GEOPHYSICS, VOL. 81, NO. 4 (JULY-AUGUST 2016); P. IM35–IM43, 6 FIGS.
10.1190/GEO2015-0297.1

Hybrid combination genetic algorithm and controlled gradient method to train a neural network

Alexander Kobrunov[1] and Ivan Priezzhev[2]

ABSTRACT

Multivariate predictive analysis is a widely used tool in the petroleum industry in situations in which the deterministic nature of the relationship between a variable that requires prediction and a variable that is used for the purposes of such prediction is unknown or very complex. For example, to perform a sweet-spot analysis, it is necessary to predict potential oil and gas production rates on a map, using various geologic and geophysical attribute maps (porosity, density, seismic attributes, gravity, magnetic, etc.) and the initial oil and gas production rates of several control or training wells located in the area of interest. We have developed a new technology that allows for building a stable nonlinear predictive operator by using the combination of a neural network, a genetic algorithm, and a controlled gradient method. The main idea behind the proposed technology is to combine stochastic and deterministic approaches during the construction of the predictive operator at the training stage. The proposed technology avoids many disadvantages of the genetic algorithm and gradients methods, such as a high level of dependency on the initial values; the phenomenon of over-fitting (overtraining), which results in creation of an operator with unstable predictability; and a low speed of decreasing error during iteration, and, as a result, a low level of prediction quality. However, the above-mentioned combination uses the advantages of both methods and allows for finding a solution significantly closer to a global minimum for the objective function, compared to simple gradient methods, such as back propagation. The combination of these methods together with Tikhonov regularization allows for building stable predictions in spatial or/and time coordinates.

INTRODUCTION

A neural network is a nonlinear operator that is widely used for predictive analysis in the petroleum industry (Ali, 1994; Roth and Tarantola, 1994; Schultz et al., 1994a, 1994b; Chen and Sidney, 1997; Russell et al., 1997; Boadu, 1998; Liu and Liu 1998; Hampson et al., 2001; Mogensen and Link, 2001; Veeken et al., 2009; Priezzhev et al., 2014). A neural network predictive operator has several significant advantages (Schultz et al., 1994a, 1994b; Bishop, 1995; Girosi et al., 1995). First, the operator allows for controlling of the nonlinearity and the degree of freedom via the number of hidden nodes and through the choice of the activation function type. Second, the operator can be used in very complex cases in which the relationship between dependent (predicted) variables and independent (input) variables is unknown or turns out to be too complex to use a deterministic approach.

The neural network predictive operator has also several disadvantages (Schultz et al., 1994a, 1994b; Bishop, 1995; Girosi et al., 1995). Among them is the phenomenon of overfitting (overtraining) resulting in creation of an operator with unstable predictability. In addition, if the training data set range does not cover the space of possible dependent variable values, then the trained operator can again result in unstable predictability.

To train a neural network, two main techniques are usually applied. First, there is a family of back-propagation algorithms, which typically implement gradient methods (Bishop, 1991, 1995; Schultz et al., 1994a, 1994b). The main problem with this type of training algorithms is that they have the following limitations: First, only a local minimum of the objective function that is close to the initial value of the neural network parameters is estimated. Second, this technique can produce very different predictive results for

Manuscript received by the Editor 29 May 2015; revised manuscript received 20 March 2016; published online 31 May 2016.
[1]Ukhta State Technical University, Ukhta, Russia. E-mail: aikobrunov@gmail.com.
[2]Schlumberger, from December 2015 acting as independent consultant, Moscow, Russia. E-mail: ivanpriez@gmail.com.

several different initial values. As a result, this technique has a very low probability of finding a global minimum.

The second type of training technique consists of finding optimal neural network parameters by using a genetic algorithm (Whitley et al., 1990; Bishop and Bushnell, 1993; Veeken et al., 2009). On the one hand, this technique allows simultaneous analysis of many variants of neural networks and thus enables us to find the global minimum of the objective function with high probability. On the other hand, it has a low convergence rate because a new generation of neural networks is created using a random process (crossover and mutation) that does not allow convergence to the best solution as efficiently, as the gradient methods. In addition, there is still no guarantee of finding a global minimum.

One of the common stabilization approaches for both training techniques is Tikhonov regularization (Bishop, 1995; Girosi et al., 1995). Tikhonov regularization requires an additional member in the objective function, which usually is the sum of squares of all coefficients in the predictive operator and which thus impedes coefficients from growing very large (Tikhonov and Arsenin, 1977; Ivanov et al., 2013).

The combination of the genetic algorithm and gradient methods (Mishra and Debroy, 2006) is considered to be a promising approach to building a stable neural network and will continue being developed in the future.

CONTROLLED GRADIENT METHOD

We propose to use the controlled gradient method developed by Kobrunov (1978, 1983, 1988, 1994) during the genetic algorithm's training iterations. The controlled gradient method (see Appendix A) can be described as follows:

$$\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^{n} + \alpha^{n}[\Im^{*}\Im]^{-1}\left[\frac{\partial \boldsymbol{F}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})}{\partial \xi}\right]^{*}[\boldsymbol{y}^{l} - \boldsymbol{F}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})], \quad (1)$$

$$\alpha^{n} = -\frac{\left\langle \boldsymbol{\phi}^{n} \middle| \frac{\partial \boldsymbol{F}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})}{\partial \xi}[\Im^{*}\Im]^{-1}\left[\frac{\partial \boldsymbol{F}(\boldsymbol{x}^{l1},\boldsymbol{\xi}^{n})}{\partial \xi}\right]^{*}[\boldsymbol{y}^{l1} - \boldsymbol{F}(\boldsymbol{x}^{l1},\boldsymbol{\xi}^{n})]\right\rangle}{\left\| \frac{\partial \boldsymbol{F}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})}{\partial \xi}[\Im^{*}\Im]^{-1}\left[\frac{\partial \boldsymbol{F}(\boldsymbol{x}^{l1},\boldsymbol{\xi}^{n})}{\partial \xi}\right]^{*}[\boldsymbol{y}^{l1} - \boldsymbol{F}(\boldsymbol{x}^{l1},\boldsymbol{\xi}^{n})]\right\|}, \quad (2)$$

where $\hat{\boldsymbol{y}} = \boldsymbol{F}(\boldsymbol{x},\boldsymbol{\xi}) : R^{N} \to R^{M}$ is a common type of the neural network operator, for example, a classic multilayer neural network with one hidden layer: $\hat{\boldsymbol{y}} = \{y_m = S(\sum_{i_h}^{N_h} b_{mi_h} \cdot S(\sum_{j=1}^{N} a_{i_h j} \cdot x_j, p_{i_h}), q_m), m = 1 \div M\}$.

