ACKNOWLEDGMENTS

This work was supported by Army Research Office contract DAAL03-92-G-0115 to the Center for Intelligent Control Systems, NSF Grant DMS-92-17655 and Office of Naval Research Contract N00014-91-J-1021.

Comment

Leo Breiman

Cheng and Titterington have most commendably brought developments in the neural network field to the attention of statisticians. It is a notable public service. Since their title is worded "...A Review from a Statistical Perspective", room is left for other statistical perspectives.

When I first heard about neural networks some years ago, I was put off by what I considered to be the hype about doing things the way the brain does. The going propaganda seemed to be that here was a set of procedures modeled after the brain that did a miraculously accurate job in a wide variety of tasks. The functioning of these procedures was coded in esoteric language based on terms borrowed from brain mechanisms. The whole thing was reminiscent of the artificial intelligence publicity a decade or two ago.

But in going to neural network meetings, reading and refereeing their articles and talking to many practitioners over the last five years, my opinion has changed. The neural network community consists of different segments. Some are concerned with constructing mathematical network models of the brain. Others are concerned with networks as mathematical entities, that is, their connectedness, dynamics, etc. Probably the largest segment consists of the people doing work on pattern recognition and other predictive problems.

1. THE CHARACTERISTICS OF THIS LATTER COMMUNITY

They are *not* a neural network community. They use any methodology that works on their problems. Often, they use CART or MARS. They experiment with nearest neighbor methods, separating surfaces gotten by using linear programming, radial basis functions, hidden Markov chains, etc. New methodologies are constantly proposed, and many of these have little resemblance to standard neural networks. Unfortunately, much of the original, and now anachronistic, terminology is retained giving misleading impressions about what is going on.

They are very pragmatic and problem oriented. In fact, the field is better defined by the nature of the problems they work on then by any particular methodology. Typical problems are speech recognition and handwritten character recognition. The range of problems is characterized by high dimensional complex data, often with very large sample sizes (10^4 to 10^7). The goal is to find accurate predictors in classification, regression and time series.

Often, the methodology they use is hand-tailored to the problem they are working on. In this respect, the neural network technology is attractive in that the network and the number of internal nodes can be tinkered with and optimized for the problem. But other methods are employed if they give better results.

Their bottom line is the error rate on the relevant data set. Proposed new methodologies are judged in terms of their error rates on banks of known data sets. But there is little systematic research into the circumstances under which some methods work better than others. This may be because the work is so oriented toward particular problem solving and tailored methodologies.

The people involved are, by background, computer scientists, engineers and physical scientists. They are generally young, energetic and highly computer literate. They have the further good fortune not to have any formal statistical training so that they feel no compulsion to engage in the futile games of modeling data or in endless asymptotics. What they have borrowed from statistics is very slight.

There are important cultural differences between the statistical and neural network communities. If a statistician analyzes data, the first question he gets asked is "what's your data model?" The NN practitioner will be asked "what's your accuracy?" In

Leo Breiman is Professor, Department of Statistics, 367 Evans Hall, University of California, Berkeley, California 94720.

statistics, high dimensionality (number of parameters estimated) is 5, maybe 20, and 100 is impressive. In NN problems, 100 is moderate while 1000 and 10,000 are more like it. Statisticians go for interactive computing. A NN member might say "what, only an overnight run? It must be a pretty small problem."

Another difference is that statisticians tend to try and develop universal methodology. That is, methodology that can be applied, virtually unchanged, in every environment. For instance, CART has been used, in untinkered form, in dozens of different fields. The NN workers, as mentioned above, tinker and tailor, cut and slice until the suit fits the data.

2. LOOKING AT THE NEURAL NETWORK METHODOLOGY

In the present prediction context, what is given is a set of data consisting of the variables to be used as predictors (usually denoted as a vector x) together with the associated values of the things (responses) to be predicted. This data is known as the training set or as the learning set. The goal is to use this data to construct a predictor of future responses based only on knowing x.

The neural network configuration most often used in prediction is called the single layer feed forward network. This has been covered by Cheng and Titterington, but I want to go through it again for several reasons. First, because it is the type of neural network most widely used in prediction. Second, because its success in some important problems was largely responsible for the surge of interest in these methods. Finally, because its structure is simple, we can hope to get some idea of its workings.

