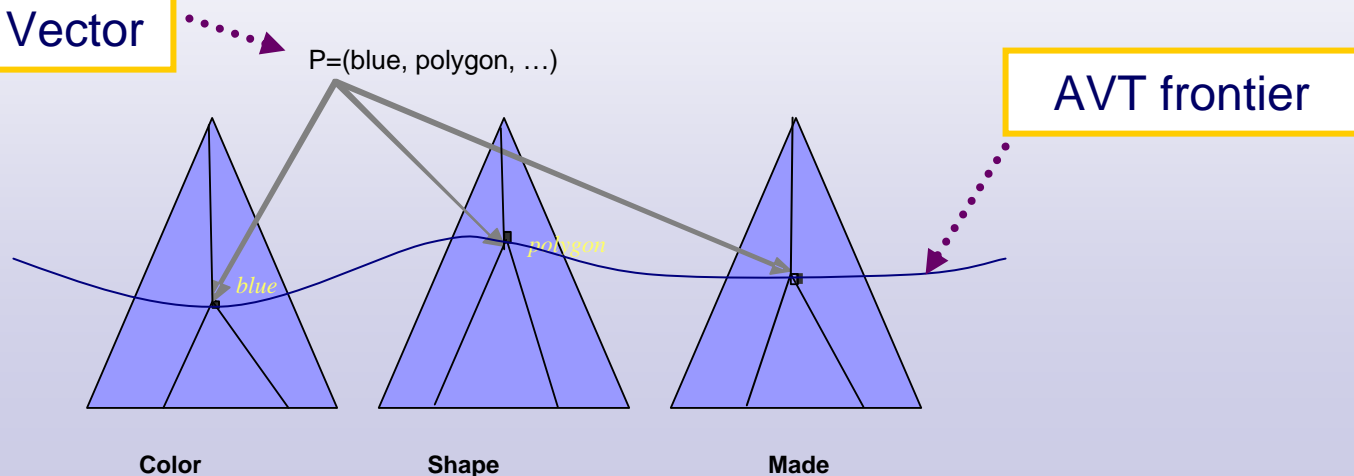
Computation of AVT based counts (Example)



AVT-DTL algorithm

- Standard decision tree algorithm considers partitions the data based on the values each candidate attribute, selecting the most informative attribute at each step
- AVT-DTL has to choose not just the attribute, but also, the appropriate level in the taxonomy which defines the values of the attribute on which to partition the data
- Top-down AVT-guided search selects the
 - Maintain a collection of pointers (*Pointing Vectors*) pointing to nodes (values) in the corresponding AVTs
 - Push the *frontier* defined by the pointing vectors *minimally* to achieve accurate classification of training data
 - Tests chosen correspond to abstract values that are sufficiently informative

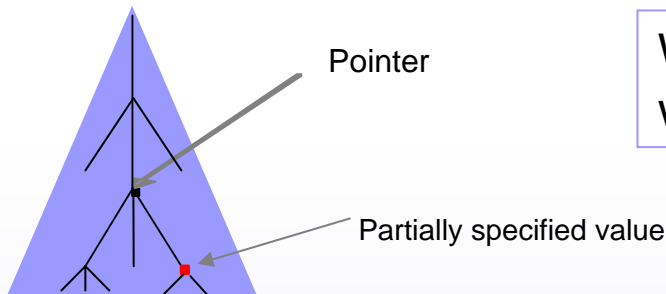
Pointing Vector



AVT-DTL algorithm (Details)

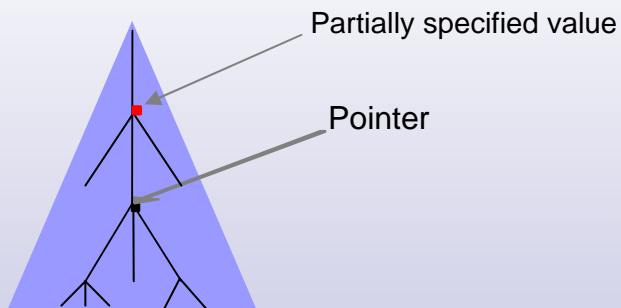
Calculation of entropy

- Case I – the value in an instance is a descendent of the node in the AVT that is pointed to by the current pointer pointing at that AVT



We can treat the value as though it were fully specified

- Case II – the value in an instance is an ancestor, or same of the node in the AVT that is pointed to by the current pointer for that AVT