The vector $\boldsymbol{\xi} = (\boldsymbol{b},\boldsymbol{p},\boldsymbol{a},\boldsymbol{q}) \in R^{K}$ holds the unknown coefficients (weights) of the neural network operator and is of length: $K = N \cdot N_h + N_h \cdot M + N_h + M$, where

- N is the number of input (independent) variables, i.e., the length of the input vector $\boldsymbol{x}$ that is used for training or prediction.
- M is the number of output (dependent) variables in output vector $\hat{\boldsymbol{y}}$, which forms the prediction of $\boldsymbol{y}$. Commonly, there will be only one prediction variable ($M = 1$), but sometimes it is advantageous to simultaneously predict several variables, such as, for example, the initial oil and gas rates of producer wells in unconventional exploration.
- N_h is the number of nodes in the hidden layer, and $S(x, p)$ is an activation function. Commonly, a sigmoid function or the identity function is used.
- $(\boldsymbol{x}^{l} = \{x_j^{l}, j = 1 \div N\}, \boldsymbol{y}^{l} = \{y_m^{l}, m = 1 \div M\}, l = 1 \div L)$ is the training data set, and $L \times M > K$ is a number of training pairs $(\boldsymbol{x}^{l}, \boldsymbol{y}^{l})$.
- α^{n} is the relaxation coefficient that is designed for maximum iteration speed (Denysiuk and Kobrunov, 1983), $\|.\|$ denotes L2 norm, $\langle .|. \rangle$ denotes the inner product, and $[.]^{*}$ denotes the conjugation or transpose for noncomplex matrix.
- $\boldsymbol{\phi}^{n} = \boldsymbol{y}^{l} - F(\boldsymbol{x}^{l},\boldsymbol{\xi})$ is a vector $\{\varphi_m^{n}, m = 1 \div (L \times M)\}$ of prediction errors at the current iteration, $l = 1 \div (L \times M)$.
- $[\partial F(\boldsymbol{x}^{l},\boldsymbol{\xi})/\partial \xi]$ is $K \times (L \times M)$ matrix and $[\partial F(\boldsymbol{x}^{l},\boldsymbol{\xi})/\partial \xi]^{*}$ *is* the transpose $(L \times M) \times K$ matrix that transfers error vector $\boldsymbol{y}^{l} - F(\boldsymbol{x}^{l},\boldsymbol{\xi})$ with $(L \times M)$ dimensions to vector $\boldsymbol{\xi}$ with K dimensions.

The matrix $[\Im^{*}\Im]^{-1}$ represents a priori information that modifies the movement direction given by the gradient, for example, according to a common trend of the objective function (Himmelblau, 1972). If it is a unit matrix, the movement has the direction to a local minimum.

Iterating according to equations 1 and 2 guarantees finding a value close to the initial value $\boldsymbol{\xi}^{0}$ and, in the direction influenced by a priori information defined by the matrix $[\Im^{*}\Im]^{-1}$, that is a local minimum of the objective function:

$$\|\boldsymbol{y}^{l} - F(\boldsymbol{x}^{l},\boldsymbol{\xi})\|^{2}_{R^{L\times M}} \to \min; \quad \xi \in R^{K}. \quad (3)$$

Euler's equation according to the objective function 3 is $[\partial F(\boldsymbol{x}^{l},\boldsymbol{\xi})/\partial \xi]^{*}[\boldsymbol{y}^{l} - F(\boldsymbol{x}^{l},\boldsymbol{\xi})] = 0$, and it can be used as a base for iteration processes 1 and 2.

Let us assume that in a neighborhood of $\boldsymbol{\xi}^{n}$, a pair of vectors $\boldsymbol{\xi}_1^{n} - \boldsymbol{\xi}_2^{n}$ is selected. For example, this pair can be the current and parent vectors during an iteration according to a classic genetic algorithm. The system matrix for this pair can be calculated as follows (Himmelblau, 1972):

$\boldsymbol{\Psi}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n}) = F(\boldsymbol{x}^{l},\boldsymbol{\xi}_2^{n}) - F(\boldsymbol{x}^{l},\boldsymbol{\xi}_1^{n})/\boldsymbol{\xi}_2^{n} - \boldsymbol{\xi}_1^{n}$ and the iterative processes 1 and 2 can be modified to

$$\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^{n} + \alpha^{n}[\Im^{*}\Im]^{-1}[\boldsymbol{\Psi}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})]^{*}[\boldsymbol{y}^{l} - F(\boldsymbol{x}^{l},\boldsymbol{\xi}_2^{n})] \quad (4)$$

$$\alpha^{n} = -\frac{\langle \boldsymbol{\phi}^{n} | \boldsymbol{\Psi}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})[\Im^{*}\Im]^{-1}[\boldsymbol{\Psi}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})]^{*}[\boldsymbol{y}^{l1} - \boldsymbol{F}(\boldsymbol{x}^{l1},\boldsymbol{\xi}^{n})]\rangle}{\|\boldsymbol{\Psi}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})[\Im^{*}\Im]^{-1}[\boldsymbol{\Psi}(\boldsymbol{x}^{l},\boldsymbol{\xi}^{n})]^{*}[\boldsymbol{y}^{l1} - \boldsymbol{F}(\boldsymbol{x}^{l1},\boldsymbol{\xi}^{n})]\|}. \quad (5)$$

If we were to use only equations 4 and 5, the result would be highly dependent on the initial value $\boldsymbol{\xi}^{0}$, as with any other gradient method, and usually this would allow finding only a close-by local minimum. Even though when using a priori information according to $[\Im^{*}\Im]^{-1}$ the probability of finding a global minimum of the objective function increases, it is still not guaranteed.

COMBINED METHOD

Our suggestion to combine a genetic algorithm with the controlled gradient method for training a nonlinear neural network operator follows the below-described procedure (Figure 1):

1) Create an initial generation of neural networks; i.e., create several realizations of the network coefficients ξ^0 by drawing them from a random function.
2) Select a small number of the best members of a generation, i.e., variants of neural networks with the smallest objective function. (This is the selection stage of the genetic algorithm.)
3) Create the next generation from the selected best objects according to the following rules:
   - Controlled gradient: Adjust the coefficients for current objects ξ^n according to their parent objects ξ^{n-1} using equations 4 and 5. It is important to take this step before the crossover and mutation operations because it can significantly move ξ^n from ξ^{n-1} and as a consequence result in a bad approximation of the derivative. This step cannot be applied for a first iteration because the parent object does not exist.
   - Crossover: Exchange coefficients between randomly chosen selected members.
   - Mutation: Add small random perturbations to the resulting coefficients of the previous step. This operation is usually applied only to the defined part of randomly selected members.

Repeat steps 2 and 3 until the objective function becomes small or until a few maximum iterations are achieved.

Using equations 4 and 5 for calculation of the objects for the next generation is rather easy because it relies only on the current and parent objects. This allows including the calculations directly in the genetic algorithm. We use calculations 4 and 5 only for selected objects with the smallest errors because the calculation cost is higher compared to the simple calculation for crossover or mutation.

Due to the stochastic nature of the predicted results, it is reasonable to calculate several realizations of the prediction and then calculate an average value, standard deviation, and P10, P50, P90 values in any predicted position. This technique allows estimating the prediction quality via a standard deviation and provides for a more stable result.

