Neural Networks in Clinical Medicine

Will Penny and David Frost

Med Decis Making 1996 16: 386
DOI: 10.1177/0272989X9601600409

Neural Networks in Clinical Medicine

WILL PENNY, PhD, DAVID FROST, MB, PhD

Neural networks are parallel, distributed, adaptive information-processing systems that develop their functionality in response to exposure to information. This paper is a tutorial for researchers intending to use neural nets for medical decision-making applications. It includes detailed discussion of the issues particularly relevant to medical data as well as wider issues relevant to any neural net application. The article is restricted to back-propagation learning in multilayer perceptrons, as this is the neural net model most widely used in medical applications. *Key words:* neural networks; medical decision making; pattern recognition; nonlinearity; error back-propagation; multilayer perceptron. **(Med Decis Making 1996;16:386–398)**

Neural networks are statistical pattern-recognition machines composed of simple nonlinear processors connected into networks. Their design is inspired by knowledge from neurophysiology, though they are not biologically realistic in detail. They are parallel, distributed, adaptive information-processing systems that develop their functionality in response to exposure to information. These attributes differ from the usual (von Neumann) computational paradigm in which computation is the execution of a sequence of programmed instructions in a central processing unit. Neural nets have been applied to many problems, including handwriting recognition, speech recognition, the prediction of protein structure, the conversion of text to speech, and the prediction of stock market prices, to name but a few.[34,48]

It is not surprising, therefore, to see neural nets being used in medical decision making. In fact, the number of such applications has mushroomed in the last four years. In a recent review article,[88] 386 neural net applications in the biomedical sciences since the beginning of 1991 are referred to. Of these, 108 are based on actual clinical data. In 1990, a similar review[39] referred to fewer than 20 studies. Special sessions on medical applications at recent neural network conferences highlight the current interest.[86,87]

A picture of neural net research in medicine may be gained from reviews of neural nets in computer-aided diagnosis,[79] in pathology and laboratory medicine,[3] in medical imaging and medical signal processing (EEG,EKG),[52] in cancer research,[67,91] and medicine as a whole in Japan.[84]

This paper is a tutorial for researchers intending to use neural nets for medical applications. It includes detailed discussion of the issues particularly relevant to medical data and wider issues relevant to any neural net application. The article is restricted to back-propagation learning in multilayer perceptrons, as this is the neural net model most widely used for medical applications.

The article refers to applications from diverse medical disciplines to illustrate points. Quantitative results from these studies are summarized in table 1.

Although the article is self-contained from the point of view of medical applications, readers requiring more introductory material or greater detail are referred to introductory texts.[1,18,33,34] Readers wishing to apply the methods illustrated in this tutorial to existing medical problems are referred to an archive of medical data sets[62] and a review of freely available and commercial software.[58]

Neural Network Basics

A neural network consists of simple processing units called neurons or nodes, which are connected together to form the network. Each neuron sends signals to other neurons via connections, analogous to the axons in the central nervous system. The signals that a neuron receives from other neurons are weighted and then summed to produce an overall activity level in the neuron. Thus, if the ith neuron in a network receives signal x_j from the jth neuron, it is weighted by an amount w_{ij}. The total activity in the ith neuron is

Received March 9, 1995, from the Department of Psychiatry, University College London Medical School, Wolfson Building, Riding House Street, London W1N 8AA. UK. Revision accepted for publication October 27, 1995. Supported by the North East Thames Regional Health Authority.

Address correspondence and reprint requests to Dr. Penny.

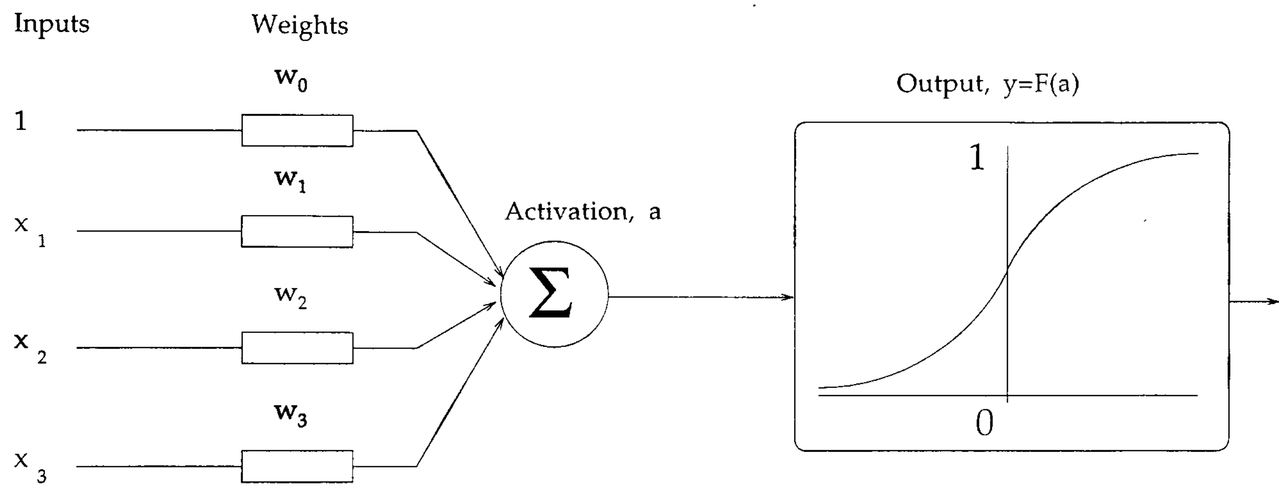

FIGURE 1. A perceptron.

$$a_i = w_{i0} + \sum_j w_{ij} x_j \tag{1}$$

where w_{i0} is a bias weight. The ith neuron responds to this activity by sending a signal

$$y_i = F(a_i) \tag{2}$$

This type of neuron, called a perceptron, is illustrated in figure 1. The standard choice for the function F is the nonlinear logistic or sigmoid function

$$F(a) = \frac{1}{1 + \exp(-a)} \tag{3}$$

which restricts the output to be between 0 and 1. If the incoming weighted activity is larger than the (negative) bias weight, the activation is positive. Positive activations cause node outputs that tendj to 1. Negative activations cause outputs that tend to 0. Thus, the bias weight acts as a threshold above which the node is active. For small activation levels, the sigmoidal function is approximately linear.

Perceptrons are the basic processing element in most neural network models. A feed-forward neural network, called the multilayer perceptron (MLP), is illustrated in figure 2. The network consists of sensory units that make up the input layer, one or more hidden layers of processing units (perceptrons), and one output layer of processing units (perceptrons). Every unit is connected to every unit in the layer below. The input signal propagates through the network a layer at a time. Because MLPs are trained with an algorithm called error back-propagation, they are also known as "backprop" networks.

There are many other types of networks, varying in node models and patterns of connectivity,[34,37,44,74] but the MLP is the network used in nearly all medical applications. Our discussion is therefore restricted to MLPs.

Overall, the MLP performs a functional mapping from the input space to the output space. The input and output spaces are multidimensional, with one dimension per input and output variable. The input–output mapping is determined by the structure of the network and the values of its weights. Changing the structure or the weights changes the function implemented.

