Neural computing in geophysics

By MICHAEL D. McCORMACK
ARCO Oil and Gas Company
Plano, Texas

The 17th century philosopher-mathematician Rene Descartes offered the statement "Cogito, ergo sum" or "I think, therefore I am" as proof of his personal existence. While modern computers using artificial intelligence technologies cannot match Descartes' profundity, they are beginning to mimic some endeavors previously thought to be completely restricted to the human domain. Neural computing is one area of artificial intelligence that has recently left the laboratory and is being used to solve real and practical problems in geophysics and geology. In this article, we explore the history, operation, and current status of neural computing, and discuss some applications of this new technology.

History. Although neural networks are relatively new to the petroleum industry, their origins can be traced back to the 1940s when psychologists began developing models of human learning. With the advent of the computer age in the 1950s, researchers began to program neural network models to simulate the complex interconnections and interactions between neuronal cells in the brain. These models successfully exhibited various types of human learning behavior. Optimism soared; it was felt that with a large enough and fast enough computer, the entire human brain could be reproduced by a massive neural network. However, in 1969, Marvin Minsky, one of the founding fathers of artificial intelligence, proved mathematically that perceptrons—the simple neural networks being studied at that time—were incapable of solving many simple problems. This revelation put a damper on neural network research for over a decade. It wasn't until the 1980s that these mathematical difficulties were overcome by the introduction of more complex neural network architectures. It was also about this time that people began to realize the potential value of neural networks as general purpose problem solvers, over and above their use as biological models. Today, there are several dozen different neural network paradigms available. The one we will discuss is the backpropagation neural network because its performance is reasonably stable and well understood mathematically. The term "backpropagation" refers to the method by which the network is adapted or modified to solve a problem.

Biological and artificial neural networks. An artificial neural network is a highly simplified computer model of the organization and operation of biological neural systems. In the brain, a neural cell, or neuron, receives inputs from many other neurons via interconnections called axons. If the energy level of the combined inputs exceeds a threshold level, then the neuron transmits an output electrical signal to other neurons. The output signal strength is modified by a special connection called a synapse before entering another neuron. It is believed that certain forms of learning occur when the synapses are trained to assume certain strengths or weights by repeated exposure to the same stimulus. An artificial neural network consists of nonlinear processors called neurodes (corresponding to neurons) linked to each other by connections analogous to the axons. Each connection has an associated scalar weight (corresponding to a synapse) whose value can be modified during training of the neural network. Figure 1 illustrates the processing that occurs at each neurode in a network. Each input signal to the neurode, I_k, is first multiplied by a scalar weight, W_{kj}, and the weighted inputs are summed together and set equal