Because a neural network operator can be trained by using the proposed scheme when combining the genetic algorithm with the controlled gradient method (equations 4 and 5), this has the following main advantages:

1) Using stochastically changed initial values for the controlled gradient method can increase the probability of finding the global minimum.
2) The controlled gradient procedure is fast because the optimal relaxation coefficient (5) is used.
3) There are no restrictions to the internal structure and complexity of the neural network because the calculation of the gradient $F(x^l, \xi_2^n) - F(x^l, \xi_1^n)/\xi_2^n - \xi_1^n$ and other components of equations 4 and 5 can be performed numerically.
4) The matrix $[\Im^* \Im]^{-1}$ allows using common a priori information in the most effective way to find a global solution with a higher probability. Our general recommendation is to apply it according to the global trend of the objective function. For example, a smoothed version of the objective function can be used as a trend. As an alternative, we propose using several directions of movement for iterations simultaneously, which can increase the probability of finding a global minimum.

The proposed procedure avoids many disadvantages of a classic genetic algorithm, especially, the low speed of the decreasing error during iteration. It also allows avoiding many disadvantages of the gradient methods, such as a high level of dependency on the initial values and a high level of error during final iterations.

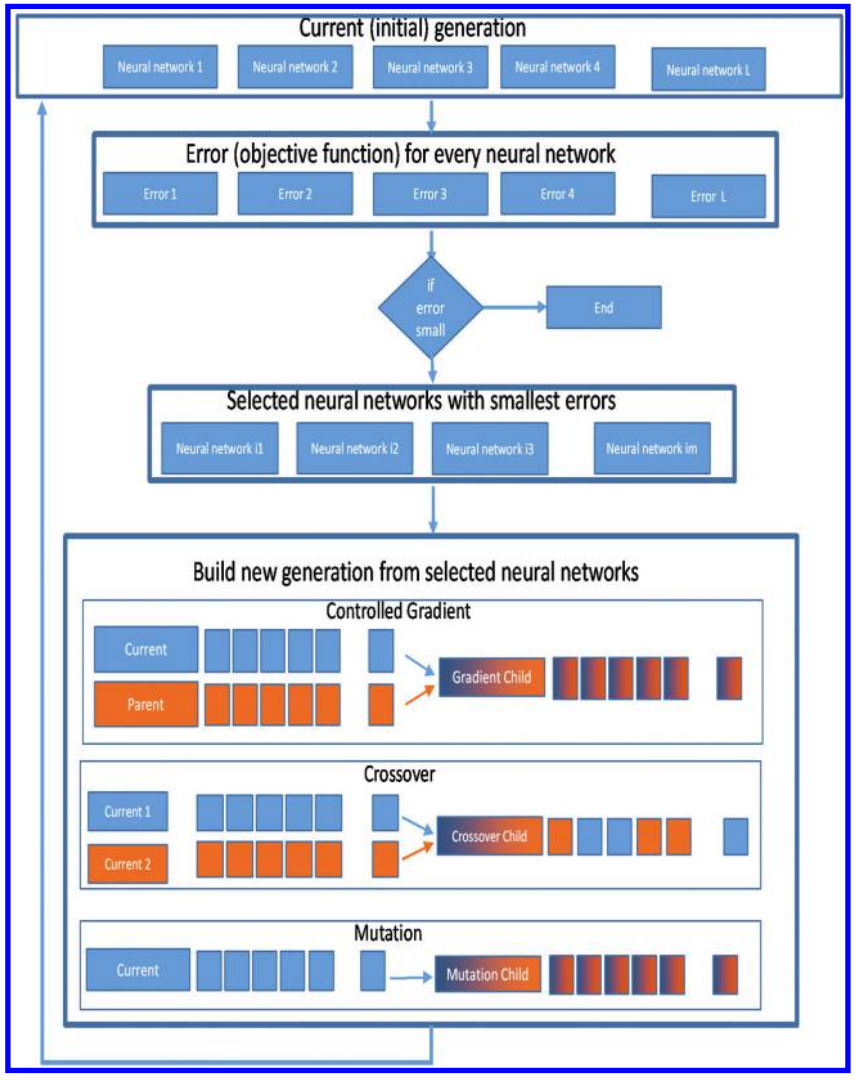

Figure 1. Neural network training scheme based on the combination of the genetic algorithm and the controlled gradient method.

EXAMPLE

For the purposes of testing the proposed technology, we used a data set from the Avalon Shale, an unconventional shale play in the Delaware Basin in New Mexico. The basis for the independent variables was a seismic cube with original amplitudes in the depth domain. In addition, we used production data that were downloaded from an IHS database. The variable to predict was the one-year initial barrel of oil equivalent normalized by the actual number of production months in the year and by the length of the horizontal part of the wells. In total, we used 84 horizontal wells that have the Avalon shale formation as a production target and that intersect with the seismic cube. We distributed production points equally in the horizontal part of the wells at the horizontal distance of 304.8 m (1000 ft.), and, accordingly, there were 658 production points used for training.

This approximation of the production profile was necessary because we had no information about the distribution of production along the horizontal part of the wells. The approximation allows restoring the production profile based on the seismic attributes during the training stage. To predict the production rate, we used the original seismic amplitudes in a vertical window around the production points with the vertical size of 107.3 m (352.1 ft. or 21 samples). In addition, we used all the neighboring traces around the central trace that created a moving window with nine seismic traces. Thus, in total, we used $21 \times 9 = 189$ input parameters in our operator. Therefore, in total, we used 189 seismic samples as the input to neural network from the original seismic cube around every production point during the training stage and then around the prediction position during the calculation stage. Only the volume of stacked seismic amplitudes was used for the purposes of the predictive analysis. The reasons for that are as follows:

1) We used a moving window input to the neural network, which means that we relied on several values from one volume distributed around the center sample of the moving window. For example, in our case, the moving window consists of nine traces and 21 samples. Therefore, we actually used 189 samples (188 samples around the center sample).
2) In this case, we believe that the neural network can use this distribution more effectively for prediction purpose compared to using several seismic attributes such as input because seismic attributes also use a moving window during the calculation, whereas the neural network has a high level of power of freedom to directly obtain the needed information from such distribution.
3) Also, we believe that using several input cubes, such as an angle stack or results of its inversion, can improve the prediction. However, elastic cubes in the data set that could be used for the prediction were not available to us in this case.
4) Moreover, this technique greatly increases the input size and crosscorrelations for input parameters, but it has the advantage of predictive accuracy (Priezzhev et al., 2009; Veeken et al., 2009; Priezzhev et al., 2014).

Three nodes were used in the hidden layer and, in total there were 574 unknown coefficients in neural network nonlinear operator ($189 * 3 + 1 * 3 + 3 + 1 = 574$, where 189*3 is the connection from 189 inputs to three hidden nodes, 1*3 is the threshold coefficients for hidden nodes, 3 is the connection from hidden nodes to one output, and 1 is the threshold coefficient for one output). The genetic algorithm used the following main parameters: The population size was 100, selection size was 10, and mutation ratio was 5%. In addition, we used the "elitism" option for one the best neural networks allowing it to pass to new generation without changes. In this case, the error will decrease or stay without changes. We used several iterations without changes as a rule for increasing or decreasing mutation ratio. If the number of iterations without changes becomes high, the mutation ratio should be decreased to ensure the convergence of the iteration process, and vice versa, if the number of iterations without changes equals zero, the mutation ratio can be increased, which will speed up the iteration convergence. The prediction was performed in a 3D volume, restricted by two surfaces, i.e., the top and the bottom of the target shale layer.

