Hastac: an Algorithm for Developing a Tree of Cuts, and its Relation to Neural Networks

James T. Linnemann[a], David Bowser-Chao, James Hughes

Michigan State University, Department of Physics, East Lansing, MI 48824, USA

We describe Hastac, a C++ program for finding criteria for separating signal and background samples by means of a tree of cuts on linear combinations of variables. The method is quite fast, and not only allows near-optimal rejection for a given efficiency, but identification of the most significant variables from a large set candidate variables. A mapping to feed-forward neural nets is possible, offering an excellent starting point for further refinement. Results are presented for test distributions where the answers are known. An interpretation of feed-forward neural nets in terms of hyperplanes is developed.

1 Hastac, a Tree of Hyperplane Cuts

A tree[1] of cuts describes successively refined separation of an initial mixed sample of events into two classes, signal (s) and background (b), much as in a game of 20 questions. Each *node* of the tree represents a cut; events passing the cut are described as s-enhanced; those which fail are b-enhanced. Each subsample is subjected to more tests. The full tree consists of t cuts, and partitions the sample into $t+1$ regions (called *leaves* in this context). The cuts used in Hastac are hyperplanes. A point x is classed as s if the signed distance from the hyperplane is positive:

$$d = (\hat{n} \cdot x - c) > 0 \tag{1}$$

where $c = \hat{n} \cdot x_0$, with x_0 any point on the hyperplane, and $\hat{n}$ is a unit normal to the hyperplane pointing to the signal-enhanced region.

A tree built of such hyperplane cuts is a generalization of the usual (hyperrectangle) cuts. The tree nodes are directly geometrical; examining the hyperplanes used in decision nodes can identify useful variables. As we shall see, there are also useful relations between the tree and feed-forward neural nets.

The tree is grown by starting with training samples of pure s and b. 2000 events each were used for training the tree (and comparison methods). A decision hyperplane for a node is chosen by adjusting $\hat{n}$ and c to optimize s/b^α in its s-enhanced leaf. In practice, tree classification performance is improved by taking (during optimization) a continuous approximation to the step function

$$\begin{aligned} \theta(\hat{n} \cdot x - c) &\rightarrow g\left(\frac{\hat{n} \cdot x - c}{T}\right) \\ g(y) &= \frac{1 + \tanh(y)}{2} \\ s \approx \sum_{i \in S} g\left(\frac{\hat{n} \cdot x_i - c}{T}\right); \quad b &\approx \sum_{i \in B} g\left(\frac{\hat{n} \cdot x_i - c}{T}\right) \end{aligned} \tag{2}$$

[a]linnemann@msupa.pa.msu.edu

This continuous approximation is equivalent to estimating the projection of the probability distributions for s and b by using a smoothing kernel[2] $K(d - d_i) = \cosh^{-2}(\frac{d-d_i}{T})$ along the projection direction $\hat{n}$. Using the kernel estimate rather than the raw data helps avoid over-training. The continuous approximation also allows application of optimization methods using derivatives.

This optimization is repeated on both leaves of the node. The method is 5-10 times faster than neural net training because the successive subsamples are smaller and smaller.

2 Comparison with Optimum Performance and other Methods

The performance of a classifier can be rigorously tested by comparing it to the *best* possible test, the Neyman-Pearson test[3]. This test is equivalent to cutting on the single variable

$$u(x) = \frac{P(x|s)P(s)}{P(x|s)P(s) + P(x|b)P(b)}. \tag{3}$$

u is defined so that it ranges between 0 and 1. When $u = .5$, the point is equally likely to be s or b. This cut defines a (possibly complicated) contour along the decision surface $u(x)$ in hyperspace. The tree attempts to fit the contour with hyperplanes; other methods approximate contours or the surface by other means. The performance of the best test may be computed by calculating the distribution of u for the known signal and background distributions. The performance of a specific cut may be characterized by

$$\epsilon_s = \int_{u_0}^{1} P(u|s)du; \quad \epsilon_b = \int_{u_0}^{1} P(u|b)du \tag{4}$$

s/b is proportional to ϵ_s/ϵ_b, the factor by which a cut improves s/b from its original value of N_s/N_b. Thus, the rejection vs efficiency performance can be summarized by the plot ϵ_s/ϵ_b vs ϵ_s.

The tree leaves have varying degrees of signal enhancement. Ordering them in descending s/b allows presentation of the performance of the tree as a classifier by defining $\epsilon_s = s/N_s$ and $\epsilon_b = b/N_b$. This comparison with the optimum for all ϵ_s is a bit unnatural for the tree, as it is now required to fit *many* decision contours with hyperplanes instead of just one. For these tests, the tree was forced to generate up to 200 leaves.

