Optimisation of Neural Network with Simultaneous Feature Selection and Network Prunning using Evolutionary Algorithm

WK Wong[1], Chekima Ali[2], Wong Kii Ing[1], Law Kah Haw[1], Vincent Lee[1]
[1]*Curtin University Sarawak, Miri, Sarawak*
[2]*Universiti Malaysia Sabah, Kota Kinabalu*
weikitt.w@curtin.edu.my

***Abstract*—Most advances on the Evolutionary Algorithm optimisation of Neural Network are on recurrent neural network using the NEAT optimisation method. For feed forward network, most of the optimisation are merely on the Weights and the bias selection which is generally known as conventional Neuroevolution. In this research work, a simultaneous feature reduction, network pruning and weight/biases selection is presented using fitness function design which penalizes selection of large feature sets. The fitness function also considers feature and the neuron reduction in the hidden layer. The results were demonstrated using two sets of data sets which are the cancer datasets and Thyroid datasets. Results showed backpropagation gradient descent error weights/biased optimisations performed slightly better at classification of the two datasets with lower misclassification rate and error. However, features and hidden neurons were reduced with the simultaneous feature /neurons switching using Genetic Algorithm. The number of features were reduced from 21 to 4 (Thyroid dataset) and 9 to 3 (cancer dataset) with only 1 hidden neuron in the processing layer for both network structures for the respective datasets. This research work will present the chromosome representation and the fitness function design.**

***Index Terms*—Neuroevolution; Feature Selection; Network Pruning; Evolutionary Algorithm.**

I. INTRODUCTION

Neural network has been applied in countless pattern recognition and have proven to be efficient in application particularly in supervised learning. The most common weight update system for a feed forward neural network is the gradient descent algorithm in which the errors with regards to the label output are backpropagated from the output to the input layers to update the weights [1][2]. However, for large networks, the phenomenon of vanishing gradient can prevent a fully optimised weight update in the neural network [3]. One solution to the vanishing gradient problem is by applying an LSTM layer to network as demonstrated in [4]. Another method of avoiding the vanishing gradient problem is to perform weights optimisation using evolutionary algorithm to select weights and biases in which the errors are not backpropagated to the back layers and hence, the issue of vanishing gradient does not arise. Such methods are generally classified as Neuroevolution methods in which the weights and the biases are encoded as chromosome in the evolutionary optimisation. The conventional Neuroevolution involves using the error as fitness to the evolutionary algorithm such as genetic algorithm. Several research work have been conducted which involves using Evolutionary Programming (EP) [5]. Other types of Neuroevolution includes features with evolving topology and weights. Neuroevolution are mostly using reinforced learning because it enables the learning to be performed via a fitness function such as for games application as demonstrated in [6] and [7]. However, more focus is applied in the optimisation of recurrent neural network using evolutionary algorithm such as NEAT (neuro Evolutionary of augmented topology) in which the connections and weights of the recurrent neural network are simultaneously selected using evolutionary process as explained in [8]. The NEAT networks are suitable for reinforced learning particularly in the inverse pendulum balancing cart as the benchmark problem. The concept of NEAT and its variants has great appeal especially on the reinforced learning due to the fact that it does not need a set of training, label instances for training. Hence, Neuroevolution is extensively studied for reinforced learning rather than for classification machine learning problem.

The research objective of this paper is to demonstrate simultaneous feature selection and network pruning for feed forward network optimisation using by proposing a fitness function that penalises selection of large features sets and hidden neurons in the processing layer. This will particularly useful to eliminate noisy features from datasets and simultaneously to selection of weights biases and number of hidden neurons. Hence, this method also reduces overfitting by pruning the network.

