Reservoir Uncertainty Assessment Using Machine Learning Techniques

Jincong He

***Abstract*— Petroleum exploration and production are associated with great risk because of the uncertainty on subsurface conditions. Understanding the impact of those uncertainties on the production performance is a crucial part in the decision making process.Traditionally, uncertainty assessment is performed using experimental design and response surface method, in which a number of training points are selected to run reservoir simulations on and a proxy model is built to give prediction for the whole design space. The quality of the response surface strongly depends on the method used to construct them. In this paper we evaluate the use of thin plate spline (TPS), artificial neural network (ANN) and support vector regression (SVR) for this application. Results show that when properly tuned ANN and SVR provide superior performance to TPS.**

I. INTRODUCTION

Reservoir uncertainty assessment is one of the most important aspects in production decision making. Because of the heterogeneity underground and also the scale of the oil fields, even with today's advanced seismic and well logging techniques, many of the reservoir parameters can not be measured accurately. Some of these parameters, such as the permeability, porosity and the residual phase saturation, may have a large impact on the oil production forecast. Therefore, quantifying the production forecast uncertainty arising from the parameter uncertainty would play an important role in monitoring the risk of investment.

Mathematically, the problem can be formulated as

$$R = h(x_1, x_2, \cdots, x_n) \tag{1}$$

x_i are uncertain reservoir parameters such as permeability multipliers, porosity and residual saturations. R is the response function of the reservoir under a specific set of parameters. It is usually taken as some economical metric of the field, such as net present value (NPV) of the development project. Given the probability distribution of x_i to be $P_i(x)$, our goal is to find the probability distribution of $P(R)$.

The ideal method for uncertainty assessment would be Monte Carlo simulation. By sampling a large number of points according to the parameter distribution and running simulations on each of them, we can calculate the statistics of results and estimate their uncertainty. However, by the law of large numbers, Monte Carlo simulation will only display $1/\sqrt{N}$ convergence-i.e. quadrupling the number of sampled points will only halve the error. On the other hand, production simulation (the process to evaluate R) for practical sized reservoir is very time consuming. One single sample may require hours to obtain. For typical problem only 50 to 100 samples would be available. Therefore, Monte Carlo method is seldom used in practice.

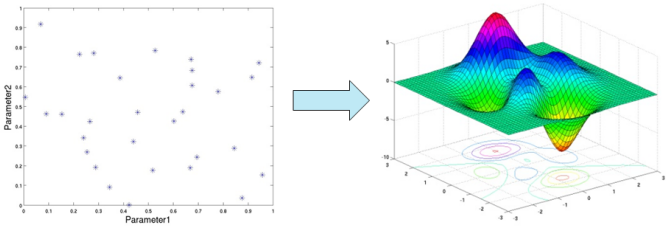

Fig. 1. Illustration of experimental design procedure

Experimental design method provides an alternative to Monte Carlo simulation. As shown in Fig. 1, the basic idea behind this methodology is to sample a few training points in the parameter space in a way that maximum inference can be attained with minimum cost. After that, a proxy model will be built based on the training points and be used to estimate the response for the whole parameter space. Finally, Monte Carlo simulation will be carried out using the proxy model to calculate the probability distribution of the response. Two key steps that affect the performance of the algorithm the most are the design step, how the training points should be chosen, and the proxy building step, how the proxy should be built. [1] reviewed and compared many of the current experimental design techniques. For the proxy building, thin plate spline and artificial neural network (ANN) are commonly used. As far as we know there has not been any investigation on the applicability of support vector regression (SVR) for this problem. This paper focuses on evaluating and comparing the three approaches. We will proceed as follows: First a brief overview of thin plate spline, ANN and SVR will be given . Then all three methods will be tested on an uncertainty assessment case for a synthetic reservoir. Finally comparisons of the results from different methods will be made in terms of reliability of their estimation on the output statistics, accuracy of their point-wise estimations and their capability in terms of estimating the influence of uncertain parameters on the economical or recovery responses.