The idea is this: let the sigmoid function $\sigma(x) = \exp(x)/(1+\exp(x))$. Then fit the data by linear combinations of σ (linear combinations of the predictor variables). In regression where the training data is of the form $(y_n, x_n), n = 1, \ldots, N$ and x has M coordinates $x_1, \ldots, x_M$, $x_1 \equiv 1$, fit the data by a sum of the form

$$\hat{y}(x) = \sum_k \alpha_k \sigma(\beta_k x).$$

In a J class problem, the training data is of the form $(j_n, x_n), n = 1, \ldots, N$, where each j_n is a class label taking value in $\{1, \ldots, J\}$. Then the conditional probability for each class is estimated by a function of the form

$$\hat{p}(j \mid x) = \sigma\left(\sum_k \alpha_{jk} \sigma(\beta_k x)\right),$$

and the decision rule is to predict the class corresponding to the vector x as j if

$$\hat{p}(j \mid x) = \max_i \hat{p}(i \mid x).$$

To estimate the coefficients in regression, the least squares error L is defined by

$$L(\alpha, \beta) = \sum_n \left(y_n - \sum_k \alpha_k \sigma(\beta_k x_n)\right)^2.$$

Then L is minimized using gradient descent. In classification, define $z_{jn} = 1$ if $j_n = j$, otherwise zero. Put

$$L(\alpha, \beta) = \sum_{j,n} \left(z_{jn} - \sigma\left(\sum_k \alpha_{jk} \sigma(\beta_k x_n)\right)\right)^2$$

and again minimize L by gradient descent. The gradient descent most commonly used is called backpropagation and consists of putting in one data case at a time and then taking a partial gradient step. The data set is circulated through until the convergence is deemed satisfactory.

This is a simple and easily programmed idea. Since its introduction, it has been used in a wide variety of important engineering and computer applications with almost universally "satisfactory" results. In fact, it has become an all purpose crank. For many hopeful users, it relieves the tedium of thinking.

For instance, consider a problem that consists of classifying 32×32 bit images with each pixel in 16 grey levels and such that there are 26 classes. Note that each prediction vector x is of dimension 1024. Before the NN technology, researchers would puzzle over the images and try to extract a few features (functions defined on each image) that would contain most of the relevant classification information. Having drastically reduced the dimensionality, some standard classification methods could be used on the feature values.

Now the procedure is to toss the data directly into the NN software, use tens of thousands of parameters in the fit, let the workstation run 2–3 weeks grinding away doing the gradient descent and, voila, out comes the result. Automatic feature selection has taken place.

There are pluses and minuses to the NN crank. Prior to the crank, the only widely available methods were nearest neighbor templates, linear methods and various kludges. The NN crank is a widely applicable nonlinear method that usually gives good results.

3. BUT ALL IS NOT TEA AND CRUMPETS

The NN crank may not work well without a lot of tuning and tinkering. A number of initial decisions have to be made to run the program. For instance, how many sigmoids to use in the fit? (In their language, how many nodes to use in the hidden layer?) Each additional sigmoid used introduces $M + 1$ additional parameters to estimate. If too many sigmoids are used, there is the possibility of overfitting the data; too few and the data may be underfit.

Another problem is what initial values of the parameters to use. Gradient descent finds a nearby local minimum and the nonlinear surface generated by sums of sigmoids is guaranteed to have many local minima. One way to find the global minimum is to run the procedure many times starting from randomly selected initial values. But the lengthy running times of neural networks rule this out.

In my discussions with many practitioners concerning this problem, I ran into two schools of thought. One was "don't worry, all local mins give about the same accuracy." The second, and more surprising, was "never run for so long that you get into a local minimum."

The latter prescription defies the usual descriptions of how neural networks methodology works. But it seems to be followed by many of the most experienced and successful practitioners. The idea is this: given that you are minimizing over thousands of parameters, if you fall into the bottom of a minimum then you are overfitting the data. The "smart thing" to do is to set aside a test set, stop the program at various times, run the test set down the current predictor and select that point in the run that gives minimum test set error.

There are other recipes for avoiding overfitting. For instance, another current recipe is the use of regularization (aka "weight decay" in NN terms). Here, instead of minimizing the error sum of squares, a penalty term is added consisting of the sums of squares of the coefficients multiplied by a parameter to be determined. This method takes repeated runs, much more computing, and does not seem to have been widely adopted. Still another recipe advanced to me by knowledgeable users is to stop the run at various times and delete "inactive" variables from the fitting procedure.

Experienced users know how to tinker, cut and paste. They have their own ways of adjusting the number of nodes in the hidden layer to get good performance, and of preventing overfitting. But most of this is folk wisdom, and there is, so far, no handbook on the sacred mysteries of neural network tinkering.

