# Oil Reservoir Porosity and Permeability Predictions with Feed-forward Neural Network and Evolutionary Algorithm

Yiwen Qin

Research School of Computer Science
Australian National University
U6688408@anu.edu.au

**Abstract:** In this paper, different types of neural networks were built for oil reservoir porosity and permeability prediction. The feed-forward networks were trained using traditional back-propagation method and evolutionary algorithms (EA). Network Reduction was also implemented. The EA methods included Evolving Neural Networks through Augmenting Topologies (NEAT). The networks were trained multiple times with each method for each problem. The results of different methods were evaluated and comparisons were made. The efficiency of the involved methods was discussed. The trained neural networks can predict the targets with acceptable loss. NEAT can generate more accurate networks, but the training is slower and the result of training is more unstable.

**Keywords:** Machine Learning ·Neural Network ·Evolutionary Algorithm ·NEAT ·Oil Reservoir

## 1. Introduction

### 1.1 Machine Learning

Machine learning are widely used to solve a variety of problems. Feed-forward neural networks of a few layers trained by back-propagation is one the most popular traditional approach [1]. However, the hidden units of the networks are usually more than what reality requires, resulting in a final network with unnecessary units for post-learning. Network Reduction is a solution to this issue. Another problem is that the traditional feed-forward neural network cannot deal with higher level problems with more complex relationship. The evolutionary algorithms (EAs) have been widely implemented in science and engineering for solving more complex problems [2]. The EA method used in this paper is NEAT, which fixes the topology of the network and learns significantly faster than the other EAs [3].

### 1.2 Oil Reservoir Background

Porosity and permeability of oil reservoir is essential for oil well engineering. The objective of this paper is to predict the Porosity and permeability of the oil wells based on the other features from the wells.

### 1.3 Dataset

The dataset used in this report is a benchmark data file for AI Research in Reservoir Characterisation at UNSW. The data set is obtained from an oil reservoir in the North West Shelf, offshore western Australia [4]. Data from four wells were available. According to the original Introduction to the Benchmark Data Set, the engaged data of the wells were separated into 3 pairs of datasets with a training set and a testing set.

**Features**

The well logs available are: GR (Gamma Ray), RDEV (Deep Resistivity), RMEV (Shallow Resistivity), RXO (Flushed Zone Resistivity), RHOB (Bulk Density), NPHI (Neutron Porosity), PEF (Photoelectric Factor) and DT (Sonic Travel Time).

**Targets**

The objective is to develop an estimator to predict porosity (PHI) and permeability (logK) from well logs.

**Others**

The data set also contains Lithofacies and FLAG. Lithofacies is defined by expert geologist who has examined the actual rock samples ("cores") and put all the samples into classes [5]. FLAG shows that goodness of the core samples. It is characterised by "Good," "OK," or "Frac." These values were not considered in this report.

## 2. Methods

### 2.1 Data Preprocessing

The features and the targets of the oil well dataset were normalised between 0 and 1.

### 2.2 Traditional Feed-forward Neural Networks

#### 2.2.1 Network Structure

The neural networks were initialized as pytorch 3-layer neuron networks with 8 input neurons, 8 hidden neurons and 1 output neuron. Sigmoid activation function and SGD optimizer was used in the network. 6 networks for 2 targets from 3 pairs of datasets have been trained. For each pair of datasets, 2 networks have been trained with the features and targets mentioned in **1.3**.

#### 2.2.2 Loss Function

The mean squared error (MSE) loss function was chosen for neural network evaluation, because both PHI and logK (targets) were numeric.

### 2.2.3 Training Process

The training was started with a high learning rate (0.01) and was adjusted observing the loss-epoch curve during training (0.0015 as the final learning rate). (Although the lowest loss that the training with learning rate = 0.01 reached was approximately the same as the one with rate = 0.0015, the curve of the latter one is more similar to the tendency of a good learning rate.) Each network was trained by 500 epochs using back-propagation of error measures.

