Permeability Prediction in Petroleum Reservoir using a Hybrid System

Y. Huang[1#], P.M. Wong[2] and T.D. Gedeon[3#]

[1]TechComm Simulation Pty Ltd, 1/53 Balfour Street
Chippendale NSW 2008, Australia (yuantuh@techsim.com.au)
[2]School of Petroleum Engineering, University of New South Wales
Sydney NSW 2052, Australia (pm.wong@unsw.edu.au)
[3]School of Information Technology, Murdoch University
Murdoch WA 6150, Australia (t.gedeon@murdoch.edu.au)

Abstract: This paper introduces and demonstrates a hybrid soft computing system for predicting reservoir permeability of sedimentary rocks in drilled wells in the petroleum exploration and development industry. The method employs Takagi-Sugeno's fuzzy reasoning, and its fuzzy rules and membership functions are automatically derived by neural networks and floating-point encoding genetic algorithms. The method is trained with known data and tested with unseen data. The results show that the hybrid system has a good generalisation capability and is effective for industrial applications.

1. Introduction

1.1 Petroleum Reservoir

A petroleum reservoir is a volume of porous sedimentary rock which has been filled with hydrocarbons, such as oil and gas. Reservoir permeability (a measure of fluid conductance in porous media) is an important parameter for characterising complex geological formation, reserves estimation and production forecasting. This property is commonly obtained from drilled wells. Electronic equipment is used to *log* the well in such a way that multi-type digital measurements or *well logs* are obtained as a function of drilled depth.

Permeability can be estimated by correlating well logs with the laboratory measured permeability data obtained from rock samples or *cores*. Retrieving cores for laboratory testing is tedious and expensive. It is only practiced at selected depths/intervals. Therefore, permeability prediction at the *un-cored* wells relies strongly on functional transformation of well logs developed at the *cored* wells.

[#] Paper prepared at the University of New South Wales, Australia.

Y. Suzuki et al. (eds.), *Soft Computing in Industrial Applications*

1.2 A Regression Problem

The problem is traditionally solved by developing simple transfer functions, which attempt to match the target permeability values by manually adjusting constants and exponents of the functions. In many cases, these equations perform unsatisfactorily. Multivariate statistical techniques have offered a new insight and provided a potential solution by multiple regression. The regression approach directly takes the target values to minimise the prediction error. The transfer functions, however, oversimplify the natural complexity of the reservoir data. The advent of soft computing techniques (e.g. neural networks, fuzzy logic and genetic algorithms) offers powerful tools for further improving permeability predictions.

The regression approach requires a set of training patterns with known inputs (well logs) and targets (permeability). These patterns can be obtained from cored wells. The permeability required at the un-cored wells can be estimated by using the same types of well logs as inputs to the transfer function developed. If multiple cored wells are available, we may develop a transfer function at each cored well. An appropriate weighting scheme can be used to weigh the importance of each of the transfer functions and obtain the final (averaged) prediction at the un-cored well.

Figure 1 illustrates the problem by using three wells, namely W1, W2 and W3, drilled in the same petroleum reservoir. W1 and W2 are cored and W3 is assumed to be un-cored. We define the input-output relation as follows:

$$y_3 = \frac{\sum_{i=1}^{2} \mu_i^{\beta_i} \cdot f(\mathbf{x}_3, \boldsymbol{\theta}_i)}{\sum_{i=1}^{2} \mu_i^{\beta_i}} \tag{1}$$

where $\mathbf{x}_3 = (x_1, ..., x_m)_3$ denotes the input log data (m types) at W3, $\boldsymbol{\theta}_i$ denotes a set of parameters of the transfer function $f(.)$ developed at cored well i, $\boldsymbol{\mu} = (\mu_1, \mu_2)$ is a weighting vector, $\boldsymbol{\beta} = (\beta_1, \beta_2)$ is a real number in $(-\infty, +\infty)$ and y_3 denotes the output permeability.