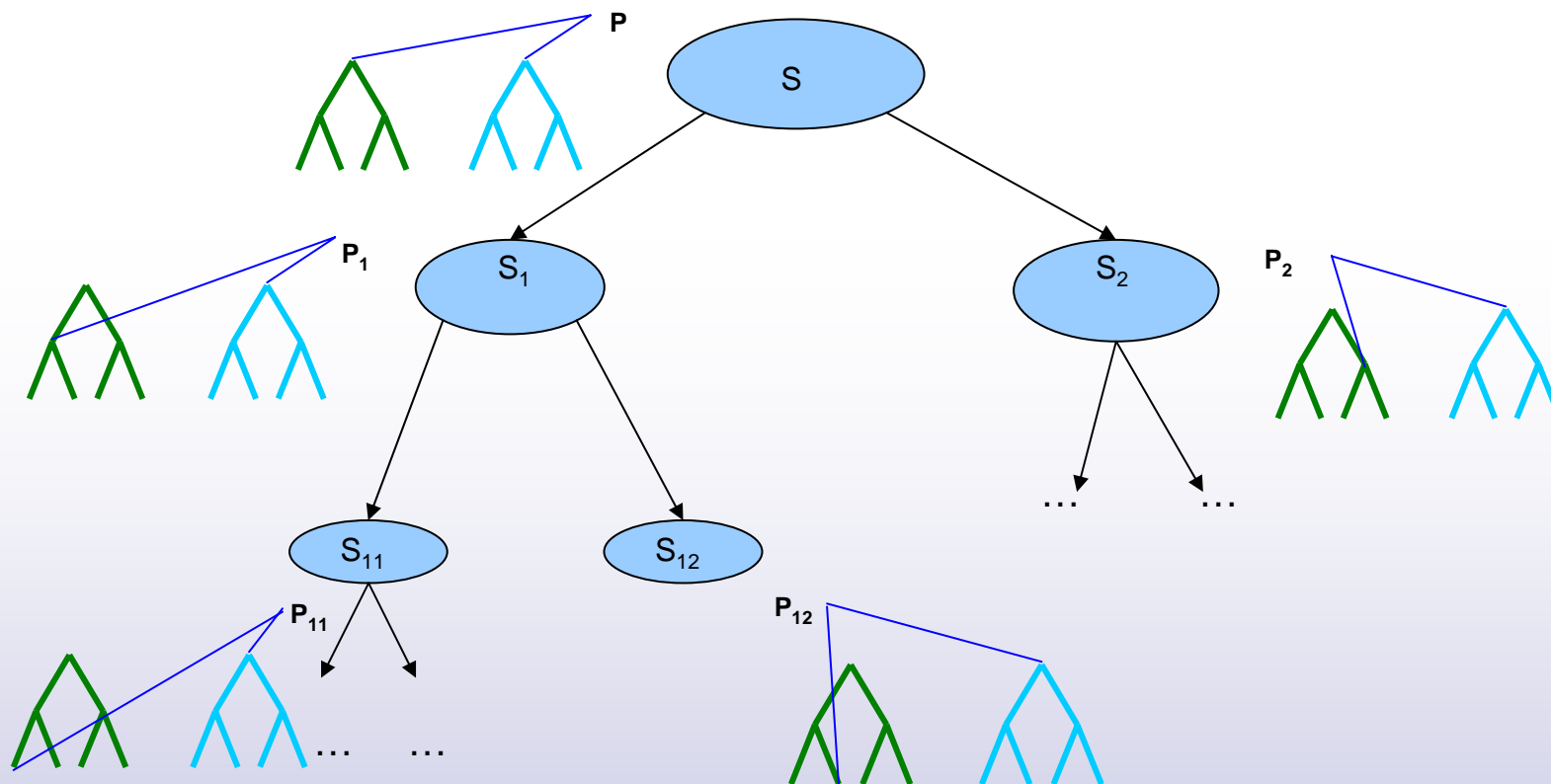
We treat the value as partially missing, filling it in probabilistically based on the pre-computed distribution of values among the descendants and use the filled in value to compute the contribution of corresponding instance to entropy calculation for the split being considered

AVT-DTL algorithm (Details)

Iterate until the desired stopping criterion is met

- Choose the most informative split among all candidate splits based on the current frontier of pointers
- Update the pointers

Building AVT-DT (example)



Decision Tree construction with corresponding Pointing Vectors

Overview

- Background and Motivation
- AVT-DTL Algorithm
- Experimental Results
- Summary

Experiments and Results

■ Questions

- How does AVT-DTL compare with standard DTL (C4.5) and C4.5 with sub-setting option with respect to compactness and comprehensibility of the resulting classifiers?
- How does AVT-DTL compare with standard DTL (C4.5) and C4.5 with sub-setting option with increasing percentage of partially missing instances?

Note – C4.5 with sub-setting searches for splits where each branch corresponds to a subset of attribute values whereas in the case of AVT-DTL, the splits are constrained by the subsets defined by the AVTs.