An MLP with a single hidden layer having H hidden units and a single output, y, implements mappings of the form

$$y = F\left(w_0 + \sum_{h=1}^{H} w_h z_h\right) \tag{4}$$

$$z_h = F\left(\beta_{0h} + \sum_{j=1}^{N} \beta_{jh} x_j\right) \tag{5}$$

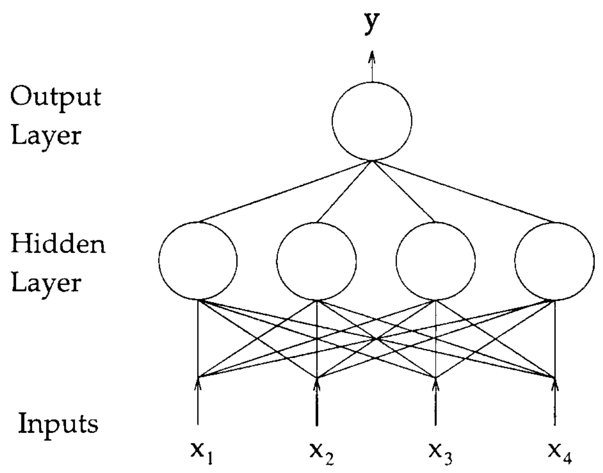

FIGURE 2. A multilayer perceptron. This is a two-layer perceptron with four inputs, four hidden units, and one output unit.

where z_h is the output of the hth hidden unit, w_h is the weight between the hth hidden unit and the output unit, and w_0 is the output bias. There are N sensory inputs, x_j. The jth input is weighted by an amount β_{jh} in the hth hidden unit. β_{0h} is the bias in that hidden unit.

The question then remains as to how the weights (w_0, w_h, β_{0h}, β_{jh}) are chosen for the network to solve the task at hand, whether it be diagnosis of myocardial infarction or prediction of cancer outcome. The answer is to choose those weights that best approximate the mapping $(x_1, x_2, \ldots, x_N) \Rightarrow y$ for every pairing of input vector and output in the training set. The network thus adapts its function according to the information it is presented with. This process is carried out by a training algorithm. The most popular training algorithm is error back-propagation, which, in conjunction with the MLP, is used in nearly all medical applications.

After a network has been trained, it may be used in the particular application environment. Its performance on a separate data set, known as the test set, indicates how well it is likely to perform.

A typical medical application is that of Baxt,[6] who used a neural net for the diagnosis of myocardial infarction. Baxt used 351 cases to train the network. Each consisted of an input vector containing 20 input predictors. These included details of the patient's history and electrocardiographic findings. Each output was a single variable signifying whether or not the patient had myocardial infarction. The network had 20 input nodes ($N = 20$), one per input variable, and a single output node. Two hidden layers of ten nodes each were used, as this was thought to give the network sufficient complexity to approximate the required mapping. This degree of complexity is unusual, in that most applications use only a single layer of hidden nodes. The network was trained with back-propagation. When tested on a further 331 cases, the network was able to diagnose both positive and negative cases more accurately than a clinician.

Any application of a neural network must address issues of linearity versus nonlinearity, training time, network structure, and network generalization. These are discussed below.

In medical applications there are further concerns. A typical medical data set is small, contains missing data items, and has low-prevalence categories. Neural nets must therefore be used with appropriate input and output representations and be assessed with suitable performance measures. Also, it is often necessary that some explanation be given as to how the network makes its decisions. How well the network performs in relation to other computational methods is also of interest. These issues are discussed later.

Linearity versus Nonlinearity: are Hidden Layers Necessary?

Interest in neural networks largely stems from their ability to implement nonlinear functions.

A perceptron can implement a nonlinear mapping by virtue of its nonlinear activation-output function, F. In effect, however, this nonlinearity acts only to monotonically rescale, or squash, the output of a linear transform. A perceptron can implement only linearly separable functions. This was pointed out in the late 1960s by Minsky and Papert,[53] whose work signalled a 20-year decline for neural net research.

Multilayer perceptrons with hidden layers, however, *can* implement nonlinearly separable functions. This was known in the 1960s, but until the mid-1980s there was no method of training them.

Multilayer perceptrons are universal approximators, which means that *any* mapping can be approximated. Specifically, an MLP with a single hidden layer and a sufficient number of hidden units can approximate any smooth function to an arbitrary degree of accuracy.[36] This includes all Boolean functions and any smooth nonlinear function.

In theory, there are some problems for which it is better to use a network with two hidden layers because the overall number of nodes will be less than it would be in a single-hidden-layer net.[37] There is a problem, however, in deciding how many units to have in a hidden layer (see below) which is compounded by having two of them. Moreover, there is no substantial practical evidence that more than one hidden layer adds to the predictive capabilities of a network. For these reasons, the majority of medical applications use single-hidden-layer networks.

Neural nets will offer better results than linear methods only if there are nonlinearities in the data. If a network with no hidden units or a single hidden unit performs as well as a network with a large number of hidden units, then the problem is linear. Not every medical data set contains nonlinearities, but many do. Of the studies listed in table 1, those of myocardial infarction,[8] low back pain,[11] anti-cancer agent discovery,[89] thyroid function,[62] and cancer outcome[13] all showed modest improvements in prediction accuracy over linear methods.

Back-propagation Training

Training algorithms for perceptrons have been around since 1960, when the delta rule was developed.[90] This is an iterative gradient-descent technique that is used to minimize the mean squared error (MSE) between target outputs, t_p, and actual outputs, y_p

$$MSE = \frac{1}{2}\sum_{p}(t_p - y_p)^2 \quad (6)$$

The perceptron is trained by first setting its weights, w_j, to random values. A single iteration of the delta rule is as follows. A training pattern, p, consisting of the input vector $(x_1, x_2, \ldots, x_N)$ and the output target t_p, is selected from the training set. The input vector is processed by the perceptron, which then produces an output y_p according to equations 1, 2, and 3. Each weight is then changed according to the delta rule for sigmoidal perceptrons

$$\Delta w_j = \alpha(t_p - y_p)x_j y_p(1 - y_p) \quad (7)$$

where Δw_j is the change in the jth weight and α is the learning rate. The other terms calculate the error gradient, $-dMSE/dw_j$. The learning rate is an adjustable factor that determines how large a step is taken in the direction of this gradient. High values give faster learning, but if the learning rate is too high the network's weights may oscillate around the solution, never quite converging to it.

The error gradient comprises $(t_p - y_p)$, otherwise known as the error signal δ; x_j, the jth input; and $y_p(1 - y_p)$, the derivative of the sigmoid function. Many iterations of the delta rule for each training pattern may be necessary before a solution is reached. Convergence to the global minimum of mean squared error is guaranteed.

Multilayer perceptrons can be trained with the *generalized* delta rule, otherwise known as error back-propagation or backprop, so-called because the error signals are propagated backwards through the net. This was popularized in 1986[69] and has led to a resurgence in neural net research.