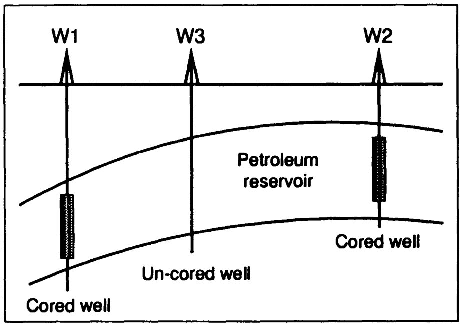

Figure 1. Permeability prediction problem.

Given the training patterns at W1 and W2, $\{\mathbf{x}_i, t_i\}, i=1,2$ where t is the target value, we may apply regression to optimise $\boldsymbol{\theta}$ by minimising the following objective function, O:

$$O(\boldsymbol{\theta}_i) = \sum_{j=1}^{n_i} (f(\mathbf{x}_i, \boldsymbol{\theta}_i)_j - t_{i,j})^2 \quad (2)$$

where n_i is the number of training patterns at cored well i.

In this paper, we propose to use a hybrid soft computing system to optimise $\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\beta}$. The system combines neural networks, fuzzy reasoning and genetic algorithms for an industrial application.

1.3 Previous Works

There are many solutions to the above problem, ranging from multiple regression, neural networks, and neural-fuzzy techniques. Both multiple regression [1][2] and backpropagation neural networks [3][4] minimise Equation (2) using a combined data set (W1 and W2) and generate y_3 without considering $\boldsymbol{\mu}, \boldsymbol{\beta}$. The major problem of the previous techniques is the use of a combined data set, which essentially averages the location-specific input-output relations.

We have avoided the averaging problem by training a separate network for each cored well [5][6]. The final prediction was obtained by using the Takagi-Sugeno's fuzzy reasoning [7] as shown in Equation (1). In [5][6], we used the separation distances between W1 and W3, and W2 and W3 as the fuzzy membership function values $\boldsymbol{\mu} = (\mu_1, \mu_2)$ and set $\boldsymbol{\beta} = (-1,-1)$. Hence, Equation (1) becomes the conventional inverse-distance estimator, which assigns a higher weighting to the estimate from the cored well closer to the un-cored well. We showed that this approach avoids the averaging problem. However, it ignores the internal similarity of the data sets among the drilled wells. It may be ineffective when applied to complex reservoirs.

In the past years, we have also looked at the optimisation of the fuzzy membership function values and the connection weights of the multi-layer perceptrons. We have used a neural-driven fuzzy reasoning method [8] to optimise the membership function values by genetic algorithms (GAs) with dramatically improved results [9]. In [10], we applied binary encoding GAs [11] to optimise the connecting weights. Our results were improved compared to weight training by the backpropagation algorithm (BP), but were computationally more expensive. One potential solution is to use the connection weights trained by BP to initialise the chromosomes in GAs. This requires floating-point encoding GAs which are fast and accurate [12], but are not popular for industrial applications to date.

1.4 Objective

The objective of this paper is to develop a hybrid soft computing system for permeability prediction which combines our previous work on neural-fuzzy estimators and floating-point encoding genetic algorithms. The major contribution of this work is the incorporation of Takagi-Sugeno's fuzzy reasoning to account for the internal similarity of data sets using $\boldsymbol{\mu} = (\mu_1, \mu_2)$.

Section 2 presents the hybrid system with a detailed description of the floating-point encoding GAs. Section 3 applies the hybrid system to predict permeability in an oil and gas well located in the North West Shelf, offshore western Australia. Data from three cored wells are available for this study. The first two data sets are used for training/validation, while the third set is treated as unseen data and is used to test the performance of the proposed method (see Figure 1). We will also compare the performance of the proposed method with the backpropagation neural networks alone.

2. Hybrid System

2.1 Neural Networks

A standard three layer (input, hidden, and output) neural network is used to generate $f(.)$ in each cored well. The number of input neurons equals to the number of well logs used, and one output neuron is used to represent permeability. The number of hidden neurons is obtained by trial and error.

In order to avoid over-fitting, we apply early-stopping by using a validation set, i.e. to terminate training when the minimum error on the validation set is reached. We use the data from one cored well as the training set and to develop $f(.)$. The data from the other cored well is used as the validation set. Swapping the use of the data sets give two generalised transfer functions. The total root mean square errors (training plus the validation errors) are used as $\boldsymbol{\beta} = (\beta_1, \beta_2)$. Note that the smaller the β value, the better the estimator, and the higher the weighting μ^β as $\mu = (0,1)$.