II. METHODS

A. Thin Plate Spline (TPS)

Thin plate spline (TPS) is a popular technique for multivariate interpolation. The name "thin plate spline", first introduced by Duchon[2] to geometric design, refers to a physical analogy involving the bending of a thin sheet of metal. Given a set of training points $\{(x^{(i)}, y^{(i)}), i = 1, 2, \ldots, m\}$, thin plate spline tries to "bend the metal sheet" so that it passes through the training points exactly with the least amount of energy. Mathematically, we try to minimize

$$E_{TPS} = \sum_{i=1}^{m} \|h(x^{(i)}) - y^{(i)}\|^2 + p \int_{\Omega} |\Delta_x h|_f d\Omega \quad (2)$$

where $y = h(x)$ is our hypotheses, which will be discussed later. p is the smoothing coefficient, Ω is the n dimensional space where x is defined, $\Delta_x h$ is the Hessian matrix and $|\cdot|_f$ is the Frobenius norm. The first term of Eq. 2 characterizes the mismatch error while the second terms, which is also called the bending energy, characterizes the smoothness of the function.

In thin plate spline, the function is parameterized as a linear combination of the thin plate spline radial basis function as follows:

$$h(x) = \sum_{i=1}^{m} c_i \phi(\|x - x_i\|) \quad (3)$$

where $\phi(r) = r^2 \log r$ and $c = [c_1, c_2, \ldots, c_m]$ are the weights.

With this parameterization, the problem in Eq. 4 is transformed to

$$c = arg\min_{c} E_{TPS} \quad (4)$$

Details for solving the problem in Eq. 4 can be found in [3]. In this work, we used the Spline Toolbox in Matlab.

B. Artificial Neural Networks (ANN)

ANN is a nonlinear regression model inspired by the structure and the functional aspects of biological neural networks. A neural network consists of input nodes, output nodes and interconnected groups of artificial neurons. A feedforward network consists two or more layers of these nodes. The first layer is the input layer, consisting of all input nodes. The last layer is the output layer, consisting of all output nodes. All the other layers are termed hidden layers, consisting of neurons. Fig. 2 shows an example network with one of each kind of layers.

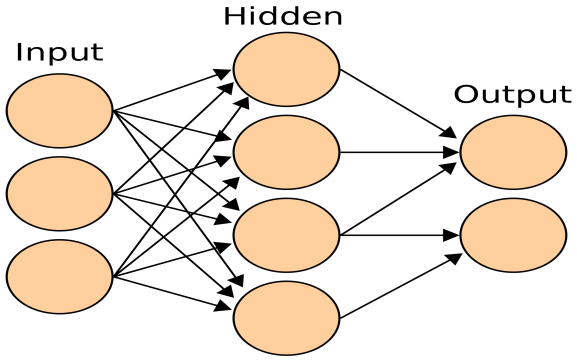

Fig. 2. An example neural network consisting of one input layer, one output layer and one hidden layer (from wikipedia)

Denote p_j^i and q_j^i to be the input and output of the jth node on the ith layer and n_i to be the number of nodes on the ith layer. Then for the input and output layers we have $q_j^1 = p_j^1$ and $q_j^n = p_j^n$. For each of the hidden layers, we have

$$q_j^{i+1} = \sum_{k=1}^{n_i} W_{jk}^i p_k^i + b_j^i \quad (5)$$

where W^i is the weight matrix and b^i is the bias vector between the ith and the $i+1$th layers. Inside the neurons, the input is processed by a transfer function, which can be any differentiable function but is usually selected to be the log-sigmoid function as follows

$$g(x) = \frac{1}{1 + e^{-x}} \quad (6)$$

To specify neural network, it is sufficient to find a set of weights and bias vectors that minimized some kind of error. In this work, this process is done by Levenberg-Marquart backpropagation algorithm. Starting with a set of randomly generated initial guess of weights and bias vectors, the algorithm alters the parameters to minimize the error between the desired and the actual output.

