FUNCTIONAL LINK NEURAL NETWORK WITH MODIFIED BEE-FIREFLY LEARNING ALGORITHM FOR CLASSIFICATION TASK

YANA MAZWIN MOHMAD HASSIM

UNIVERSITI TUN HUSSEIN ONN MALAYSIA
FUNCTIONAL LINK NEURAL NETWORK WITH MODIFIED BEE-FIREFLY LEARNING ALGORITHM FOR CLASSIFICATION TASK

YANA MAZWIN MOHMAD HASSIM

A thesis submitted in fulfillment of the requirement for the award of the Doctor of Philosophy.

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

AUGUST, 2016
ACKNOWLEDGEMENT

All praise to the almighty Allah, for the good health and wellbeing that were necessary to complete this study. I would like to express my sincere gratitude to Assoc. Prof. Dr. Rozaida Ghazali, my supervisor, for the continuous support of my PhD study and research, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

I take this opportunity to express gratitude to Faculty of Computer Science and Information Technology (FSKTM) and Centre for Graduate Studies (CGS) of Universiti Tun Hussein Onn Malaysia (UTHM) for providing good facilities and inspiring environment for me to complete this study.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their helping hand in this study.
ABSTRACT

Classification is one of the most frequent studies in the area of Artificial Neural Network (ANNs). The ANNs are capable of generating a complex mapping between the input and the output space to form arbitrarily complex nonlinear decision boundaries. One of the best-known types of ANNs is the Multilayer Perceptron (MLP). MLP usually requires a large amount of available measures in order to achieve good classification accuracy. To overcome this, a Functional Link Neural Networks (FLNN) which has a single layer of trainable connection weights is used. The single layer property of FLNN also make the learning algorithm used less complicated compared to MLP network. The standard learning method for tuning weights in FLNN is Backpropagation (BP) learning algorithm. However, the algorithm is prone to get trapped in local minima which affect the performance of FLNN network. This work proposed the implementation of modified Artificial Bee Colony with Firefly algorithm for training the FLNN network to overcome the drawback of BP-learning algorithm. The aim is to introduce an improved learning algorithm that can provide a better solution for training the FLNN network for the task of classification.
ABSTRAK

Pengkelasan adalah salah satu kajian yang paling kerap dilakukan dalam bidang Artificial Neural Networks (ANNs). ANNs mampu menjana pemetaan kompleks di antara ruang input dan output untuk membentuk sempadan keputusan tidak linear yang kompleks. Salah satu model yang paling terkenal dalam ANNs adalah Multilayer Perceptron (MLP). MLP biasanya memerlukan beberapa langkah yang disediakan untuk mencapai ketepatan pengelasan yang baik. Untuk mengatasi masalah ini, Functional Link Neural Network (FLNN) sejenis rangkaian neural yang mempunyai hanya satu lapisan pemberat digunakan untuk menggantikan MLP. Sifat FLNN yang hanya mempunyai satu lapisan pemberat juga membuat algoritma pembelajaran yang digunakan untuk melatih rangkaian menjadi kurang rumit berbanding dengan rangkaian MLP. Kaedah pembelajaran piawai untuk mengemaskini pemberat FLNN adalah dengan menggunakan algoritma backpropagation (BP). Namun begitu, algoritma BP mudah terperangkap di local minima yang mana ianya memberi kesan kepada prestasi rangkaian FLNN. Kajian ini mencadangkan pelaksanaan algoritma Artificial Bee Colony bersama dengan algoritma Firefly sebagai skim pembelajaran untuk melatih rangkaian FLNN bagi mengatasi kelemahan algoritma BP. Tujuannya adalah untuk memperkenalkan skim pembelajaran alternatif yang boleh memberikan penyelesaian yang lebih baik bagi melatih rangkaian FLNN untuk tugas pengkelasan.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>DECLARATION</th>
<th>ii</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>iii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>ABSTRAK</td>
<td>v</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>x</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xiv</td>
</tr>
<tr>
<td>LIST OF SYMBOLS AND ABBREVIATIONS</td>
<td>xvii</td>
</tr>
<tr>
<td>LIST OF PUBLICATIONS</td>
<td>xix</td>
</tr>
</tbody>
</table>