2.2 Fuzzy Reasoning

To generate $\boldsymbol{\mu}$, a similar multi-layer perceptron is used. The inputs are identical, but we use two output neurons. The following targets (s_1, s_2) are used:

$$(s_1, s_2) = \begin{cases} (1,0) & \text{if} \quad \mathbf{x} \in \mathrm{W1} \\ (0,1) & \text{if} \quad \mathbf{x} \in \mathrm{W2} \end{cases} \tag{3}$$

This training scheme estimates the degree of membership of each input pattern "belonging" to each of the cored wells. The value of the membership function is defined as the output of the trained neural network, i.e., $\mu_i = \hat{s}_i$, where $\hat{s}_i$ denotes the output from the trained network.

The above weighting scheme examines the similarity of the inputs of the training data sets and the inputs of the unseen data set. It is superior to the one proposed in [5][6] which used only the inverse-distances.

2.3 Genetic Algorithms

Genetic algorithms (GAs) mimic processes observed in nature evolution, and are stochastic global search methods. Individuals in a population are called *chromosomes*. A genetic representation for a potential solution to a problem is encoded as a chromosome. A good initial population of potential solutions can result in fast convergence with higher accuracy for real world problems. The steps of a typical GA are listed as follows:

a. Initialise a population of chromosomes within the range of potential solutions.
b. Evaluate each chromosome in the population based on a *fitness* function.
c. Select the parent chromosomes to reproduce (according to the fitness values).
d. Apply genetic operators to the parent chromosomes to produce children so as to generate a new population.
e. Evaluate the chromosomes in the new population.
f. Stop and return the best chromosomes as the final solution if a termination condition is satisfied; otherwise, go to step c.

Generally speaking, the parent chromosome selection for both binary and floating-point encoding is similar. However, the genetic operators used for floating-point encoding are different from these for binary encoding. In our problem we will use the connection weights trained by BP as one of the chromosomes in the initial population, and hence, floating-point encoding is required. The following steps outline the general structure of floating-point encoding GAs:

2.3.1 Population Initialisation

Let $\boldsymbol{\theta} = (w_1, w_2, ..., w_n)$ denotes a parameter vector of connection weights trained by BP, where n is the number of total connection weights in the network. We define a real number parameter "swing" as Δw. The use of this parameter avoids the need to define the upper and lower bounds for the parameters to be optimised. Typically the value of Δw for our problem is $[\Delta w_{\min} = 0.5, \Delta w_{\max} = 10]$. Assuming the size of a population is m and we keep $\boldsymbol{\theta}$ as one of the chromosomes in the initialised population, the remaining $m-1$ chromosomes are randomly generated, and the range of the i^{th} parameter in a chromosome is $[w_i - \Delta w, w_i + \Delta w], i = 1,...n$.

2.3.2 Performance Evaluation

A fitness function is used to evaluate the performance of each of the chromosomes. A typical fitness function is defined as follows:

$$F(\boldsymbol{\theta}_j) = \frac{10}{1 + E(\boldsymbol{\theta}_j)} \tag{4}$$

and,

$$E(\boldsymbol{\theta}_j) = \sum_{k=1}^{N} (f(\mathbf{x}_k, \boldsymbol{\theta}_j) - t_k)^2 \tag{5}$$

where $F(\boldsymbol{\theta}_j)$ with $\boldsymbol{\theta}_j = (w_{j1}, w_{j2}, ..., w_{jn})$ is the fitness value for the j^{th} $(j = 1, ..., m)$ chromosome, $E(\boldsymbol{\theta}_j)$ is the sum of squared errors of the target data t_k $(k = 1, ..., N)$ and the predictions $f(.)$ and N is the total number of training patterns. The higher the $F(\boldsymbol{\theta}_j)$ value, or the lower the $E(\boldsymbol{\theta}_j)$, the better the solution.