In this study we employ the neural network toolbox in matlab and use a feedforward artificial neural network with one hidden layer of log-sigmoid neurons. The number of nodes on the input layer, N_{in} equals the number of uncertain parameters and there is only one node on the output layer, i.e. $N_{out} = 1$.

C. Support Vector Regression (SVR)

Among the many proxy building approaches, support vector regression is the least investigated for reservoir uncertainty. In ϵ-SV regression, our goal is to find a function $f(x)$ that has at most ϵ deviation at the training points, and at the same time is as flat as possible. The detail derivation of various kinds of support vector regression was presented in [4] and [5]. For completeness an abbreviated description is also included here.

For a linear case where $y = f(x) =< w, x > +b$, this translates into the following constraint optimization problem.

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \quad (7)$$

$$\text{subject to } \begin{cases} y_i - < w, x^{(i)} > -b \leq \epsilon + \xi_i \\ -y_i + < w, x^{(i)} > +b \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \quad (8)$$

Here, $< \cdot, \cdot >$ is the inner product of two vector. $\|w\|^2 =< w, w >$ measures the flatness of the proxy and the constraints requires the proxy approximates all training points with ϵ precision. ξ_i, ξ_i^* are slacks variables introduced to allow trade-offs between the flatness of the function and the compliance with the ϵ deviation constraints. C characterizes the amount of penalty for violating the constraint.

By formulating the Lagrangian, the dual optimization problem of Eq. 7 can be written as

$$\text{minimize} \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*) < x^{(i)}, x^{(j)} > \\ -\epsilon\sum_{i=1}^{m}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{m} y_i(\alpha_i-\alpha_i^*) \end{cases} \tag{9}$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{m}(\alpha_i-\alpha_i^*)=0 \\ 0 \le \alpha_i, \alpha_i^* \le C \end{cases} \tag{10}$$

Eq. 9 is a convex optimization problem which can be solved efficiently by many optimization package, e.g. OSL, CPLEX, or MINOS. With α_i, α_i^* solved from Eq. 9. Our proxy function can be written as follows

$$h(x) = \sum_{i=1}^{m}(\alpha_i-\alpha_i^*) < x^{(i)}, x > +b \tag{11}$$

See [6] for more detail on the computation of the constant bias term b.

It's straightforward to kernelize the complete algorithm as the training problem in Eq. 9 and the prediction problem in Eq. 11 are all in terms of the inner product of two features. Therefore by replacing $< x^{(i)}, x^{(j)} >$ with any kernel function $K(x^{(i)}, x^{(j)})$, we can introduce nonlinear features without the need to calculate x or w explicitly. The Gaussian kernel in Eq. 12

$$K(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right) \tag{12}$$

In this work we used libSVM [7] for our SVR implementation. The effect of various kinds of kernels and different choices of parameters are also investigated for our problem.

III. TEST RESULT

The three methods discussed in section II are tested on a reservoir production uncertainty assessment problem. Figure 3 presents the top view of the reservoir model used in this problem. The model is generated using the sequential Gaussian simulation method. There are totally 8 producers (marked by dots) and 8 injectors (marked by triangles). Flow simulations by reservoir simulator GPRS [8] are run from day 0 to day 800 and the total oil production Q_o, water production Q_w and water injection Q_{inj} are recorded. We study the effect of the seven parameters presented in Table III on the uncertainty on net present value (NPV), as defined below.

$$\text{NPV} = Q_o p_o + Q_w p_w + Q_{inj} p_{inj} \tag{13}$$

Here, p_o, p_w and p_{inj} are the prices of the oil and cost of produced water treatment and cost of injected water. They are set to be \$80/bbl, -\$5/bbl and -\$3/bbl respectively.

Six training data sets with the sizes of 25, 50, 75, 100, 125 and 150 are generated using Latin hypercube sampling [9] on the parameter space. Each training result is generated by running the GPRS simulator and therefore is considered to be true without error. A test set of 10000 samples is generated in the same way. The three methods presented in Section II are trained on each of the training set and are evaluated on the test set. The evaluation metric is the mean prediction error of the test data set.