Figure 2 shows a comparison of the errors during the training iterations for five realizations of the genetic algorithm only, five realizations of the method combining the genetic algorithm and controlled gradient scheme, and five variants of initial values for the controlled gradient method. This clearly demonstrates that the combined method converges faster than the classic genetic algorithm and achieves a smaller predictive error. The controlled gradient method shows the dependency on initial values and is restricted to finding only a local minimum. Figure 3 shows the prediction quality of the production points used for neural network training. Red points and regression on the crossplot show the correlation between measured and predicted values for the production points used for training. The correlation coefficient between measured and predicted production rate is greater than 0.67, showing a good training quality. Blue points and regression on the crossplot show the correlation between measured and predicted values for the production points that were not used during the training and provide for a quality control of the prediction. The correlation coefficient is greater than 0.61, showing good prediction results. The green points and regression on the crossplot show the correlation between the measured and pre-

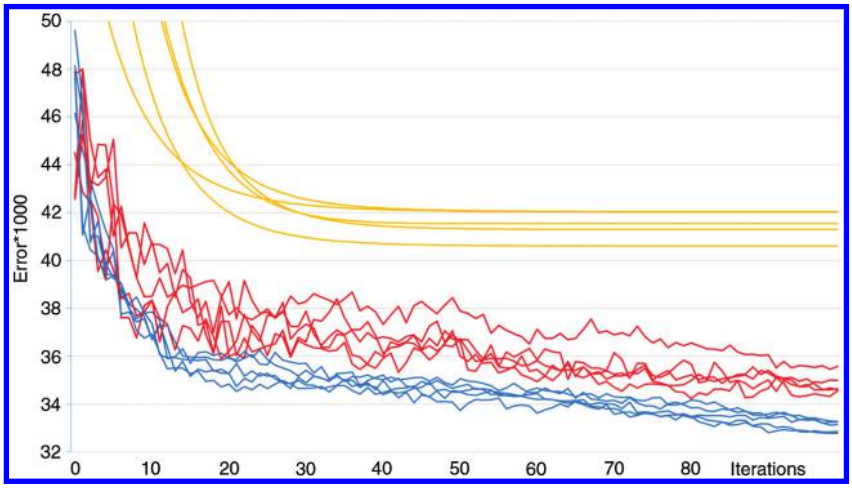

Figure 2. Training rate, i.e., error versus iterations for five variants of the initial values of the controlled gradient (yellow curves), for five realizations of the genetic algorithm (red curves), and for five realizations of the combined genetic algorithm and controlled gradient methods (blue curves).

dicted values of all the available production points obtained as a result of using prediction operator with training based on only the classic genetic algorithm. The correlation coefficient is approximately 0.6 and is lower than the quality control correlation for the neural network created by the combined method.

The relatively small correlation coefficient can be explained by the complex nature of the production rate that is dependent on several independent factors. The first and the main factor impacting the production rate is the reservoir quality or its geologic potential for production, such as an upper limit for possible production rate in a particular point. The second factor is the completion and drilling quality, which can significantly decrease the production rate relative to the geologic potential. It should be noted that our goal is to predict only the geologic potential of the formation, and only this part has a statistical relation to the seismic data because the completion quality and the drilling quality have no relevance to the seismic data. This was the reason for using the neural network with a high level of power of freedom of predictive operator that allows separating these two factors. Additionally, using equal production values over the horizontal part of the wells may introduce wrong training points.

Figure 4 shows a cross section through the input seismic cube and the result cube. In Figure 5, we demonstrate the predicted 3D cube (only for the target shale layer) that is calculated via the predictive operator. The results can be used for new well placement. Figure 6 shows the estimation of P50, P10, P90, and the standard deviation based on the calculation of 10 realizations.

DISCUSSION

In some cases, the number of training pairs $(\boldsymbol{x}^l, \boldsymbol{y}^l)$, $l = 1 \div (L \times M)$ can be extremely high. For example, the number of horizontal wells in the Eagle Ford Formation is more than 12,000 (Priezzhev et al., 2014). This can create a very large matrix $[\partial F(\boldsymbol{x}^l, \boldsymbol{\xi})/\partial \xi]$ and error vector $y_m^l - F(\boldsymbol{x}^l, \boldsymbol{\xi})$ that can significantly prolong the calculation time. Especially, it can be significant in the case of deep neural network with more the one hidden layer that can increase the number of unknown coefficients. In this case, we can recommend reducing the number of training pairs by applying or combining of the following approaches:

1) Divide the training set in two parts. The first part will be used for training, and the second one will be used only for quality control. This division can be randomly changed for each iteration.
2) Divide the area of investigation into several parts containing a smaller number of training pairs. These parts can intersect to create a relatively continued result.
3) Combine some training pairs if they are similar or equal. This technique can be based on well-known automatic classification algorithms such as K-means or SOM.

We also propose to use the matrix $[\mathfrak{J}^* \mathfrak{J}]^{-1}$ for setting the priority of the coefficients $\boldsymbol{\xi}^{\text{pred}}$ predefined in the previous neural network. In this case, the matrix will calculate the direction of the iteration

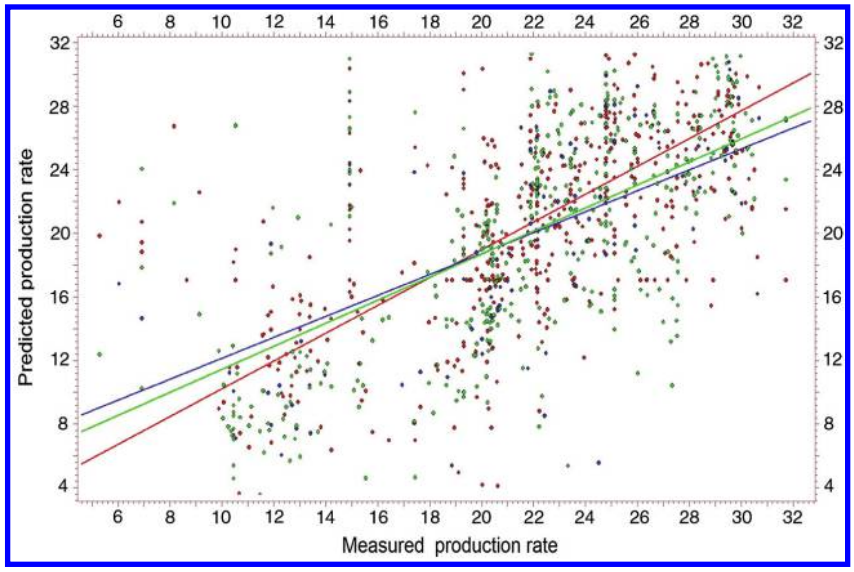

Figure 3. The neural network training quality cross section shows the measured production rate versus the predicted rate. Red circles and regression (correlation coefficient 0.67) show the neural network prediction results based on production points used for training (75% from all production points). Blue circles and regression (correlation coefficient 0.62) are the 25% production points that are not used for neural network training, but used only for quality control. Green circles and regression (correlation coefficient 0.60) show the neural network prediction results obtained by applying only the classic genetic algorithm.