2.3.3 Reproduction

Selecting parents for reproduction is a very important aspect of GAs. There are many selection methods. A parent solution can be selected more than once. The most popular selection method is the Goldberg's roulette wheel parent selection [13]. The roulette wheel has slots sized according to the fitness of each chromosome. The purpose is to give more reproductive chances to those population members who are the most fit. After the roulette wheel parent selection, we copy the best chromosome twice to replace two of the worst.

In order to accelerate convergence, before crossover and mutation we dynamically update the swing and the range of the parameters for the chromosomes as follows:

$$\Delta w = \Delta w_{\min} + \frac{(M - k) \cdot (\Delta w_{\max} - \Delta w_{\min})}{M} \tag{6}$$

where M is the desired number of iterations and $k (\leq M)$ is the current iteration counter. Let $\boldsymbol{\theta} = (w_1, w_2, ..., w_n)$ be the best chromosome, then the new range of the i^{th} parameter in a chromosome for a new population is $[w_i - \Delta w, w_i + \Delta w], i = 1, ... n$.

2.3.4 Crossover

Crossover produces offspring by exchanging genetic information between the selected parent solutions. The selection criteria are based on a user-defined probability for crossover, P_c (generally between 0.5 to 0.9). This probability defines the number of candidates for crossover. For example, if P_c is 0.7, it means that 70% of the parent chromosomes in the population will be selected randomly and mated in pairs. In this paper, we use two-point arithmetical crossover.

Let the i^{th} chromosome $(w_{i1}, w_{i2}, ..., w_{in})$ and the j^{th} chromosome $(w_{j1}, w_{j2}, ..., w_{jn})$ be selected for crossover between the p^{th} and the q^{th} parameters $(1 \leq p \leq q \leq n)$. The offspring becomes $(w_{i1}, ..., w_{ip-1}, w_{ip}^{j}, ..., w_{iq}^{j}, w_{iq+1} ..., w_{in})$ and $(w_{j1}, ..., w_{jp-1}, w_{jp}^{i}, ..., w_{jq}^{i}, w_{jq+1} ..., w_{jn})$ where $w_{ik}^{j} = \alpha \cdot w_{ik} + (1 - \alpha) \cdot w_{jk}$ and

$w^{i}_{jk} = \alpha \cdot w_{jk} + (1-\alpha) \cdot w_{ik}$ for $p \le k \le q$ and α is a uniform random number in [0,1].

2.3.5 Mutation

The reproduction and crossover operation would only exploit the known regions in the solution space, which could lead to premature convergence for the fitness function with the consequence of missing the global optimum by exploiting some local optimum. Mutation is a genetic process to avoid such a problem. This process allows the introduction of new characteristics to the offspring, which are unrelated to the parent solutions. It first requires a user-defined probability for mutation P_m (generally between 0.01 to 0.2). Here, we use non-uniform arithmetical mutation. Let the j^{th} parameter in the i^{th} chromosome w_{ij} is selected for mutation. Thus, the new parameter is $w^{*}_{ij} = \alpha \cdot w_{ij} + (1-\alpha) \cdot v$, where v is a uniform random number in $[w_i - \Delta w, w_i + \Delta w]$. More details about genetic operators for floating-point encoding can be found in [14].

2.4 Neural-Driven Fuzzy Reasoning

After generating the functions $f(.)$ with optimised $\boldsymbol{\theta}$, we can extract two fuzzy rules from the two cored wells, W1 and W2:

Rule 1: If $\mathbf{x}_3 \in$ W1 then $y_1 = f(\mathbf{x}_3, \boldsymbol{\theta}_1)$

Rule 2: If $\mathbf{x}_3 \in$ W2 then $y_2 = f(\mathbf{x}_3, \boldsymbol{\theta}_2)$

Similar optimisation routines can be run to obtain the membership function values $\boldsymbol{\mu} = (\mu_1, \mu_2)$ in Equation (3). Equation (1) can then be applied to obtain the final estimate.