The comparison with the optimum for the tree (and other methods[4]) was done by fixing the parameters of the classifier with the training samples with standard control parameters and testing on an independent sample of 2000 each s and b events.

Test Distributions All distributions chosen for the test were 3-dimensional. Table 1 gives their characteristics. The first 4 distributions have 3 independent distributions. The first 3 distributions correspond to the "poor separation" case of ref. 5. CLEAN is FAR with s and b reversed. FRAC is a sawtooth in f; the x, y, z projections are flat, but peaks in s/b lie along $x + y + z = \frac{1}{2}, \frac{3}{2}$. RING has clearly

Table 1: Test distributions

SAME	s	flat(1.1)	Normal(0,1.2)	LogNormal(0,1.3)
	b	flat(1.1)	Normal(0,1.2)	LogNormal(0,1.3)
NEAR	s	flat(1.1)	N(0,1.2)	LNor(0,1.3)
	b	flat(1.4)	N(0,1.5)	LNor(0,1.6)
FAR	s	N(0,1.1)	N(0,1.2)	LNor(0,1.3)
	b	N(.2,1.7)	N(.3,1.8)	LNor(.4,1.9)
CLEAN	s	N(.2,1.7)	N(.3,1.8)	LNor(.4,1.9)
	b	N(0,1.1)	N(0,1.2)	LNor(0,1.3)
FRAC[6]	s	$\|f-.5\|; f=(x+y+z) \bmod 1, \quad x,y,z \in (0,1)$		
	b	$.5-\|f-.5\|$		
RING	s	torus($\overline{r}=1, \sigma_r, \sigma_z = .3$) in (x,y)		
	b	torus($\overline{r}=1, \sigma_r, \sigma_y = .3$) in (x,z)		

distinguishable regions and regions of equal s and b density. The corresponding author (JTL) is interested in hearing of other useful test distributions.

Results Typical results are shown in Fig. 1. For CLEAN, all methods somewhat underperform the optimum in the high ϵ_s region, and all have worse problems in the low ϵ_s region. For FRAC, PDE is unable to achieve high rejection, mainly because it has a tendency to oversmooth sharp features such as found in FRAC. The Neural Net also had trouble, needing a higher setting of learning parameters to follow the difficult shape. The tree, while not fully optimal, does a better job than PDE of tracing the optimal curve, which in this instance is calculable: $\epsilon_s/\epsilon_b = \frac{2}{\epsilon_s}(1+\sqrt{1-\epsilon_s})-1$. Our conclusion is that Hastac's algorithm needs tuning to better subdivide s-enhanced regions, and to have the objective function also consider the situation in the b-enhanced leaf. An earlier version of Hastac was shown[1] to be competitive to Neural Nets on realistic problems. Generally, the problems at small ϵ_s warn that adequate statistics are needed to measure high rejection: count s and b in the signal region and compute error bars for the expected rejection, just as you would for classic cuts!

3 Transcription of a Tree to a Feedforward Neural Net

A completed decision tree may be mapped into a 2 hidden-layer feedforward neural net[7] as follows. All nodes in the neural net start out with $T \approx 0$, that is as step function units. The first hidden layer consists of t units each testing one hyperplane used by the corresponding tree node. The second layer consists of $t+1$ units each representing a leaf path. The weights for the connections to first hidden layer are $+1$ if the path to a leaf required a s-enhanced decision from the corresponding tree node, -1 if a b-enhanced decision was required, and 0 if the tree node decision was irrelevant to the leaf. The output node is connected to the second hidden layer with weights 1 for all s-enhanced leaves. If the resulting net is to be used without alteration to implement the tree (rather inefficiently), the b-enhanced leaves could be

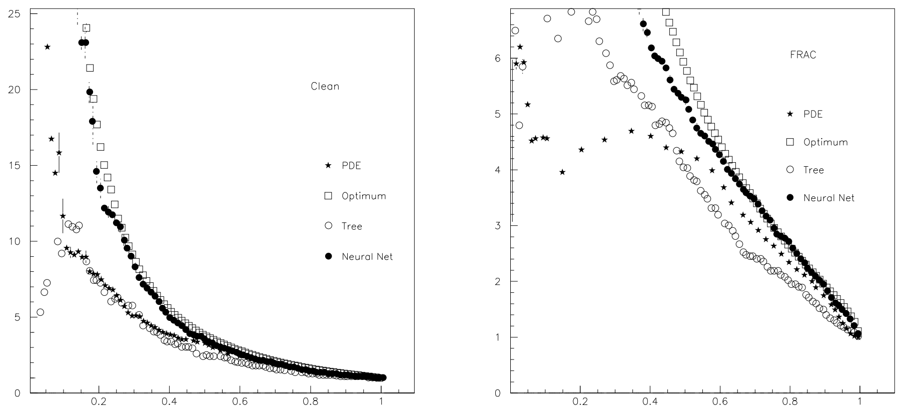

Figure 1: ϵ_s/ϵ_b vs ϵ_s for CLEAN and FRAC

omitted from the second layer. A more interesting use of the transcription would be to take the resulting net as a starting point for net training, with the nodes converted from step function units to sigmoids of finite temperature. A more radical alternative would be to ignore the second hidden layer and use the node hyperplane directions as starting points for the training of a 1-layer net with sigmoidal units, since the intent of the tree was to locate good hyperplane directions.