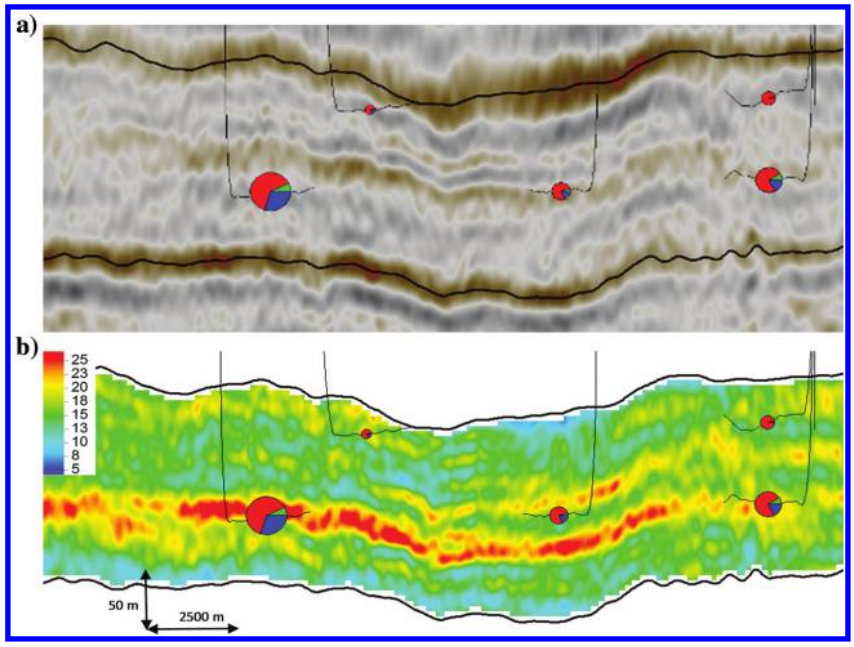

Figure 4. (a) Crossline within the source seismic cube and (b) the result cube. The bubbles show a one-year production of oil (green), gas (red), and water (blue), and their size shows the cumulative value. The borehole projection from the crossline is ±300 m.

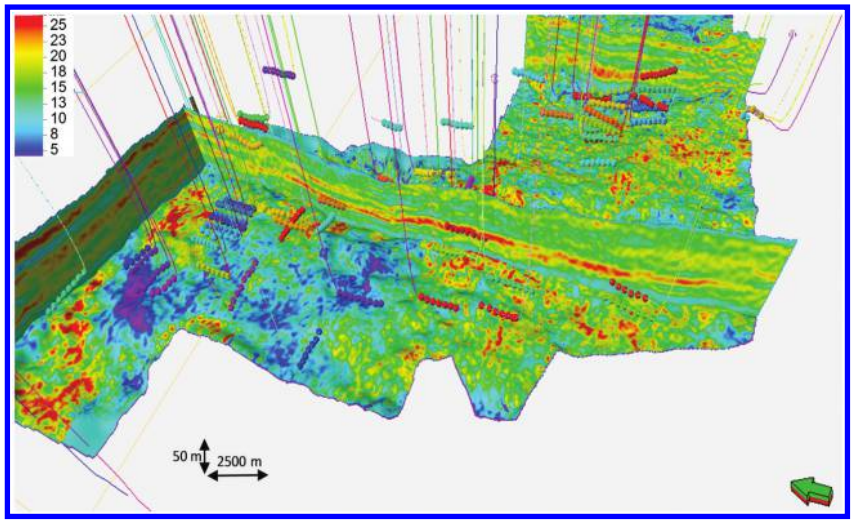

Figure 5. The 3D cube with the prediction results (3D sweet spot) using a nonlinear neural network operator for the shale layer. The resulting cube is shown with the inline, crossline, and stratigraphic slice along the bottom of the target shale layer. Production points are distributed along the horizontal part of the wells with an equal, approximately 300 m, distance.

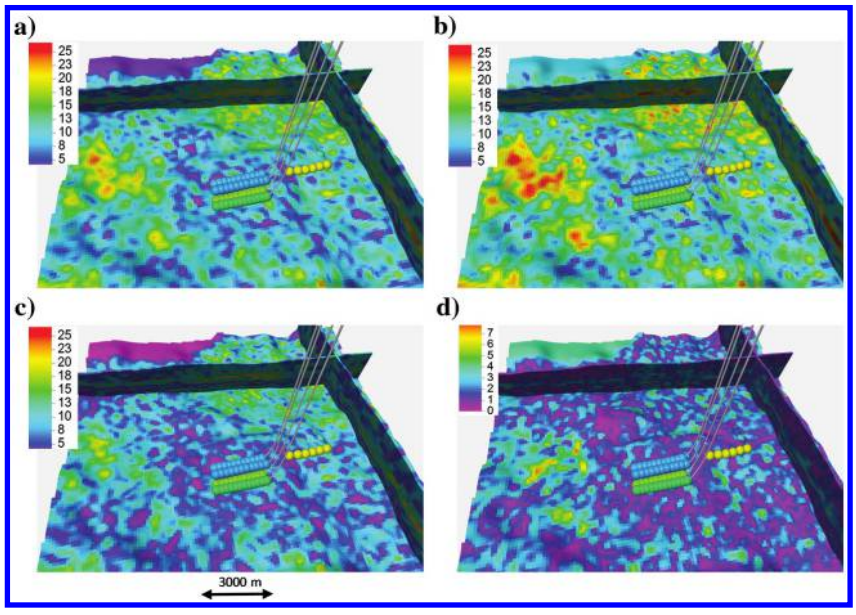

Figure 6. Part of the 3D cube with the results of the calculations of several realizations: (a) P50, (b) P10, (c) P90, and (d) the standard deviation.

from the current values of $\boldsymbol{\xi}^n$ to these coefficients. Here, the matrix will work as the memory for the previously trained neural networks and will allow retraining, if we will get additional train information for it. In the future, we plan to apply this technique in practice for the purposes of training and retraining deep neural networks as an alternative to using $\boldsymbol{\xi}^{\text{pred}}$ as initial values.

CONCLUSIONS

The proposed method for training a nonlinear neural network based on the combination of the genetic algorithm and the controlled gradient method allows creating a predictive operator with smaller prediction error.

The combination of a genetic algorithm and the controlled gradient methods is more robust and obtains a better overall result. The deterministic "controlled-gradient method" works better in combination with the stochastic "genetic algorithm" because it uses stochastically changed initial values and thus achieving a smaller predictive error. The stochastic genetic algorithm works better in combination with the deterministic controlled-gradient method because it uses an optimal gradient element during the new generation creation.

The proposed predictive technology shows the effectiveness in sweet-spot tasks and can also be used for various other purposes and applications.

ACKNOWLEDGMENTS

The authors would like to thank the Ukhta State Technical University and Schlumberger for the opportunity to spend the time necessary to develop this technology and for the permission to publish the results of the work. In addition, the authors would like to thank WesternGeco for the access to the seismic data sets. This work also includes production data supplied by © IHS Energy Log Services Inc., 2014.

