Fusing the Results of Diverse Algorithms

John F. Elder IV, Ph.D.

Elder Consulting
1006 Wildmere Place
Charlottesville VA 22901
elder@stat.rice.edu

Abstract

Structurally adaptive methods -- from decision trees and polynomial networks, to projection pursuit models, additive networks, and cascade correlation neural networks -- iteratively add components in an attempt to construct models with complexity appropriate to the data. The diverse basis functions and search strategies employed usually lead to a distribution of results (with rankings hard to predict *a priori*) yet, robustly combining the output estimates can achieve a consensus model with properties often superior to the best of the individual models. Additionally, other information learned by some of the modeling techniques can be shared to the benefit of the fused system, including identification of key variables to employ and outlying cases to ignore.

This paper describes the fused model developed for a small but challenging classification dataset (where one infers the species of a bat from its chirps) and introduces a robust method of combining the outputs of diverse models.

Inductive Modeling Toolbox

Though new inductive methods are introduced continually, they appear to be motivated by only a handful of key underlying ideas, and it is likely that a suite of methods can be identified which will essentially span "method space". In a readable empirical survey, Michie, Spiegelhalter and Taylor (1994)[1] suggested such a sufficient toolbox would include linear discrimination, decision trees, and *k*-nearest neighbors, as well as the more recent methods of projection pursuit and radial basis functions. Elder and Pregibon (1996) further suggest adding polynomial networks and (perhaps) adaptive splines and neural networks to the set of covering methods, due to some relatively unique properties of those algorithms. Four of these methods -- decision trees, *k*-nearest neighbors, neural networks, and polynomial networks -- were selected to be used, in a multi-stage procedure, to address a challenging classification problem.

[1] Reviewed by Elder (1996).

Application Example: Identifying Bat Species

Tracking populations of potentially endangered bats can be simplified if similar species can be distinguished using features of their sophisticated echolocation signals. Data recently collected and analyzed by Kaefer et al. (1996) demonstrated that the necessary information is present in the signals, but that a fused model (described here) seems required to extract it. The data was obtained by capturing and labeling bats, recording several of their in-flight biosonar calls, extracting Fourier and time-frequency features from such "chirps", and iterating the signal feature extraction phase after analyzing early returns from individual induction algorithms charged with constructing discriminating classifiers. The data consists of 93 cases representing 18 different bats (with 3-8 signals per individual) from 5 species (some quite related physically)[2] all of which emit calls in the FM range (a spectrum area previously unexplored for the purposes of discrimination).

Because there were multiple signals from each individual, a voting mechanism could be employed to arrive at a final classification for each set (bat). That is, a bat was counted as correctly classified if a plurality of its acceptable signals were correctly identified. (The acceptance process is defined below.)

For each algorithm, cross-validation accuracy was measured for both case-wise and bat-wise identification. That is, each method trained a model using 17 of the bat signal sets, then tested it on the 18th set (with this repeated 18 times so each set could be the hold-out sample in turn) and the collection of test results accumulated. Thus, the results reflect the accuracy of each algorithm (represented by a bundle of models) when applied to *new* individuals, as is required to estimate potential field performance.

Decision Trees

Decision tree algorithms (e.g., CART; Breiman, et al., 1984) try to discriminate between classes by recursively

[2] Physically distinguishing one pair of species, for example, apparently requires examination of the underside of the bat's ear.

partitioning the data until the resulting groups are as pure as can be sustained. They start with all the training data and find the single (typically univariate) threshold split which divides the sample into two maximally pure parts. Each part is then similarly recursively subdivided until either no splits are possible, the classification is perfect, or the partitions reach some minimum size. As this maximal tree is typically too complex to be useful on new cases (it overfits), CART then prunes back the tree, using cross-validation to decide how much to simplify. (Other methods use a complexity penalty to determine a forward stopping point.)

