Published in 1994, *Proceedings of the National Conference on Artificial Intelligence (AAAI-94)*, p. 1463, AAAI/MIT Press, Cambridge, MA

Finding Multivariate Splits in Decision Trees Using Function Optimization

George H. John*
Computer Science Department
Stanford University
Stanford, CA 94305
gjohn@cs.Stanford.EDU

We present a new method for top-down induction of decision trees (TDIDT) with multivariate binary splits at the nodes. The primary contribution of this work is a new splitting criterion called soft entropy, which is continuous and differentiable with respect to the parameters of the splitting function. Using simple gradient descent to find multivariate splits and a novel pruning technique, our TDIDT-SEH (Soft Entropy Hyperplanes) algorithm is able to learn very small trees with better accuracy than competing learning algorithms on most datasets examined.

The process of finding a splitting function at a node of a decision tree is a search problem, and we choose to view it as unconstrained parametric function optimization over the space of hyperplane weight vectors $w \in R^n$. Our objective function is soft entropy, a new continuous approximation to the entropy measure (Quinlan 1986). Soft entropy was chosen for two reasons. First, it is well-established that *entropy* is a good splitting criterion (Buntine & Niblett 1992). Second, *softness* is important to get good generalization in continuous spaces, as shown in Figure 1. Related work is similar overall, but the OC1 algorithm of Murthy *et al.* (1993) uses entropy as a criterion, and Brodley and Utgoff (1992) describe algorithms using error, also a hard splitting criterion.

Figure 1: All four splits shown on the left have equivalent entropy and error. On the right we show the split found by minimizing soft entropy.

The overall learning algorithm is simply the standard TDIDT method (Quinlan 1986). To choose a split at a node, it uses gradient descent to find the hyperplane with minimal soft entropy. To prune the resulting tree, it uses a new pruning technique which we call *iterative re-filtering*, a general regularization algorithm that we are investigating further.

Dataset	Vote	Vote1	Monks2	Monks3
See	Buntine & Niblett		Thrun et al.	
#Train/Test	200/135	200/135	135/432	122/422
SEHpr	**98.5 (3)**	**94.8 (3)**	**100 (9)**	90.7 (3)
SEHp	**98.5 (3)**	90.4 (13)	**100 (9)**	91.2 (5)
SEH	96.3 (15)	85.9 (29)	**100 (9)**	93.8 (9)
OC1p*	95.6 (11)	92.6 (17)	97.9(5)	92.6 (3)
C4.5p*	97 (7)	92.6 (27)	75.9(39)	**100 (9)**
C4.5*	94.8 (12)	92.6 (33)	75.5 (135)	97.2 (17)

Table 1: Test set accuracy and number of nodes (in parentheses) in the induced decision tree for several datasets. "p" indicates use of pruning, "r" indicates re-filtering. "*" indicates several versions were run with different parameter settings, and the best results for each dataset are presented.

Table 1 shows results on various datasets. SEH achieves the best test-set accuracy on all datasets except for Monks3, which was difficult because the algorithm pruned away four of the data points which turned out not to be noise. On the rest of the domains, SEH was able to achieve high accuracy using very small trees.

*Inquiries are welcome. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

Brodley, C. E., and Utgoff, P. E. 1992. Multivariate versus univariate decision trees. Technical Report COINS TR 92-8, Department of Computer Science, University of Massachusetts, Amherst, MA, 01003.

Buntine, W., and Niblett, T. 1992. A further comparison of splitting rules for decision-tree induction. *Machine Learning* 8:75–85.

Murthy, S.; Kasif, S.; Salzberg, S.; and Beigel, R. 1993. OC1: randomized induction of oblique decision trees. In *AAAI-93: Proceedings, Eleventh National Conference on Artificial Intelligence*, 322–327. MIT Press.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Thrun, S. B., et al. 1991. The monk's problems – a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, CMU School of Computer Science.