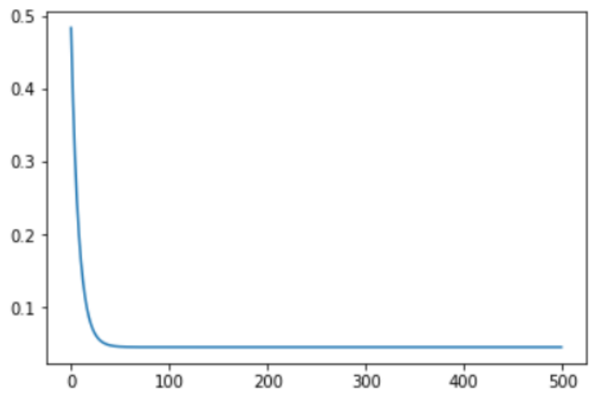

**Fig.1.** Loss-epoch curve (Dataset 1, PHI, learning rate = 0.01). The curve shows the change of the MSE loss (y-axis) with respect to the epochs (x-axis)

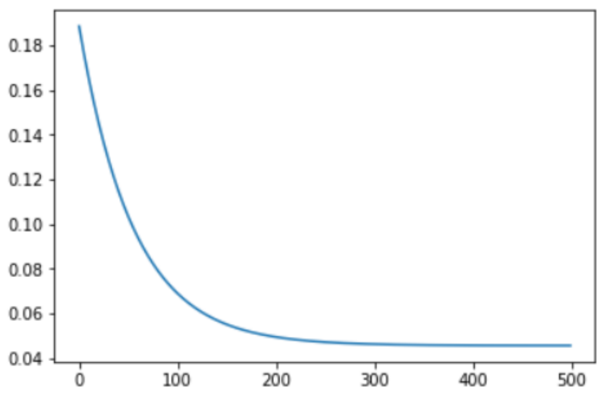

**Fig.2.** Loss-epoch curve (Dataset 1, PHI, learning rate = 0.0015)

### 2.2.4 Network Reduction

The hidden units of the networks are usually more than what reality requires, resulting in a final network with unnecessary units for post-learning. To solve this issue, a developed algorithm in the previous study for detecting and removing undesirable hidden units were implemented to the trained neural networks. This algorithm is based on the distinctiveness between the expression of the hidden units on input patterns [6] and it can generally maintain functionality of networks.

## 2.3 NEAT

### 2.3.1 Overview

The aim of traditional genetic algorithms is to optimize the connection weights that determine the functionality of a network. However, the topology of neural networks also affects their functionality [3]. Fixing the topology of the network is a time-wasting progress with few standards. However, while training with NEAT, the network structure is evolved such that topologies are minimized and grown incrementally, significant gains in learning speed result [3].

NEAT is an example of a topology and weight evolving artificial neural network (TWEANN) which attempts to simultaneously learn weight values and an appropriate topology for a neural network. It is based on applying three key techniques: tracking genes with history markers to allow crossover among topologies, applying speciation (the evolution of species) to preserve innovations, and developing topologies incrementally from simple initial structures.

**Genes Tracking with History Markers**

The genetic encoding scheme of NEAT is designed to allow corresponding genes to be easily lined up when two genomes cross over during mating [3]. Genomes are linear representations of network connectivity. An innovation number is provided by each node gene for corresponding genes finding [3].

When a new gene appears through structural mutation, a global innovation number is incremented and assigned to this gene. Therefore, the innovation numbers represent a chronology of the appearance of every gene in the system and the historical origins are saved [3]. historical markings allow NEAT to perform crossover using linear genomes without expensive topological analysis [3].

**Speciation, Topologies Developing and Competing Conventions**

Speciating is to divide the population into species such that similar topologies are in the same species [3]. This way, topological innovations are protected in a new niche where they have time to optimize their structure through competition within the niche [3].

