Abstract. In petroleum geology, lithofacies information is important for estimating porosity and permeability values from wireline logs at the un-cored well intervals; however, predicting lithofacies from logs is not an easy task. The resuits of lithofacies prediction from wireline log signals were compared using two different ıpervised classification methods: backpropagation neural network (BPNN) and a simplified version of Fuzzy ARTMAP called SFAM. The SFAM method gives results similar to those of BPNNs, but does not suffer BPNN's problems of excessive training time and the need for prior specification of network topology. If training time and the effort required to fine-tune BPNN parameters are acceptable, in some cases BPNN can provide significantly improved performance. However, to achieve this ›erformance will generally require some degree of skill and a process of trial and error. Porosity and permeability predictions can be estimated by BPNNs; however, SFAM is currently unable to perform this task, and this requires further study.

Fuzzy ARTMAP: A New Tool for Lithofacies Recognition

P.M. Wong,[1] T.D. Gedeon,[2] and I.J. Taggart[1,3]

Accurate determination of hydraulic properties of porous media from wireline log measurements is a central problem in petroleum well-log interpretation. The amount of oil recovered from oil wells is related to both the amount of oil-in-place and its ability to flow through the rocks. Two important measures of this amount are the *porosity* of the rock, which indicates the size of void spaces or pores in the rock, and the *permeability* of the rock, which is related to the connectedness of individual pores into flow channels. The quality of these values has a strong impact on reservoir modeling and the subsequent management scheme to maximize oil production.

Recent studies have shown that identification of rock types or lithofacies prior to calculating the hydraulic properties can lead to improved estimates (Silseth et al. 1990, Alabert and Massonnat 1990, Wong et al. 1994, 1995b) because each lithofacies has its own lithohydraulic characteristics, such as the observable lithofacies-specific porosity-permeability relations (Jian et al. 1994). Some other key concepts and practices relevant to petroleum reservoirs can be found in Wong et al. (1995a). Each of the lithofacies is characterized by its textural, diagenetic, and petrophysical properties.

[1]Centre for Petroleum Engineering
The University of New South Wales
Sydney, NSW 2052, Australia

[2]School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2052, Australia

[3]Now with West Australian Petroleum Pty. Ltd., GPO Box S1580, WA 6001, Perth, Australia.

Supervised and Unsupervised Classification

There are generally two approaches to lithofacies classification from wireline logs: unsupervised and supervised. The unsupervised approach, such as the use of cluster analysis or its derivatives (Wolff and Pelissier-Combescure 1982), requires only the logging data (i.e., inputs) and forms clusters based on the distribution of the sample data. The outcome using this approach is usually referred to as "electrofacies" (Serra and Abbott 1980), defined as the set of log responses that characterizes a sediment and permits the sediment to be distinguished from others. Although electrofacies can be obtained from logs, it does not guarantee the presence of its specific hydraulic properties, such as the porosity-permeability relation. Hence, it does not have a strong geological foundation.

The supervised or "genetic" approach (Jian et al. 1994, Wong et al. 1995c) is a newer concept that seeks to identify and treat each dominant lithofacies type individually. This approach requires use of a training data set which forms the prior knowledge of lithofacies present in the reservoir (i.e., target groupings). The classification is usually obtained based on textural, diagenetic, and petrophysical properties of the reservoir. The identified groupings are used as the training data to teach the model to relate the inputs and the target groupings. In this paper, we will assume that by adopting data from a variety of sources, such as petrophysical, depositional, and diagenetic features, a suitable set of lithofacies has been defined and confirmed by experienced geologists, and each lithofacies has a characteristic log signature.

Backpropagation neural networks (BPNNs) have been used for supervised well-log signal classification (Smith et al. 1991, Rogers et al. 1992, Wong et al. 1995c). A number of studies have shown that BPNNs generally perform better than statistical methods such as discriminant analysis (Parikh et al. 1991, Yoon et al. 1993). The BPNNs, however, also have disadvantages. Recently, a new neural network-based pattern classification technique, using fuzzy arithmetic and adaptive resonance theory and called Fuzzy ARTMAP, has been proposed (Carpenter et al. 1992) to address problems faced by use of BPNNs. This technique has not been widely used to solve engineering problems, and the literature includes only a few documented examples (Ham and Han 1993, Shahla et al. 1993, Srinivasa and Ziegert 1993).

The aim of this paper is to determine via actual field examples how this method performs in relation to the BPNN method in predicting lithofacies from wireline logs. We start with a brief review of the BPNN and Fuzzy ARTMAP methods in signal classification and three examples from different reservoirs. Each reservoir is a geographically distinct area and contains a number of wells. Core data are available for two wells in each reservoir. The core data is derived by actually measuring properties at selected locations in the well by extracting rock samples. In each example, one well provides the core data, which are the known outputs used in the training phase, and the methods are then applied to a second well, where predictions of lithofacies based on the wireline logs are compared to core data from the second well, which were withheld during the training phase.