The form of the learning rule is the same as the delta rule, but the way in which the error signals are generated differs. The errors for the nodes in the output layer are calculated exactly as before and the delta rule is implemented. These errors are then passed backwards to the previous layer, are weighted by the connections along which they are sent, and are then summed to produce total error contributions for each unit in the layer below. The delta rule is then implemented in these units with this modified error signal. For networks with more than one hidden layer, this process continues until all nodes have been updated. Full implementation details and descriptions of other training algorithms are available in introductory neural network texts.[33]

Convergence to a global minimum of mean squared error is, however, *not* guaranteed. This is because an MLP's weight space contains local error minima in which the back-propagation algorithm may become trapped. Different initial random-weight settings may therefore lead to different solutions. Thus, it is advisable to train networks a number of times from different initial settings.

A variant of backprop incorporates a momentum term in which a proportion of the previous weight-change value is added to the current value. This adds "momentum" to the algorithm's trajectory through weight space, which may prevent it from becoming trapped in local minima.

The time it takes to train a neural net increases exponentially with the number of network inputs,[80] and the number of network nodes,[40] and polynomially with the number of training examples.[81] A large network, with 200 inputs trained on a few thousand examples, takes about four hours to train on a SUN SPARC.[71] There is great utility, therefore, in including only those inputs and examples that seem relevant to the task at hand. This is not a significant problem for medical decision-making networks, as they are generally smaller. A typical medical net has 20 inputs and is trained on 350 examples (median values from table 1).

Generalization, Network Structure, and Stopped Training

After a network has been trained, it will be required to operate in its given problem domain. The generalization ability of a network is a measure of its performance on data not present in the training set. To assess how well the network will perform on novel data, the original data set is partitioned, usually into *two* sets; a training set and a generalization or test set.

The performance that the network achieves on its training set, however, does not necessarily reflect its performance on the test set or on future data. Unless care is taken, neural networks will learn features in the training set that are not present in the wider population of cases. This is known as fitting to noise or overfitting.

Overfitting will occur if the complexity of the network is greater than the complexity of the function being estimated. Thus, there is a link between the structure of the network and generalization performance.

The vast majority of studies use networks with one hidden layer, as these are sufficient to approximate any smooth nonlinear function. The neural net designer has only to decide, therefore, how many units to put in the hidden layer so as to define the network structure completely. Rigorous theoretical results relating the number of weights in a network, and thus the number of hidden units, to the number of training examples and likely generalization performance do exist.[5] This sort of analysis, however, allows the examples to come from any, possibly abnormal, probability distribution and so overestimates the number of examples required in practice.[16] There are, however, a number of rules of thumb.[5] Livingstone and Manallack's empirical work,[46] for example, suggests that

$$D = \frac{mo}{w} \tag{8}$$

where m is the number of training examples, o is the number of network outputs, and w is the total number of network weights, be greater than 3 to ensure good generalization and avoid memorization of the training set. This is typical of values used in practice (see table 1).

Thus, if there are 240 training cases and a single network output, the network should not have more than 80 weights. In a network with ten inputs this corresponds to having a single hidden layer with six units.

One strategy for choosing network structures is to start with the above number of hidden units, then reduce it until a generalizing network is obtained. Networks with zero or one hidden units should be tried to see whether or not the problem is linear. This procedure requires the examples to be split into *three* sets: a training set $\{L\}$, a validation set $\{V\}$, and a testing set $\{T\}$. The validation set is a pseudo-test set used to establish which network generalizes best. The final network performance is measured on $\{T\}$.

A different way to achieve matching of network complexity to data complexity is the stopped-training method.[25,56,62] It allows lower values of D to be used. Stopped training has been widely used in medical applications.[2,35,49,72,76,91]

Stopped training involves first choosing an overlarge network and setting its weights to *small* random values. The examples are again split into three sets: $\{L\}$, $\{V\}$, and $\{T\}$. The validation set is used to monitor the generalization error during training. In networks that are too large for the data, the validation error reaches a minimum before the training error does. Training is stopped at this earlier point and the minimum validation error is recorded. The network is then retrained on the pooled training and validation set until the previously recorded minimum validation error is reached. Network performance is then evaluated on the completely unseen test set.

Stopped training works by effectively placing a limit on the magnitude of the weights.[25] This corresponds to a limit on the complexity of the network function, because large weights are necessary for nodes to operate in their nonlinear range.

A third class of model-selection methods includes the weight decay or complexity regularization algorithms. These give each weight a tendency to decay to zero. Thus, connections that are not otherwise strengthened by backprop will disappear. The simplest algorithm sets the new weight to be a fraction, e, of the old weight, before backprop is implemented. Although this restricts weight size in a more direct way than stopped training, a suitable value for e must be found. This requires use of a validation set.

Finoff et al.[25] compare stopped training with complexity regularization and examine hybrid approaches.

Representation of Network Inputs

The variables in a medical problem will be a mixture of information from various sources; clinical diagnosis, patient records, examination results, laboratory tests, etc. These variables will be measured with various levels of intrinsic precision. There are three such levels: nominal, ordinal, and interval. The weakest level is nominal, where the values assumed by the variable simply indicate different categories. The variable "diagnostic group," for example, is nominal, since by assigning the numbers 0, 1, and 2 to "psychosis," "neurosis," and "organic" we can differentiate between categories. There is no ordering. An ordinal variable, however, allows ordering of categories. The variable "severity of depression," for example, is ordinal because it may take on one of a number of values indicating ordered severity, e.g., 1 = mildly depressed, 2 = moderately depressed, and 3 = severely depressed. Differences in the value of an ordinal variable, however, cannot be interpreted as distances between categories. Thus, someone with severe depression is not 3/2 times as depressed as someone with moderate depression. Interval variables such as age do have this property.

The way in which variables are presented to a neural net can be an important determinant of network performance. Apart from entering the input variables exactly as they are measured, there are two main coding schemes.

The first is to use a *1-out-of-C* code where a variable with C categories is converted into C Boolean inputs, each of which is high for a certain category or range. The second is a thermometer code, where an ordinal or interval variable is represented by $C - 1$ Boolean inputs, of which the leftmost k has value 1 to represent the kth category while all others are 0.[62]

The above coding schemes produce networks with more input units but possibly fewer hidden units. This is because hidden units that were previously needed to decode the more complex input representation are no longer necessary.[74]

Missing values can be represented by adding an extra input that is 1 if a variable is missing. This is somewhat extravagant, however, and should be used only when that variable is missing for a large proportion of cases. The strategy is also appropriate if the extra input can account for many variables' being missing together. For example, if arterial blood gas analysis is not ordered, the extra input "ABG not ordered" will account for all the variables pH, Pco_2, Po_2, etc., being missing.

An alternative strategy is to set missing Boolean inputs to the value 0.5. A less biased approach is to set missing inputs to their average non-missing values.[89] A fourth, but more computationally intensive, method is to set missing values to a value found by the expectation-maximization (EM) algorithm.[19] This uses a probabilistic model of the input space where the parameters of the model (means, covariances, etc.) are initially guessed at. The values of the missing inputs are then calculated from these parameter values and other input values. This is the expectation step. The parameters that maximize the likelihood of observing these data given the model are then calculated. This is the maximization step. The two steps are repeated until the algorithm converges. The EM algorithm is impractical unless it can be automated in software.

A further issue concerns scaling. Rescaling the input variables to have zero mean and unit variance can reduce training time.[74] Certainly, no learning can take place in weights connected to inputs that are zero because the corresponding weight change, according to the delta rule, is zero. After training, the weights can be resealed to the original range of the training inputs, allowing test inputs to be applied without any transformation.