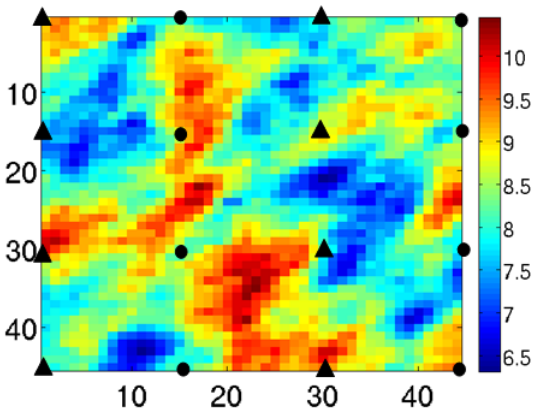

Fig. 3. Top view of the reservoir model used, log transmissibility in x-direction is shown

TABLE I
UNCERTAIN PARAMETERS

Param.	Param. Description	Min	Max
H	Net thickness, ft	20	40
PORO	Porosity	0.2	0.3
XPERMMUL	x-permeability multiplier	0.1	2
YPERMMUL	y-permeability multiplier	0.1	2
S_{wr}	Residual water saturation	0.15	0.25
S_{or}	Residual oil saturation	0.15	0.25
C_r	Rock compressibility	1e-6	1e-5

A. Base case

There are two parameters (C and ϵ) in the SVR method and one (number of hidden neurons) in the ANN method. Their effect on the proxy will be investigate in section III-B. For the base case, we set $C = 10000$, $\epsilon = 0$ and we use 5 hidden neurons for the ANN method. Figure 4 shows the learning curves for all three methods. It is clear that the error of all three method decreases as the number of training data increase. However, using the same amount of training data, the error from TPS is always at least one order of magnitude higher than the error of SVR and ANN. Comparing SVR and ANN, when the number of training data are small, SVR provides superior result to ANN. As the amount of the available training data increases, the error of the neural network decrease faster than that of the SVR, and ANN outperform SVR when the number of training data is 75 or higher. For practical applications, the maximum number of simulation can be run normally ranges from 50 to 100. Therefore, both SVR and ANN are good candidates for building a good proxy on training data of that size. It is also noticeable that the error of ANN does not decrease monotonically. This may in part due to the fact that the initial guess of the weights in ANN are randomly generated and the optimization problem of the weights may have multiple local minima.