II. METHODOLOGY

The research objective is to demonstrate simultaneous feature selection and network pruning. The network will consist of a single hidden layer consist of 3 neurons. Hence, the network will be allowed to select from the range of 1 to 3 neuron in the hidden layer and the *n*, number of features in the datasets. The output layer consist of the categories of the class as indicated in the dataset ie. 2 classes for breast cancer datasets and 3 classes for thyroid datasets. The *i*,*j* *k* layer will denote the input feature layer, hidden layer and output layer respectively. The following

Equation 1 is the transfer function for the hidden and output neurons. Equation 2 and 3 is the activation for the summation of weight and features at the jth layer and kth output layer respectively. Both arethe tangent sigmoid cost function with biases b_j and b_k respectively. Equation 4 and 2 are the summation of weights product of the neuron output at hidden layer and output layer.

$$f(u_j) = \frac{1}{1+e^{-(U_j)}} \tag{1}$$

$$u_j = \sum_{j=1}^{3} w_{ij}.xi + b_j \tag{2}$$

$$y_k = f(u_k) = \frac{1}{1+e^{-(U_k)}} \tag{3}$$

$$u_k = \sum_{k=1}^{n} w_{jk}.u_{jk} + b_k \tag{4}$$

$$RMSE = \sqrt{\frac{1}{m}\sum_{t=1}^{m}\sum_{k=1}^{n}(z_k(t) - y_k(t))^2} \tag{5}$$

where m is the number of instances in training, n is the number of output class, U the neuron input to the sigmoid transfer in the j,kth neuron. Z_k and y_k are the training output label and network output for the m instances. x^i is the feature vector where $xi \in \boldsymbol{R}^o$,o is the number of features in the dataset. b_j and b_k are the individual biases of the neurons of the j and k layer.

Figure 1 shows the structure of the neural network for optimised by the evolutional Algorithm (Genetic Algorithm). The hidden layer, j neurons consist of a maximum of 3 neurons and switching mechanism allowed to turn of the j neurons respectively. The switching for the individual j neuron consist of a binary vector swj where swj is a binary vector, R^3.

Similarly for the i layer where the vector x_i which are the individual features from the data sets, a switching consist of binary sw_i in which each entry in the binary vector coresponds to the the switch to turn on or off the individual feature, $\boldsymbol{R}^o$, o is the number of features for the dataset.

The product of the the weight and the features are multipled by the switching vector for turning the feature on and off. Hence the weighted product for the activation function is multiplied by the switching vector as shown in Equation 6 and 7.

$$u_j = \sum_{j=1}^{3} w_{ij} \text{ x } (xi.swi) + b_j \tag{6}$$

$$u_k = \sum_{k=1}^{n} w_{jk} \text{ x } (swj\ .f(u_j)) + b_k \tag{7}$$

The genetic algorithm uses a vector of decimal number,*f* to represent the weights, biases and the switching.The switching vector swi and swj are binarized from the decimal number data type using Equation 8.

$$\text{swi} = \begin{cases} 1, f(i) > 0 \\ 0\ f(i) \leq 0 \end{cases} \tag{8}$$

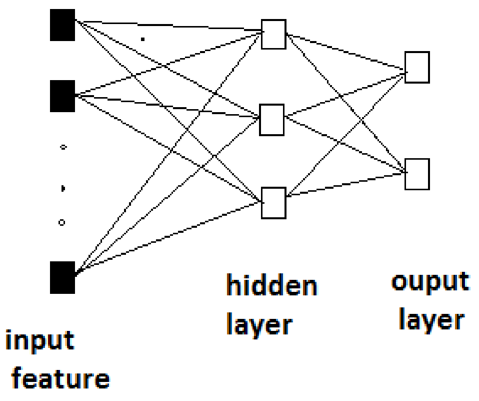

Figure 2.1 : Structure of proposed neural network

The chromosome vector, *f* consist of vector of signed precision number, where *f* is a vector $R^{(m*n)+(o*n)+o+2n+m}$ where *m* is the number of total feature, *n* is the number of unprunned hidden neuron and *o* is the number of output class in the output layer.

Table 1
Details of Genetic Algorithm

Classes	Cancer dataset
Crossover function	Crossover immediate, 0.5
Mutation function	Uniform mutation , 0.01
Population size	100
Elite children guaranteed for Crossover to next generation	3
Individuals generated from cross over function over the entire generations	0.9

The dataset used for testing are the thyroid data set [9] and cancer data set [10]. Table 1 shows the details of the datasets. The data sets are separated into 90%- 10% partition for training and testing with no crossvalidation. Table 2 shows the partition of datasets for training and validation.