3. Field Example

In this case study, data from three wells, W1, W2 and W3, located in the North West Shelf, offshore western Australia, were used. The well logs available for the analyses were: gamma ray (GR), deep resistivity (LLD), sonic travel time (DT), bulk density (RHOB) and neutron porosity (NPHI). The 11 rock classifications were converted to 11 values spread evenly within the interval [0,1]. Permeability measurements were available at selected well depths. There are a total of six inputs and one output. All the well log data were normalised in the range of [0,1]. All the permeability values were normalized in the range of [0.1,0.9]. The number of data pairs in each well was 152, 156, and 140 points, respectively. We aim to predict the permeability values at W3 using the transfer functions developed at W1 and W2.

In this study, a three-layer neural network with five hidden neurons was found to be the best structure. Due to the presence of the bias weights (only in the input layer), $f(.)$ had a total of 40 connection weights. For the network of $\boldsymbol{\mu} = (\mu_1, \mu_2)$, the use of two output neurons resulted in 45 connection weights. The GA configuration is shown in Table 1.

Table 1: GA configuration.

Maximum iteration	5,000
Population size, m	20
Probability for crossover, P_c	0.8
Probability for mutation, P_m	0.1

Table 2 shows the training and validation errors for $f(.)$, optimised by the floating-point encoding GAs using BP trained weights as the initial population. The results using BP alone are also tabulated in the same table. From these results, the errors from BP followed by GAs were smaller (10% and 7% lower respectively) and hence the rules were more reliable for prediction. The corresponding $\boldsymbol{\beta} = (\beta_1, \beta_2)$ was also calculated. Note that the similarity of the β values showed that the two transfer functions had similar prediction reliability.

Table 2. Root mean square errors from BP followed by GA.
The bracketed values are results from BP alone.

	Training error	Validation error	β value
W1 for training W2 for validation	0.065 (0.072)	0.075 (0.084)	0.140 (0.156)
W2 for training W1 for validation	0.073 (0.075)	0.076 (0.086)	0.149 (0.161)

Figure 2 shows the trained hyper-surface membership function values $\boldsymbol{\mu} = (\mu_1, \mu_2)$ obtained by GAs. These values were calculated by using $\mathbf{x}_3$ (140 points in total), the input patterns of W3 in the fuzzy reasoning step. The plots show the degree of membership of W3 patterns "belonging" to W1 and W2 respectively. We can see that the majority of the W3 patterns are similar to the W2 patterns.

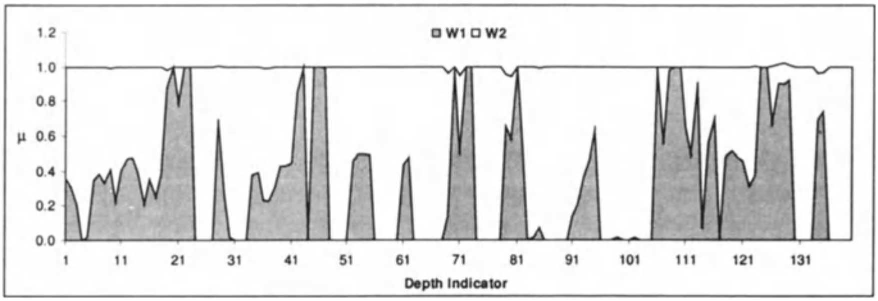

Figure 2. Membership of the W3 patterns "belonging" to W1 and W2.

4. Results

The results from the hybrid system are tabulated in Table 3. The results from using the separate rules trained by BP alone are also shown in the same table. Clearly, the hybrid system gave the smallest total sum of error squares (TSS) and the highest correlation coefficient (r^2). When comparing the predictions from those of the separate rules, the TSS from the hybrid system was 26% and 20% smaller than rule 1 and rule 2 respectively. Similarly, the r^2 was 11% and 16% higher respectively. Figure 3 shows the predictions at W3 along with the actual permeability data. The predictions matched very well with the actual values.

Table 3. Comparison of TSS and r^2 of 140 data points at W3 using the hybrid system and the separate rules developed by BP alone.