Backpropagation Neural Networks (BPNNs)

Classifying signals using neural nets can be done in a supervised or unsupervised manner. Supervised classification requires a set of training data with known input-output pairings; the unsupervised method requires only input data. As discussed previously, the genetic approach (i.e., the supervised method) uses a training data set which, in turn, requires the use of a learning rule to match the inputs to the known (or target) outputs. Thus, unsupervised techniques, such as the self-organizing algorithm (Baldwin et al. 1990), are not suitable for implementing the genetic approach. Back-error propagation, or backpropagation, is the most widely used learning rule in supervised neural nets. This technique aims to approximate functions by minimizing the errors between the predicted and the target outputs (Rumelhart et al. 1986).

A typical BPNN contains three kinds of layers: input, hidden, and output. Each layer consists of a number of neurons. The numbers of input and output neurons depend on the dimension of the input vector and the number of categories in the problem of interest, respectively. The optimum number of neurons in the hidden layer is a difficult question that usually requires a trial-and-error method to answer. Each neuron in a layer connects to all neurons in the next layer. The connection between biological neurons is known as a synapse. During learning, the strength of each synapse is changed, either strengthened or weakened. In BPNNs, the strengths of these synapses are simulated by weights. The aim of training BPNNs is to determine the weights in each connection in such a way that the errors between the predicted and target outputs are minimized. When a suitable minimum is found, the weights are fixed and the training stage is terminated. Details of BPNNs can be found in Dayhoff (1990).

Practical Difficulties in Using BPNNs

Before beginning training, a pre-specified network topology (such as the number of hidden layers and the number of neurons in each hidden layer) is required. Too few layers and neurons will not learn satisfactorily, and too many of these elements will memorize the training patterns (Dayhoff 1990). The amount of training time (measured in number of iterations, or epochs) is also important because excessive training time can also lead to memorization of input patterns. Therefore, a careful monitoring of training time is required. This is usually done by using a test data set (i.e., a set of known input-output data withheld from the training process) to validate the model and terminate training before generalization capability degrades. Also, the weight values in all connections must usually be initialized to some small random numbers before training starts to break the symmetry between oth-rwise identical hidden neurons. Randomizing the weights in an inappropriate range of values will lead to slow convergence (Wessels and Barnard 1992) because large initial weights impose a large random bias for the network to unlearn. The values of the other parameters, such as learning rate and the momentum term, are also important in the training phase (Dayhoff 1990).

Because a large number of parameters can be adjusted to improve the performance of BPNNs, it is very difficult to obtain an optimum set of parameters that gives the best results. Another problem in using BPNNs is that, once the neural net is trained, it is not easy to incorporate new knowledge, and it is necessary to re-train the neural net with the incremental knowledge—a time-consuming process. A new supervised classification technique, called Fuzzy ARTMAP, addresses some of these problems.

Fuzzy ARTMAP

Application of adaptive resonance theory (ART) in signal processing was proposed by Grossberg (1976a, b) and was originally used for unsupervised classification. Recently, Carpenter et al. (1991) developed a model (a hybrid method), called ARTMAP, by combining supervised and unsupervised methods for mapping known input-output patterns. At a later stage, fuzzy set theory (Zadeh 1965, Kosko 1986) replaced the traditional set theory used in the supervised ARTMAP method and, hence, the name Fuzzy ARTMAP (Carpenter et al. 1992) was coined. Kasuba (1993) also reported a simplified version of Fuzzy ARTMAP, called SFAM; however, the application of Fuzzy ARTMAP or SFAM for signal processing is poorly documented. In this paper, SFAM is used for the comparison study because of its simplicity compared to the Fuzzy ARTMAP.

The SFAM model is based on a simple neural network. It has all the advantages of BPNNs but does not suffer from the problems (such as weight values initialization, network topology optimization, and the amount of training time required for the iteration process) that were discussed previously. SFAM has three layers: input, prototype, and output (or category) layers. The input layer is composed of neurons, the number of which depends upon the dimension of the input vector. No initial neurons in the prototype layer are required to start the network. The number of neurons grows depending upon the complexity of the problem during supervised learning. Each category is assigned to a neuron in the output layer and, hence, the number of output neurons is the same as the number of known categories in the problem.

SFAM Learning Procedure

When the training patterns are presented to the network, each neuron in the input layer receives the signals and distributes this information to the prototype layer. If the category of the input signals has not been seen by the network, it will create a neuron in the prototype layer to represent this category and will connect it to the appropriate output neuron. The weights connecting the input neurons and the newly formed prototype neuron are set equal to the values of the input vector. For the prototype-to-output connections, a logical OR operation is performed. The output layer is used only to record the identity of the prototype neurons.