4 Geometric Interpretation of Feedforward Neural Net

A single layer feed-forward neural net may be described as follows. D inputs are combined in M hidden units with outputs

$$Y_k = g(\omega_k \cdot x - a_k) \tag{5}$$

where g is a sigmoidal function (2). The hidden layer outputs are then combined by the output unit

$$u(x) = g(W \cdot Y - A) \tag{6}$$

From our study of the tree algorithm, we can rewrite the weight vectors as

$$\vec{\omega_k} = \hat{n}/T_k \qquad a_k = c_k/T_k. \tag{7}$$

The first expression defines T_k and $\hat{n}$; the second defines c_k. That is, we can understand the operation of the hidden layers as picking a hyperplane direction and position to maximize signal and background separation. The purpose of the choice of hyperplane direction is to do a tomographic projection of the s and b distributions and fit $(s/(s+b)?)$ with a sigmoid of transition width T_k. The hidden layer weights contain information on the useful variables which is just as geometrical as that in the tree decision nodes hyperplane directions!

What about output layers? One way of looking at it is that it merely repeats the process in the space of the extracted features $Y_k \in (0,1)$, again telling which features are important. An interesting symmetry becomes apparent when we realize that one can force the weights W_k positive by demanding that the hidden layer hyperplanes all be oriented towards the good-signal region, that is, that $Y_k = 1$ always be s-enhanced. The symmetry $Y_k \rightarrow 1 - Y_k$ is

$$\begin{aligned} \omega_k &\rightarrow -\omega_k, & a_k &\rightarrow -a_k \\ W_k &\rightarrow -W_k, & A &\rightarrow A - W_k \end{aligned} \tag{8}$$

In the $W_k > 0$ orientation, we may view the output node as a soft voting machine, with A as the number of weighted projections which, if driven to 1.0, would force u to $u > .5$, the s-enhanced region. This symmetry shows that the net has 2^M completely equivalent states for the learning algorithm to seek.

With this perspective, we also gain some insight into the number of hidden nodes needed. M must be at least the number of projections needed to pick up important features (and correlations) of the distributions to be separated. However, this number may be boosted further if the profile along an important projection is not well represented by a single step function; multiple transition points c_k may be needed for one projection direction $\hat{n}_k$.

Acknowledgments

The presenter (JTL) thanks Alberto Santoro, CBPF/LAFEX, MSU, and the NSF for assistance in attending this conference.

References

1. David Bowser-Chao and Debra L. Dzialo, *Phys. Rev.* D **47**, 1900 (1993); D. Chao et. al., MSU-HEP/50327, hep-ph/9503453; J. Friedman, IEEE Trans. on Comp. p404 (1977); L. Breiman et. al. *Classification and Regression Trees* (Chapman & Hall, 1984); J. Dorfan, Mark II Note, 1979.
2. D. W. Scott, *Multivariate Density Estimation* (John Wiley, 1992); L. Holmström et. al., Comp. Phys. Comm., to be published. The $cosh^{-2}$ kernel resembles a Gaussian, but has longer tails.
3. W.T. Eadie et. al., *Statistical Methods in Experimental Physics* (North Holland, 1971), p 224; S. Brandt *Statistical and Computational Methods in Data Analysis* (North Holland, 1970), p 174. $u(x)$ is closely related to $r/(1+r)$ in the notation ref. 5.
4. The P.D.E. (probability density estimation) program was provided by P. Virador and H. Miettinen[2]; Harrison Prosper ran JetNet3.0 for the Neural Net trials.
5. J. Linnemann, "How Hard Should you Cut when the Data Sample is Finite?", these proceedings.
6. L. Garrido et. al. Comp. Phys. Comm. **84**, 297 (1994)
7. R. P. Brent, "Fast Training Algorithms for Multi-layer Neural Nets," Australian National Laboratory report (unpublished); see also J. Hertz et al, *Introduction to the Theory of Neural Computation* (Addison Wesley 1991) p 159 which claims a mapping into a 1-layer network.