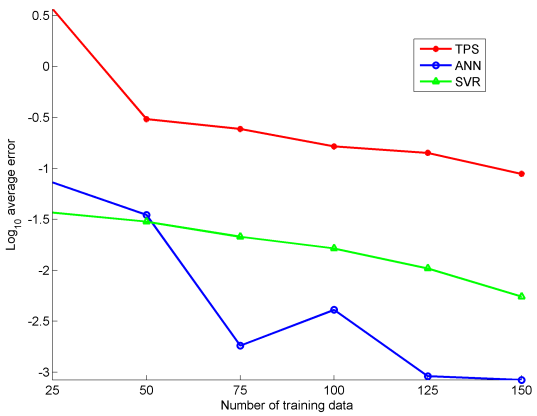

Fig. 4. Learning curve: average prediction error vs training set size

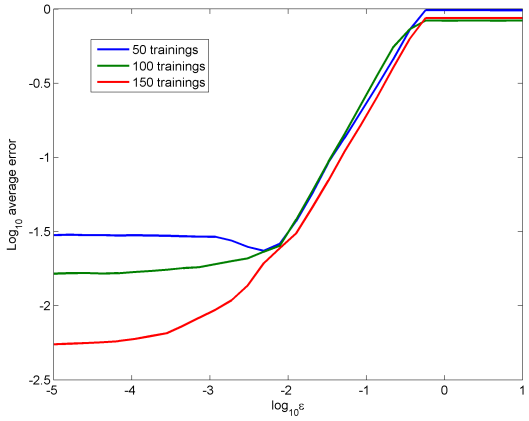

Fig. 5. Effect of ϵ on the SVR model accuracy

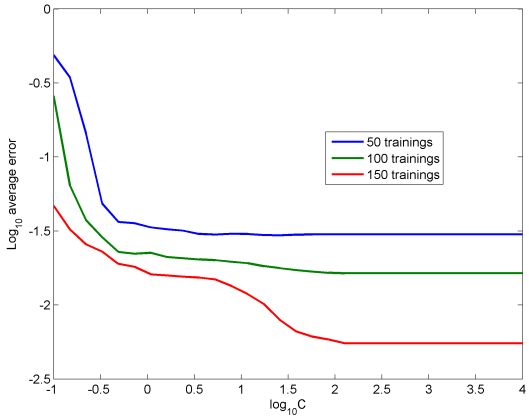

Fig. 6. Effect of C on the SVR model accuracy

B. Parameter investigation

As mentioned above, there are two parameters (C and ϵ) in the SVR method and one (number of hidden neurons) in the ANN method. It would be of interest to investigate the effect of those parameters on the model accuracy and provide guidelines for practical uses.

Figure 5 is a log-log plot of the average prediction error of the SVR model for different value of ϵ for three training sets of different size (50,100 and 150). It can be seen that the average error decreases with smaller ϵ, and when ϵ is small enough, the error stays constant. As ϵ is the tolerance by which the proxy can deviate from the training points, the result of Figure 5 indicates that the proxy should approximate the training point perfectly. This is reasonable for our application because the training data are generated by simulation and therefore have no error. If the training data were collected from experiments with errors, the optimal choice of ϵ would be larger than 0. Figure 6 shows the average prediction error of the SVR model for different value of ϵ for three training sets of different size (50,100 and 150). It is clear that the average error decrease as C increase, and stays at constant value when C is large enough. Note that C is the coefficient of the slack variables and it characterizes the penalty for the proxy to violate the C deviation at training points. Figure 6 indicates that the penalty for large deviation at training points should be very high, or in other word, the slack variables should not be introduced. This is, again, because we have perfect training data from a simulator. The above observations suggest that for our application the introduction of slack variable may not be necessary and the inequality constraints in Eq.7 can be simplified to be equality constraints. This would be an interesting topic for future work.

Figure 7 shows the effect of using different number of hidden neurons on the average prediction error. The fluctuation of the result dues in part to the random components in the ANN method, and in part to the sensitivity of the method to the number of neural used. Despite the fluctuation, it can still be observed that there is an optimal choice of the number of hidden neurons for each cases. The result deteriorates if too many or too few neurons are used. There is still not consensus in the literature on how the optimal number of neurons should be chosen. [10] proposed the following formula (and rounding it to the next integer).

$$N_{\text{neurons}} = \sqrt{N_{\text{input parameters}} \times N_{\text{output parameters}}} \tag{14}$$

In our case, the above formula would suggest using only 3 hidden neurons, which is clearly not optimal. More sophisticated method should optimize the parameters by some cross-validation techniques such as leave-one-out cross validation.

Figure 8 shows the cumulative distribution of the 10000 results from the true simulations, and the three proxy models. It can be seen that results from ANN and SVR match with the true solution reasonably well (they almost overlap), while the result of TPS deviate slightly. Table III-B shows the relative error of the estimation of 10%, 50%, 90% percentiles (P10, P50, P90) for the three methods. These three quantities are of significance because they represent the pessimistic, average and optimistic estimations of the production response. It can be seen in the table that ANN and SVR provides