Interpretation of Network Outputs

The majority of medical applications of neural nets involve decision making. Thus, the target outputs are nominal or ordinal variables. When there are n distinct output categories, n output nodes can be used. If the ith target output is 1 when the ith category is present and the other targets are 0, we have a *1-out-of-n* coding. If this scheme is used, the outputs of a trained network may be interpreted as posterior probabilities.[66] That is, output i is the probability that the category is i given that the input vector x has been observed, $p(i/x)$. If a novel input is presented and the ith output is the highest, a commonsense decision rule is to assign that pattern to the ith category. This corresponds to picking the category with the highest posterior probability and results in a network with minimum classification-error rate.[20,66] The accuracy with which the posterior probabilities are approximated is dependent upon training set size and requires that the training algorithm find a global minimum of the cost function.[66]

If the prevalence (prior probability) of an output category in the training set is low, however, then information about that category will be largely ignored by the network. Specifically, in a neural network trained on a *1-out-of-n* code, the transform implemented in the hidden layer weights each class in proportion to the *square* of the number of patterns in that class.[47] Thus, the network disproportionately (that is, not in linear proportion) misclassifies those patterns in low-prevalence classes in order to correctly classify patterns in high-prevalence classes.

In many medical situations it is difficult to obtain a large amount of data about a particular outcome or diagnosis of interest. Thus, it is inevitable that low-prevalence classes are common, even though we may wish the network to classify these classes just as accurately as more prevalent ones.

To accommodate this type of situation, Smith[74] suggests using a weighted mean-squared-error criterion during training, where the error from underrepresented class patterns is magnified by a weighting factor. This corresponds to simply replicating underrepresented class patterns. The converse approach, namely that of reducing the number of overrepresented class patterns, throws away information unnecessarily.

Lowe and Webb[47] take a more general and rigorous approach that, by manipulation of error weightings and target values, allows arbitrary importance to be placed on each class. One suggestion is the use of a $1/\sqrt{p_i}$*-out-of-n* output coding where p_i is the prevalence of the ith class. This ensures that the network constructs a discriminant function in which the classes are weighted linearly according their prevalences (rather than in proportion to the square).

Further, if the relative costs of different types of misclassification are known, they can be used directly to train the network. If a pattern in class i is presented during training, then the output targets are t_j where t_j is the cost of misclassifying a class i pattern as a class j pattern. In testing, one assigns a pattern to the class having the smallest network output. This output coding scheme produces networks that minimize the misclassification *cost*.[47]

These latter two schemes require linear activation–output functions in the output nodes because the targets may be greater than 1. For *1-out-of-n* coding schemes, the output nodes use logistic functions. Perturbation of these target values by a small amount prevents the weights from assuming large magnitudes and so speeds learning.[33]

Performance Measures

The majority of medical applications are classification problems. A network's performance is therefore measured by how well it discriminates between classes. To force the network into decisions, its final output is compared with a decision threshold* that is chosen to maximize performance on the training set.

Measures such as the mean squared error or cross entropy,[33] which are used to train the network, indicate how well the outputs are *calibrated* to their targets, but do not necessarily measure the network's *discrimination* ability.

A more suitable measure is the network's overall classification rate. But if quoted on its own it can be misleading. The performances of two very simple decision algorithms should also be quoted. The first is "always choose the class having the maximum prior probability." This is known as the no-data rule or the default rule. The second is "make a random decision where class i is chosen with a probability equal to the prior probability of class i." This rule classifies at the chance agreement rate.

These precautions are necessary because in medical applications the prior probabilities of different classes are usually quite uneven. This is illustrated in table 1, where the median prior probability of the most prevalent class (in two-class problems) is 71%.

These issues are illustrated in an application that attempted to predict psychiatric stays as short or long.[17] An MLP achieved an overall classification rate of 74%. This appears significant until one finds out that 73% of the examples were short stays. Thus, a 73% classification rate is achievable with the default rule, so the network's performance is clearly unimpressive.

For networks that have a single output, classification rates on positive target outputs, the sensitivity, and classification rates on negative target outputs, the specificity, can be quoted.[14] Furthermore, as each of these values is dependent upon the choice of decision threshold, a graph of sensitivity versus 1 minus specificity, known as the receiver operating characteristic (ROC), can be plotted through various decision threshold values.[50] The area under this curve (AUROCC) then gives a definitive measure of the classifier's discrimination ability that is not dependent on the choice of decision threshold. It is identical to the probability that, given a positive case and a negative case, the network output will be higher for the positive case. Hanley and McNeil[29] describe algorithms for calculating the AUROCC and describe its relation to other measures of rank correlation.

It is also useful to measure the proportion of cases that the network classifies as positive that actually are positive, the positive predictive value, and the proportion of cases that the net classifies as negative that are actually negative, the negative predictive value. These quantities are, however, dependent on the prevalence of positive cases.[14] Even for classifiers that have high sensitivity and specificity, it is impossible to get a high positive predictive value if the prevalence of positive cases is low. The best predictive value is achieved when the prevalence of the class of interest is 0.5.

Other summary measures of the network's discrimination ability are the kappa value and the information gain. Unlike the AUROCC, these require an output threshold to be chosen. Kappa is the actual improvement in classification rate over the chance rate divided by the maximum possible improvement over the chance rate.[73] A value of 1 indicates perfect classification and a value of 0 indicates classification at the chance rate. The information gain is the decrease in classification uncertainty after having observed the network output.[75] Somoza and Mossman[75] describe an algorithm for finding the output threshold that maximizes the information gain.

If the relative costs of different types of misclassification, i.e., the costs of false-negative and false-positive decisions, are known, then an overall cost measure can be calculated. Alternatively, the network can be trained to output these values directly,[47] as described above.

For networks with many outputs, individual performance measures can be quoted separately or averaged across all outputs.

Cross-validation and Bootstrapping

The performance measures described above should be made on both the training set and the test set. Only if the test set has in no way been used to set the network's weights or evaluate its structure will it reflect the network's performance on future data. In the statistics literature, the procedure of splitting data into a training set and a test set is called cross-validation. If there is plenty of available data, this procedure is suitable.

Medical data sets, however, are often small. When this is the case, the method of v-fold cross-validation is applicable.[78] The data are split into v roughly equal-sized parts where, typically, $5 \leq v \leq 10$. The network is then trained and tested v times. On the ith iteration, the network is trained on all parts except i and then tested on the ith part. The cross-validated performance measures are then the average test performances across all v iterations. If v is equal to the number of data items, the technique is called leave-one-out cross-validation, or the jackknife.

Another statistical technique for small data sets is

*This is not the same as the bias weight in the output node.

the bootstrap.[22] One of its uses is for estimating the value of the use of a performance measure, θ, on future data. All available examples are first placed in a training set. A network is then trained and θ is measured. *B* samples are then generated, each one of size *m* drawn with replacement from the *m* training examples. For the *b*th bootstrap sample, a network is trained and θ is measured. The θ value obtained when this *b*th network is tested on the original sample is also measured. The difference between these two values is recorded. The average difference across all *B* bootstrap samples is then subtracted from the original θ value to give an estimate of θ for future data.‡

The bootstrap method has been used to estimate prediction errors in decision-tree classifiers[22] but has only recently appeared in the neural network literature.[82] A drawback of the bootstrap is that $20 \leq B \leq 200$ networks must be trained.