APPENDIX A

CONTROLLED GRADIENT METHOD THEORY

The approach described below is titled the controlled-gradient method, developed by Kobrunov (1978, 1983, 1988, 1994). It allows using formal a priori information to avoid nonuniqueness of the solution and is based on the Tikhonov optimization methods theory (Tikhonov and Arsenin, 1977).

A neural network predictive operator with any complexity of the internal structure, for example, a classic neural network with one hidden layer, is usually based on a nonlinear weighted sum according to the following:

$$\boldsymbol{F}(\boldsymbol{x},\boldsymbol{\xi}):R^N\rightarrow R^M;$$

$$\hat{\boldsymbol{y}}=\left\{y_m=S\left(\sum_{i_h}^{N_h}b_{mi_h}\cdot S\left(\sum_{j=1}^{N}a_{i_hj}\cdot x_j,p_{i_h}\right),q_m\right),m=1\div M\right\};$$

$$\boldsymbol{\xi}=(\boldsymbol{b},\boldsymbol{p},\boldsymbol{a},\boldsymbol{q})\in R^K;$$

$$K=N\cdot N_h+N_h\cdot M+N_h+M$$

$$\dim(\boldsymbol{b})=\dim(\boldsymbol{p})=N_h;\dim(\boldsymbol{a})=N;\dim(\boldsymbol{q})=M, \quad \text{(A-1)}$$

where N *is* the number of input (independent) variables in the vector $\boldsymbol{x}$ that is used for prediction; M is the number of output (dependent) variables in the vector y that is necessary to predict; often in our

application, there is only one output variable; however, sometimes it is convenient to predict several variables simultaneously, for example, initial oil and gas rates; N_h is the number of nodes in the hidden layer; K is the number unknown coefficients (weights) of the neural network operator;$S(x, p)$ is an activation function; usually it is a sigmoid function $S(x, p) = 1/1 + \exp -(x - p)$ or the identity function $S(x) = x$.

Therefore, it is necessary to find the following operator:

$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\xi}) = \hat{\boldsymbol{y}}, \tag{A-2}$$

based on the training data set: $(\boldsymbol{x}^l = \{x_j^l, j = 1 \div N\}, \boldsymbol{y}^l = \{y_m^l, m = 1 \div M\}, l = 1 \div L)$ where $L \times M > K$ is the number of training pairs $(\boldsymbol{x}^l, \boldsymbol{y}^l)$, $l = 1 \div (L \times M)$ and with the following objective function:

$$\|\boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi})\|^2_{R^{L\times M}} \to \min; \quad \xi \in R^K. \tag{A-3}$$

The Euler's equation according to equation A-3 is the following:

$$\left[\frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi})}{\partial \xi}\right]^* [\boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi})] = 0, \tag{A-4}$$

where $[\partial F(\boldsymbol{x}^l, \boldsymbol{\xi})/\partial \xi]$ is $K \times (L \times M)$ matrix and $[\partial F(\boldsymbol{x}^l, \boldsymbol{\xi})/\partial \xi]^*$ is the transpose $(L \times M) \times K$ matrix that transfers error vector $y_m^l - F(\boldsymbol{x}^l, \boldsymbol{\xi})$ with $(L \times M)$ dimensions to vector $\boldsymbol{\xi}$ with K dimensions.

According to equation A-4, to find a local extreme close to $\boldsymbol{\xi}^0$, we can use the following iterations:

$$\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^n + \alpha^n \left[\frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* [\boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)]. \tag{A-5}$$

The convergence of this process to the solution (equation A-4) requires regularity of the operator $F(\boldsymbol{x}^l, \boldsymbol{\xi})$ in the vicinity of the desired solution $\boldsymbol{\xi}$, which includes all $\boldsymbol{\xi}^n$.

Let us request the fastest minimization in iterations A-5 of the error vector $\boldsymbol{\phi}^n = \boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi})$ with $(L \times M)$ dimensions:

$$\begin{aligned}
\boldsymbol{\phi}^n &= \boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi}) = \boldsymbol{y}^l - F\left(\boldsymbol{x}^l, \boldsymbol{\xi}^n + \alpha^n \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* \right.\\
&\left. \times [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)]\right)\\
&\approx \boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi}^n) + \alpha^n \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)]\\
&= \boldsymbol{\phi}^n + \alpha^n \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)],
\end{aligned} \tag{A-6}$$

where index $l = 1 \div (L \times M)$ and additional index $l1 = 1 \div (L \times M)$ define order of the operations.

It is important to keep in mind that the error vector $\boldsymbol{\phi}^n$ and vector $\boldsymbol{y}^l$ have the same dimension $L \times M$. Hence,

$$\begin{aligned}
\|\boldsymbol{\phi}^{n+1}\|^2 &= \left\|\boldsymbol{\phi}^n + \alpha^n \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)]\right\|^2\\
&= \|\boldsymbol{\phi}^n\|^2 + 2\alpha^n \left\langle \boldsymbol{\phi}^n \left| \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* \right.\right.\\
&\left.\left. \times [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)] \right\rangle + (\alpha^n)^2 \left\| \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* \right.\\
&\left. \times [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)] \right\|^2 = \|\boldsymbol{\phi}^n\|^2 * q(\alpha^n),
\end{aligned} \tag{A-7}$$

where

$$\begin{aligned}
q(\alpha) &= 1 + 2\alpha^n \left\langle \boldsymbol{\phi}^n \left| \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* \right.\right.\\
&\left.\left. \times [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)] \right\rangle / \|\boldsymbol{\phi}^n\|^2 +\\
&(\alpha^n)^2 \left\| \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)] \right\| / \|\boldsymbol{\phi}^n\|^2.
\end{aligned} \tag{A-8}$$

Function $q(\alpha)$ describes the change of error function with coefficient relaxation α. A request to maximize the speed of error minimization means minimization of function $q(\alpha)$ by α:

$$\alpha^n = -\frac{\left\langle \boldsymbol{\phi}^n \left| \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)] \right\rangle}{\left\| \frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)}{\partial \xi} \left[\frac{\partial F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)}{\partial \xi}\right]^* [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)] \right\|^2}. \tag{A-9}$$

Iterations A-5 will find the vector $\boldsymbol{\xi}$ close to $\boldsymbol{\xi}^0$ as the local minimum of the objective function A-3 in a fastest way by using the adaptive relaxation coefficient in equation A-9. In cases in which the objective function A-3 has more than one local minimum and the task is to find the global minimum, we need to repeat the iteration A-5 for many different initial values $\boldsymbol{\xi}^0$.

Using the matrix form for the training data sets defined as $\boldsymbol{X} = \{x^l\} = \{x_j^l, j = 1 \div N, l = 1 \div L\} \varepsilon R^{N \cdot L}$ and $\boldsymbol{Y} = \{y^l\} = \{y_m^l, m = 1 \div M, l = 1 \div L\} \varepsilon R^{M \cdot L}$, operator 2 can be rewritten in vector form:

$$\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{\xi}) = \boldsymbol{Y}. \tag{A-10}$$

The solution does not exist, in a strict mathematical sense, for any pairs $(\boldsymbol{X}, \boldsymbol{Y}) \in R^{N \cdot L} \times R^{M \cdot L}$. To solve the problem, let us consider the variety $\mathfrak{M} = \mathrm{Im}\, \boldsymbol{F}(\boldsymbol{X}, \boldsymbol{\xi}) \subseteq \boldsymbol{Y}$ for operator A-10 with the defined matrix $\boldsymbol{X}$ and use ξ in all R^K and Im notate image of function in its domain. Let us describe the quasi-solution for operator A-10 as follows:

$$\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{\xi}) = P(\mathfrak{M}, \boldsymbol{Y}), \tag{A-11}$$

where $P(\mathfrak{M}, \boldsymbol{Y})$ is projection of vector $\boldsymbol{Y}$ to consider the variety $\mathfrak{M}$.

It is natural to suggest that the quasi-solution of equation A-11 is not unique in the sense that the same pair $\{\boldsymbol{X}, P(\mathfrak{M}, \boldsymbol{Y})\}$

corresponds to the variety $\mathfrak{A}$ vectors $\boldsymbol{\xi}$, which are $\forall\, \xi \in \mathfrak{A} : \boldsymbol{F}(\boldsymbol{X}, \xi) = P(\mathfrak{M}, \boldsymbol{Y})$. The structure of the variety is not defined and can be presented as a nonlinear variety in R^K. However, if we assume that the area of nonuniqueness is very complex and nontrivial, then we must apply an additional rule to find the solution from this area $\mathfrak{U}$ ($\mathfrak{U}$ contains more than one solution). Let us define this rule as a request to find the minimum of the functional:

$$\|\mathfrak{J}[\boldsymbol{\xi} - \boldsymbol{\xi}^0]\|_{R^K} \to \min; \quad \boldsymbol{\xi} \in \mathfrak{U}, \tag{A-12}$$

where ξ^0 is the initial values of the operator coefficients and $\mathfrak{J}$ is the multiplicities component of the criteria A-12 that defines the direction of moving from ξ^0 to find the solution from $\mathfrak{U}$. Iterations A-5 define the direction of moving exactly according to the gradient direction. Combine equations A-11 and A-12:

$$\begin{aligned} &\boldsymbol{F}(\boldsymbol{X}, \varsigma) = P(\mathfrak{M}, \boldsymbol{Y}); \\ &\|\mathfrak{J}[\boldsymbol{\xi} - \boldsymbol{\xi}^0]\|_{R^K} \to \min. \end{aligned} \tag{A-13}$$

The necessary condition for equation A-13 is the following:

$$\boldsymbol{\xi} = \boldsymbol{\xi}^0 + [\mathfrak{J}^*\mathfrak{J}]^{-1}\left[\frac{\partial \boldsymbol{F}(\boldsymbol{X}, \varsigma)}{\partial \varsigma}\right]^* \lambda, \tag{A-14}$$

where λ is some element from conjugate space $P(\mathfrak{M}, \boldsymbol{Y})$.

An equivalent to equation A-14 can be the following:

$$\begin{aligned} &\boldsymbol{\xi} = \boldsymbol{\xi};^0 + [\mathfrak{J}^*\mathfrak{J}]^{-1}\left[\frac{\partial F(\boldsymbol{x}^l, \boldsymbol{\xi})}{\partial \xi}\right]^* [\lambda^l]; \\ &\lambda = \{\lambda^l, l = 1 \div L\}. \end{aligned} \tag{A-15}$$

Finally, equation A-15 can be written as the following iterative procedure that generalizes equation A-5 and allows finding the solution for equation A-13:

$$\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^n + \alpha^n [\mathfrak{J}^*\mathfrak{J}]^{-1}\left[\frac{\partial F(\boldsymbol{x}^l, \xi^n)}{\partial \xi}\right]^* [\boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)], \tag{A-16}$$

$$\alpha^n = -\frac{\left\langle \boldsymbol{\phi}^n \middle| \frac{\partial F(\boldsymbol{x}^l, \xi^n)}{\partial \xi}[\mathfrak{J}^*\mathfrak{J}]^{-1}\left[\frac{\partial F(\boldsymbol{x}^{l1}, \xi^n)}{\partial \xi}\right]^* [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)] \right\rangle}{\left\| \frac{\partial F(\boldsymbol{x}^l, \xi^n)}{\partial \xi}[\mathfrak{J}^*\mathfrak{J}]^{-1}\left[\frac{\partial F(\boldsymbol{x}^{l1}, \xi^n)}{\partial \xi}\right]^* [\boldsymbol{y}^{l1} - F(\boldsymbol{x}^{l1}, \boldsymbol{\xi}^n)] \right\|}, \tag{A-17}$$

where $\boldsymbol{\phi}^n = [\boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)]$ error vector with $(L \times M)$ dimensions.

In the multiple nonuniqueness condition for the task,

$$\begin{aligned} &\|\boldsymbol{F}(\boldsymbol{X}, \varsigma) - \boldsymbol{Y}\|_{R^{M \cdot L}} \to \min, \\ &\varsigma \in R^k. \end{aligned} \tag{A-18}$$

Equations A-16 and A-17 allow finding the solution according to the initial values ξ^0 and matrix $[\mathfrak{J}^*\mathfrak{J}]^{-1}$ that defines the direction of movement of the iterations and can be calculated based on common a priori assumption.

The problem (equation A-11) is unstable due to the problem of finding the vector $\lambda = \{\lambda^l, l = 1 \div L\}$. However, the next operation on this vector, given by equation A-15, smooths the oscillations in the vector λ associated with errors in the input data $(\boldsymbol{X}, \boldsymbol{Y}) \in (R^{N \cdot L} \times R^{M \cdot L})$. We can assume that the errors are concentrated in the vector $\boldsymbol{Y}$, and in fact, this vector is the sum $\boldsymbol{Y}^\delta = \boldsymbol{Y} + \delta(\boldsymbol{Y})$; $\|\delta(\boldsymbol{Y})\| \le \delta$ where δ can be defined. The gradients of the iterative processes applied to the solution of these kinds of equations have regularizing properties. The number of step iterations plays the role of the regularization parameter (equation A-16) based on the general principles of the regularization theory (Ivanov et al., 2013) that the choice of the regularization parameter on the basis of the residual provides for the regularizing properties of the algorithm. This means that the principle of stopping the iterative process (equation A-16) would be the achievement of discrepancy $\boldsymbol{\phi}^n$ level, $\|\boldsymbol{\phi}^n\| = [\boldsymbol{y}^l - F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)]$. Alternatively, other rules can be used to choose an optimal regularization parameter in the absence of information about the level of residual δ. Thus, the stop role can be, for example, the principle of quasi-optimality based on the control of $\Delta^{n+1} = \|\boldsymbol{\xi}^{n+1} - \boldsymbol{\xi}^n\|$ and stopping the iterative process (A-13) on the principle of achieving a minimum value of Δ^{n+1}/n.