The number of prototype neurons is at least equal to the number of output neurons (i.e., number of categories). For the trained network, the number of prototype neurons for each category may be larger than one (Figure 1). The actual number depends on the complexity of the data set. For example, if the number of prototype neurons for "sandstone" created after training is three, it may mean that these three neurons represent three subclasses of sandstone, such as "fine-grained sandstone," "medium-grained sandstone," and "coarse-grained sandstone." In BPNNs, using only one output neuron to represent the category "sandstone" in order to learn and incorporate differences among grain sizes may take a considerable amount of training time.

When the category of a new pattern has been seen by the network, new activation functions and match functions are determined for all of the prototype neurons. These functions calculate the degree to which the weight vector is a fuzzy subset of the input vector (i.e., the activation function value, *T*) and the degree to which the input vector is a fuzzy subset of the weight vector (i.e., the match function value, *M*). For details of the calculation of degrees of fuzzy subset membership, see Kasuba (1993). Using fuzzy logic, it is possible to calculate the degree one vector is a fuzzy subset of another vector; this is analogous to finding the degree of similarity between the vectors.

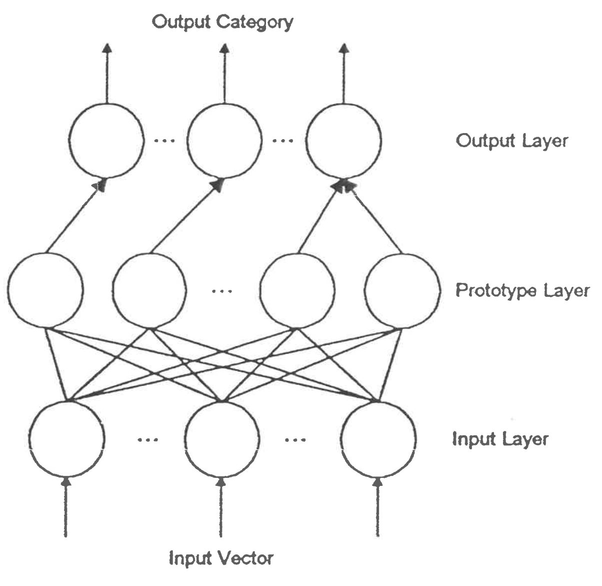


Figure 1. *Schematic diagram of an SFAM network.*

The *M* value of the prototype neuron with the maximum *T* value (i.e., the winning neuron) is then compared to a parameter called the vigilance term, ρ. The vigilance term ranges from 0 to 1, and must be specified before training starts. If the *M* value is less than the ρ value, the network is said to be in a state of *mismatch reset*, which means the winning prototype neuron is not good enough to encode the current input pattern and, hence, another prototype neuron with the same category as that pattern must be created. On the other hand, when the *M* value is greater than the ρ value, the network state is called *resonance* (or *reset*), which means the winning prototype neuron is good enough to encode the current input pattern, and the weights connecting the input neurons and the winning prototype neuron can be updated, provided the category of the winning neuron is the same as the category of the input pattern. If, however, the category of the winning prototype neuron is not the same as the category of the input pattern (which we know because this is training data and labeled with the category), the state of the network is called category mismatch, which means the chosen vigilance (i.e., the baseline or lowest value) is too small for the system. The network will then increase the vigilance value by an appropriate amount, and another test on the next highest *M* value is done. The amount of increase required can be readily calculated automatically so that the category mismatch is no longer true for the current pattern. Therefore, the value of the vigilance term is monotonically increasing during learning. Generally speaking, a high vigilance causes a greater number of prototype neurons to form. If the vigilance is set too high initially, larger numbers of prototypes are formed, which reduces the ability of the network to generalize. A simple flow chart for SFAM in an epoch is shown in Figure 2.

The weight update in the SFAM model is based on the fuzzy AND (Zadeh 1965) operator of the input and the weight vectors:

$$w_{ij}(t+1) = \beta\,[X \wedge W_j(t)]_i + (1 - \beta) \tag{1}$$

where w_{ij} is the weight connecting the input neuron

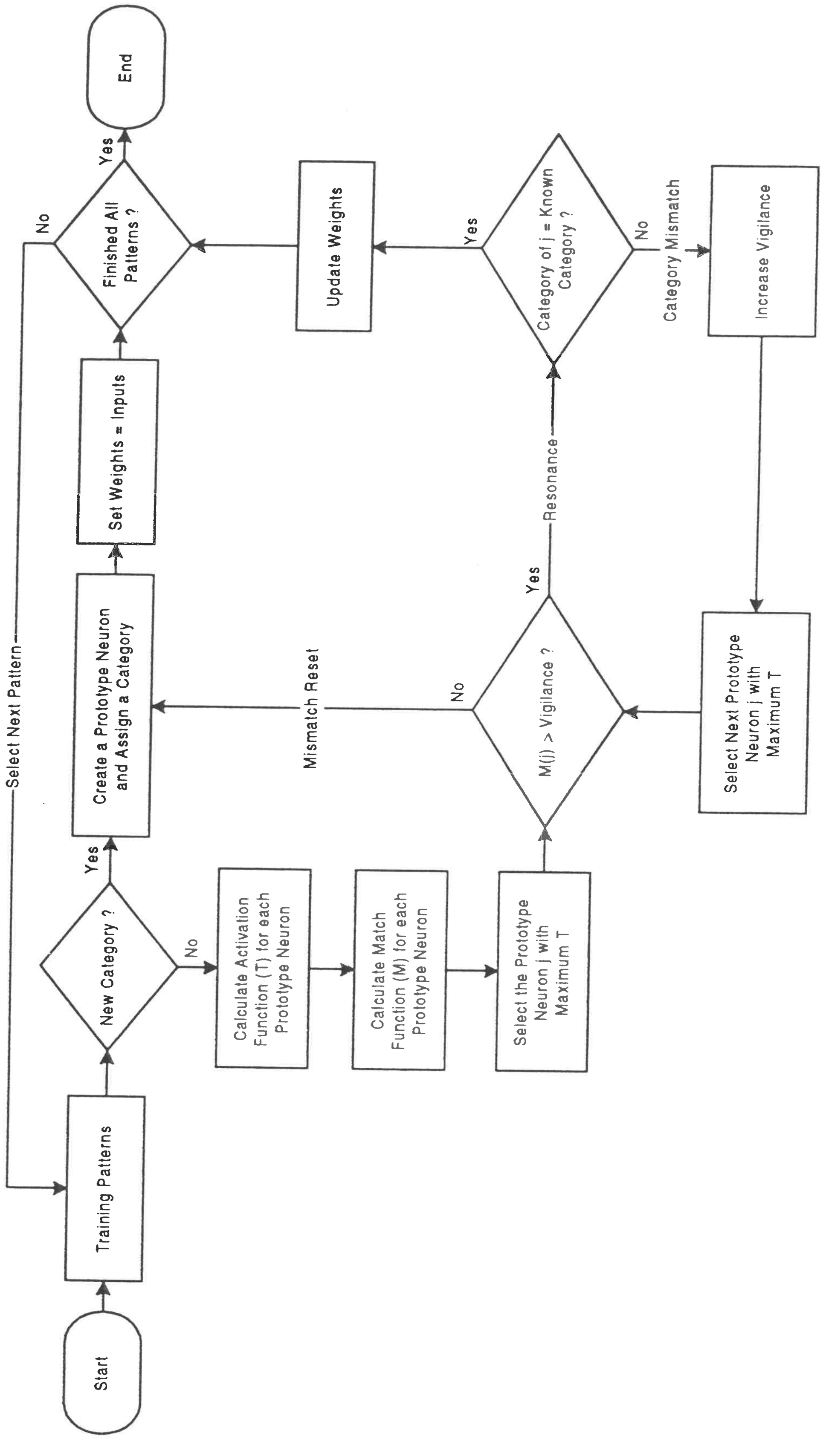

Start
Training Patterns
New Category ?
Yes
No
Create a Prototype Neuron and Assign a Category
Set Weights = Inputs
Finished All Patterns ?
Yes
No
End
Select Next Pattern
Calculate Activation Function (T) for each Prototype Neuron
Calculate Match Function (M) for each Prototype Neuron
Select the Prototype Neuron j with Maximum T
M(j) > Vigilance ?
No
Mismatch Reset
Yes
Resonance
Category of j = Known Category ?
Yes
Update Weights
No
Category Mismatch
Increase Vigilance
Select Next Prototype Neuron j with Maximum T

Figure 2. *Flow chart for SFAM in an epoch of training.*

i and winning prototype neuron *j*, *X* is the input vector, W_j is the vector of weights connecting inputs to prototype neuron *j*, β is the learning rate (analogous to the learning rate of BPNNs), and *t* is incremented by 1 for each update. Note that the learning rate and the baseline vigilance term are the only user-selectable parameters in the SFAM model. The "∧" is the fuzzy AND operator. It is simply the minimum of each of the components of two vectors:

$$[X \wedge W_j(t)]_i = \min\,[x_i, w_{ij}(t)] \qquad (2)$$

where x_i is the corresponding component of the vector *X*. Further details can be found in Kasuba (1993). The main similarities and differences of BPNNs and SFAM models for pattern classification are shown in Table 1.

After training, all the weights on the input-to-prototype connections are fixed, and each prototype neuron has a category assigned to it (Figure 1). For patterns with unknown categories, each input pattern can be simply presented to the network, and the activation function value *T* is calculated for each prototype neuron. A winning prototype neuron (i.e., the one with the highest *T*) is chosen, and the network assigns the category of the connected output neuron to this input pattern.