Tree methods use a "greedy" search scheme, optimizing each successive step. Yet, it is well known that greedy algorithms may not result in the best classifier. The TX2step decision tree algorithm (Elder, 1994) enhances this aspect of decision trees by looking two steps ahead in choosing the split, often thereby improving performance (though at the cost of squaring the KN runtime for K variables and N cases).

As expected, the 2-step trees were more accurate on training and usually more complex). Yet they also fared better on evaluation, where the one-step trees got 46% of the cases correct, and only 7 of the 18 bats. The 2-step trees correctly identified 58% of the cases and 11 of 18 bats.

Backpropagation Neural Networks

Decision trees form sharp decision boundaries, transitioning immediately from one class to another. Smoother, more probabilistic, transitions can be more robust, however. Backpropagation Neural Networks (BNNs) are nonlinear estimators which can be applied to the problem of classification, typically by defining a network with an output node representing each class, an input node for each input variable, and one or more intermediate (hidden) layers. The nodes are nonlinear -- weighted sums followed by a logistic transformation. Using the training data, the weights in the network are adjusted according to a type of gradient search to cause the correct output node to take on a value near unity and the other output nodes to approach zero. When classifying an unknown case, its input variable values are run through the network, and the class with the largest output node value is selected. (Note that, though not used here, "optimal scoring" -- a superior way to use the output of estimative methods for classification, has recently been introduced by Hastie, Tibshirani, & Buja, 1994.) Though quite slow to train, BNNs do have the ability to construct nonlinear classification functions of greater variety than simpler methods such as linear regression or decision trees.

We[1] employed networks with three layers and varying K, with the number of nodes in the hidden layer proportional to $log(K)$. (The internal network training parameters were set after some experimentation to be: learning rate = 0.3, momentum = 0.8, tolerance = 0.1, iterations = 40,000.) As the logistic activation function for each node does not reach its extreme values of 1 or 0 without infinitely large weights, we instead set targets of 0.9 and 0.1.

The first set of BNNs trained using the 35 original variables was 52% accurate on the cases, and showed signs of overfit. Therefore, correlation and visualization information was employed to reduce the number of input variables to 17, improving the case-wise cross-validation (CV) accuracy to 63%. Further reducing the input set to the 8 variables selected by the decision tree methods boosted the BNN results to 69%. After voting, this last BNN set got 14 of 18 bats correct.

Nearest Neighbors

Exemplar-based statistical techniques, such as nearest neighbors and kernel methods, attempt classification by more direct comparison of test data with the training data itself. In k-nearest neighbors (k -NN), a new case is labeled with the class of the majority of the k closest examples in the training set, usually using Euclidean distance as the metric. This simple non-parametric method is very flexible and straightforward; furthermore, it has been proven that, given enough training data, the error rate is less than twice that of the theoretical optimum (Cover and Hart, 1967).

However, moreso than with methods which select variables, k-NN is sensitive to the dimensions chosen, as any unnecessary one contributes to the distance calculations and thereby obscures the true relationships. So, the 8 dimensions found useful by the tree methods were employed, and a 5-NN algorithm was 70% accurate on the cases and hit 14.5 of 18 bats (counting ties as half right). Further, when all subsets of these 8 variables were exhaustively explored, a set of 5 dimensions were found with 80% accuracy, indicating the importance of dimension selection for the algorithm. (However, it is not valid to use this performance number, as the CV accuracy was maximized through a search; another level of CV would be required to get an honest estimate of performance on truly new data.)

Although the input set and the resulting case accuracy of the k-NN and BNN methods are equivalent, they disagreed on 35% of the case labels. (Also, the k-NN took almost no time to run, whereas the Matlab implementation of the BNNs suite required about a day of processing power.) This degree of difference suggests that some cases are easier for one method to handle than the other and that combining them (below) would likely be fruitful.