## CHAPTER 1 INTRODUCTION

| 1.1 Background of Research          | 1  |
| 1.2 Problem Statements              | 3  |
| 1.3 Aim and Objective of Research   | 6  |
| 1.4 Scope of Research               | 6  |
| 1.5 Significance of Research        | 7  |
| 1.6 Chapter Summary                 | 7  |
| 1.7 Organization of Thesis          | 7  |

## CHAPTER 2 LITERATURE REVIEW

<p>| 2.1 Introduction                    | 9  |
| 2.2 Classification Task             | 10 |
| 2.3 Artificial Neural Networks       | 14 |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1</td>
<td>Components of Artificial Neural Networks</td>
<td>15</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Different Models of Neural Networks</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Feedforward Neural Networks</td>
<td>19</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Multilayer Perceptron</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>Higher Order Neural Networks</td>
<td>22</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Properties of HONNs</td>
<td>23</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Types of HONNs</td>
<td>28</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Functional Link Neural Networks (FLNN)</td>
<td>31</td>
</tr>
<tr>
<td>2.4.4</td>
<td>The FLNN Learning Algorithm</td>
<td>35</td>
</tr>
<tr>
<td>2.5</td>
<td>Swarm Intelligence Optimization</td>
<td>37</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Artificial Bee Colony Algorithm</td>
<td>40</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Firefly Algorithm</td>
<td>46</td>
</tr>
<tr>
<td>2.6</td>
<td>Hybridization of Metaheuristic Algorithms</td>
<td>49</td>
</tr>
<tr>
<td>2.7</td>
<td>Discussion: Scenario Leading to the Research Framework</td>
<td>50</td>
</tr>
<tr>
<td>2.8</td>
<td>Chapter Summary</td>
<td>53</td>
</tr>
</tbody>
</table>

**CHAPTER 3 RESEARCH METHODOLOGY**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>54</td>
</tr>
<tr>
<td>3.2</td>
<td>Research Framework</td>
<td>54</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Phase 1: Data Preparation</td>
<td>56</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Phase 2: FLNN Training Procedure</td>
<td>65</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Phase 3: Results Analysis</td>
<td>69</td>
</tr>
<tr>
<td>3.3</td>
<td>Environment</td>
<td>71</td>
</tr>
<tr>
<td>3.4</td>
<td>Parameters Setting</td>
<td>71</td>
</tr>
<tr>
<td>3.4.1</td>
<td>BP-learning Algorithm</td>
<td>71</td>
</tr>
<tr>
<td>3.4.2</td>
<td>ABC Learning Algorithm</td>
<td>72</td>
</tr>
<tr>
<td>3.4.3</td>
<td>MABC Learning Algorithm</td>
<td>73</td>
</tr>
<tr>
<td>3.4.4</td>
<td>MBF Learning Algorithm</td>
<td>74</td>
</tr>
<tr>
<td>3.5</td>
<td>Chapter Summary</td>
<td>75</td>
</tr>
</tbody>
</table>

**CHAPTER 4 MODIFIED BEE-FIREFLY LEARNING ALGORITHM**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>76</td>
</tr>
<tr>
<td>4.2</td>
<td>FLNN as objective function</td>
<td>77</td>
</tr>
<tr>
<td>4.3</td>
<td>ABC learning algorithm for FLNN (FLNN-ABC)</td>
<td>78</td>
</tr>
</tbody>
</table>
CHAPTER 5 SIMULATION RESULTS AND DISCUSSION 114