Under the current equivalence, and probably a wider approximate called practical or quasi-equivalence, many local minima exist in the problem

$$\begin{aligned} &\|\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{\xi}) - \boldsymbol{Y}\|_{R^{M \cdot L}} \to \min, \\ &\boldsymbol{\xi} \in R^k. \end{aligned} \tag{A-19}$$

The iterative process (equation A-13) converges to a local minimum of iterative process (equation A-16), the properties of which are controlled by two parameters: the choice of the zero approximation $\boldsymbol{\xi}^0$, and the matrix $[\mathfrak{J}^*\mathfrak{J}]^{-1}$ determining the direction of the iteration process to a local minimum. If this is a unit operator, the direction of movement is determined by the gradient of the residual functional (equation A-19). However, wide opportunities arise if to vary this matrix while relying on the assumptions of the correlation proximity $\boldsymbol{\xi}^0$ and a local minimum in equation A-19. Nevertheless, Himmelblau (1972) gives a sufficiently high number of heuristics algorithms in which a gradient (equation A-19) is replaced by its approximate, sufficiently generalized, approximation calculated so as to exclude in the area of local minima and to move toward a trend.

Let us suppose that in a neighborhood of $\boldsymbol{\xi}^n$, a pair of vectors $\boldsymbol{\xi}_1^n, \boldsymbol{\xi}_2^n$ is selected. This pair can be, for example, a pair of iterative two-step processes to minimize gradient (equation A-19) during the genetic algorithm. The system matrix matrices can be calculated for this pair as follows: $\boldsymbol{\Psi}(\boldsymbol{x}^l, \boldsymbol{\xi}^n) = F(\boldsymbol{x}^l, \boldsymbol{\xi}_2^n) - F(\boldsymbol{x}^l, \boldsymbol{\xi}_1^n)/\xi_2^n - \xi_1^n$ and the iterative process (equation A-16) is modified as

$$\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^n + \alpha^n [\mathfrak{J}^*\mathfrak{J}]^{-1} [\boldsymbol{\Psi}(\boldsymbol{x}^l, \boldsymbol{\xi}^n)]^* [y^l - F(\boldsymbol{x}^l, \boldsymbol{\xi}^n)]. \tag{A-20}$$

REFERENCES

Ali, J. K., 1994, Neural networks: A new tool for petroleum industry?: European Petroleum Conference, SPE-27561-MS.

Bishop, C. M., 1991, Improving the generalization properties of radial basis function neural networks: Neural Computation, **3**, 579–588, doi: 10.1162/neco.1991.3.4.579.

Bishop, C. M., 1995, Training with noise is equivalent to Tikhonov regularization: Neural Computation, **7**, 108–116, doi: 10.1162/neco.1995.7.1.108.
Bishop, C. M., and M. J. Bushnell, 1993, Genetic optimization of neural network architectures for color recipe prediction: Proceedings of the International Joint Conference on Neural Networks and Genetic Algorithms, Innsbruck, Austria, 719–725.
Boadu, F. K., 1998, Inversion of fracture density from field seismic velocities using artificial neural networks: Geophysics, **63**, 534–545, doi: 10.1190/1.1444354.
Chen, Q., and S. Sidney, 1997, Seismic attribute technology for reservoir forecasting and monitoring: The Leading Edge, **16**, 445–448, doi: 10.1190/1.1437657.
Denysiuk, R. P., and A. I. Kobrunov, 1983, On the choice of the relaxation parameter in the solution of the inverse problem of gravity: Geophysical Journal, **5**, 63–68.
Girosi, F., M. Jones, and T. Poggio, 1995, Regularization theory and neural networks architectures: Neural Computation, **7**, 219–269, doi: 10.1162/neco.1995.7.2.219.
Hampson, B., J. Schuelke, and J. Quirein, 2001, Use of multi-attribute transforms to predict log properties from seismic data: Geophysics, **66**, 220–236, doi: 10.1190/1.1444899.
Himmelblau, D. M., 1972, Applied Nonlinear Programming: McGraw-Hill.
Ivanov, V. K., V. V. Vasin, and V. P. Tanana, 2013, Theory of linear ill-posed problems and its applications: De Gruyter, Inverse and ill-posed problems series.
Kobrunov, A. I., 1978, Optimization method to inverse gravity data: Earth Physics USSR Academy of Science, **8**, 73–78.
Kobrunov, A.I., 1983, Extreme classes in problems of gravity and their use for building density models of geological environments: D.Sc. dissertation, The Institute of Geophysics of the National Academy of Sciences of Ukraine.
Kobrunov, A. I., 1988, Theory of gravity data interpretation for layers model (uniform optimization): Earth Physics USSR Academy of Science, **8**, 33–34.
Kobrunov, A. I., 1994, The theory of interpretation of gravity data for media of intricate structure: Academic Express, Geophysical Express, **1**.
Liu, Z., and J. Liu, 1998, Seismic-controlled nonlinear extrapolation of well parameters using neural networks: Geophysics, **63**, 2035–2041, doi: 10.1190/1.1444496.
Mishra, S., and T. Debroy, 2006, A genetic algorithm and gradient-descent based neural network with the predictive power of a heat and fluid flow model for welding: Welding Journal, **85**, 231–242.
Mogensen, S., and C. Link, 2001, Artificial neural networks solutions to AVO inversion problems: 71st Annual International Meeting, SEG, Expanded Abstracts, 316–319.
Priezzhev, I., A. Scollard, and Z. Lu, 2014, Regional production prediction technology based on gravity and magnetic data from the Eagle Ford Formation: SEG.
Priezzhev, I., L. Shmaryan, and P. Veeken, 2009, Genetic seismic inversion using a non-linear, multi-trace reservoir modeling approach: 71st Annual International Conference and Exhibition, EAGE, Extended Abstracts, P018.
Roth, G., and A. Tarantola, 1994, Neural networks and inversion of seismic data: Journal of Geophysical Research, **99**, 6753–6768, doi: 10.1029/93JB01563.
Russell, B., D. Hampson, J. Schuelke, and J. Quirein, 1997, Multi-attribute seismic analysis: The Leading Edge, **16**, 1439–1444, doi: 10.1190/1.1437486.
Schultz, P. S., S. Ronen, M. Hattori, and C. Corbett, 1994a, Seismic-guided estimation of log properties: Part 1(a): A data-driven interpretation methodology: The Leading Edge, **13**, 305–310.
Schultz, P. S., S. Ronen, M. Hattori, and C. Corbett, 1994b, Part 2(b): Using artificial neural networks for nonlinear attribute calibration: The Leading Edge, **13**, 674–678, doi: 10.1190/1.1437020.
Tikhonov, A. N., and V. Y. Arsenin, 1977, Solutions of ill-posed problems: V. H. Winston and Sons.
Veeken, P. C. H., I. I. Priezzhev, L. E. Shmaryan, Y. I. Shteyn, A. Y. Barkov, and Y. P. Ampilov, 2009, Non-linear multi-trace genetic inversion applied on seismic data across the Shtokman field (offshore northern Russia): Geophysics, **74**, no. 6, WCD49–WCD59, doi: 10.1190/1.3223314.
Whitley, D., T. Starkweather, and C. Bogart, 1990, Genetic algorithms and neural networks: Optimizing connections and connectivity: Parallel Computing, **14**, 347–361, doi: 10.1016/0167-8191(90)90086-O.