Table 2
Details of datasets

Classes	Cancer dataset	Thyroid data set

Instances	720(total)
	630(training)
	70 (testing)
		7200(total)
		6480(training)
		720(testing)

Features	9	21
Classes /Categories	2	3

The following are the stopping criteria for the Genetic Algorithm to stop running.

i) **Generations** — The algorithm stops when the number of generations reaches the value of **600 generations**.
ii) **Fitness limit** — The algorithm stops when the value of the fitness function for the best point in the current population is less than or equal to **Fitness limit**.

iii) **Stall generations** — The algorithm stops when the average relative change in the fitness function value over **Stall generations** is less than **Function tolerance**.
iv) **Function Tolerance** — The algorithm runs until the average relative change in the fitness function value over 200 generations is less than 0.01

The fitness functions is as shown in Equation 10 is used to generate a score for the chromosome configuration evaluation. The Genetic Algorithm continually evaluates the configurations until the stopping criteria is reached. The fitness score equations is as expressed in Equation 10. The immediate cross over function as expressed in Equation 9 creates children in each generation from the parent chromosome , parent 1 and parent 2 as shown in Equation 9.

$$\text{child} = \text{parent1} + \mathbf{0.5} * (\text{parent2} - \text{parent1}) \tag{9}$$

$$\begin{aligned} &fitness\ score \\ &= RMSE + \frac{Missclassified\ samples}{total\ samples} \\ &+ w\left(\frac{total\ selected\ features}{total\ features}\right) \\ &+ w\left(\frac{total\ selected\ neurons\ in\ hidden\ layer}{total\ range\ of\ neurons\ in\ hidden\ layer}\right) \end{aligned} \tag{10}$$

where W =0.01

III. RESULTS

Table 3 shows the comparison between the evolutionary optimised neural networks and the backpropagation optimised neural network. The backpropagation neural network is trained with the matlab neural network toolbox with 0.001 learning rate and data partition of (75%, 15%, 10%) for training/validation/testing. The equivalent training/validation/ testing RMSE is shown. For classification rate, an output of >0.5 is considered a positive output (1.00) and ≤ 0.5 is considered an negative output (0.00).

As shown in table 3.1, the backpropagation gradient descent weight configurations showed slightly higher misclassification rate for cancer data sets. For thyroid data set, the backpropagation trained network showed slightly better classification rate.

Although the switching mechanism manage to reduce the number of features applied for classification, almost similar test performance were acquired for both data sets but the network with switching performed slightly better with the cancer dataset as shown with lower misclassification samples.

The structure for the Cancer data set neural Network selected was 3 -1-2. Only 1 neurons was selected in the hidden neuron (processing layer) and 3 features selected with the tradeoff weightage,w of 0.01. The network structure for the thyroid data set neural Network selected was 4 -1-3. Only 1 neurons was selected in the hidden neuron (processing layer) and 4 features selected with the weightage of 0.01.

When comparing the backpropagation neural network and the neuroevolution with feature and neuron switching, back propagation ANN performance was almost similar compared to the neuroevolution. However, the only improvement of the neuroevolution is the reduced number of features and reduced neuron in the hidden layer.

Table 3
Comparison of recognition based on thyroid and cancer data sheets

Thyroid data set			
	Back propagation	Neuroevolution Without switching	Neuro evolution with switching
RMSE	(Training) 0.1797, (validation) 0.1775	0.2432	0.2604
Samples missclassified	59/5040 (validation), 310/1080 (training	(481/6480) samples	(481/6480) samples
Test sample	0.1857	0.2432	0.2583
Test RMSE	46 samples	52/720 samples	53/720 samples
Cancer data set			
	Backpropagation	Neuroevolution Without switching	Neuroevolution with switching
RMSE	0.2041 (training), 0.0853 (validation)	0.22162	0.1159
Samples missclassified	(23/504) sample (training), (1/105) sample (validation)	(28/730) samples)	(21/730) samples)
Test sample missclassified	(6 /70) samples	(8/72) sample	(5/72) samples
Test RMSE	0.2919	0.4210	0.1757

The results for the proposed feed forward optimisation is compared to the benthmark test in [5] as shown in Table 4. [5] is the neural network trained with Evolutionary Programming (EP). The misclassification rate in [5] is the mean of the several attempt in testing.

Table 4
Benchmarking with other research work

Thyroid data set		
	Neuroevolution with EP [5] (mean)	Neuroevolution with switching
Missclassification rate for training/validation	0.00823(training) 0.01174 (validation)	0.0742
Missclassification rate for training/validation	0.01376	0.0736
Cancer data set		
Missclassification rate for training/validation	0.03773 (training) 0.00590 (validation)	0.02876
Missclassification rate for training/validation	0.01376	0.00694

IV. CONCLUSION AND FUTURE WORKS

The results showed neuro evolution performance in simultaneous selection of weights, biases, feature subset and number of hidden neurons in the hidden layer. The slight performance difference could be due to the network pruning and feature selection performed. However, more work needs to be performed to improve the neuroevoution weight selection due to the heuristic nature of Genetic Algorithm.

This research is primarily introducing feature selection encoding and network pruning. As future works, multi later hidden neurons will be examined in which layers can be added and subtracted and the network weights and biases simultaneously configured for feedforward network.

Future works will also be focused primarily on feedforward network since reccurent neural network optimisation using evolutionary algorithm is well documented using the NEAT configurations. A mixed chromosome design can also be considered incorporating binary and decimal point type chromosomes to reduce complexity in search space. This will reduce the number of possible chromosome encoding the structure and hence reduce processing time.

Backpropagation feed forward neural network is very efficient method of optimising the weights. However, when large network is applied, the phenomenon such as 'vanishing gradient' will effect the optimisation of the weights. The network shown in this research is just as single hidden layer network. Hence, the heuristic method of weight selection is shown to be slightly inefficient when compared the gradient descent method. A multilayer Neuro evolution structure can be further examined to further reveal the optimisation of weights with neuro evolution with feature and neuron switching. Neuroevolution does not suffer from vanishing gradient which primarily effects gradient descent weights optimisation but is known to be computationally intensive and time consuming. However, an efficient neuroevolution with feature and network pruning is worthwhile research for highly noisy datasets. The weightage w, as indicated in Equation 10 needs to be kept at a small value in which it can be a functional tradeoff forreduction of features and maintain classification rate in training sets. A slightly higher value can be considered for w in highly noisy feature sets. The weights can be further untied for feature reduction and neuron pruning. This will require more testing with other data sets.

REFERENCE

[1] Stuart Russell, Peter Norvig, Artificial Intelligence A modern approach, pg 578

[2] Rumenlhart. David E; Hinton G; Williams Ronald J, 1986, 'Learning representations by back-propagating errors', Nature 323(6088):533-536

[3] Hochreiter. S; Bengio Y ; Fransconi P; Schmidhuber J. 'Gradient Flow in Recurrent nets, the difficulty of learning long term dependencies, A field guide to dynamical Recurrent Neural Networks IEE press, 2001

[4] Hochreiter. S; Schmidhubber Jurgen, 1997, "Long short Term Memory", Neural Computation 9 (8):1735-1780

[5] Xin Yao, Yong Li u, 1997, 'A new Evolutionary System for evolving Artificial neural networks, IEEE transactions on neural networks 8(3)

[6] Kenneth O. Stanley ; Bobby D. Bryant and Risto Miikukuilainen, 2005, 'Evolving Neural network Agents in the Nero Video Game', Evolutionary Computation, vol9(6) : 653-668

[7] Reeder J.; Miquez R.; Sparks J.; Georgipoulus M. and Anagnostopoulos G., 2008,'Interactively evolved modular neural networks for game and agent control', Computational Intelligence and Games:167-174

[8] Kenneth O Stanley, Risto Miikkuilainen, 2002 'Evolving Neural Networks through augmenting Topologies', Evolutionary computation 10(2):99-107

[9] www.ics.uci.edu/pub/ml-repos/machine-learning-database/throid-disease

[10] http://archiev.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29