There is nothing wrong with tinkering, but not enough is known about how best to tinker. There is not enough known about performance of neural networks on simple simulated data. We need to know more about the whys and wherefores, ifs and buts of NN performance.

4. ALTHOUGH SOME METHODS ARE USUALLY GOOD, NO METHOD IS ALWAYS BEST

Neural networks cannot satisfy the desire for ultimate optimality. It has become increasingly clear to the NN community that no one prediction method will be universally most accurate on all data and that what is best depends on the structure of the data. Because of this, a cottage industry in the invention of new methods has risen.

The methods generally fall into one of two categories. The first I call global. These methods (like neural nets) use the training data to estimate a global prediction surface. Local methods make a local prediction for each new vector x. For instance, in classification, the class predicted for x may be the class of its nearest neighbor in the training data.

To understand what is different and new about neural networks, we give a brief and selective overview of global methods currently used in nonlinear analysis.

5. GLOBAL METHODS

All current global predictive methods use selection of elements from a large set of basis elements. That is, one specifies a set of basis functions $\{B(x,\theta)\}, \theta\varepsilon\Theta$, defined on the space of predictor vectors such that "most" functions of x are in the span of the basis. Then, in regression, one uses a predictor function of the form

$$\hat{y}(x) = \sum_k \alpha_k B(x, \theta_k).$$

In classification, conditional probabilities are estimated as

$$\hat{p}(j \mid x) = G\left(\sum_k \alpha_{jk} B(x, \theta_k)\right)$$

for some specified function G. Here are some examples:

Neural Nets: $\{B(x,\theta)\} = \{\sigma(\theta \cdot x)\}, \Theta = \{E^M\}$.

CART: $\{B(x,\theta)\} = \{I(x\varepsilon R); R$ a rectangle in E^M, I an indicator function$\}$ Basis elements $I(x\varepsilon R_k)$ are selected such that the $\{R_k\}$ are disjoint with union E^M.

MARS: $\{B(x,\theta)\} = \{\Pi_i(\pm(x_{mi} - \theta_{mi})^+\}$, i.e., basis elements are products of a finite number of univariate linear splines.

For all of these sets of basis functions, various completeness theorems are known. These have the form: for all $f(x)$ of some specified smoothness and any $\varepsilon > 0$, there exists $K, \{\alpha_k, \theta_k\}$, $k = 1, \ldots, K$ such that

$$\|f(x) - \sum_k \alpha_k B(x, \theta_k)\| < \varepsilon.$$

This is comforting, but it leaves open questions important to applications:

> What are good sets of basis functions?
> How can a "good" subset of basis functions be selected?

There are drawbacks to the basis elements used in CART and MARS. CART can lose accuracy because its basis elements are discontinuous and are aligned with the coordinate axes. The MARS basis elements are continuous but unbounded. Also, with a high dimensional data set, it is not computationally feasible to include basis functions that are products of more that a few univariate splines.

The strategies familiar to statistics for selecting basis elements consists of stepwise "optimal" addition. For instance, in CART each current basis function is "optimally split" to give two new basis functions. A similar strategy is used in MARS. Because of this stepwise add-one-at-a-time approach and some clever algorithms, the basis selection procedure goes very rapidly. On the other hand, neural networks optimize the choices over all basis elements simultaneously using backpropagation.

6. WHAT IS UNIQUE AND DIFFERENT ABOUT NEURAL NETS?

Having come this far, we are in a position to venture some guesses as to why neural networks seem to give good results over a wide range of data bases. There may be two contributing factors. The first is that the basis element have desirable properties.

They are very smooth functions of linear functions and nicely bounded above and below. Their form, being close to zero in one portion of the space and close to one in another portion make them particularly good for approximating conditional probabilities and for approximating local ripples.

Another property may explain why NN users can throw in thousands of parameters and not have catastrophic overfitting. Usually, in starting the NN fit, one uses small random coefficients for the linear combinations in each sigmoid. If all of the coefficients in β are small, then $\sigma(\beta x) \cong .5 + .25\beta x$. Then, the sum over all sigmoid functions whose coefficients remain small collapses into a single linear function with the number of equivalent parameters equal only to the number of coordinates in the x-vector.

The other unique element in neural nets is the idea of simultaneously selecting all basis elements using backpropagation. My first impression of this method was that it was bound to fail by winding up in poor local minima. This does not seem to happen and the why is mysterious. It may be wound up in the nature of backpropagation. By this, I mean the particular procedure of entering one case at a time and then taking a partial gradient step.

