

# Occam's Razor and a Non-Syntactic Measure of Decision Tree Complexity

Goutam Paul, Department of Computer Science, State University of New York, Albany  
 goutam@cs.albany.edu

## Motivation

### William of Occam (1285-1349)

"*Puritas non est ponenda sine neccesitate*": plurality should not be assumed without necessity

### Machine Learning Interpretation

If two models have the same performance on the training set, choose the simpler

### Justifications

For parsimonious reasons and predictive accuracy

### Decision Trees

- Information Gain - preference to shorter trees
- Shorter trees have better generalization accuracy (e.g. pruning)
- Some evidence (Webb, JAIR 1996) apparently against Occam's Razor

### C4.5 vs C4.5X

- C4.5X has more nodes and branches, and therefore is considered more complex
- Both C4.5 and C4.5X have the same training set accuracy
- C4.5X has better generalization accuracy for some data sets, contrary to Occam's Razor

### Our approach

- A decision tree partitions the instance space into a set of regions
- The C4.5X tree maybe more complex in representation, but the description of its partitions may be less complex
- Occam's Razor holds in our non-syntactic measure of complexity of the partitions

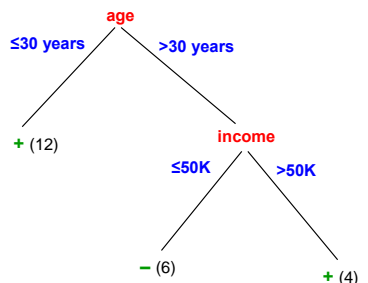
## Partitioning of the Instance Space by Decision Tree

### Data Set

- A set of instances
- Each instance has  $m$  attributes or features and a class label (which is known for the training data set and unknown for the test data set)

### Decision Tree

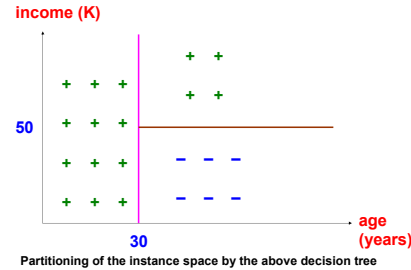
- Each node represents a test of an attribute
- Each branch corresponds to a value of that attribute
- Each leaf node represents a class



An example of a decision Tree. The numbers within the parenthesis represent the no. of instances belonging to the corresponding leaf node.

### The Geometry

- Each training or test instance is a point in an  $m$ -dimensional instance space
- Each test of an attribute places a hyper-plane in the instance space, perpendicular to that attribute axis
- The hyper-planes partition the instance space
- A tree  $T$  with  $n$  leaf nodes partitions the instances into  $n$  subsets:  $B_1, B_2, \dots, B_n$



## Complexity of Decision Tree Partitioning

### Traditional Measure

- Complexity of a tree  $T$  with  $n$  nodes and height  $h$  is traditionally measured as a function of  $n$  and  $h$ , and the simplest and the most popular such measure is

$$C(T) = n$$

### Our Measure using Kolmogorov Complexity

- Any object  $x$  can be represented by a binary string
- Definition of Kolmogorov Complexity of a binary string  $x$ :

$$K(x) = \min\{l(p); p \text{ is a program for } x\}$$

- where  $l(p)$  denotes the length of the program  $p$
- Important features of this complexity:
  - It is representation Independent (Non-syntactic)
  - It measures the complexity of the object, not of the source, unlike information theoretic measures
  - A substring of a string may have more Kolmogorov Complexity than the whole string, and hence a C4.5X tree may have less Kolmogorov Complexity than its C4.5 subtree

- If  $f$  is a recursive bijective mapping from pairs of binary strings to binary strings, the Kolmogorov Complexity of a pair of strings  $x$  and  $y$  is defined as follows: if

$$f(x, y) = \langle x, y \rangle$$

then

$$K(x, y) = K(\langle x, y \rangle)$$

- Extending the above definition to  $n$  such strings  $x_1, x_2, \dots, x_n$ :

$$K(x_1, x_2, x_3, \dots, x_n) = K(\langle \langle \langle x_1, x_2 \rangle, x_3 \rangle, \dots, x_n \rangle)$$