Competing Conventions means having more than one way to express a solution to a weight optimization problem with a neural network [3]. The main insight in NEAT is that the historical origin of two genes is direct evidence of homology if the genes share the same origin. NEAT performs artificial synapsis based on historical markings, allowing it to add new structure without losing track of which gene is which over the course of a simulation [3].

### 2.3.2 Fitness Function

I took the sum of the differences between the prediction and the truth value as the fitness function for each individual (species), because both the targets were numeric. Although the fitness was lower than 0, the algorithm worked as expected.

### 2.3.3 Training Process

An existing python algorithm in the NEAT package and its visualization package were imported and used for training. The training parameters for NEAT were listed in an independent file. Since the size of each dataset was small, the training parameters were modified in order to conduct more mutation. 300 generations with 300 pops were trained. 6 networks for 2 targets from 3 pairs of datasets have been trained. For each pair of datasets, 2 networks have been trained with the features and targets mentioned in **1.3**.

## 3. Results

### 3.1 Traditional Feed-forward Neural Networks

Each network was trained and finished within 10 seconds. The training loss and testing loss of 6 neural networks of 2 targets of 3 datasets are shown as below.

| Target | Dataset 1 | Dataset 2 | Dataset 3 |
|---|---|---|---|
| PHI | 0.0455 | 0.0318 | 0.0321 |
| logK | 0.0230 | 0.0270 | 0.0325 |

**Table.1.** Traditional Feed-forward Neural Networks Training Loss (MSE). Each estimator was trained and tested 10 times. The table shows the best results. the max difference between the best and the worst results were 0.0003.

| Target | Dataset 1 | Dataset 2 | Dataset 3 |
|---|---|---|---|
| PHI | 0.0555 | 0.0322 | 0.0409 |
| logK | 0.0244 | 0.0291 | 0.0374 |

**Table.2.** Traditional Feed-forward Neural Networks Testing Loss (MSE). Each estimator was trained and tested 10 times. The table shows the best results. the max difference between the best and the worst results were 0.0151.

### 3.2 Reduced Neural Networks

The testing loss of 6 neural networks of 2 targets of 3 datasets are shown as below.

| Target | Dataset 1 | Dataset 2 | Dataset 3 |
|---|---|---|---|
| PHI | 0.0594 | 0.0395 | 0.0649 |
| logK | 0.0328 | 0.0289 | 0.0487 |

**Table.3.** Traditional Feed-forward Neural Networks Training Loss (MSE). Each estimator was trained, tested and reduced 10 times. The table shows the best results. the max difference between the best and the worst results were 0.0205.

## 3.3 NEAT

Each network was trained and finished within 5 minutes. The shortest training time was 30 seconds. The training loss and testing loss of 6 neural networks of 2 targets of 3 datasets are shown as below.

| Target | Dataset 1 | Dataset 2 | Dataset 3 |
|---|---|---|---|
| PHI | 0.0302 | 0.0112 | 0.0103 |
| logK | 0.0129 | 0.0036 | 0.0106 |

**Table.4.** NEAT Training Loss (MSE). Each estimator (a network for a certain problem) was trained and tested 5 times. The table shows the best results. the max difference between the best and the worst results were 0.0238

| Target | Dataset 1 | Dataset 2 | Dataset 3 |
|---|---|---|---|
| PHI | 0.0376 | 0.0108 | 0.0205 |
| logK | 0.0225 | 0.0051 | 0.0114 |

**Table.5.** NEAT Testing Loss (MSE). Each estimator was trained and tested 5 times. The table shows the best results. the max difference between the best and the worst results were 0.0514.

The training result visualization are shown as below. (Since the figures are not intuitive, only 2 figures of the logK – Dataset 2 estimator are shown. The other figures of NEAT were saved as files.)