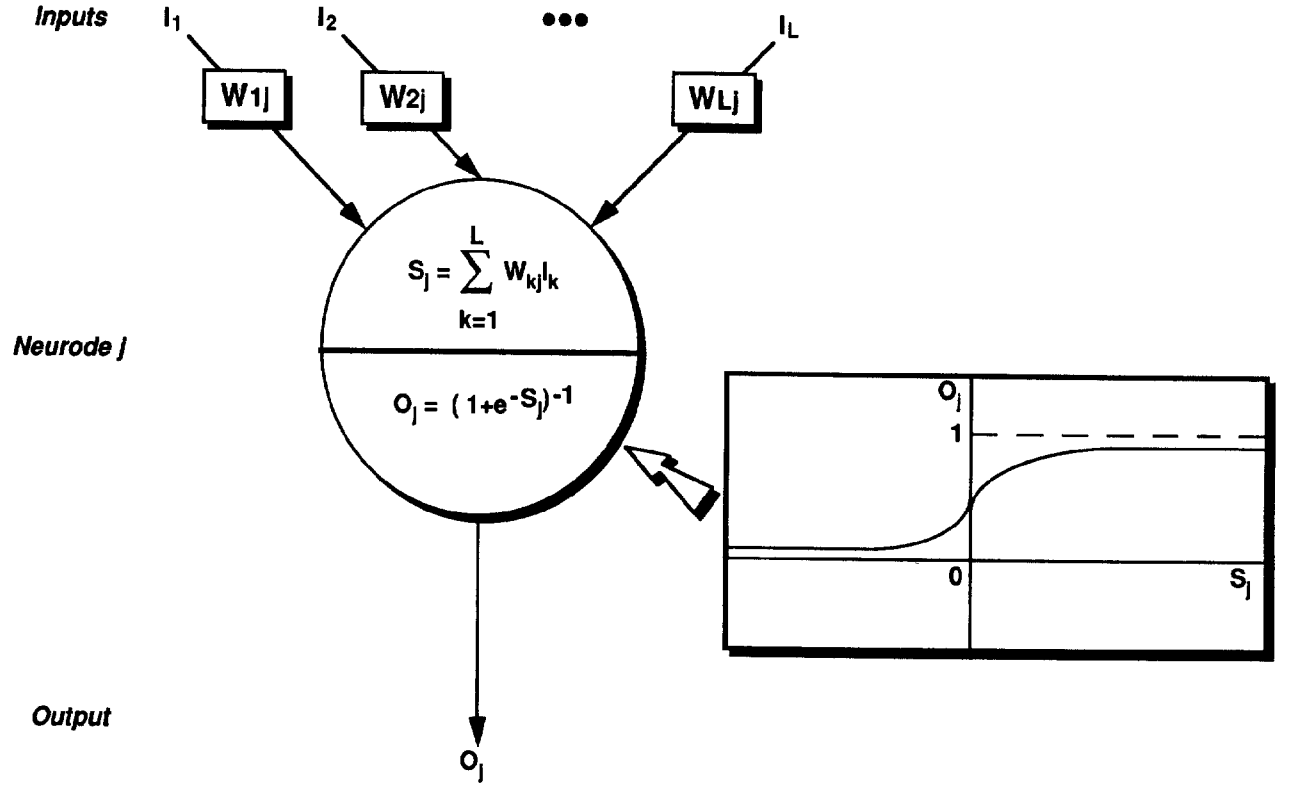

Figure 1. The neurode—basic processing element of a neural network—showing inputs (I_k), interconnection weights (W_{jk}), and output (O_j). The sigmoidal activation function that thresholds the inputs is also shown in the sidebox.

to S_j. The subscript notation "kj" refers to the connection between the kth and jth neurodes. The sum, S_j, is next modified by a nonlinear "activation" function to yield an output signal, O_j. The mathematical form of the activation threshold function is somewhat arbitrary, but in general the function is selected on the basis of its ability to mimic the thresholding behavior observed in biological neurons, and computation simplicity. The most commonly used function is the S-shaped sigmoid:

$$F(S_j) = (1 + e^{-S_j})^{-1}$$

which has asymptotic limits of 0 and 1 as S_j approaches negative and positive infinity (Figure 1). The output signal from the neurode serves as an input to other neurodes, or as the final output answer, depending on the location of the neurode in the overall network architecture. Figure 2 depicts a simple three-layer neural network showing the neurodes as circles, and the interconnections with their associated weights, W_{kj}, represented by lines. Each line connecting two neurodes has a unique scalar weight. Normally, the neurodes are organized into layers, so that there is a layer receiving input signals, one or more middle layers, and an output layer. The middle layers are sometimes referred to as "hidden" layers because they are not visible to the input or output signals. The signals in a backpropagation neural network flow unidirectionally from the input to the output layer; other types of network structures incorporate feedback loops, connections between multiple layers, and delay lines. These complexities will not be discussed here.