Polynomial Networks

Another powerful adaptive modeling method escapes the limitations of linearity through employing compositions of nonlinear polynomials. Polynomial networks (PNs) (e.g., GMDH, Farlow 1984; ASPN, Elder & Brown, 1992) use linear least squares regression to build nodes (groups of nonlinear terms) from combinations of the inputs. ASPN

[1] Neural network runs were performaed by Oliver Kaefer and Doug Jones of the Univ. Illinois Urbana/Champaigne EE Dept.

selects the best several nodes for membership in the current layer using a criterion which penalizes complexity as well as error (to avoid overfit). Within each node, reverse elimination is used to pare away terms not contributing sufficiently to its accuracy. The building of layers continues until the modeling criterion can no longer be improved. A great strength of the method is that the structure of the network (its inputs, connections, and terms) does not have to be pre-specified, as is required with neural networks, but instead adapts automatically toward the structure of data. The resulting multiple layers of diverse polynomial nodes, each with several terms, is a flexible compound function which smoothly interpolates the sample region but, because of its use of polynomials, extrapolates poorly beyond the data boundaries.

The suite of PNs was 62% accurate on the cases but actually performed best of all individual methods on getting bats right: 15 of 18, *when outliers were ignored*, as described below.

Case Elimination

Three of the methods -- decision trees, BNNs, and *k*-NN -- all degrade rather gracefully compared to the last; since PNs employ global polynomials, they can easily "explode" on extrapolated cases. This is normally considered a great negative (e.g., Friedman, 1991) but can, in a voting situation, be turned into an important advantage. With nonlinear PNs, cases sufficiently distant from the convex hull of the training data space can have estimates many orders of magnitude more positive or negative than the target values [0-1]. (Experiments showed that a few orders of magnitude off could be tolerated.) When this occurs, it can signal the system that a "don't know" answer is more appropriate than the typical approach of limiting the response to the nearest boundary.

Further, when all the output nodes are close to a zero value, it is generally better to leave the case out of the voting scheme, than to "split hairs" among uncertain estimates. Here, a PN output summation threshold of 0.1 eliminated 12% of the cases. When this extra information from the PN was employed within the *other* methods, the results of each improved slightly.

Combining Outputs

It is an ancient statistical principle[1] that merging a set of independent estimates can reduce the variance of the output. First, to smooth the contributions of the boundary-based methods, the leaf nodes of the trees were labeled not with the winning class, but with the sample distribution. Similarly for the 5-NN method; each sample neighbor contributed a mass of 0.2 to the distribution. (Late in the experiment it was discovered that simply averaging the 5-NN distributions over a set of signals from a single individual and *then* taking the maximum worked quite well, getting 16 of 18 bats correct. However, for the following experiments, the outputs were treated like the other methods, labeling each case with its maximum, then voting among the accepted cases making up one set.)

Simply averaging the outputs of the four techniques before voting did improve the CV case accuracy to 74%. When the weights for each method's outputs were made proportional to their individual performance, this increased to 78%. Then, when the polynomial network was allowed to identify and remove outliers for all four techniques, 82% accuracy was obtained.

Advisor Perceptrons

Weighted sums can be sensitive to outlying estimates which can dominate any combination. To bound the influence of each method, the technique of "advisor perceptrons" (AP) -- originally developed to induce binary models (Elder, 1992) -- was employed. APs are motivated by the scenario of a decision maker faced with a binary choice (e.g., buy/sell, class 1/not class 1) who had d advisors offering binary recommendations. It was shown that there are a finite number of possible weight sets which lead to distinct decision maps, and that a linear program could generate the list. For example, with four advisors, there are only 12 sets to enumerate: corresponding to giving double weight to one advisor (2111, 1211, 1121, 1112), ignoring one advisor (0111, 1011, 1101, 1110), or employing only one advisor (1000, 0100, 0010, 0001). (For five inputs the list grows to a size of 81, and is in the billions by $d = 9$.)

Recently, Lee (1996) employed APs to combine five methods -- logistic regression, decision trees, projection pursuit regression, BNNs, and learning vector quantization -- to achieve results superior to averaging on each of four diverse data sets. This is somewhat surprising, as one is throwing away some probability information by discritizing each estimate. However, the added robustness can be quite useful on new data.

Here, there were three perceptron weight sets with similar strong performance (correct on 16 of 18 bats), corresponding to, respectively: doubling the BNN influence, or removing either the tree or PN contributions to the output estimation. (These results were not subject to CV checking, so must be relied on with caution. They are essentially equivalent though to the performance obtained by optimizing the combination weights in hindsight, which suggests that the upper limit of accuracy on this data is hitting 16.5 bats out of 18.)

Conclusions

Diverse induction methods have different strengths which can often be tapped by combining their information. Such *fusion* requires merging their outputs, either say, by weighting (based on individual performance) or using the robust technique of "advisor perceptrons". But it also

[1] Proverbs 24:6b: "... in a multitude of counselors there is safety."

involves sharing information learned about the data between techniques. Here, on a challenging species classification problem, the methods which select variables (decsion trees and polynomial networks) were able to improve the cross-validation performance of those which cannot (backpropagation neural networks, and k-nearest neighbors). Also, a hitherto negative property of polynomial networks -- their tendancy to "explode" in extrapolation situations -- was harnessed to identify cases which should not contribute to the vote among a set, and thereby turned into a benefit.

References

Breiman, L., J. H. Friedman, R. A. Olshen, & C. J. Stone 1984. *Classification and Regression Trees*. Wadsworth & Brooks, Pacific Grove, CA.

Cover, T. M., & P. E. Hart 1967. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory* 13: 21-27.

Elder, J. F. IV 1992. Optimal Discrete Perceptrons for Graded Learning, *International Systems, Man, and Cybernetics Conf*, Chicago, IL, Oct. 18-21.

Elder, J. F. IV 1994. Inducing Models less Greedily, *International Systems, Man, and Cybernetics Conf.*, San Antonio, TX, Oct. 2-5, pp. 908-912.

Elder, J. F. IV 1996. A review of *Machine Learning, Neural and Statistical Classification*, (eds. Michie, Spiegelhalter & Taylor; Ellis Horwood, 1994), *J. American Statistical Assoc.* 91, no. 433: 436-437.

Elder, J. F. IV & D. E. Brown 1992. Induction and Polynomial Networks, Univ. VA Tech. Rpt. IPC-TR-92-9. (Forthcoming as Chapter 3 in *Advances in Control Networks and Large Scale Parallel Distributed Processing Models* (Vol. 2), Ablex: Norwood, NJ.

Elder, J. F. IV & D. Pregibon 1996. A Statistical Perspective on Knowledge Discovery in Databases, Chapter 4 in *Advances in Knowledge Discovery and Data Mining*, eds. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI/MIT Press.

Farlow, S. J., ed. 1984. *Self-Organizing Methods in Modeling: GMDH Type Algorithms*. Marcel Dekker, NY.

Friedman, J. H. 1991. Multiple Adaptive Regression Splines (with discussion). *Annals of Statistics* 19: 1-141.

Hastie, T., R. Tibshirani, & A. Buja 1994. Flexible Discriminant Analysis by Optimal Scoring. *J. American Statistical Assoc.* 89, no. 428: 1255-1270.

Kaefer, O., J. F. Elder IV, C. J. Condon, K. R. White, A. S. Feng, & D. L. Jones 1996. Classification and Discrimination of the Echolocation Signals of Six Species of Vespertilionid Bats from Illinois. Forthcoming.

Lee, S. S. 1996. Combining Neural and Statistical Classifiers Via Perceptron, *Proc. 2nd Annual Knowledge Discovery in Databases Conf.*, Aug. Forthcoming.

Michie, D., D. J. Spiegelhalter, & C. C. Taylor eds. 1994. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, NY.