	TSS	r^2
Hybrid system	0.939	0.73
Rule 1 from BP alone	1.273	0.66
Rule 2 from BP alone	1.180	0.63

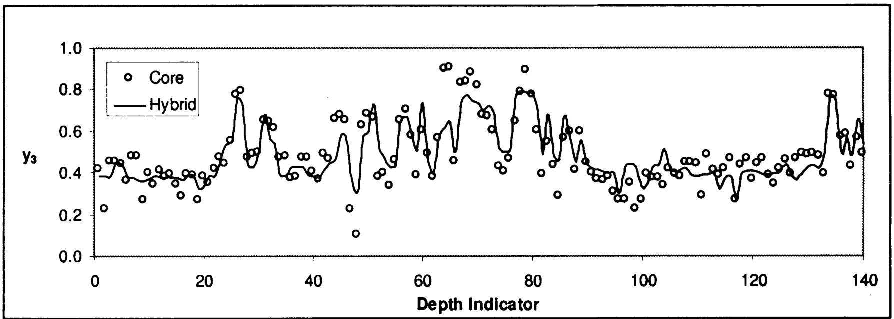

Figure 3. Permeability profiles at W3.

The performance of the hybrid system was good compared to our previous approach. It has not only the intrinsic advantages of the neural-driven fuzzy reasoning, but also incorporates the floating-point encoding GAs to further optimise the neural networks trained by BP. Because the chromosomes of the initial population in GAs come from the BP-trained weights, the final results are never worse than those obtained by BP alone. Therefore, the combination of the GAs with BP can provide fast and accurate results.

5. Conclusions

In this paper, we present the use of a hybrid soft computing system to predict permeability in a petroleum reservoir in offshore western Australia. The system uses the neural-driven fuzzy reasoning combining with the floating-point encoding genetic algorithms. It is a robust and flexible estimator for industrial applications. In the field example, the results showed that the hybrid system provided the smallest error and the highest correlation coefficient on the unseen data compared to the conventional method using backpropagation neural networks alone.

References

[1] Bloch, S., 1991, Empirical prediction of porosity and permeability in sandstones. *AAPG Bulletin*, **75**, 1145-1160.
[2] Jian et al., 1994, A genetic approach to the prediction of petrophysical properties. *Journal of Petroleum Geology*, **17**, 71-88.
[3] Wong, P.M. Taggart, I.J. and Gedeon, T.D., 1995, Use of neural network methods to predict porosity and permeability of a petroleum reservoir. *AI Applications*, **9**(2), 27-38.
[4] Mohaghegh et al., 1996, Petroleum reservoir characterization with the aim of artificial neural networks. *Journal of Petroleum Science and Engineering*, **16**, 263-274.
[5] Huang, Y., Wong, P.M. and Gedeon, T.D., 1997, Spatial interpolation in log analysis using neural-fuzzy technique. *59th EAGE Conference & Technical Exhibition*, Geneva, Extended Abstracts, vol. 1, P174.
[6] Gedeon et al., 1997, Two dimensional neural-fuzzy interpolation for spatial data. *Proceedings of GIS AM/FM ASIA '97 & Geoinformatics '97*, Taipei, Taiwan, vol. 1, 159-166.
[7] Takagi, T. and Sugeno, M., 1985, Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man & Cybernetics*, **15**(1), 116-132.
[8] Takagi, H. and Hayashi, I., 1991, NN-driven fuzzy reasoning. *International Journal of Approximate Reasoning*, **5**, 91-212.
[9] Huang Y., Wong, P.M. and Gedeon T.D., 1998, Neural-fuzzy-genetic-algorithm interpolator in log analysis. *60th EAGE Conference and Technical Exhibition*, Leipzig, Germany, Extended Abstracts, vol. 1, P106.
[10] Huang, Y., Wong, P.M. and Gedeon, T.D., 1998, Prediction of reservoir permeability using genetic algorithms. *AI Applications*, **12**(1-3), 67-75.
[11] Holland, J., 1975, *Adaptation in Natural and Artificial Systems*. Ann Harbor: University of Michican Press.
[12] Wang, X. and Elbuluk, M., 1996, Neural network control of induction machines using genetic algorithm training. *Proceedings of the 31st IEEE IAS Annual Meeting*, 3, 1733-1740.
[13] Goldberg, D.E., 1989, *Genetic Algorithms in Search, Optimization, and Machines Learning*. Addison-Wesley, Reading, MA.
[14] Michalewicz, Z., 1994, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin.