Submitted to *Advances in Neural Information Processing Systems 7*

Robust Soft-Entropy Neural Network Trees

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

Abstract

We present a new method for the induction of tree-structured recursive partitioning classifiers that use a neural network as the partitioning function at each node in the tree. Our technique is appropriate for pattern recognition tasks with many continuous inputs and a single multivalued nominal output. This paper presents two main contributions: 1) a novel objective function called *soft entropy*, which is used to train each neural net to give the optimal partitioning of the data, and 2) a novel but simple method for removing outliers called *iterative re-filtering*, which boosts performance on many datasets. These two ideas are presented in the context of a single learning system called SENNT-PIRE (Soft Entropy Neural Net Trees with Pruning and Iterative RE-filtering).

1 INTRODUCTION

Recursive partitioning classifiers (Friedman 1977, Breiman, Friedman, Olshen & Stone 1984), or decision trees, are an important nonparametric function representation. Their wide and successful use in fielded applications and their simple intuitive appeal make decision tree learning algorithms an important area of study. In this paper we present a study of decision trees in which a neural network is employed at each node to recursively split the data. Previous objective functions for neural net training (*e.g.*, mean squared error) were found deficient, so we derived a new objective function called *soft entropy* based on the entropy criterion used in early and recent decision tree methods (Morgan & Messenger 1973, Quinlan 1986).

Since optimizing performance on an unseen test set is the real problem we're trying to solve, many authors have proposed *pruning* the decision tree in an attempt to address the bias/variance tradeoff. Pruning involves removing an entire subtree and replacing it with a leaf. Implicit in this operation is the assumption that the input patterns which caused the subtree to be built are locally either unimportant or harmful. We suggest that these patterns might be *globally* unimportant or harmful, and thus the induction algorithm should remove such patterns from the training set and build a new tree. This is a general technique applicable to any classifier: train the classifier, perform some regularization step, see which patterns in the training set it now classifies incorrectly, and retrain with those patterns removed.

The rest of this paper explains the SENNT and PIRE components in more detail, and then gives performance comparisons and discussion.

2 BUILDING SOFT-ENTROPY NEURAL NET TREES

The tree-building algorithm is the common recursive partitioning algorithm – take the whole set of data, partition it somehow into two subsets, and continue partitioning until all subsets are *pure*, that is, all patterns in the subset come from the same class. Each node in our classification tree is a neural network with one output unit. If the output of net net is greater than some threshold, the instance is sent to the left subset, otherwise it is sent to the right subset. We may specify this formally in the following way, where $left(x)$ and $right(x)$ denote the extent to which x is a member of the left or right subset:

$$left(x) = \begin{cases} 1 & net(x) \geq .5 \\ 0 & net(x) < .5 \end{cases} \tag{1}$$

$$\text{right}(\vec{x}) = 1 - \text{left}(\vec{x}) \tag{2}$$

The problem of finding the optimal neural network is a function optimization problem. Such problems may be described by the domain, the objective function which evaluates elements of the domain, and a method of using the objective function to find the optimal element within the domain. Our domain: The neural net topology is fixed for a given run of the algorithm, so the domain is simply $\mathbb{R}^n$, where n is the total number of weights in the network. Our optimization algorithm: We used gradient descent, with a learning rate of $0.1/|\nabla f|$ and momentum of 0.8. The interesting part of the tree construction algorithm is the objective function f used to evaluate how well a neural net partitions a set of data, which we discuss further below.

2.1 THE OBJECTIVE FUNCTION: SOFT ENTROPY

Our objective function is the soft entropy function, a continuous approximation to the entropy criterion used previously in decision tree induction. Previous work in neural trees has generally used off-the-shelf training algorithms for neural networks, thus minimizing a mean squared error criterion(Sankar & Mammone 1991, Brodley & Utgoff 1992). Soft entropy is to be preferred over MSE for several reasons. Figure 1 shows a dataset and the hyperplanes (neural nets with no hidden units) minimizing MSE and soft entropy.

Because the number of grey patterns in the middle is outnumbered by both clusters of black patterns on either side, any split in the middle of the data assigning the "grey" class on

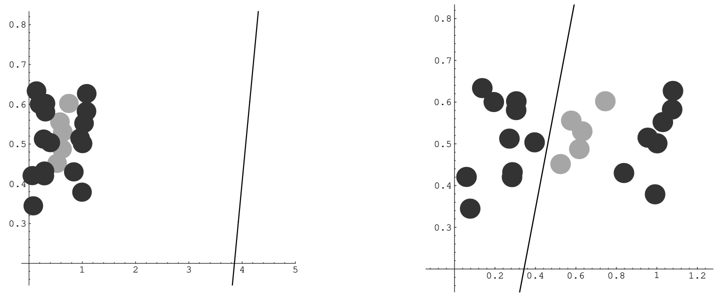

Figure 1: Left: The hyperplane split that minimizes MSE. Right: The hyperplane split that minimizes soft entropy.

Figure 2: All four splits shown on the left have equivalent entropy. On the right, the split that minimizes soft entropy is shown.

one side and "black" on the other will be worse than simply assigning "black" to the entire dataset, which is exactly what the MSE-minimizing hyperplane does. Clearly the objective of partitioning is not to minimize error immediately (which is what the MSE solution gives) but rather to divide the dataset in such a way that a small tree is found which minimizes errors. We would expect entropy to perform better in this role than MSE. In this example the entropy-minimizing split also maximizes the Kolmogoroff-Smirnoff distance (Friedman 1977).

The entropy criterion is described in Morgan & Messenger (1973). Soft entropy is quite similar, but is to be preferred. Figure 2 shows several candidate splits of a simple dataset. All of the splits shown have equivalent entropy, but clearly the hyperplane that divides the space down the middle is preferred. Thus we believe that in continuous spaces a good splitting criterion requires more granularity than entropy provides. Additionally, since soft entropy is differentiable it is amenable to the efficient use of a wider variety of function optimizers.

2.1.1 Definition of Soft Entropy

Soft entropy is a significant but simple modification of entropy, so we begin by discussing entropy and then show how it is modified to give the soft entropy criterion. As in Quinlan (1986), we are concerned with $E(net)$, the expected mutual information of the class label and the partition described by the network.

$$E(net) \quad = \quad \sum_{i \in \{l,r\}} \frac{p_i + n_i}{p + n} I(p_i, n_i) \tag{3}$$

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n} \tag{4}$$

p_i and n_i refer to the number of positive and negative instances in the left ($i = l$) or right ($i = r$) subsets[1]. p and n are similarly defined on the entire dataset to be split.

$$p = \sum_{\vec{x}\in\text{dataset}} 1_{\text{class}(\vec{x})=\text{p}} \tag{5}$$

$$p_l = \sum_{\vec{x}\in\text{dataset}} \text{left}(\vec{x}) \times 1_{\text{class}(\vec{x})=\text{p}} \tag{6}$$

$$p_r = \sum_{\vec{x}\in\text{dataset}} \text{right}(\vec{x}) \times 1_{\text{class}(\vec{x})=\text{p}} \tag{7}$$

and similarly for n, n_l, n_r.

So far we have constrained $left(x) \in \{0,1\}$, but this is only by convention. Allowing instances to be partially assigned to both the left *and* right subsets amounts to simply allowing $left(x) \in [0,1]$ by defining

$$left(x) = net(x) \ ,$$

that is, the net output is taken to be the extent to which x is a member of the left subset. As before, $left(x) + right(x) = 1$.

The soft entropy of the net, $SE(net)$, is simply $E(net)$ when $left$ is defined as above. We only approximate the threshold split with this soft split during optimization of the net. When an optimal net is found and the data is partitioned, instances are sent wholly either left or right.

Note that the approach here is more general than only defining a soft version of entropy – any splitting criterion which depends only on the statistics p, p_i, n, n_i (e.g., GINI (Breiman et al. 1984), C-SEP (Fayyad & Irani 1992)) can be similarly "softened" by taking $left(x)$ and $right(x)$ to be continuous in $[0,1]$ and using the above equations for these statistics.

2.2 THE DERIVATIVE

To use steepest descent or any other function optimizer requiring first derivative information, we must first differentiate the soft entropy function SE with respect to the parameter vector $\vec{w}$. Since p_r and n_r are determined from p, p_l and n, n_l respectively, we first rewrite SE as a function of only p_l, n_l, p, n, and thus by the chain rule:

$$\frac{d\,SE}{d\,w_i} = \frac{\partial SE}{\partial p_l}\frac{\partial p_l}{\partial w_i} + \frac{\partial SE}{\partial n_l}\frac{\partial n_l}{\partial w_i} + \frac{\partial SE}{\partial p}\frac{\partial p}{\partial w_i} + \frac{\partial SE}{\partial n}\frac{\partial n}{\partial w_i} \tag{8}$$

$$= \frac{\partial SE}{\partial p_l}\frac{\partial p_l}{\partial w_i} + \frac{\partial SE}{\partial n_l}\frac{\partial n_l}{\partial w_i} \quad \text{since } p, n \text{ are constant} \tag{9}$$

[1]To simplify the presentation we will only discuss binary classification tasks, but extension to multiple classes is straightforward.

$$\frac{\partial SE}{\partial p_l} = \frac{\log_2 \frac{p_r}{p_r+n_r}}{p_r+n_r} - \frac{\log_2 \frac{p_l}{p_l+n_l}}{p_l+n_l} \tag{10}$$

$$\frac{\partial p_l}{\partial w_i} = \sum_{\vec{x}\in\text{dataset}} \frac{\partial net}{\partial w_i} \times 1_{\text{class}(\vec{x})=\text{p}} \tag{11}$$

$$\frac{\partial SE}{\partial n_l} = \frac{\log_2 \frac{n_r}{p_r+n_r}}{p_r+n_r} - \frac{\log_2 \frac{n_l}{p_l+n_l}}{p_l+n_l} \tag{12}$$

$$\frac{\partial p_l}{\partial w_i} = \sum_{\vec{x}\in\text{dataset}} \frac{\partial net}{\partial w_i} \times 1_{\text{class}(\vec{x})=\text{n}} \tag{13}$$

Since $\frac{\partial net}{\partial w_i}$ is exactly what backpropagation calculates, we have thus fully specified the first derivative calculation. This concludes the section on building a SENNT. At each step, we find the locally optimal neural network using gradient descent. We then do a hard split on the data, sending each instance to either the left or right subset, and these are recursively partitioned until purity.

3 REGULARIZATION: PRUNING AND TRAINING PATTERN FILTERING

While studying the trees induced by the algorithm, we found that typical leaf subsets contained around 50 to 100 instances, while other leaves contained only a few. Clearly some sort of pruning is called for. In our learning system, pruning simply replaces a decision node with a leaf whenever the number of positive or negative instances matching that node drops below a fixed parameter. (The value 5 was used in all experiments[2].)

Pruning is a method of local pattern selection, since instances are effectively removed from nodes deep in the tree. A novelty in our iterative re-filtering approach is to *rebuild* the tree using the reduced training set, the training set with outliers removed. The hypothesis is thus that data which is *locally* uninformative is also *globally* uninformative. In statistics, estimators resistant to the effect of outliers, *robust* estimators, have been studied in some depth. Still, Hubel (1977) states "I am inclined to... prefer technical expertise to any 'statistical' criterion for straight outlier rejection." In this vein, Guyon, Matić & Vapnik (1994) has proposed an interesting method for making use of human expertise in outlier removal to "clean" a dataset. Straight "statistical" outlier rejection is exactly what iterative re-filtering amounts to – essentially we replace the human expert with our own learning system in making such outlier rejection decisions.

The selection of patterns for training is an ubiquitous problem in pattern recognition. Many methods are *passive*, accepting data from a training sample in some random order. Other methods (Matan 1994, Aha 1991) actively accept or reject training patterns from a temporal sequence presented to them. The difference between our method and theirs is somewhat akin to the difference between forward and backward feature subset selection (John, Kohavi & Pfleger 1994) or forward and backward (construction/pruning) search methods over neural net or decision tree architectures. In general, unless time is a very limited resource, the best results are achieved by starting "big" and then shrinking (Breiman et al. 1984). This should

[2]A more sensible strategy will be implemented shortly.

Table 1: Test set accuracy and concept size (number of nodes) results on several datasets.

	Breast	Vote	Monks1	Monks2	Monks3
SENNT-PIRE	**99.3 (3)**	98.5 (3)	**100 (7)**	**100 (5)**	91.7 (5)
SENNT-P	95.6 (7)	**99.2 (5)**	**100 (7)**	**100 (5)**	91.7 (5)
SENNT	95.6 (13)	97.8 (13)	**100 (7)**	**100 (5)**	93.8 (11)
OC1-Pruned	95.6 (7)	95.6 (3)	82.9 (5)	96.3 (5)	**97.7 (3)**
OC1	98.5 (19)	97.0 (17)	91.2 (9)	94.2 (5)	90.3 (5)
Cascor	97.4 (5.9)	97.9 (1.2)	96.1 (1)	98.2 (1)	91.4 (1)

apply to pattern selection as well, and thus we suspect backward pattern selection to give greater performance than forward selection; however, this hypothesis has not been tested.

4 EXPERIMENTAL RESULTS

We ran SENNT, SENNT-PIRE, OC1 (Murthy, Kasif, Salzberg & Beigel 1993*b*, Murthy, Salzberg & Kasif 1993*a*), and Cascade Correlation (Fahlman & Lebiere 1990) on several datasets (Thrun et al. 1991, Murphy & Aha 1994), and results appear in Table 1. Note that our trees require continuous inputs, so a local encoding was used for all nominal input values.

We give results for the unpruned, pruned, and retrained versions of the SENNT algorithm. In all cases, the reported results are for neural nets with no hidden units. Because of problems with attractive local minima, trees built using larger nets almost always had exactly the same structure and accuracy. For example, a single neural net with 2 hidden units is able to completely separate the data in Monk2, but this network is only found in roughly one out of five runs. We currently initialize weights using a uniform(-.5,.5) distribution, but clearly a better initialization is called for.

SENNT-PIRE is the best or second best algorithm on both real datasets. SENNT with pruning was the winner on the congressional voting database, indicating that perhaps the hypothesis underlying iterative-refiltering is not valid for this dataset. In Monks3 the algorithm performed poorly becuase the pruning step actually threw out more real patterns than noisy ones! This supports Hubel's position.

A related algorithm is OC1(Murthy et al. 1993*b*, Murthy et al. 1993*a*), which is quite similar to the cycliclal optimization algorithm for finding hyperplane splits used in CART, except that OC1 adds an attempt to jump out of local minima by moving in a random direction in weight space. Experiments with OC1 used the `info-gain` (entropy) impurity measure and all parameters set to default values, except for the number of iterations and number of restarts, which we increased to 25. We show results for the pruned and unpruned version. Although both OC1 and SENNT (with no hidden units) learn hyperplanes, SENNT appears to perform better on the test data, generating smaller and better trees. That smaller trees were generated by SENNT suggests that the cyclical optimization in OC1 might still be getting stuck in local minima. From these results we cannot claim that soft entropy outperforms entropy, since totally different splits seem to have been found.

We ran Cascade Correlation (Fahlman & Lebiere 1990) ten times on each dataset with the default parameter settings. Results were competitive with the other algorithms, but worse than the reported performance of Cascade Correlation in (Thrun et al. 1991). We did not use the parameter values reported there because it is unclear to what extent those parameters were determined by "training on the test set," that is, using the test set performance to select good values for the parameters. In addition, the parameter setting recommended in that report gave worse performance on the other two datasets.

5 RELATED AND FURTHER WORK

Although this work was developed independently, Brent (1991), discusses essentially the same technique for softening a related splitting criterion, using conjugate gradients for the optimization. This work did not deal with pruning, nor did it present comparisons with similar work. Other related work has been discussed in the text where appropriate.

We are currently pursuing the use of soft entropy, rather than MSE, as the objective function for the early nodes in Cascaded neural networks (Fahlman & Lebiere 1990, Littmann & Ritter 1993) which add neurons as training progresses. Cascaded networks are quite similar to decision trees in the way information flows through the nodes, and thus the same desiderata apply – early nodes should try to pass along as much information as possible rather than trying to minimize error themselves.

