Combining Neural and Statistical Classifiers Via Perceptron

Sauchi Stephen Lee
Division of Statistics
University of Idaho
Moscow, ID 83843
Tel: (208) 885-7701, Fax: (208) 885-5843
steve@laplace.mathstat.uidaho.edu

Abstract

Many classification models are presently available from both the Neural Networks and Statistics communities. Recent research has revealed that, instead of selecting the "best" single model from a set of plausible candidates, integrating a set of "heterogeneous" models will maximize overall accuracy. Disagreements between these candidate models enable one to pick and choose their relative strengths in an optimal manner. Many statistical models, due to their intrinsic different model structures, could complement neural networks. In classification problems, encouraging results were obtained by integrating some statistical and neural classifiers via perceptron.

Keywords: Classifiers, logistic regression, classification and regression tree, projection pursuit regression, learning vector quantization.

Introduction

The recent decade has witnessed the flourish and bloom of applying artificial neural networks in many disciplines such as economic forecasting, medical diagnosis, environmental monitoring, and financial market prediction. Despite their apparent success and widespread acceptance, statisticians are slow in using these emerging technologies (Cheng & Titterington 1994; Ripley 1993). On the other hand, people are challenged by these powerful networks which have demonstrated their potential applicability to many practical domains (Cheng & Titterington 1994; Ripley 1994; Ripley 1996).

It is sad, but real, to see a polarization toward the neural networks and the statistics paradigms. Attitudes ranging from *Statistics is doomed to fail because it cannot handle complex nonlinear real world problems* to *Neural networks are for people who cannot do statistics.* It is unnecessary to subscribe to such a dichotomy. People from both communities should/could work and complement each other. In fact both fields share many common goals in mining data, in trying to extract information and recognizing patterns in data sets, and in fitting appropriate models to data for classification, interpretation, and prediction (Elder & Pregibon 1995; Cherkassky, Friedman, & Wechsler 1995; Michie, Spiegelhalter, & Taylor 1994).

In attempting to build bridges between the two paradigms, we will present an empirical study to integrate some neural and statistical classifiers. Instead of selecting the best model among a set of candidate classifiers, a hybrid system which appropriately combines various models is believed to be more accurate in classifying new data, especially when the candidate models originate from different families (Hashem, Schmeiser, & Yih 1994; Jacobs et al. 1991; Perrone & Cooper 1993; Wolpert 1992). Recently, there has been considerable interest in building such hybrid systems. The techniques can be broadly divided into two main areas: generating a set of heterogeneous models; and integrating the learned model-outputs. It appears that in classification problems, much attention has been focused on how to integrate homogeneous classifiers (Romero & Obradovic 1995; Tumer & Ghosh 1995; Xu & Jordan 1993), and not much effort has been devoted to combine heterogeneous classifiers. Attempts to generate multiple classifiers could be done by either using different training heuristics (e.g. initializing with different connection weights (Battiti & Colla 1994; Hansen & Salamon 1990), different network architectures, different stopping and pruning criteria) and/or bootstrapping the training data with noise (Raviv & Intrator 1995) or without noise (Breiman 1994). It is believed that many classification models from the neural and statistics communities will potentially be a rich source to generate multiple heterogeneous classifiers. They could be easily combined via a special perceptron as described in the next section.

Methods

Suppose we have c classifiers' outputs $y_1, \cdots, y_c$ and want to combine them to form an integrated output z. We first discretize $y_1, \cdots, y_c$ into binary ± 1 and input them into the "unbiased" perceptron

$$z = \begin{cases} -1 & \text{if } \sum_{j=1}^{c} w_j y_j < 0 \\ +1 & \text{if } \sum_{j=1}^{c} w_j y_j > 0, \end{cases}$$

where the weights $\boldsymbol{w} = \{w_1, \cdots, w_c\}$ are chosen such that $\sum_{j=1}^{c} w_j y_j \neq 0$. We further constraint the weights $\boldsymbol{w}$ to be non-negative because the exclusion of negative weights assumes no classifier is deliberately misleading and one should at worst ignore the classifier's advice.

Given a data set $\{(\boldsymbol{y}_i, z_i), i = 1, \cdots, n\}$, the weights $\boldsymbol{w}$ for the "best" unbiased perceptron (just called perceptron for simplicity) are not unique. The weights space can be partitioned into regions, and there are finitely many distinct perceptrons, one for each region (Elder 1992). The "best" perceptron is chosen from this finite collection of distinct perceptrons. We may represent each region by choosing a representative point $\boldsymbol{w}^*$ closest to the origin. Muroga *et al.* (Muroga, Tsuboi, & Baugh 1970) showed that for $c \leq 7$, all $\boldsymbol{w}^*$'s are unique and take only integer values. He found out that the number of distinct perceptrons for $c = 1$ to 9 are as follows:

c	Perceptrons	New "templates"
1	1	1
2	2	0
3	4	1
4	12	1
5	81	4
6	1,684	14
7	123,565	114
8	33,207,256	2,335
9	34,448,225,389	172,958

Table 1: Number of distinct perceptrons

The sets of new "templates" for $c = 1$ to 5 are $\{1\}, \{\}, \{111\}, \{2111\}, \{11111, 22111, 31111, 32211\}$. Permutations of these "templates" can enumerate the whole weights space. For instance, the four weight sets of a three-inputs perceptron have the two templates $\{100\}$ and $\{111\}$ which generate the four representative weights $\boldsymbol{w}_1^* = \{1, 0, 0\}$, $\boldsymbol{w}_2^* = \{0, 1, 0\}$, $\boldsymbol{w}_3^* = \{0, 0, 1\}$, and $\boldsymbol{w}_4^* = \{1, 1, 1\}$. The perceptron with $\boldsymbol{w}_4^*$ corresponds to majority voting, and the other three perceptrons ignore the advice of two inputs and follow the third input.

We decided to integrate 5 classifiers by carrying out an exhaustive enumeration to find the best perceptron. Since there are only 81 such perceptrons and it is expected that the exhaustive enumeration will be faster and more reliable than searching.

The basic idea of the 5 chosen classifiers is briefly described below.

We use x_{ji} to represent the j-th attribute for pattern i, and $y_i, \hat{y}_i$ to denote the true class and estimated (model) class, respectively, for pattern i. Let $P(\mathit{event})$ denote the probability of *event*.

Logistic Regression (LR) The model is

$$\mathit{logit}(P(y_i = 1)) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ji},$$

where $\mathit{logit}(z) = \mathit{log}\frac{z}{1-z}$, and $\boldsymbol{\beta}$'s are the parameters to be estimated via maximum likelihood (Myers 1990). If $P(y_i = 1) \geq .5$, then pattern i is classified as "class +1", otherwise as "class -1".

Classification And Regression Trees (CART) A tree partition the features (input variables) space into locally constant regions, often hypercubes parallel to the features axes. There are many different schemes for estimating trees. The idea is to recursively choose a feature or combination of features and to split the features space on that value. In this paper, we will use Breiman's CART (Breiman et al. 1984).

Projection Pursuit Regression (PPR) This is a statistical procedure originally proposed by Friedman and Stuetzle (Friedman & Stuetzle 1981) to interpret high dimensional data through well-chosen lower-dimensional projections. The model is of the form

$$\mathit{logit}(P(y_i = 1)) = \beta_0 + \sum_{m=1}^{M_0} \beta_m \phi_m(\sum_{j=1}^{p} \alpha_{jm} x_{ji}),$$

where three sets of parameters, projection directions $\boldsymbol{\alpha}_m$'s, the unknown smooth activation functions ϕ_m's, and the projection "weights" $\boldsymbol{\beta}$'s are estimated in a sequential manner. The number of projection directions M_0 is user-defined.

Neural Network (NN) We used the supervised feed-forward single-hidden-layer neural network

$$\hat{y}_i = \phi_0(w_0 + \sum_h w_h \phi_h(w_{0h} + \sum_{j=1}^{p} w_{jh} x_{ji})),$$

where ϕ_0 and ϕ_h's are some fixed monotonic differentiable functions. Unlike projection pursuit regression which estimates the parameters sequentially, all network training algorithms estimate the weights $\boldsymbol{w}$'s simultaneously.

Learning Vector Quantization (LVQ) This method has its origin as a self-organizing map which has been traditionally considered as a neural network method, although it is very distinct from the supervised feedforward multi-layer-perceptron. The idea is to replace the example patterns by a "representative" set called codebook vectors. To be useful as a classifier, these codebook vectors are grouped into classes. Learning vector quantization adjusts the codebook vectors to produce optimal class boundaries. The classification rule is to choose the class of the nearest codebook vector.

These 5 classifiers outputs serve as inputs to the perceptron for integration. The last classifier **LVQ** gives binary -1 or +1 output directly. The first four produce real-valued outputs and they have to be discretized to binary values. Sensitivity is traded for robustness in the perceptron integration process.

Experiments and Results

We applied the methods to four data sets: *Advisor, Kyphosis, Birth Wt,* and *Diabetes.* The first data set is simulated data and the other three are real-world data. These data sets were chosen because we want to perform the experiments on data with diverse attributes: simulated *vs* real-world, small sample (*Kyphosis*) *vs* large sample (*Advisor*), and few input variables (*Kyphosis*) *vs* many input variables (*Birth Wt/Diabetes*).

In order to estimate the "true" misclassification rate on new data, the test data should be sufficiently large. Sample size is not a problem for the simulated data. For the real-world data, we randomly partition the whole data set into three portions: a classifiers-training set, a perceptron-training set, and a test set. The data were divided into three instead of two because some classifiers may overfit the training set and would mislead the perceptron by the "seemingly" low misclassification rates if the same training data is used again to train the perceptron. The random partitions were repeated 100 times for each of the 3 real-world data sets and the overall misclassification rates were calculated.

The four data sets are described as follows:

Advisor The data are generated from a rule base adapted from (Luger & Stubblefield 1989) and is shown in Table 2. It is a simplified system trying to illustrate issues involved in real-life financial advising. The system consists of 5 input variables, shown in bold in Table 2, and a binary response variable *invest-stocks.* We simulated 1,000 patterns for the classifiers-training set, 4,500 patterns for the perceptron-training set, and another 4,500 for the test set.

Kyphosis The data consist of measurements on 81 children following corrective spinal surgery (Chambers & Hastie 1992). There are three input variables:

(1)	if (saving-adequate and income-adequate)
then	*invest-stocks*
(2)	if dependent-saving-adequate
then	saving-adequate
(3)	if assets-high
then	saving-adequate
(4)	if dependent-income-adequate
then	income-adequate
(5)	if debt-low
then	income-adequate
(6)	if (**saving** $\geq$ **dependents**$\times$5000)
then	dependent-saving-adequate
(7)	if (**income** $\geq$ 2500 + 4000$\times$**dependents**)
then	dependent-income-adequate
(8)	if (**assets** $\geq$ **income** $\times$ 10)
then	assets-high
(9)	if (**annual-debt** $<$ **income** $\times$ 0.3)
then	debt-low

Table 2: Financial advisor rule base

Age of the child in months, **Number** of vertebrae involved in the operation, and **Start**, the beginning of the range of vertebrae involved. The response variable *kyphosis* indicates the presence or absence of the postoperative deformity. The data-splitting is 65/8/8 (means 6500/800/800 when repeated 100 times) for the classifier-training/perceptron-training/testing sets, respectively.

Birth Wt The data consist of 189 births at a US hospital, with the main interest in classifying low birth weight, given 8 input variables which represent the medical information and history of expectant mothers (Hosmer & Lemeshow 1989). The data-splitting is 139/25/25 for the three portions.

Diabetes This is a data set gathered among the Pima Indians by the National Institute of Diabetes and Digestive and Kidney Diseases. It is made available in the UCI repository (ftp ics.uci.edu /pub/machine-learning-databases). The data set consists of 768 patterns and 8 input variables. The input variables are medical measurements and pregnancy information on each patient. The response variable is one of two classes: tested positive for diabetes (268) or negative (500). The data-splitting is 568/100/100 respectively.

The misclassification rates (in %) on test data for the four data sets are tabulated in Table 3. The first five rows are the error rates for the five individual classifiers. The means were obtained by averaging the real-valued and binary-valued classifiers' outputs.

We noticed that the integrated classification rates via the optimal perceptron (labeled "percep.") are better than the means in all four data sets. This indicates that discretization of the classifiers' real-valued outputs helped to increase robustness. We can interpret the perceptron integration as a weighted average. For example, the perceptron integration is equivalent to the single best **NN** classifier for the *Advisor* data; for the *Kyphosis* data, **LR**'s vote weights twice as important as **CART**, **PPR**, and **NN**, while **LVQ**'s vote is ignored; and so on. The results are encouraging.

It is interesting to note that for *Advisor* and *Birth Wt* data, the perceptron just chose the classifier with the lowest misclassification rate. Why this is the case is not clear. By inspecting the error rates on the perceptron-training data, the differences between the lowest and second lowest error rates for *Advisor* and *Birth Wt* data were more significant than the other two data sets. This may cause the perceptron to pick the best.

Classifiers	*Ad.*	*Kyph.*	*Bir. Wt*	*Diab.*
LR	25.1	19.9	31.0	23.8
CART	9.7	23.4	37.8	28.9
PPR	17.4	22.9	35.4	25.3
NN	5.6	22.5	35.4	27.1
LVQ	41.3	23.0	32.3	32.7
mean	7.2	21.0	31.2	25.3
percep.	5.6	18.8	31.0	23.7
optimal $\boldsymbol{w}$	00010	21110	10000	31111

Table 3: Misclassification rates on test data for the 4 data sets

The experiments were carried out in the S-PLUS version 3.2 environment (Chambers & Hastie 1992; Venables & Ripley 1994). The three statistical classifiers are standard S-PLUS functions. Many S users have written their own S-PLUS functions and have generously organized these functions into libraries and made them publically available. We were benefited from Brain Ripley's nnet and classif libraries from the "StatLib" archive (e-mail: statlib@lib.stat.cmu.edu, or URL address: http://lib.stat.cmu.edu/S/) in implementing the Neural Network and Learning Vector Quantization classifiers, respectively. All model parameter settings were chosen either in an adhoc manner (e.g. number of projections in **PPR**, number of hidden units in **NN**), or at the default S-PLUS function values (e.g. number of terminal nodes in **CART**, number of codebook vectors in **LVQ**).

Acknowledgements

We are grateful to John Elder for providing the 81 perceptron weight sets and to Zoran Obradovic and Pedro Romero for providing the simulated financial advisor data set.

References

Battiti, R. and Colla, A. M. 1994. "Democracy in neural nets: Voting schemes for classification," *Neural Networks*, vol. 7, pp. 691–707.

Breiman, L. 1994. "Bagging Predictors," Technical Report No. 421, University of California at Berkeley.

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

Chambers, J. M. and Hastie, T. 1992. *Statistical Models in S.* Pacific Grove, California, Wadsworth & Brooks.

Cheng, B. and Titterington, D. M. 1994. "Neural Networks: A Review from a Statistical Perspective," *Statistical Science*, vol. 9, pp. 2–54.

Cherkassky, V., Friedman, J. H., and Wechsler, H. 1995. *From Statistics to Neural Network: Theory and Pattern Recognition Applications.* Springer.

Elder, J.F. 1992. Optimal Discrete Perceptrons for Graded Learning, International Systems, Man, and Cybernetics Conf., Chicago, Illinois, October 18-21.

Elder, J. F. and Pregibon, D. 1995. "A Statistical Perspective on Knowledge Discovery in Databases," In *Advances in Knowledge Discovery and Data Mining*, eds U.M.Fayyad, G.Piatetsky-Shapiro, P.Smith, and R.Uthurusamy. AAAI/MIT Press.

Friedman, J. H. and Stuetzle, W. 1981. "Projection pursuit regression," *Journal of the American Statistical Association*, vol. 76, pp. 817–823.

Hansen, L. K. and Salamon, P. 1990. "Neural Networks Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993–1001.

Hashem, S., Schmeiser, B., and Yih, Y. 1994. "Optimal Linear Combinations of Neural Networks: An Overview," *Proceedings of the 1994 IEEE International Conference on Neural Networks*, vol. 3, pp. 1507–1512.

Hosmer, D. W. and Lemeshow, S. 1989. *Applied Logistic Regression.* New York, John Wiley and Sons.

Jacobs, R. A., Jordans, M. T., Nowlan, S. J., and Hinton, G. E. 1991. "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, pp. 79–87.

Kohonen, T. 1995. *Self-organizing Maps.* Springer-Verlag, Heidelberg.

Luger, G. F. and Stubblefield, W. A. 1989. *Artificial Intelligence and the Design of Expert Systems.*

Benjamin/Cummings.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C. 1994. *Machine Learning, Neural and Statistical Classification.* New York, Ellis Horwood.

Muroga, S., Tsuboi, T., and Baugh, C. R. 1970. "Enumeration of Threshold Functions of Eight Variables," *IEEE Transcations on Computers*, September.

Myers, R. H. 1990. *Classical and Modern Regression with Applications.* Boston, PWS-KENT.

Perrone, M. P. and Cooper, L. M. 1993. "Learning from What's been Learned: Supervised Learning in Multi-Neural Networks Systems," *Proceedings of World Congress on Neural Networks*, vol.III, pp. 354–357.

Quinlan, J. R. 1993. *C4.5: Program for Machine Learning.* San Mateo, Morgan Kaufmann.

Raviv, Y. and Intrator, N. 1995. "Bootstrapping with Noise: An Effective Regularization Technique," Technical Report, Tel-Aviv University.

Ripley, B. D. 1993. "Statistical aspects of neural networks," In *Networks and Chaos – Statistical and Probabilistic Aspects*, eds O.E.Barndorff-Nielsen, J.L.Jensen and W.S.Kendall. pp. 40-123. London: Chapman and Hall.

Ripley, B. D. 1994. "Neural Networks and Related Methods for Classification (with discussion)," *Journal of Royal Statistical Society, B*, vol. 56, pp. 409–456.

Ripley, B. D. 1996. *Pattern recognition and neural networks.* Cambridge University Press.

Romero, P. R. and Obradovic, Z. 1995. "Comparison of Symbolic and Connectionist Approaches to Local Experts Integration," *Proceedings of IEEE Technical Application Conference.* pp. 105–110, Portland, OR.

Tumer, K. and Ghosh, J. 1995. "Theoretical foundations of linear and order statistics combiners for neural pattern classifiers," Unpublished Manuscript, University of Texas at Austin.

Venables, W. N. and Ripley, B. D. 1994. *Modern Applied Statistics in S-PLUS.* New York, Springer.

Wolpert, D. 1992. "Stacked Generalization," *Neural Networks*, vol.5, pp. 241–259.

Xu, L. and Jordan, M. I. 1993. "EM Learning on a Generalized Finite Mixture Model for Combining Multiple Classifiers," *Proceedings of World Congress on Neural Networks*, Portland, OR, vol.IV, pp. 227–230.