Bootstrapped estimates of performance measures are less variable than cross-validated estimates but can be (optimistically) biased. The choice of validation method depends on the data set and software used.

We re-emphasize that the test set may not be used in any way during the training process.

‡This is the bootstrap-pairs method.

Sensitivity Analysis and Rule Extraction

Because the path between an input and an output is so complex, it is difficult to appreciate exactly how an output is dependent on any individual input. Also, as this relationship is embodied in a mathematical equation rather than a set of rules, the workings of the network seem somewhat obscure.

It has been argued[32] that other machine learning algorithms such as C4.5 and CART are more likely to be accepted in actual clinical settings because they produce sets of rules that are easy to interpret.

Another point of view[28] is that there should be no problem for clinicians to put their faith in "black box" systems because they already do in everyday clinical practice; expert clinicians making intuitive judgments are often unable to explicitly pass on their thought processes to others. Neural networks should therefore be judged on their performances on prospective data and field trials just as any other new clinical technology is.

However, the workings of a neural set *can*, to some extent, be explained. For example, in using a network to perform survival analysis on censored data, Delaurentis and Ravdin[43] explored the interactions between predictive variables and survival rate. They concluded that the network "was not a black box but could lead to useful insights into the roles played by different prognostic variables in determining patient outcome."

One method is to rank inputs according to how much the testing error increases when the individual input is not used in the training or testing of the network. This leave-one-out method has been used, for example, to find which factors were predictive of admission into a psychiatric ward from a network trained on data from a psychiatric emergency room.[76] The method, however, may require considerable computation. For a network with *n* inputs, an extra *n* networks must be trained and tested. Furthermore, this method does not indicate the polarity of the relationships.

A different approach is sensitivity analysis.[68,85] Each input is varied, and the corresponding change in output is measured. The ratio, change in output over change in input, $\delta y/\delta x_i$, is then averaged over all examples to produce a sensitivity measure for each input $\langle \delta y/\delta x_i \rangle$. The inputs can then be ranked according to sensitivity.

The method has been used to find which factors were predictive in psychiatric outcome,[54] breast and ovarian cancer,[91] early prediction of heart attack,[31] and diagnosis of myocardial infarction in emergency room patients.[7] The method is particularly suited to psychiatric data as these data sets often have large numbers of inputs (see table 1), only a few of which turn out to be useful.

Sensitivity analysis, however, may be misleading. This is because *y* is not necessarily a linear function of x_i. Thus dy/dx_i and therefore $\delta y/\delta x_i$ will not be constant across x_i. The resulting sensitivity measures are therefore dependent on the particular examples used to train the network. They are also dependent on the initial weight-setting of the network, because different initial weight settings may result in different solutions. Average values from a number of network runs with different training examples and initial conditions are therefore more reliable.[10]

In the myocardial infarction application, results from a single trained network showed that the network was sensitive to electrocardiographic variables that had in the past been shown to be good predictors. It also identified three other predictors not before known for their predictive ability. In a later study, Baxt and White[9] used bootstrap estimates of sensitivity measures to eliminate uncertainty due to random-sampling variation. They concluded that only two of the three new predictors were truly predictive.

Gallant[27] and Saito and Nakano[70] describe methods for extracting "if–then" rules from neural networks. Saito and Nakano's "rules from network" method is based on the idea of growing regions around training examples that the network correctly classifies as positive by expanding the range of one input dimension at a time about its original value until further expansion would result in negative classification by the network. The resulting hyper-rectangle is then operated on by subtracting out regions around negative examples. All resulting terms are then joined by logical "ors." The algorithm produces rules of the form "if (x between x1 and x2 and y between y1 and y2) or (x between x3 and x4 and y between y3 and y4), then diagnosis = positive."

Gallant also describes an approach, called "key-factor justification," that is used to explain individual decisions. Each input is, in turn, reversed. If the output decision is reversed as a consequence, then we have found a key factor. Explanations are of the form "myocardial infarction has been diagnosed because rales (an abnormal chest sound indicative of fluid in the lungs) are heard." If no single key factor can be identified, then pairs of variables or triples are reversed together. To go beyond triples may require too much computation.

Other Paradigms

There are many paradigms for computerized decision making other than neural networks. These include the methods of regression,[41] discriminant analysis,[20,41] pattern recognition,[20] probabilistic reasoning and Bayesian decision theory,[60] decision tree classifiers,[12,26,64] knowledge-based expert systems,[38] curve fitting,[24] and fuzzy logic.[42] There are also similarities between neural nets and a number of these methods.

Univariate regression, correlation measures, or t-tests[14] can be used as baseline performance measures of how well the output variable can be predicted. They can also be used to select inputs for other models.

Multivariate regression methods are applicable for predicting continuous variables. They can be categorized into linear, logistic, and polynomial or other nonlinear schemes. The regression function is that function that minimizes the squared error and is found by matrix inversion. The nonlinear scheme requires that nonlinear terms be entered explicitly as separate inputs. This is often impractical, as the relationship between variables is unknown a priori and the number of possible nonlinear interactions grows exponentially with the number of inputs. A perceptron that has a linear activation–output function trained with the delta rule approximates the linear regression method.[18] Similarly, a perceptron with a logistic activation–output function approximates logistic regression.[77] Multilayer perceptrons are known as "model-free estimators" because they can learn the same functions as nonlinear regression methods but do not require the nonlinear terms to be entered explicitly.

Discriminant-function analysis is appropriate for predicting nominal variables. It is otherwise very similar to regression analysis. The discriminant function is chosen to maximize class separation and is found by matrix inversion. Linear discriminant functions may be implemented in a perceptron and higher-order functions, such as quadratic functions, by an MLP. Again, the MLP has the advantage of being a model-free estimator.

A common pattern-recognition method is the nearest-neighbor classifier. This works by calculating the distance, usually the Euclidian distance, between the pattern to be classified and the patterns already seen. The new pattern is then assigned to the class of its nearest neighbor or to that class most frequently represented among the *k* nearest neighbors. These classifiers are practical when large amounts of memory and sufficient computation power are available. This is not a significant problem for medical data sets, which are often small, but the problem here is in finding a distance metric that works with nominal input variables.

Bayesian classifiers estimate class probability density functions from training patterns and a priori information and then use Bayesian decision theory to classify patterns. Research has focused on parametric classifiers where the form of probability distribution is assumed to be known and only the parameters need be estimated. A frequent assumption is that examples are drawn from a multivariate Gaussian distribution. Hence, only the mean and covariances need by calculated from the training data. A further assumption is that groups of inputs or all of the inputs are independent. A Bayesian classifier for independent Gaussian inputs can be implemented with a perceptron.[44] Multilayer perceptrons have been shown to be more accurate than Bayesian methods in a number of empirical studies.[45] This is not always so, however, as shown by the head-injury study in table 1, in which a Bayesian classifier was more accurate.