6 CONCLUSION

We have discussed SENNT, a new algorithm for finding recursive partitioning classification trees composed of neural networks at the decision nodes. We have presented a softening approach which would allow one to use any of the common decision tree impurity measures as an objective function. In this paper we only looked at soft entropy and motivated its use as an objective function. The SENNT learning system performed well without any pruning. Iterative re-filtering was proposed as a method of backwards training pattern elimination. The difference in performance between SENNT-PIRE and SENNT-P on the datasets examined does not allow us to confirm or reject our hypothesis – more experiments are needed. Performancewise, SENNT-PIRE compares favorably with Cascade Correlation and OC1, generally giving better accuracy on a test set and generating small trees.

7 Acknowledgements

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. We would like to thank Wray Buntine, Jerome Friedman, Pat Langley, Ronny Kohavi, Nils Nilsson and Scott Benson for their comments on earlier drafts of this paper. Thanks to Sreerama Murthy for supporting the OC1 software.

References

Aha, D. W. (1991), 'Instance-based learning algorithms', *Machine Learning* **6**(1), 37–66.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984), *Classification and Regression Trees*, Chapman & Hall, New York.

Brent, R. P. (1991), 'Fast training algorithms for neural networks', *IEEE Transactions on Neural Networks* **2**(3), 346–354.

Brodley, C. E. & 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.

Fahlman, S. E. & Lebiere, C. (1990), The cascade-correlation learning architecture, *in* D. Touretsky, ed., 'Advances in Neural Information Processing Systems', Vol. 2, Morgan Kaufmann.

Fayyad, U. M. & Irani, K. B. (1992), The attribute selection problem in decision tree generation, *in* 'AAAI-92: Proceedings of the Tenth National Conference on Artificial Intelligence', American Association for Artificial Intelligence, AAAI Press / The MIT Press, pp. 104–110.

Friedman, J. H. (1977), 'A recursive partitioning decision rule for nonparametric classification', *IEEE Transactions on Computers* pp. 404–408.

Guyon, I., Matić, N. & Vapnik, V. (1994), Discovering informative patterns and data cleaning, *in* 'AAAI-94: Proceedings of the Twelfth National Conference on Artificial Intelligence', American Association for Artificial Intelligence, AAAI Press / The MIT Press. To Appear.

Hubel, P. J. (1977), *Robust Statistical Procedures*, Society for Industrial and Applied Mathematics, Pittsburgh, PA.

John, G., Kohavi, R. & Pfleger, K. (1994), Irrelevant features and the subset selection problem, *in* H. Hirsh & W. Cohen, eds, 'Machine Learning: Proceedings of the Eleventh International Conference', Morgan Kaufmann. To Appear.

Littmann, E. & Ritter, H. (1993), Generalization abilities of cascade network architectures, *in* S. J. Hanson, J. Cowan & C. L. Giles, eds, 'Advances in Neural Information Processing Systems', Vol. 5, Morgan Kaufmann, pp. 188–195.

Matan, O. (1994), On-site learning, or learning "on the job", Submitted to these proceedings.

Morgan, J. N. & Messenger, R. C. (1973), *THAID: a sequential analysis program for the analysis of nominal scale dependent variables*, University of Michigan.

Murphy, P. M. & Aha, D. W. (1994), 'UCI repository of machine learning databases', Available by anonymous ftp to `ics.uci.edu` in the `pub/machine-learning-databases` directory.

Murthy, S. K., Salzberg, S. & Kasif, S. (1993*a*), 'OC1', Software available by ftp to `blaze.cs.jhu.edu`.

Murthy, S., Kasif, S., Salzberg, S. & Beigel, R. (1993*b*), OC1: Randomized induction of oblique decision trees, *in* 'AAAI-93: Proceedings of the Eleventh National Conference on Artificial Intelligence', American Association for Artificial Intelligence, AAAI Press / The MIT Press, pp. 322–327.

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

Sankar, A. & Mammone, R. J. (1991), Speaker independent vowel recognition using neural tree networks, *in* 'IJCNN-91-SEATTLE: International Joint Conference on Neural Networks', IEEE Press, Seattle, WA, pp. II: 809–14.

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.