- For a decision tree  $T$ , instead of measuring its complexity by the description of its splits, we measure its complexity by the description of the partitions of the instance space as induced by  $T$ ; if the instances are partitioned into  $n$  subsets (blocks)  $B_1, B_2, \dots, B_n$ , we define the **K-complexity** of  $T$  as

$$C(T) = K(B_1, B_2, \dots, B_n)$$

		Pima	Iris	New Thyroid	Glass	Credit Screening	Hypothyroid
Average Training Set Error (%)	R	15.12	1.7	1.486	0	4.916	0.486
	X	15.12	1.7	1.486	0	4.916	0.486
Average Test Set Error (%)	R	26.812	6.464	7.688	3.057	16.839	1.314
	X	26.478	5.564	7.432	3.483	16.222	1.257
Average Tree Size (No. of Nodes)	R	49.9	8.2	14.2	11	143.01	23.16
	X	72.14	14.34	22.12	15.58	210.87	31.98
Average Number of Partitions	R	19.43	3.7	5.67	5.72	45.17	9.01
	X	20.95	4.06	6.14	6.21	45.71	9.25
Average K-complexity	R	2706.04	502.21	617.28	1280	2807.88	6424.94
	X	2705.98	503.17	616.13	1278.81	2805.96	6422.93
% times R and X partitioning differs		98	92	84	96	100	68
% times Occam's Razor holds		72	76	83	80	83	87

Comparison of Regular and Webb's extended C4.5 (denoted by R & X respectively). For simplicity, instances with missing attribute values are ignored.

## Experiments

### Experimental Set-up

- Same set-up as C4.5X (JAIR 1996)
- 100 random divisions of each dataset into training set (80% instances) and test set (20% instances)

### Measuring the Complexity

- We measure the complexity of partitioning on the test set
- Blocks of instances  $B_1, B_2, \dots, B_n$  concatenated together
- Additional *newline* character as a delimiter between successive blocks
- Linux utility *Compress* to approximate Kolmogorov Complexity
- This is a crude measure, but similar methods have been employed to measure similarity between strings (Cilibiasi and Vitanyi, 2003)

## Discussions

### Occam's Razor

- The idea behind Occam's Razor is the hope that the simpler model will give better predictive accuracy
- In our experiments, we define Occam's Razor as holding, if and only if for two classifiers  $T_i$  and  $T_j$ , either

$$C(T_i) \leq C(T_j) \text{ and } GA(T_i) \geq GA(T_j)$$

or

$$C(T_i) \geq C(T_j) \text{ and } GA(T_i) \leq GA(T_j)$$

- otherwise we say that Occam's razor fails (GA means generalization accuracy)

- Out of 100 experiments, more than 70% support Occam's razor in all the datasets

### Training Set / Test Set Issues

- We have measured the complexity of partitioning on the test set, rather than on the training set, because C4.5 and C4.5X require the same partitioning on the training set and both have the same training set accuracies

- Our complexity measure is generic, and hence we can also measure the complexity based on partitioning on the training set, when, for example, we are considering two different decision tree generating algorithms having different training set partitioning

## Conclusions and Future Work

### Summary

- Occam's Razor says to prefer less complex models
- How to measure the complexity is an important issue
- Ideally, complexity measure should be non-syntactic
- Occam's Razor is typically not violated in our non-syntactic measure of decision tree complexity

### Two Further Applications

- Choosing between two or more decision trees based on the minimum complexity criterion
- Prediction of a new test instance  $x$  as follows: predict the class of  $x$  as the most occurring class label amongst the instances in block  $B_j$ , where

$$j = \operatorname{argmin}_{i=1,2,\dots,n} \{K(B_i U \{x\}) - K(B_i)\}$$

In other words, We can set the class label of  $x$  to be the most occurring class label amongst the instances in the block which has the least increase in complexity, when we add the instance  $x$  to that block

- For prediction, instead of using difference in complexity measure, use of other similarity metrics are also possible, such as the ratio of complexities, or the relative increase in complexity etc.

### Future Plans

- Explore the above applications
- Compare the Minimum Message Length / Minimum Description Length approaches with our work