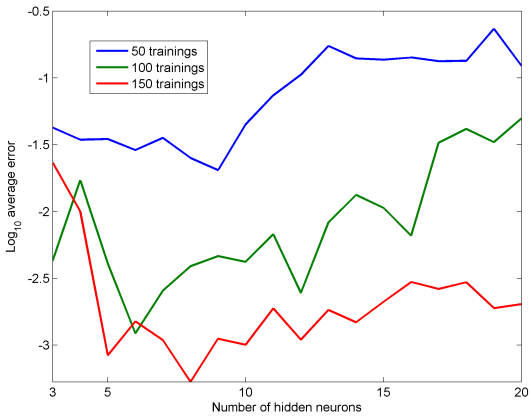

Fig. 7. Effect of the number of hidden neurons on the ANN model accuracy

very accurate estimation of these quantities. In terms of computational efficiency, the generation of 10000 samples using reservoir simulator takes more than 1 days while the three proxy models only take several seconds to generate 10000 predictions. Therefore, uncertainty analysis based on these proxies is very efficient.

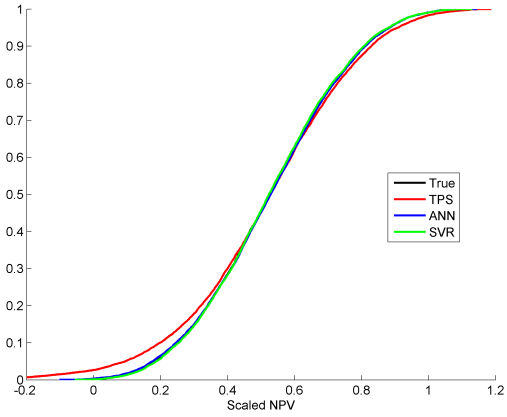

Fig. 8. The cumulative distribution of the results from different methods

TABLE II
RELATIVE ERROR FOR ESTIMATIONS OF P10, P50 AND P90

	Spline	ANN	ϵ-SVR
P10	-21.0%	-1.9%	1.39%
P50	0.51%	0.60%	0.37%
P90	2.59%	0.43%	0.11%

IV. CONCLUSION AND DISCUSSION

In this study, we investigated the use of thin plate spline (TPS), artificial neural networks (ANN) and support vector regression (SVR) for uncertainty assessment of the reservoir production activities. Using a small amount of training data from a simulator, these three methods were used to build a proxy of the reservoir response with respect to uncertain parameters for Monte Carlo simulation. Results shows that using the same amount of training data, SVR and ANN always outperform TPS when the parameters of the methods are properly selected. When less training data is available, SVR seems to be more accurate than ANN.

The parameter sensitivity of SVR and ANN is also investigated. For SVR, it is suggested that C should be taken to be very large and ϵ should be close to 0 for the application we are interested in. For ANN, the performance of the method depends heavily on the the number of hidden neurons, and a good choice of this parameters may require optimization using cross validation techniques.

REFERENCES

[1] B. Yeten, A. Castellini, B. Guyaguler, and W. H. Chen. A comparison study on experimental design and response surface methodologies. In *2005 SPE Reservoir Simulation Symposium*, The Woodlands, Texas, USA, January 2005.

[2] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In Walter Schempp and Karl Zeller, editors, *Constructive Theory of Functions of Several Variables*, volume 571 of *Lecture Notes in Mathematics*, pages 85–100. Springer Berlin / Heidelberg, 1977. 10.1007/BFb0086566.

[3] B. Li and F. Friedmann. Novel multiple resolutions design of experimental/response surface methodology for uncertainty analysis of reservoir simulation forecasts. In *2005 SPE Reservoir Simulation Symposium*, The Woodlands, Texas, USA, January 2005.

[4] A. J. Smola and B. Scholkopf. A tutorial on support vector regression. *NeuroCOLT Technical Report*, (TR-98-030), 2003.

[5] B. Scholkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2005.

[6] S. S. Scholkopf, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murty. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13:637–649, 2001.

[7] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[8] H. Cao. *Development of Techniques for General Purpose Simulators*. PhD thesis, Stanford University, 2002.

[9] K. Fang. Uniform design: application of number theory in test design. *ACTA Mathematicae Applicatae Sinica*, 1980.

[10] T. Master. *Practical neural network recipes in C++*. Academic Press, 1993.