Case Examples

The data used in this study came from three oil-bearing reservoirs located in the Carnarvon Basin of the North West Shelf, Australia. In each example, two wells from each reservoir were used to provide the wireline log and core readings. The data from the first well was used to construct the training data

Table 1. Comparison of BPNNs and SFAM models.

	BPNNs	SFAM
Classification system	Supervised learning; non-linear; hyper-plane classifier.	Hybrid (supervised and unsupervised) learning; non-linear; hyper-rectangle classifier.
Model assumption	None.	None.
Network topology	Pre-set number of neurons in the hidden layer; one output neuron for each category.	Number of prototype neurons for each category depends on complexity of the problem; self-organizing.
Initial weights	Random initialization.	None.
Learning rate	Present.	Present.
Momentum term	Present.	Absent
Vigilance term	Absent.	Present.
Data normalization	Required.	Required.
Updating method	On-line or off-line.	On-line or off-line.
Incremental learning	Difficult to incorporate new knowledge; have to re-train the whole data set.	Easy to incorporate new knowledge.
Learning time	Slow.	Fast.
Neuron algorithm	Linear combination of input signals with weights; use of a sigmoid transfer function.	Calculate the degree to which the weight vector is a fuzzy subset of the input vector (i.e., the activation function value).
Weight modification	Backpropagation of error.	Using the fuzzy AND operator.

set, which was then used to classify the log signals from the other well where core data was available for comparison purposes; hence, the performance of the different methods can be evaluated.

Data Sets and Procedure

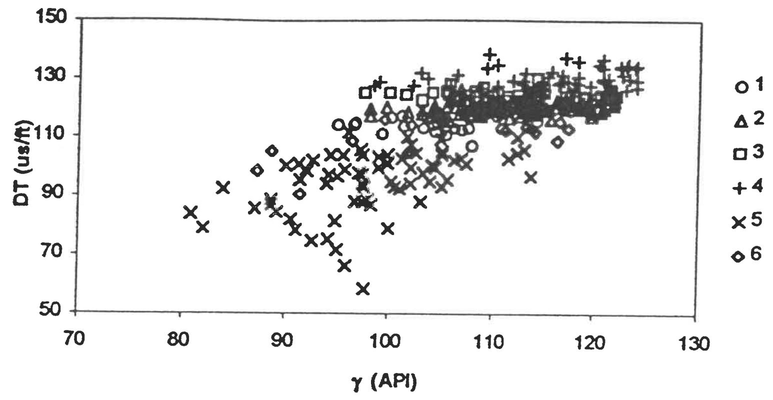
Table 2 summarizes the data used in this comparison. This table shows the number of data points used in the training phase and the test phase, the wireline logs used, and the description of each lithofacies for each of the three reservoirs (A, B, and C). The lithofacies were identified by a geologist in a manner consistent with the genetic concepts defined earlier. Note that the data used for training were from one well and the test data were from the other well in the same reservoir. Figure 3 displays some data cross-ts for these reservoirs. The crossplots show that some rock types can distinguished from the other rock types. However, overall there is significant overlap in the values in any pairs of input variables; this is primarily due to the presence of heterogeneities.

In the BPNN experiment, the numbers of input and output neurons were, respectively, three (the three log variables) and six (the six lithofacies). Six hidden layer neurons gave the best performance for all cases by trial and error. Weights can be updated in either on-line (stochastic updating) or off-line (epoch updating). For the on-line method, each weight is updated on a pattern-by-pattern basis, while for the off-line method, the weight changes are accumulated for all the training patterns in an epoch before updating is done. Because the on-line method requires more computational time in BPNNs, off-line learning was used. The learning rate and momentum term were both set to 0.1. Each of the three training data sets was trained for 10,000 epochs, which was chosen to be well past the optimum stopping point. Each network was run 10 times with different sets of initial weights. The corresponding test data were also used to record the highest classification accuracy during the training phase. Only the results with the highest classification accuracy on the test data set will be shown. This process of selecting the network training time as the point when the error on the test set is minimum is called cross validation. Cross validation maximizes the generalization performance as the test data is used to stop the network or, in our case, to mark the epoch and weight values to be used. In practice, it is always necessary to save the minimum network state and continue training in order to avoid selecting the first local minimum as the stopping point.

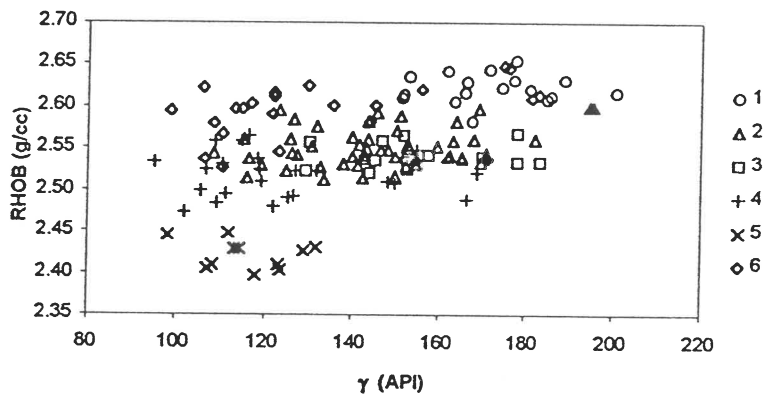
In the SFAM method, the learning rate and baseline vigilance terms were set to 0.5 and 0.4, respectively. These parameters were chosen by trial and error. Note that SFAM does not require the prior specification of the number of prototype neurons. Since SFAM requires much less computational time, the on-line weight updating was chosen. The performance of this learning method depends on the presentation sequence of the input patterns and,

Table 2. *Description of data sets. Note that γ, PEF, ILD, and DT represent the gamma ray, photoelectric adsorption ʾex, deep induction resistivity, and sonic travel times, respectively.*

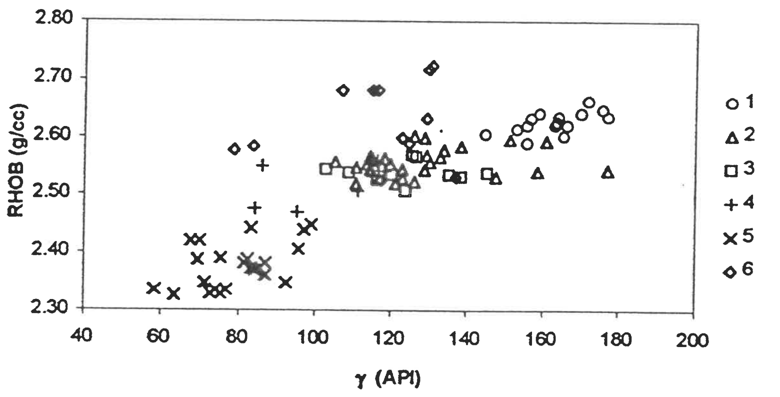
Reservoir	Train (Well 1)	Test (Well 2)	Logs Used	Lithofacies
A	181	155	γ, ILD, DT	"1": Mudstone. "2": Sandy Mudstone. "3": Muddy Sandstone. "4": Sandstone. "5": Carbonate-Cemented Beds. "6": Carbonate Concretions.
B	87	53	γ, PEF, RHOB	"1": Mudstone. "2": Sandy Mudstone. "3": Interbedded Sand/Mud. "4": Muddy Sandstone. "5": Sandstone. "6": Carbonate-Cemented Beds.
C	56	41	γ, PEF, RHOB	Same as reservoir B.



(a) Reservoir A: DT versus γ.



(b) Reservoir B: RHOB versus γ.



(c) Reservoir C: RHOB versus γ.

Figure 3. *Wireline logs and core data for different reservoirs. Note that* γ, *PEF, ILD, and DT represent the gamma ray, photoelectric adsorption index, deep induction resistivity, and sonic travel times, respectively.*

hence, improved performance can be achieved by training the system several times using different orderings of the patterns. In this study, 10 sets of different pattern orderings were used. Each training data set was run for 10 epochs (chosen to be well past the optimum of epochs), and the test data were used to record the classification accuracy during training. Again, only the results with the best performance are reported.

Results

The results of the three examples are shown in Table 3. SFAM has the capability of pattern memorization, and a high recognition rate (%) is the usual result for the training data. This is not an impor-·t issue, as the performance of the different methous was measured on the test patterns, not the training patterns. In reservoir A, the BPNN performed the best; however, it took more than 5,300 epochs to achieve this result. Although the SFAM method performed less well, it took only one epoch (with 36 prototype neurons resulting) to obtain results. In reservoir B, the SFAM method gave the best performance in two epochs (with 40 prototype neurons). In reservoir C, the SFAM method provided the best results (with 21 prototype neurons) on the test set compared to the BPNN.

Table 4 shows the number of prototype neurons automatically created for each lithofacies after training in the SFAM network. It is important to note that the numbers displayed relate directly to the complexity of the problem domain. For example, the lithofacies "1" for reservoir C requires only 1 .ron. That means that this lithofacies has very distinct properties, and the network is able to easily discriminate it from other groups. On the other hand, lithofacies "2" in reservoir C requires 7 neurons to represent, and hence this lithofacies is very difficult to resolve (see also Figure 3c).