Unlike traditional computer programs incorporating a fixed algorithm to solve a particular problem, neural networks utilize a learning technique to develop an appropriate solution. The neural network is "trained" by repeatedly presenting examples of the inputs and desired outputs of the problem to be solved. As each example is entered into the network, the difference between the actual

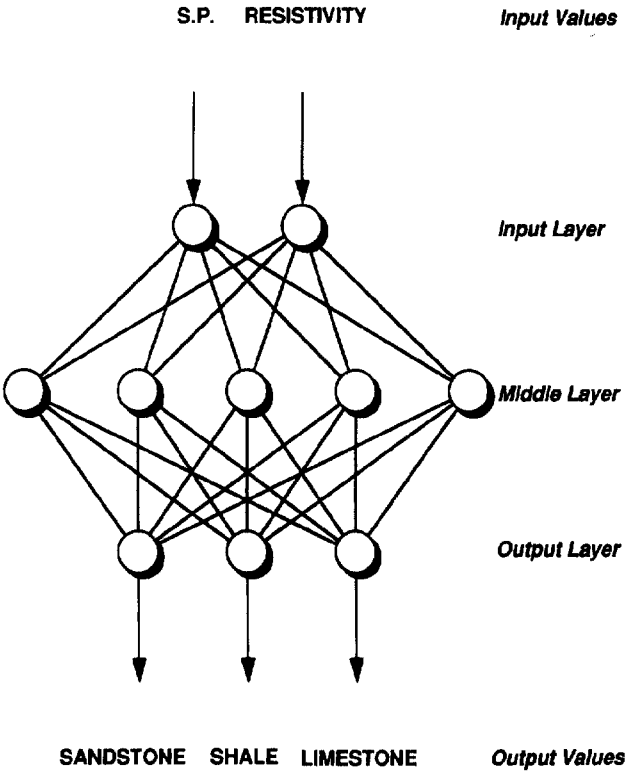

Figure 2. A three-layer, fully connected neural network.

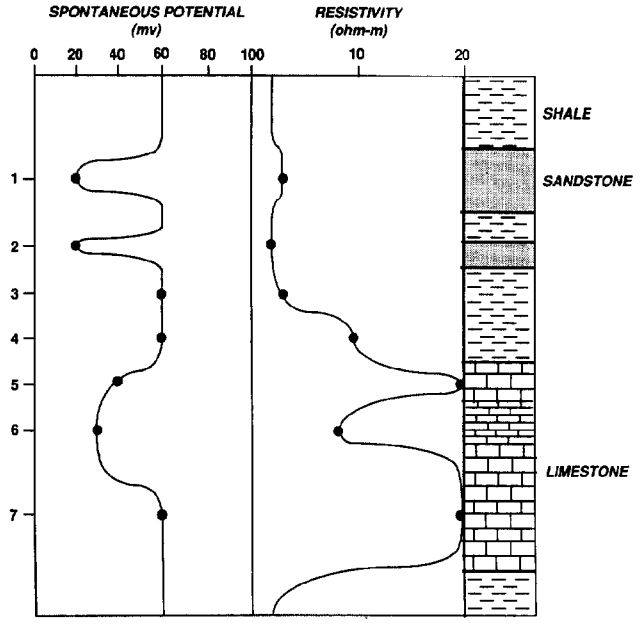

Figure 3. Synthetic spontaneous potential (SP) and resistivity logs showing the depth locations of the seven pairs of SP-resistivity log values used to train a backpropagation neural network. The lithology log computed by the trained neural network is illustrated on the right.

ouput of the network and the desired output is used to modify the weights for each interconnection. Training of the neural network (or, equivalently, changing the values of the interconnection weights) continues until the actual output of all the training examples matches the desired outputs to within some specified tolerance. At this point the neural network is said to be "trained" and is ready to accept new inputs for which output answers are desired.