Decision-tree classifiers such as classification and regression trees (CART) and C4.5 have been developed extensively over the past ten years. These classifiers extract rules from data by building decision trees. The simplest classifier builds a decision tree by computing inequalities on inputs, an input at a time. They are much faster to train than MLPs, their operation is easily understood in terms of "if-then" rules, and they are of comparable accuracy. Like regression methods, however, they may require certain features to be inputted explicitly rather than be discovered. This is exemplified in the problem of

Table 1 • 25 Neural Network Studies in Medical Decision Making*

	No. of Examples					Accuracy§	
Subject	Training	Test	P†	Network	D‡	Neural	Other
Breast cancer[4]	57	20	60	9–15–2	0.6	80	75
Vasculitis[2]	404	403	73	8–5–1	8.0	94	—
Myocardial infarction[6]	351	331	89	20–10–10–1	1.1	97	84
Myocardial infarction[8]	356	350	87	20–10–10–1	1.1	97	94
Low back pain[11]	100	100	25	50–48–2	0.2	90	90
Cancer outcome[13]	5,169	3,102	—	54–40–1	1.4	0.779	0.776
Psychiatric length of stay[17]	957	106	73	48–400–4	0.2	74	76
Intensive care outcome[23]	284	138	91	27–18–1	0.5	0.82	0.82
Skin tumor[21]	150	100	80	18	—	80	90
Evoked potentials[35]	100	67	52	14–4–3	3.8	77	77
Head injury[47]	500	500	50	6–3–3	20	66	77
Psychiatric outcome[54]	289	92	60	41–10–1	0.7	79	—
Tumor classification[55]	53	6	38	8–9–3	1.4	99	88
Dementia[57]	75	18	19	80–10–7–7	0.6	61	—
Pulmonary embolism[59]	607	606	69	50–4–1	2.9	0.82	0.83
Heart disease[62]	460	230	54	35–16–8–2	3	83	84
Thyroid function[62]	3,600	1,800	93	21–16–8–3	22	98	93
Breast cancer[62]	350	175	66	9–4–4–2	10	97	96
Diabetes[62]	384	192	65	8–4–4–2	12	77	75
Mycardial infarction[63]	2,856	1,429	56	291–1	9.8	85	—
Hepatitis[65]	39	42	38	4–4–3	3.3	74	79
Psychiatric admission[76]	319	339	85	53–1–1	6.0	91	—
Cardiac length of stay[83]	713	696	73	15–12–1	3.5	0.70	—
Anti-cancer agents[89]	127	141	25	60–7–6	1.5	91	86
Ovarian cancer[91]	75	98	—	6–6–2	2.6	84	81
MEDIAN VALUE	350	175	71	20	2.8		

*For reference citations, see the reference list
†P = prior probability of most prevalent category.
‡D = ratio of training examples to weights per output.
§A single integer in the accuracy column denotes percentage overall classification rate and a single real number between 0 and 1 indicates the AUROCC value. Neural = accuracy of neural net, Other = accuracy of best other method.

differential identification of fatty liver and two types of hepatitis on the basis of laboratory tests.[65] CART required that the ratio of two inputs be entered explicitly as a third input. Without this extra input, CART would not classify as accurately as a neural net.

Knowledge-based expert systems have been widely used in the medical domain. The difficulty in eliciting rules from experts and the inconsistency and brittleness of resulting systems have been their main drawbacks. Neural networks offer a more direct approach but have the disadvantage that their workings are not readily interpreted.

Curve-fitting methods such as generalized spline fitting are similar to regression methods. A difference is that the data may be approximated by many local functions, which are then combined to form a single global nonlinear function.

Fuzzy-logic systems implement general nonlinear functions, which are initialized by heuristic, expert knowledge. They are based on readily understood but vague linguistic rules, which are given precise meaning via algebraic operators called "membership functions."

Curve-fitting[61] and fuzzy-logic methods[30] are similar to a type of neural network called a "radial basis function network." This is a two-layer network with Gaussian activation–output functions in the hidden layer and linear functions in the output layer.

Considerable research effort is being devoted to systems involving combinations of the above-mentioned methods and neural networks. A recent selection of studies involving such "hybrid" systems for medical reasoning is given by Cohen and Hudson.[15]

Table 1 shows how accurate neural nets are in comparison with other methods. Michie et al.[51] compare machine learning, neural nets, and statistical classifiers on a variety of data sets, including classifications of heart disease, head injury, and diabetes.

Conclusion

Certain issues must be addressed for neural networks to truly perform well in medical applications. These include choosing input and output representations and performance measures that are suitable for the low-prevalence categories and missing data items often found in medical data sets. If the data set is small, then the statistical techniques of folded cross validation and bootstrapping allow a more accurate assessment of the network's performance.

Table 2 • Checklist for Applying Neural Networks to Problems in Clinical Medicine

Step 1	Data preparation
	• Expert selection of relevant clinical predictors. • Choose input coding from 1-of-C, thermometer, or original values. • Missing values: replace by average, have extra input that is high for missing values, or use EM algorithm. • Rescale all inputs to have zero mean and unit variance. • Choose output coding from ordinal, 1-of-N, $1/\sqrt{p_i}$-out-of-n, or cost targets.
Step 2	Baseline performance measures
	• Univariate t-tests or correlations. • Calculate default and chance classification rates. • Linear, logistic regression and other classification methods.
Step 3	Preliminary training experiments
	Select network NET with h hidden units such that number of weights = (no. of examples × no. of outputs)/3. • Train NET on whole data set with backprop to find good values of learning rate, momentum, number of training epochs, and other parameters. Use this parameter set in all further experiments.
Step 4	Select validation method
	For large data sets use cross validation: split data into three sets $\{L\}$, $\{V\}$, $\{T\}$ • For small data sets use v-fold cross-validation: split data into $\{X\}$ and $\{T\}$. Then split $\{X\}$ into v roughly equal-sized parts. On ith fold, ith part is $\{V\}$, remainder is $\{L\}$. • The bootstrap is also suitable for small data sets.
Step 5	Model selection
	Choose performance measures such as overall classification rate, AUROCC, kappa, or information gain. Use one of three model-selection methods:
	• Vary number of hidden units: Starting with NET, Train on $\{L\}$, test on $\{V\}$. Repeat for various h. Select h that gives best performance on $\{V\}$.
	• Stopped training: Select network LARGENET with at least as many hidden units as NET. Train on $\{L\}$. After each training epoch, test on $\{V\}$. Stop training at epoch that gives best performance on $\{V\}$.
	• Weight decay/complexity regularization: With LARGENET train on $\{L\}$, test on $\{V\}$ for various weight decay or regularization parameters, e. Select value of e that gives best performance on $\{V\}$.
	Test resulting network model on $\{T\}$ to give final result.
Step 6	Model explanation
	Use sensitivity analysis, rule extraction, key factor justification, or the leave-one-out method to explain how the network works.

Sensitivity analysis and rule-extraction methods are available to explain how a neural network's decisions are made. Performance comparisons with linear classifiers allow assessment of the nonlinearity in the data. Table 2 summarizes the steps involved.