Discussion

From the case examples presented, the SFAM method provides good results compared to the BPNN method. The BPNN method in pattern recognition is a powerful technique; however, it suffers from some practical problems such as the long training time requirement and the large number of parameters modifiable for improving the network performance. In most cases, obtaining an optimum set of network parameters is a difficult task. In the reservoir B and C examples, the BPNN did not give the best results, but it does not mean that it could not. We can alter some other network parameters, such as the transfer function used in each neuron and the use of on-line training method (and different pattern orderings), to improve its performance in terms of both the training time requirement and classification accuracy. However, in practice, time is usually an important constraint for professionals to analyze a large amount of data.

In contrast, the SFAM method gave very good results within only a small number of iterations (usu-

Table 3. *Lithofacies classification results (measured in percent correct classification, %). The "n" in the "Remarks" column represents the number of prototype neurons automatically created after SFAM training.*

		Train (Well 1)	Test (Well 2)	Remarks
Reservoir A	No. data pts	181	155	
	BPNN	76.2	76.1	5,382 epochs.
	SFAM	85.6	64.5	1 epoch; n=36.
Reservoir B	No. data pts	87	53	
	BPNN	67.8	58.5	416 epochs.
	SFAM	96.6	60.4	2 epochs; n=40.
Reservoir C	No. data pts	56	41	
	BPNN	69.6	73.2	250 epochs.
	SFAM	96.4	75.6	1 epoch; n=21.

Table 4. *Number of prototype neurons created for each lithofacies during SFAM training.*

	Lithofacies						
Reservoir	1	2	3	4	5	6	**Total**
A	5	5	10	6	5	5	36
B	5	13	3	9	3	7	40
C	1	7	3	4	1	5	21

ally less than five). Moreover, the number of network parameters is much less compared to BPNNs. This means that getting the best performance takes a relatively short time, and the end result is a fast method for pattern recognition.

References

Alabert, F.G., and G.J. Massonnat. 1990. Heterogeneity in a complex turbiditic reservoir: Stochastic modeling of facies and petrophysical variability. SPE 20606, Pages 775-790 in: Proceedings, 65th Annual Technical Conference and Exhibition of Society of Petroleum Engineers, New Orleans.

Baldwin, J.L., R.M. Bateman, and C.L. Wheatley. 1990. Application of neural networks to the problem of mineral identification from well logs. The Log Analyst 3: 279-293.

Carpenter, G.A., S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen. 1992. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Transactions on Neural Networks 3: 698-713.

Carpenter, G.A., S. Grossberg, and J.H. Reynolds. 1991. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Networks 4: 565-588.

Dayhoff, J.E. 1990. Neural Network Architectures: An Introduction, Van Nostrand Reinhold, New York.

Grossberg, S. 1976a. Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. Biological Cybernetics 23: 121-134.

Grossberg, S. 1976b. Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, and illusions. Biological Cybernetics 23: 187-202.

Ham, F., and S. Han. 1993. Cardiac arrhythmia classification using Fuzzy ARTMAP. Proceedings, Annual Conference on Engineering in Medicine and Biology 15(1): 288-289.

Jian, F.X., C.Y. Chork, I.J. Taggart, D.M. Mckay, and R.M. Barlett. 1994. A genetic approach to the prediction of petrophysical properties. Journal of Petroleum Geology 17: 71-88.

Kasuba, T. 1993. Simplified Fuzzy ARTMAP. AI Expert 8 (11): 18-25.

Kosko, B. 1986. Fuzzy entropy and conditioning. Information Sciences 40: 165-174.

Parikh, J.A., J.S. DaPonte, M. Damodaran, A. Karageorgiou, and P. Podaras. 1991. Comparison of backpropagation neural networks and statistical techniques for analysis of geological features in Landsat imagery. SPIE Vol. 1469, Applications of Artificial Neural Networks II: 526-538.

Rogers, S.J., J.H. Fang, C.L. Karr, and D.A. Stanley. 1992. Determination of lithology from well logs using a neural network. American Association of Petroleum Geologists Bulletin 76: 731-739.

Rumelhart, D.E., G.E. Hinton, and R.H. Williams. 1986. Learning representations by backpropagating errors. Nature 323: 533-536.

Serra, O., and H.T. Abbott. 1980. The contribution of logging data to sedimentology and stratigraphy. SPE 9270, In: Proceedings, 55th Annual Technical Conference and Exhibition of Society of Petroleum Engineers, Dallas.

Shahla, K., D. Ajaya, and L.C. Rabelo. 1993. Evaluation of the performance of various artificial neural networks to the signal fault diagnosis in nuclear reactor systems. Pages 1719-1723 in: Proceedings, IEEE International Conference on Neural Networks.

Silseth, J.K., A.C. MacDonald, J. Alvestad, A.T. Buller, and S.B. Torp. 1990. Impact of flow unit reservoir description on simulated waterflood performance: A sensitivity study based on a synthetic 3D geological model. SPE 20603, Pages 759-774 in: Proceedings, 65th Annual Technical Conference and Exhibition of Society of Petroleum Engineers, New Orleans.