Network training example. The best way to illustrate this training procedure is by an example which uses two synthetic logs to estimate lithology. Suppose we have spontaneous potential (SP) and resistivity logs for a well, and wish to train a neural network to generate a lithology log using measured values from these two electric logs (Figure 3). Lithologies will be grossly clustered into three categories: sandstones, shales, and limestones. The neural network structure used for this problem is shown in Figure 2. It consists of two input neurodes (one neurode to accept a value for the SP log, and the second to receive a value for the resistivity log at the same depth), and three output neurons—one for each of the three possible lithologies. Output neurodes 1-3 represent sandstone, shale, and limestone, respectively. The signal from each output neurode is either a zero, indicating this lithology is *not* present, or a one, indicating this lithology is present. Thus an input pair of SP-resistivity log values should produce (1 0 0) at the output neurodes if the lithology is sandstone, (0 1 0) if shale is present, and (0 0 1) to indicate limestone. A middle layer consisting of five neurodes completes the network architecture. Selecting the number of middle layers (there can be any number, including zero!) and the number of neurodes in each middle layer is an art. Too few middle layer neurodes makes it difficult to train the network; too many neurodes causes the network to "memorize" the training examples and perform poorly when evaluating new inputs.

Table 1 contains seven pairs of SP-resistivity logs values that were selected from the indicated depths on Figure 3 for the three lithologies.

Table 1. SP-resistivity log values

No.	Lithology	SP (mv)	Resistivity (ohm-m)	Desired output
1	Sandstone	17	3	(1 0 0)
2	Sandstone	20	2	(1 0 0)
3	Shale	60	2	(0 1 0)
4	Shale	60	8	(0 1 0)
5	Limestone	40	20	(0 0 1)
6	Limestone	25	6	(0 0 1)
7	Limestone	60	20	(0 0 1)

To begin the training process, the weights for each network interconnection are set to small random values. As you might guess, when the SP-resistivity pairs are input to this network which has random weight values, the output signals (and hence the lithology estimate) are completely wrong. Thus, when we enter the inputs (for example 1), the actual values for the three output neurodes may be (0.1 0.8 0.4), rather than the desired output of (1 0 0). The difference between the actual and desired values at each output neurode is the output error. These error differences are used to modify the network weights, so that if we were to reenter the inputs for example 1, the actual output values would now be closer to the desired outputs. The errors at each neurode in the network are propagated backwards from the output layer to the input layer with changes made to each weight in proportion to (1) the contribution that connection made to the overall error, and (2) the magnitude of the signal traveling along that connection. The neural network name "backpropagation" is derived from the method of modifying the interconnection weights by propagating the error backwards. (For those interested in the mathematical details of this iterative procedure, see the list of additional reading at the end of this article.) Examples 2-7 from Table 1 are then sequentially input to the network. Changes to the interconnection weight occur after each example is presented to the network if the actual and desired outputs differ significantly. It normally takes several hundred, or thousands, of iterations before the network weights have been adjusted so the output errors for all seven examples are less than a specified limit. For the example shown in Table 1, 845 iterations were required to adjust the interconnection weights so the desired and actual output neurode values differed by no more than 10 percent. On a 25 MHz IBM clone with a 386 processor, total training time was about 60 s for this network. At this point, the neural network is said to be trained and the interconnection weights are saved for later use.

After training is completed, new SP-resistivity values for other depths in the well can be input to the neural network to generate a lithology log for the entire well. The resulting lithology log is shown on the right side of Figure 3. The neural network runs very fast in this mode of operation since the interconnected weights are fixed and data moves forward from the inputs to the outputs (i.e., there is no time-consuming backpropagation of error). Figure 4 is a plot showing how the trained neural network can map any given SP-resistivity input pair into one of the three lithology regions. The boundaries between the different lithology regions are automatically determined by the examples selected during the training session and are encoded in the interconnection weights. If the log analyst is not happy with this result, he can retrain the network by providing additional examples to move the boundaries to their proper positions.

Seismic trace editing. To demonstrate that neural networks

can be applied to a more challenging, real world problem than the simple and contrived logging example, we show the output of a prototype backpropagation network designed to detect noisy traces in real seismic data sets (Figure 5). The network structure for this trace editor has 517 input nodes fully connected to two output nodes. There are no middle layer neurodes. The input nodes consist of 512 samples for the fast Fourier transform amplitude spectrum of the trace being evaluated and five statistical parameters (the average trace frequency, the average absolute amplitude, the average trace energy, the maximum cross-correlation value of the trace being analyzed with adjacent traces, and the energy decay rate). The two output neurodes are trained to produce one of two states: (1, 0) if the trace is "good", and (0, 1) if the trace is "bad." The user selects several examples of good (G) and bad (B) traces for the data set being edited to train the network. After training has been completed, the neural network edited the remaining traces on this shot record using lower case g and b to indicate good and bad trace edits. The performance of this prototype system was good—the neural network trace edits agreed with those of the user in more than 80 percent of the cases. In cases where there was disagreement, the network usually classified the questionable trace as "bad" whereas the user called it "good."

Other applications. Neural networks are a form of automated pattern recognition in which a set of input patterns is related to an output by a transformation encoded in the network weights. They are particularly appropriate for applications in which the physics producing an output from a given input is not well understood or is unknowable (i.e., we know what the input is and we know what the output answer should be, but we cannot write the mathematical equations which will get us from the input to the output). Seismic processing and interpretation abound with examples of this. We can all learn to identify a first break refraction event on a seismic shot record, but it is extremely difficult to describe algorithmically how we do this. In addition to first break picking and the previously discussed seismic trace editing, neural networks can be trained to perform velocity analyses, control the quality of seismic processing, track seismic reflection events across fault blocks, analyze gas chromatograms, and monitor drill bit wear. These are, of course, just a few of the many potential geophysical applications of neural network technology being investigated in industry R&D labs.

Strengths and weaknesses of neural networks. Neural networks are not the solution to all computing problems. They do have some limitations which must be considered in deciding whether or not to use this technology. Current limitations and problems with neural networks include:

- Slow learning rates. For problems requiring a large and complex network architecture or having a large number of training examples, the time needed to train the network can become excessively long; 10^4–10^5 training cycle iterations, lasting minutes or hours, are common for moderately complex problems. This limits the usefulness of neural networks for processes requiring real time response, such as interactive interpretation.
- Trapping in local minima. The training of a backpropagation neural network is a search for the set of interconnection weights that minimizes the error difference between the actual and desired output neurode values. The search strategy uses a gradient descent technique to find the weights associated with this minimum error. There is no way to know, however, whether the minimum found by this procedure is a global minimum or a local one (which could be larger than the global). A network that becomes trapped in a local minimum will not provide an optimum solution. This trapping problem is analogous to trying to find the lowest elevation in North America (Death Valley, California) given that one starts from any arbitrary location. A good search strategy would be to always move in the direction that has the maximum downward slope—the gradient descent method. While this approach guarantees that one will always end up at a minimum elevation, it may not be the continental minimum (e.g., if you start in Kansas, you'll probably end up in New Orleans rather than Death Valley—which may be preferable for a number of reasons but clearly doesn't meet your original goal!).
- "Forgetfulness." A network tends to forget old training examples as it is presented with new ones. A previously trained neural network that must be updated with new information must be retrained using both the old and new examples—there is currently no known way to incrementally train a network.
- Imprecision. Neural networks do not provide precise numerical answers, but rather relate an input pattern to the most probable output state. Conventional programming techniques should be used when precision is needed. A neural network should not be employed to balance your checkbook, but it might be used to analyze your spending patterns to determine if you are headed for bankruptcy.
- "Blackbox" approach to problem solving. Neural networks can be trained to transform an input pattern to an output, but provide no insights into the physics behind the transformation. It is not possible to assign any physical significance to the magnitude or sign of the interconnection weights for most neural network paradigms. However, recent research on backpropagation neural networks concluded that the interconnection weights, while still having no physical significance per se, can be used to determine the relative contribution of each input to each output.

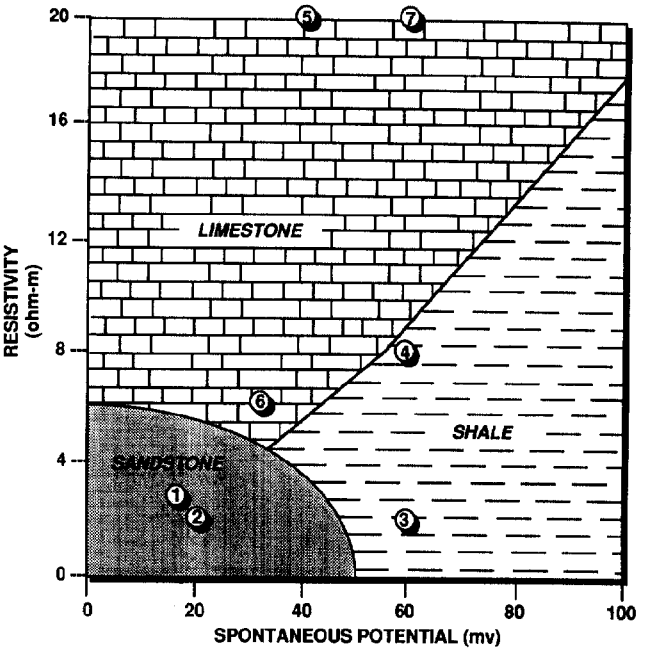

Figure 4. SP-resistivity crossplot showing the decision surface for the three lithologies developed during training with the seven examples in Table 1. The location of the seven training examples is also indicated on the crossplot.

Despite the numerous deficiencies and limitations noted above, neural networks do have significant advantages over expert systems, another artificial intelligence technology, and conventional programming methods. Expert systems, which encode expertise about a knowledge domain in a series of IF-THEN rules, cannot easily handle complex pattern recognition problems such as those encountered in visual or speech recognition. Neural networks are well suited for these low level tasks. Furthermore, expert systems are not robust where there is missing or incomplete knowledge about the physics of a problem, and must resort to heuristics and rules-of-thumb to generate acceptable solutions. Again, neural networks provide a practical and convenient pathway to a solution. Conventional programs generally use a fixed, static algorithm

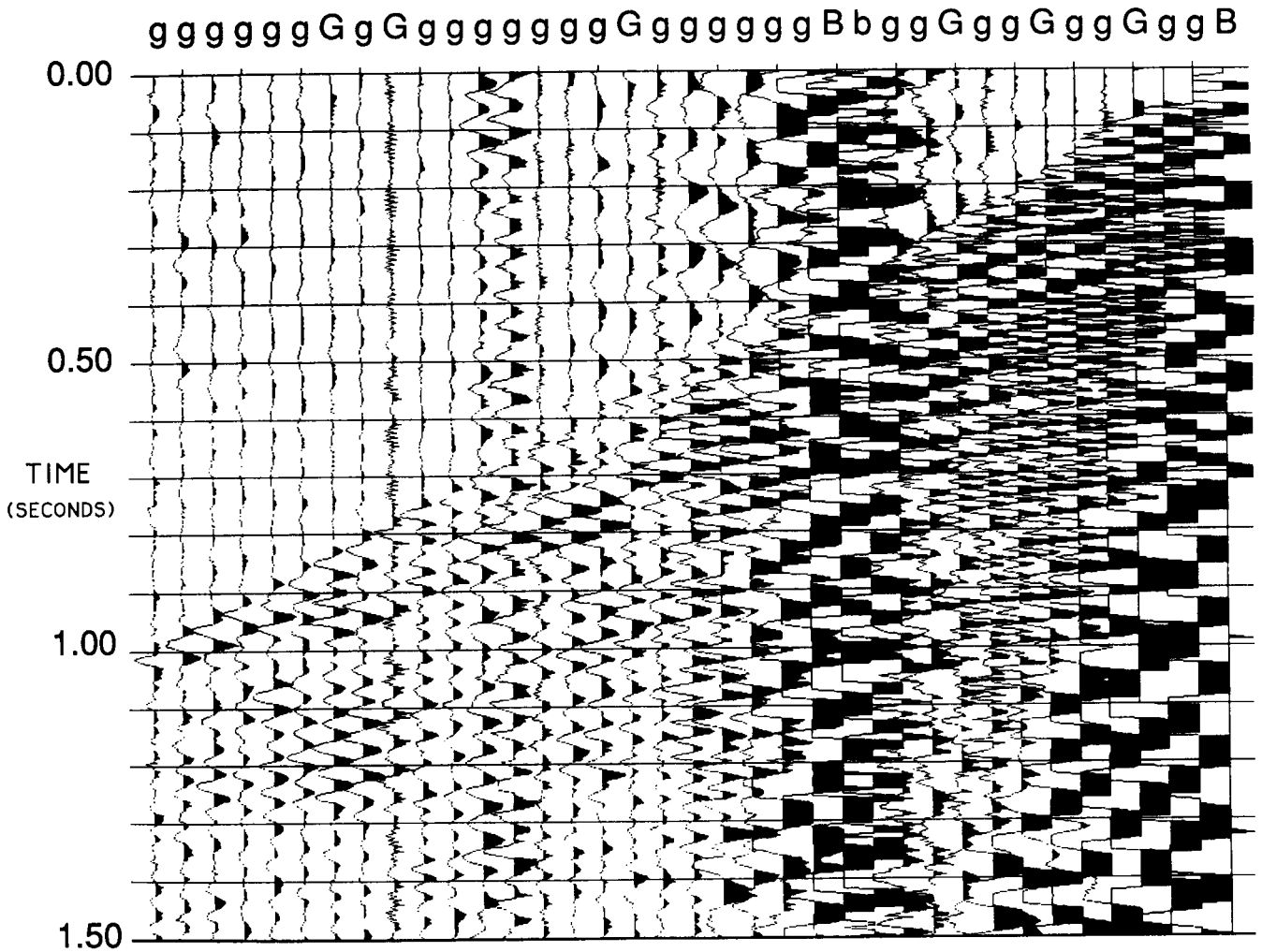

Figure 5. An edited seismic shot record with the good (G) and bad (B) traces used to train the neural network. Good and bad trace edits performed by the neural network after training are indicated by g and b, respectively.

based on a process model to provide numerically precise results. For example, spiking decon programs assume that the spectrum of the reflection coefficients of the layers in the earth is "white", the seismic waveform is unchanging with time, and so on. Catastrophic errors can result when these assumptions are violated. Neural networks, on the other hand, degrade gracefully when the input data become noisy or incomplete, or even when a portion of the network itself is destroyed. Finally, neural networks should enjoy a high rate of acceptance by the end user community. Since the network is trained using a set of examples the user has selected, the trained network will mimic the user's personal preferences and unique style of analysis. In a sense, to disagree with the output answers of a neural network one has trained is to indict oneself as a poor teacher!

Summary. Neural networks are being exploited to solve exploration and production tasks that could previously only be done by humans. These networks have strengths and weaknesses that need to be considered in selecting appropriate applications. While standalone neural networks can and will be used in many geophysical systems, the most leveraging applications will occur when they are married with expert systems and conventional programs that can take advantage of their sophisticated pattern recognition capabilities.

Suggestions for further reading. The recently published book *Naturally Intelligent Systems* by M. Caudhill and C. Butler (MIT Press, 1990) provides an excellent introduction to neural networks that goes into surprising depth without getting into mathematics. For those who are seriously investigating some of the basic neural network paradigms, *Neural Computing: Theory and Practice* by P.D. Wasserman (Van Nostrand Reinhold, 1989) and *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* by D.E. Rummelhart and J.L. McClelland (MIT Press, 1986), discuss many practical implementation and programming issues. *Adaptive Pattern Recognition and Neural Networks* by Y.H. Pao delves a little deeper into the mathematics behind neural networks and also provides C code for the backpropagation paradigm in the appendix. LE

Michael D. McCormack has been a principal research geophysicist with ARCO Oil and Gas Company since 1980. He received his bachelor's in physics from Gonzaga University, a master's in physics from the Oregon Graduate Institute, and a Ph.D. in geophysics from the University of Houston. McCormack worked in the geophysical research group at Geosource in 1972-80. His current research interests include exploration applications of artificial intelligence technologies and shear wave seismology.