Of the studies we have examined, the decisions made by neural networks are in most cases as accurate as a clinician's decisions. Nonlinear neural networks offer a modest increase in classification accuracy over linear classifiers. This indicates that there are nonlinearities in clinical data and that it is worthwhile using a neural net to model them. Other nonlinear methods make medical decisions as accurately as neural networks. In comparison with these other methods, neural networks seem most useful in those areas where there is little a priori knowledge, because of their ability to detect nonlinearities that are not explicitly formulated as inputs. They are thus able to identify factors that were not previously known to have predictive ability.

Bibliography

1. Aleksander I, Morton H. An Introduction to Neural Computing. Chapman and Hall, 1991.
2. Astion ML, Wener MK, Thomas RG, Hunder GG, Bloch DA. Overtraining in neural networks that interpret clinical data. Clin Chem. 1993;39:1998–2004.
3. Astion ML, Wilding P. The application of backpropagation neural networks to problems in pathology and laboratory medicine. Arch Path Lab Med. 1992;116:995–1001.
4. Astion ML, Wilding P. Application of neural networks to the interpretation of laboratory data in cancer diagnosis. Clin Chem. 1992;38:34–8.
5. Baum EB, Haussler D. What size net gives valid generalization? Neural Comput. 1989;1:151–60.
6. Baxt WG. Use of an artificial neural network for the diagnosis of myocardial infarction. Ann Intern Med. 1991;13:843–8.
7. Baxt WG. Analysis of the clinical variables driving decision in an artificial neural network trained to identify the presence of myocardial infarction. Ann Emerg Med. 1992;21:1439–44.
8. Baxt WG. A neural network trained to identify the presence of myocardial infarction bases diagnostic decision on nonlinear relationships between input variables. Neural Comput Appl. 1993;1:176–82.
9. Baxt WG, White H. Bootstrapping confidence intervals for clinical input variable effects in a network trained to identify the presence of myocardial infarction. Neural Comput. 1995; 7:624–38.
10. Belue LM, Bauer KW. Determining input features for multilayer perceptrons. Neurocomputing. 1995;7:111–22.
11. Bounds DG, Lloyd PJ, Mathew BG. A comparison of neural network and other pattern recognition approaches to the diagnosis of low back disorders. Neural Networks. 1990;3: 583–91.
12. Breiman L, Friedman J, Olshen R, Stone C. Classification and Regression Trees. Belmont, CA: Wadsworth, 1984.
13. Burke HB, Goodman PH, Rosen DB. Artificial neural networks for outcome prediction in cancer. Proc World Conference on Neural Networks, San Diego, CA, June 5–9, 1994, 53–6.
14. Campbell MJ, Machin D. Medical Statistics: A Commonsense Approach. New York: John Wiley and Sons, 1993.

15. Cohen ME, Hudson DL. Advances in Fuzzy Systems—Applications and Theory. Vol. 3. Comparative Approaches to Medical Reasoning. World Scientific, 1995.
16. Cohn D, Tesauro G. How tight are the vapnik-chervonenkis bounds? Neural Comput. 1992;4:249–69.
17. Davis G, Lowell WE, Davis GL. A neural network that predicts psychiatric length of stay. MD Comput. 1993;10:87–92.
18. McClelland JL, Rumelhart DE. Parallel Distributed Processing. Cambridge, MA: MIT Press, 1986.
19. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the em algorithm. J R Statist Soc B. 1977;39:1–38.
20. Duda RO, Hart PE. Pattern Classification and Scene Analysis. London, U.K.: Wiley-Interscience, 1973.
21. Durg A, Stoecker WV, Cookson JP, Umbaugh SE, Moss RH. Identification of variegated coloring in skin tumors. IEEE Eng Med Biol. September 1993, pp 71–4.
22. Efron B, Tibshirani RJ. An Introduction to the Bootstrap. London, U.K.: Chapman and Hall, 1993.
23. Doig GS, et al. Modeling mortality in the intensive care unit: comparing the performance of back-propagation, associative-learning neural network with multivariate logistic regression. Proc Annu Symp Comput Appl Med Care. 1993; pp 361–5.
24. Eubank RL. Smoothing Splines and Nonparametric Regression. New York: Marcel Dekker, 1988.
25. Finoff W, Hergert F, Georg Zimmermann HG. Improving model selection by nonconvergent methods. Neural Networks. 1993;6:771–83.
26. Friedman JH. A recursive partitioning decision rule for nonparametric classification. IEEE Trans Comput. 1977;16:404–8.
27. Gallant S. Neural Network Learning and Expert Systems. Cambridge, MA: MIT Press, 1994.
28. Guerriere MRJ, Detsky AS. Neural networks: what are they? Ann Intern Med. 1991;115:906–7.
29. Hanley JA, McNeil BJ. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology. 1982;143:29–36.
30. Harris CJ, Moore CG, Brown M. Intelligent Control: Some Aspects of Fuzzy Logic and Neural Networks. World Scientific, 1993.
31. Harrison RF, Marshall SJ, Kennedy RL. The early diagnosis of heart attacks: a neurocomputational approach. International Joint Conference on Neural Networks, V1, 1991, pp 1–5.
32. Hart A, Wyatt J. Evaluating black-boxes as medical decision aids: issues arising from a study of neural networks. Med Informatics. 1990;15:229–36.
33. Haykin S. Neural Networks: A Comprehensive Foundation. New York: Macmillan, 1994.
34. Hertz J, Krogh A, Palmer RG. Introduction to the Theory of Neural Computation. Reading, MA: Addison–Wesley, 1991.
35. Holdaway RM, White MW, Marmarou A. Classification of somatosensory-evoked potential recorded from patients with severe head injuries. IEEE Eng Med Biol. 1990;9:43–9.
36. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks. 1989;2:359–66.
37. Hush DR, Horne BG. Progress in supervised neural networks. IEEE Signal Processing Magazine. January 1993, pp 8–39.
38. Jackson P. Introduction to Expert Systems. Reading, MA: Addison–Wesley, 1990.
39. Jones D. Neural networks for medical diagnosis. In Maren AJ, Harston CT, Papp RM (eds). Handbook of Neural Computing Applications. San Diego, CA: Academic Press, 1990, pp 309–17.
40. Judd SJ. Neural network Design and the Complexity of Learning. Cambridge, MA: MIT Press, 1991.
41. Kleinbaum DG, Kupper LL, Muller KE. Applied Regression Analysis and Other Multivariable Methods. Boston, MA: PWS–Kent, 1988.
42. Kosko B. Fuzzy Thinking: The New Science of Fuzzy Logic. London, U.K.: Flamingo, 1994.
43. De Laurentiis M, Ravdin PM. A technique for using neural network analysis to perform survival analysis of censored data. Cancer Lett. 1994;77:127–38.
44. Lippmann RP. An introduction to computing with neural nets. IEEE ASSP Magazine. 1987;4(2):4–22.
45. Lippmann RP. Pattern classification using neural networks. IEEE Communications Magazine. November 1989, pp 47–64.
46. Livingstone DJ, Manallack DT. Statistics using neural networks: chance effects. J Med Chem. 1993;36:1295–97.
47. Lowe D, Webb AR. Exploiting prior knowledge in network optimization: an illustration from medical prognosis. Network: Computation in Neural Systems. 1990;1:299–323.
48. Maren AJ, Harston CT, Papp RM (eds). Handbook of Neural Computing Applications. San Diego, CA: Academic Press, 1991.
49. Marshall SJ, Harrison RF, Kennedy R. Neural classification of chest pain symptoms: a comparative study. IEEE Artificial Neural Networks, 1991, pp 200–4.
50. Meistrell ML. Evacuation of neural network performance by receiver operating characteristic (ROC) analysis: examples from the biotechnology domain. Comput Meth Programs Biomed. 1990;32:73–80.
51. Michie D, Spiegelhalter DJ, Taylor CC. Machine Learning, Neural and Statistical Classification. Ellis Horwood, 1994.
52. Miller AS, Blott BH, Hames TK. Review of neural network applications in medical imaging and signal processing. Med Biol Eng Comput. 1992;30:449–64.
53. Minsky ML, Papert SA. Perceptrons. Cambridge, MA: MIT Press, 1969.
54. Modai I, Stoler M, Inbar-Saban N, Saban N. Clinical decisions for psychiatric inpatients and their evaluation by a trained neural network. Meth Inf Med. 1993;32:396–9.
55. Molnar B, Szentirmay Z, Bodo M, Sugar J, Feher J. Application of multivariate, fuzzy set and neural network analysis in quantitative cytological examinations. Analyt Cell Pathol. 1993;5:161–75.
56. Morgan N, Boulard H. Generalization and parameter estimation in feedforward nets: some experiments. Neural Information Processing Systems 2. San Mateo, CA: Morgan Kauffman, 1990, pp 630–7.
57. Mulsant BH, Servan-Schreiber E. A connectionist approach to the diagnosis of dementia. Proc Annu Symp Comput Appl Med Care. 1988, pp 245–50.
58. Murre JMJ. Neurosimulators. In Arbib MA (ed). Handbook of Brain Research and Neural Networks. Cambridge, MA: MIT Press, 1995.
59. Patil S, Henry J, Rubenfire M, Stein P. Neural network in the clinical diagnosis of acute pulmonary embolism. Chest. 1993; 104:1685–9.
60. Pearl J. Probabilistic Reasoning in Intelligent Systems. San Mateo, CA: Morgan Kauffman, 1988.
61. Poggio T, Girosi F. Regularization algorithms for learning that are equivalent to multilayer networks. Science. 1990;247: 978–82.
62. Prechelt L. Proben 1—a set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fakultat fur Informatik, Universitat Karsruhe. Available via ftp.ira.uka.de in directory /pub/papers/techreport/1994, 1994.
63. Quandt K, Waschulzik T, Lewis M, Hormann A, Engelbrecht R, Brauer W. Evaluation of epidemiological data with neural networks. Proc World Conference on Neural Networks, San