Smith, M., N. Carmichael, I. Reid, and C. Bruce. 1991. Prediction of lithofacies types and qualities using a distributed neural network. Pages 482-492 in: Proceedings, IEEE Workshop on Neural Networks for Signal Processing.

Srinivasa, N., and J. Ziegert. 1993. Real-time learning of thermal errors in machine tools using a fuzzy logic based neural network. Pages 235-240 in: Proceedings, 1993 ASME Winter Annual Meeting of the American Society of Mechanical Engineers, New Orleans, Production Engineering Division, volume 64.

Wessels, L.F.K., and E. Barnard. 1992. Avoiding false local minima by proper initialization of connections. IEEE Transactions on Neural Networks 3: 899-905.

Wolff, M., and J. Pelissier-Combescure. 1982. FACIOLOG: Automatic electrofacies determination. Society of Professional Well Log Analysts, 23rd Annual Logging Symposium, Paper FF.

Wong, P.M., I.J. Taggart, and F.X. Jian. 1994. Integration of seismic data with wireline logs and core data into a genetic characterisation framework. Australian Petroleum Exploration Association Journal 34(1): 337-349.

Wong, P.M., T.D. Gedeon, and I.J. Taggart. 1995a. Use of neural network methods to predict porosity and permeability of a petroleum reservoir. AI Applications 9(2): 27-38.

Wong, P.M., T.D. Gedeon, and I.J. Taggart. 1995b. An improved technique in porosity predictions: A neural network approach. IEEE Transactions on Geoscience and Remote Sensing 33(4): 971-980.

Wong, P.M., I.J. Taggart, and F.X. Jian. 1995c. A critical comparison of neural networks and discriminant analysis in lithofacies, porosity and permeability predictions. Journal of Petroleum Geology 18(2): 191-206.

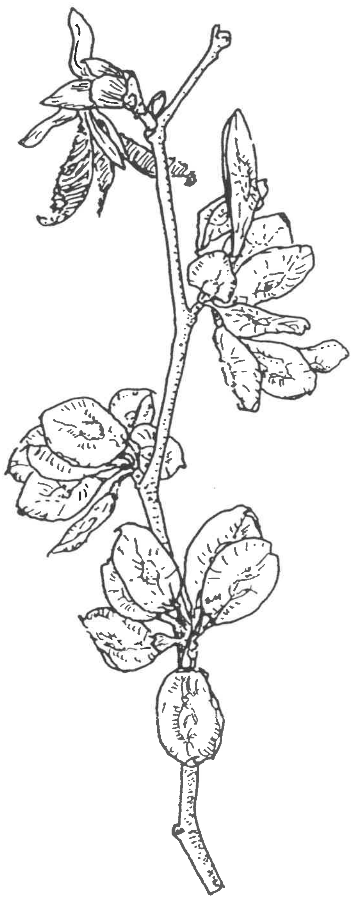
Yoon, Y., G. Swales, Jr., and T.M. Margavio. 1993. A comparison of discriminant analysis versus artificial neural networks. Journal of the Operational Research Society 44: 51-60.

Zadeh, L. 1965. Fuzzy sets. Information and Control 8: 338-353.

Patrick M. Wong received a B.E. (Hons.) and a Ph.D. in petroleum engineering in 1993 and 1996, respectively, from the University of New South Wales. He is currently a Lecturer in Petroleum Engineering at the same university. His research interests are in the area of petrophysics, geostatistics, and AI techniques. He was formerly with Petroconsultants ıstralasia Pty. Ltd. in Sydney.

Tamás D. Gedeon received a B.Sc. (Hons.) and a Ph.D. in computer science in 1981 and 1989, respectively, from the University of Western Australia. He is currently a Senior Lecturer in Computer Science and Engineering at the University of New South Wales and a Visiting Research Fellow at the Centre for Computers in Law and Finance, Brunel University, London. His research interests are in information retrieval and index generation, extracting knowledge (data mining) from trained neural networks, and fuzzy logic ᵔᵖd neural network applications.

Ian J. Taggart received a B.Math. (Hons.) in mathematical physics at the Newcastle University in 1979, and a Ph.D. degree in reservoir engineering from the University of New South Wales in 1992. He is currently a Reservoir Engineer with West Australian Petroleum Pty. Ltd. Prior to this, he was a Lecturer in Petroleum Engineering at the University of New South Wales, where he conducted research into reservoir characterization.



SUBSCRIPTION INFORMATION

AI Applications is published three times a year in April, July, and October. Prepaid individual subscriptions $37; government agencies, university departments, libraries, private companies, and foreign subscriptions (including Canada) US$72 (US$16 extra optional for air mail to foreign addresses). Send subscription requests to: AI Applications, P.O. Box 3066, Moscow, Idaho 83843 USA (telephone 208-885-7033, fax 208-885-6226). Back issues are also available. Please write for information.