For instance, the general wisdom is that one-case-at-a-time works better than putting in all of the data and doing "batch" gradient descent. Certainly, there are much faster methods for nonlinear optimization than gradient descent. But while these are faster, it is not known if they produce the accuracy given by backpropagation.

There is some research that claims to establish a link between backpropagation and stochastic optimization methods known to converge a.s. to the global optimum. If this is even partially true, then the method's largest drawback, its painfully slow running time, may also be a source of its consistent accuracy. Unfortunately, this is largely unexplored territory.

A possibility that Jerry Friedman and I are exploring is stepwise entry of sigmoid basis functions. We have designed a fast algorithm for stepwise entry of sigmoid functions patterned after the stepwise entry of hinge functions given in Breiman (1993). The procedure produces fits to the data in several orders of magnitude less running time than backpropagation. We have not done enough testing to know if the accuracy is competitive with neural networks using backpropagation.

7. CODA

I am fond of the saying "give a man a hammer and every problem looks like a nail". The NN community has their hammer. But they are also hard at work devising pliers, saws, chisels and a full repertory of tools, large and small. Interesting new methods are spawned at an almost alarming rate.

Among many recent results, here are a few that impressed me: A smart new metric leads to a nearest neighbor misclassification rate on optical character recognition about half that of a well-tinkered neural net procedure (Simard, Le Cun and Denker, 1993). Coding problems involving many classes into a sequence of two class problems results in significant decreases in error rates (Dietterich and Bakiri, 1991). Combining ("stacking") dissimilar classifiers also gives reduced error rates (Wolpert, 1992). Using linear programming methods to get nonlinear

separating boundaries between classes gives error rates on optical character recognition lower than neural nets (Boser, Guyon and Vapnik, 1992).

Often the analogies and language used in the NN community obscure the data analytic reality. There is a lack of reflective introspection into how their methods work, and under what data circumstances. But these lapses are more than offset by the complexity, interest, size and importance of the problems they are tackling; by the sheer creativity and excitement in their research; and by their openness to anything that works.

Comment: Neural Networks and Cognitive Science: Motivations and Applications

James L. McClelland

Artificial neural networks have come and gone and come again—and there are several good reasons to think that this time they will be around for quite a while. Cheng and Titterington have done an excellent job describing that nature of neural network models and their relations to statistical methods, and they have overviewed several applications. They have also suggested why neuroscientists interested in modeling the human brain are interested in such models. In this note, I will point out some additional motivations for the investigation of neural networks. These are motivations arising from the effort to capture key aspects of human cognition and learning that have thus far eluded cognitive science.

A central goal of congnitive science is to understand the full range of human cognitive function. During the 1960s and 1970s, when symbolic approaches to human cognition dominated the field, great progress was made in characterizing mental representations and in capturing the sequential thought processes needed, for example, to solve arithmetic problems, to carry out deductive reasoning tasks, even to prove theorems of logic from given axioms. Indeed, by 1980 a general computer program for solving integro-differential equations had been written. These accomplishments are certainly very valuable, yet they still leave many scholars of cognition with the very strong feeling that something very important is missing. Efforts in machine recognition of spoken and visual input, machine understanding of language, machine comprehension and analysis of text, not to mention machine implementation of creative or insightful thought, all continue to fall short. A huge gap remains between the capabilities of human and machine intelligence.

The interest in the use of neural networks among cognitive scientists springs largely from the hope that they will help us overcome these limitations. Although it is true that there is much to be done before this hope can be fully realized, there are nevertheless good reasons for thinking that artificial neural networks, or at least computationally explicit models that capture key properties of such networks, will play an important role in the effort to capture some of the aspects of human cognitive function that have eluded symbolic approaches. In what follows I mention two reasons for this view.

The first reason arises in the context of a broad class of topics that can be grouped under the rubric of "interpretation." A problem of interpretation arise whenever an input is presented to the senses, be it a printed digit, a footprint, a scientific argument or a work of creative expression such as a poem or a painting. The problem is to determine what the thing is or what it is intended to signify. The problem is difficult because the direct data is generally insufficient so that the ability to determine the correct interpretation depends on context.

Let us consider two examples. The first, shown in Figure 1, is from Massaro (1975) and illustrates the role of context in letter recognition. The same input gives rise to two very different interpretations depending on the context in which it occurs. The second comes from very simple stories of a kind studied by Rumelhart (1977):

> Margie was playing in front of her house when she heard the bell on the ice

James L. McClelland is Professor of Psychology and Professor of Computer Science, Department of Psychology, Carnegie Mellon University, Baker Hall 345-F, Pittsburgh, Pennsylvania 15213.