Diego, CA, June 5–9, 1994, pp 257–60.
64. Quinlan JR. Induction of decision trees. Machine Learning. 1986;1:81–106.
65. Reibnegger G. Neural networks as a tool for utilizing laboratory information: comparison with linear discriminant analysis and with classification and regression trees. Proc Nat Acad Sci. 1988;88:11426–30.
66. Richard MD, Lippmann RP. Neural network classifiers estimate Bayesian a posteriori probabilities. Neural Computation. 1992;4:461–83.
67. Rogers SK, Ruck DW, Kabrisky M. Artificial neural networks for early detection and diagnosis of cancer. Cancer Lett. 1994;77:79–83.
68. Ruck DW, Rogers SK, Kabrisky M. Feature selection in feedforward neural networks. Neural Network Comput. 1990;20: 40–8.
69. Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. In Rumelhart DE, Mclelland JL (eds). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Cambridge, MA: MIT Press, 1994, pp 318–64.
70. Saito K, Nakano R. Rule extraction from facts and neural networks. In International Neural Network Conf, Paris, France, 1990, pp 379–82.
71. Sejnowski TJ, Rosenberg CR. Parallel networks that learn to pronounce English text. Complex Systems. 1987;1:145–68.
72. Shen Z. An Application of Neural Networks for the Detection of Coronary Heart Disease. PhD thesis, Department of Electrical Engineering, Brunel University, Brunel, Borneo, 1994.
73. Shrout PE, Spitzer RL, Fleiss JL. Quantification of agreement in psychiatric diagnosis revisited. Arch Gen Psychiat. 1987; 44:172–7.
74. Smith M. Neural Networks for Statistical Modeling. New York: Van Nostrand Reinhold, 1993.
75. Somoza E, Mossman D. Comparing and optimizing diagnostic tests: an information-theoretical approach. Med Decis Making. 1992;12:179–88.
76. Somoza E, Somoza JR. A neural network approach to predicting admission decisions in a psychiatric emergency room. Med Decis Making. 1993;13:273–80.
77. Spackman KA. Combining logistic regression and neural networks to create predictive models. Proc Annu Symp Comput Appl Med Care, 1992, pp 456–59.
78. Stone M. Cross-validation choice and assessment of statistical predictions. J R Statist Soc B. 1974;36:111–47.
79. Stubbs DF. Multiple neural network approaches to clinical expert systems. SPIE Applications of Artificial Neural Networks, 1990, pp 433–41.
80. Tesauro G. Scaling relationships in back-propagation learning. Complex Systems. 1988;2:39–44.
81. Tesauro G, Janssens B. Scaling relationships in back-propagation learning: dependence on training set size. Complex Systems. 1987;1:367–72.
82. Tibshirani R. A comparison of some error estimates for neural network models. Technical Report, Department of Preventive Medicine and Biostatistics, University of Toronto, Toronto, Ont., Canada, 1994.
83. Tu JV, Guerriere RJ. Use of a neural network as a predictive instrument for length of stay in the intensive care unit following cardiac surgery. Comput Biomed Res. 1993;26:220–9.
84. Usui S, Toda N. Biomedical application of neural networks in Japan. Proc World Conference on Neural Networks, San Diego, CA: June 5–9, 1994, pp 63–8.
85. Utans J, Moody J. Selecting neural network architectures via the prediction risk: application to corporate bond rating prediction. Proc First Int Conf Artif Intell Applications on Wall Street. Washington, DC: IEEE Computer Society Press, 1993, pp 35–41.
86. [Various]. Special session on biomedical applications. Proc World Conference on Neural Networks, San Diego, CA, June 5–9, 1994, pp 37–138.
87. [Various]. Special session on medical applications. IEEE Int Conf Neural Networks, June 27–29, 1994, pp 3437–559, 1994.
88. Weinstein JN, Myers T, Casciari JJ, Buolamwini J, Raghavan K. Neural networks in the biomedical sciences: a survey of 386 publications since the beginning of 1991. Proc World Conf Neural Networks, San Diego, CA, June 5–9, 1994, pp 121–6.
89. Weinstein, JN, Rubinstein, LV, Koutsoukos AD, et al. Neural networks in the discovery of new treatments for cancer and AIDS. Proc World Conf Neural Networks, San Diego, CA, June 5–9, 1994, pp 111–6.
90. Widrow B, Hoff ME. Adaptive switching circuits. In IRE WESCON Convention Record, part 4, 1960, pp 96–104.
91. Wilding P, Morgan MA, Grygotis AE, Shoffner MA, Rosato EF. Application of backpropagation neural networks to diagnosis of breast and ovarian cancer. Cancer Lett. 1994;77:145–53.