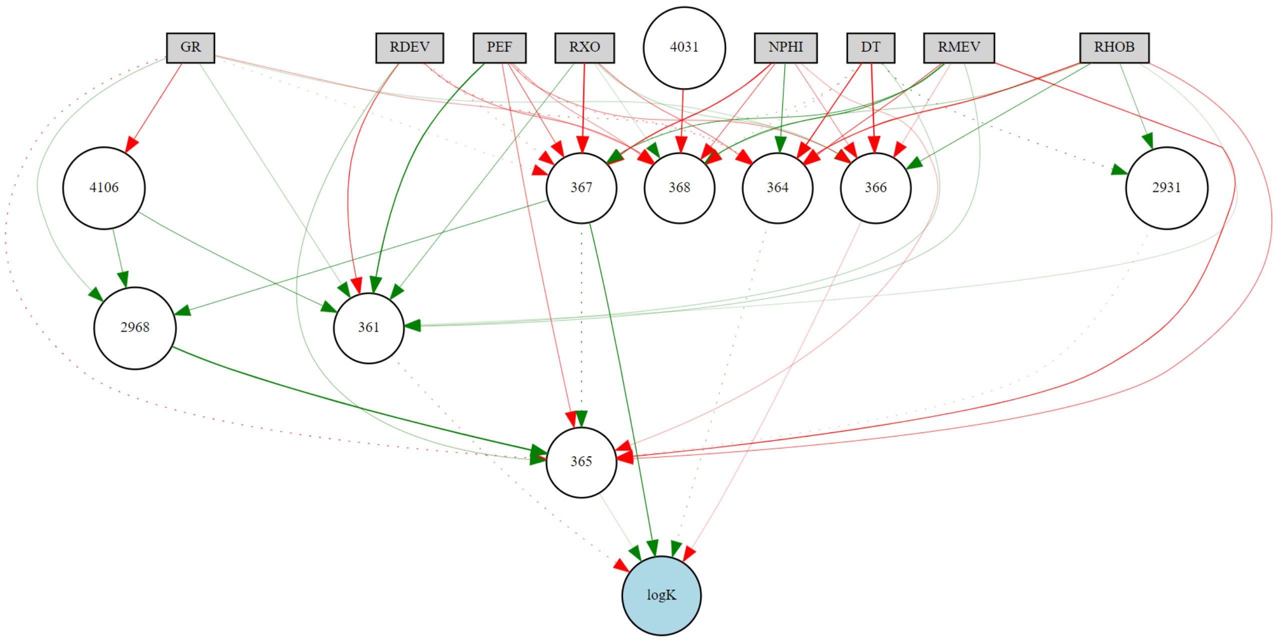

**Fig.3.** The network trained by NEAT (PHI, Dataset 2). The grey blocks are the inputs (features) and the blue sphere is the output (target). The arrow lines are the connections. Green arrow lines pass positive connection and red arrow lines pass negative connection. The dotted arrow lines are disabled.

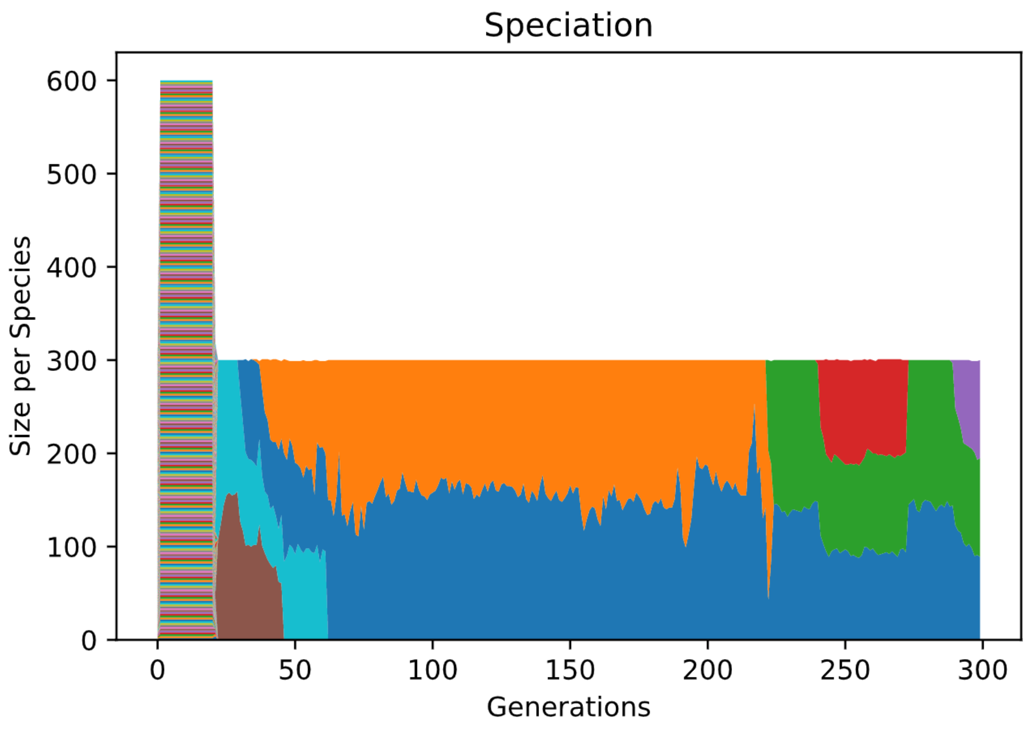

**Fig.4.** The change of pops and types of species with respect to generations (PHI, Dataset 2). Each pattern stands for a species. The normal amount of the pops was set as 300. However, 300 species were initially generated. Each species had at least 2 pops. Therefore, At the beginning of the evolution, the amount of pops was about 600.

## 4. Discussion

The traditional feed-forward neural networks can predict the two targets with acceptable loss. The estimators (networks for a certain problem) were stable. The differences between the performance of the best and the worst estimators were small. The reduced network generally maintained the functionality and stability.

Comparing with the traditional feed-forward neural networks, networks with better performance can be trained with NEAT. However, the training time of NEAT was much longer although the training time was unstable (30 seconds ~ 5 minutes). The unitability was the result of the uncertainty of the species extinction and creation. A generation with fewer species requires less calculation. Moreover, the estimators trained by NEAT were unstable. The differences between the performance of the best and the worst estimators can be very large (0.0514). Although better estimators can be trained by NEAT, some estimator of very low quality can also be generated.

## 5. Conclusion and Future Work

Neural networks were trained multiple times with different methods for oil reservoir porosity and permeability prediction. The networks trained with tradition back-propagation methods were capable for the prediction of the two targets after network reduction. NEAT can generate networks with even lower loss. However, the training time was much longer and the performance of the trained networks was more unstable.

The hyper parameters for NEAT were modified many times in this paper. However, it is probably not the optimal settings and further adjustment is necessary. Study has raised are other extended NEAT algorithms including rtNEAT, HyperNEAT, etc., which can also be implemented to oil well porosity and permeability predictions.

## Reference

[1] McClelland, J. L., Rumelhart, D. E., & PDP Research Group. (1986). Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2, 216-271.

[2] Dasgupta, D., & Michalewicz, Z. (Eds.). (2013). *Evolutionary algorithms in engineering applications*. Springer Science & Business Media.

[3] Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, *10*(2), 99-127.

[4] Kuo, H., Gedeon, T. D., & Wong, P. M. (1999). A clustering assisted method for fuzzy rule extraction and pattern classification. *In ICONIP'99. ANZIIS'99 & ANNES'99 & ACNN'99. 6th International Conference on Neural Information Processing. Proceedings (Cat. No. 99EX378)* (Vol. 2, pp. 679-684). IEEE.

[5] Rumelhart, D. E., Durbin, R., Golden, R., & Chauvin, Y. (1995). Backpropagation: *The basic theory. Backpropagation: Theory, architectures and applications,* 1-34.

[6] Gedeon, T. D., & Harris, D. (1991). Network reduction techniques. *In Proceedings International Conference on Neural Networks Methodologies and Applications* (Vol. 1, pp. 119-126).