Experiments and Results

■ Data sets:

- Mushroom Toxicology (17/22 attributes have AVTs), Nursery (6/8 attributes have AVTs)
- Different percentage (5%, 10%, 20%, 30%, 40%, 50%) of partially missing values were generated

■ Comparisons between:

- AVT-DTL w/o pruning
- C4.5 w/o pruning
- C4.5 with 'subsetting' (-s) w/o pruning

■ Error rate and tree size estimated using 10-fold cross validation

AVT-DTL yields substantially lower error rates than C4.5 and C4.5 with sub-setting on partially specified data

P- Pruning, S-Sub-setting, T-Totally missing values, P-Partially missing values

Note: Except by AVT DTL, Partially missing values are treated as though they are totally missing

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	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

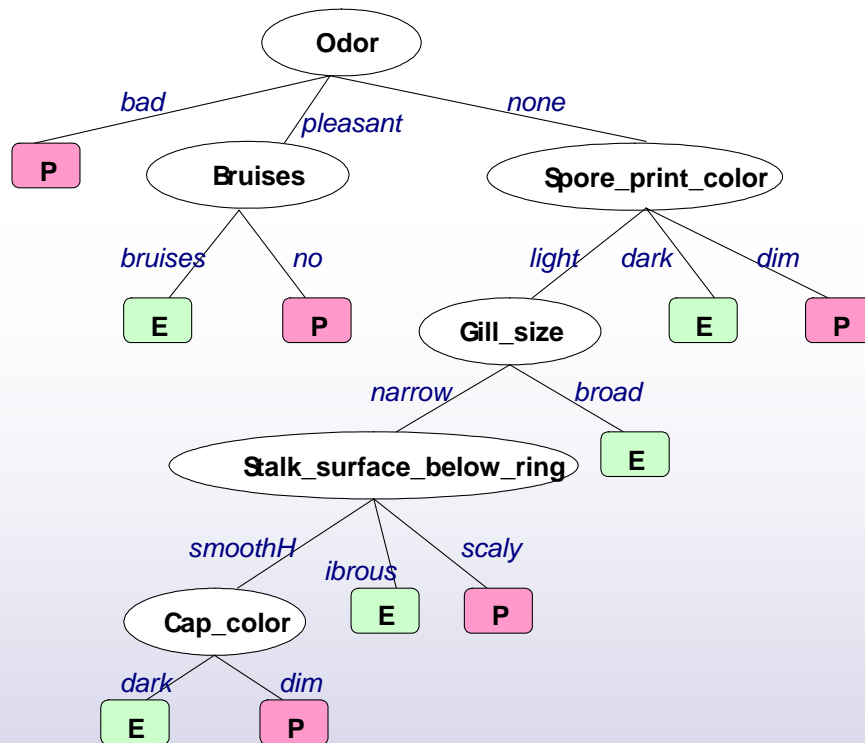
AVT-DTL yields substantially lower error rates than C4.5 and C4.5 with sub-setting on partially specified data

P- Pruning, S-Sub-setting, T-Totally missing values, P-Partially missing values

Note: Except by AVT DTL, Partially missing values are treated as though they are totally missing

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						
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

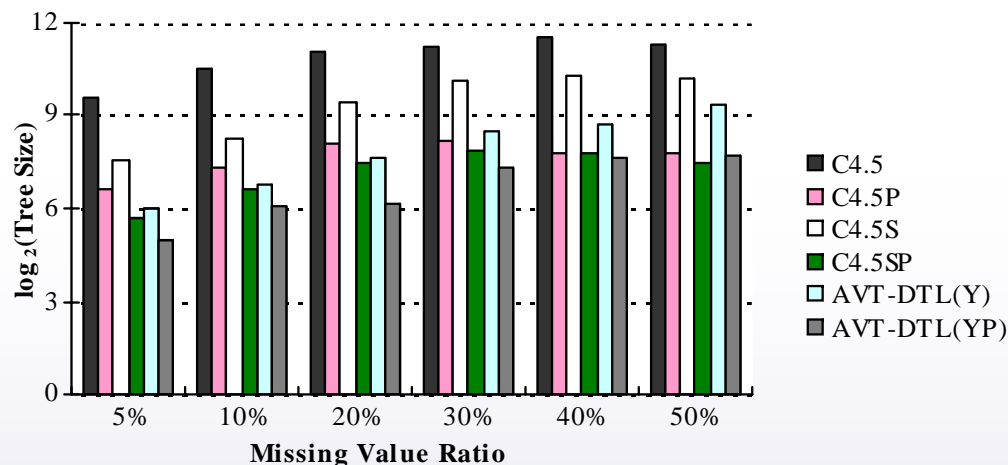
AVT-DTL produces compact, easy-to-comprehend classifiers



	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

Decision tree use tests that branch on abstract attribute values --
Summarize several separate rules generated by C4.5

Comparison of the sizes of the induced trees with different percentage of partially missing values



Tree Sizes for Mushroom data



Tree Sizes for Nursery Data

- AVT-DTL even without pruning generates trees that are smaller than those generated by C4.5 (with/without pruning).
- C4.5 with sub-setting and pruning produces trees that are smaller than those by AVT-DTL

Summary

- AVT-DTL yields robust, compact, high accuracy classifiers from partially specified data
- Unlike C4.5 with sub-setting or RIPPER (with its set valued attributes) AVT-DTL is readily able to exploit user-supplied AVT resulting in more comprehensible classifiers

Future Work

- Development AVT-based variants of other machine learning methods
- Development of algorithms that exploit other types of ontologies over attribute values and class labels, or multiple competing AVTs
- Integration of AVT-DTL with approaches to automated construction of AVT



Thank You!