5.1 Introduction 114
5.2 Experimental Design 114
  5.2.1 Dataset Division 115
  5.2.2 Network Model Topology 115
5.3 Best Average Simulation Results 116
5.4 Analysis of Results based on Datasets 123
  5.4.1 Results of Proposed FLNN-MBF Model with Standard FLNN-BP and MLP-BP Models 123
  5.4.2 Results on Improved Performance of FLNN 127
  5.4.3 Classification Performance of Proposed Learning Algorithm 133
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5 Statistical Performance Evaluation</td>
<td>138</td>
</tr>
<tr>
<td>5.6 Threat to Validity</td>
<td>139</td>
</tr>
<tr>
<td>5.7 Chapter Summary</td>
<td>140</td>
</tr>
</tbody>
</table>

**CHAPTER 6 CONCLUSION AND FUTURE WORK**  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Introduction</td>
<td>142</td>
</tr>
<tr>
<td>6.2 Research Summary</td>
<td>142</td>
</tr>
<tr>
<td>6.3 Contribution</td>
<td>146</td>
</tr>
<tr>
<td>6.4 Future Works</td>
<td>146</td>
</tr>
<tr>
<td>6.5 Closing Remarks</td>
<td>148</td>
</tr>
</tbody>
</table>

**REFERENCES**  

**VITA**
LIST OF TABLES

Table 2.1: FLNN application with BP-learning algorithm 36
Table 2.2: Properties of self-organization in swarms (Bonabeau et al., 1999) 38
Table 2.3: The main components of honey bee swarms (Karaboga, 2005) 41
Table 2.4: Application areas of ABC algorithm 44
Table 2.5: Applications of FA algorithms 48
Table 3.1: Summary of datasets 60
Table 3.2: Summary of missing value information in datasets 61
Table 3.3: Datasets after data cleaning 62
Table 3.4: Parameters setting in BP-learning algorithm 72
Table 3.5: Parameters setting in ABC learning algorithm 73
Table 3.6: Parameters setting in MABC learning algorithm 73
Table 3.7: Parameters setting in MBF learning algorithm 74
Table 4.1: Experimental result on XOR dataset 86
Table 4.2: Experimental result on 3-bit Parity problem 87
Table 4.3: Experimental result on 4-bit Encoder-Decoder problem 88
Table 4.4: Pilot Experimental result on Balance Scale dataset 89
Table 4.5: Pilot Experimental result on Haberman’s Survival dataset 89
Table 4.6: Pilot Experimental result on Mammographic Mass Dataset 89
Table 4.7: Pilot Experimental result on Breast Cancer Wisconsin (Original) Dataset 89
Table 4.8: Pilot Experimental result on Credit Approval Dataset 90
Table 4.9: Pilot Experimental result on ILPD (Indian Liver Patient Dataset) Dataset
Table 4.10: Pilot Experimental result on Iris Dataset
Table 4.11: Pilot Experimental result on Liver Disorders (BUPA) Dataset
Table 4.12: Pilot Experimental result on Lymphographic Dataset
Table 4.13: Pilot Experimental result on MONK's Problems Dataset (MONK-2)
Table 4.14: Pilot Experimental result on Page Blocks Classification Dataset
Table 4.15: Pilot Experimental result on Zoo Dataset
Table 4.16: Pilot Experimental result on Pima Indians Diabetes Dataset
Table 4.17: Pilot Experimental result on Thyroid Disease Dataset
Table 4.18: Binomial/binary classification datasets
Table 4.19: Multiclass classification datasets
Table 5.1: Best Average Result from MLP-BP
Table 5.2: Best Average Result from FLNN-BP
Table 5.3: Best Average Result from FLNN-ABC
Table 5.4: Best Average Result from FLNN-MABC
Table 5.5: Best Average Result from FLNN-MBF
Table 5.6: Number of trainable weights and biases used in all network models
Table 5.7: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Balance Scale dataset
Table 5.8: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Haberman’s Survival dataset
Table 5.9: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on mammographic mass dataset
Table 5.10: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Breast Cancer Wisconsin dataset

Table 5.11: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Credit Approval dataset

Table 5.12: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on ILPD dataset

Table 5.13: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Iris dataset

Table 5.14: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on BUPA liver disorders dataset

Table 5.15: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Lymphographic dataset

Table 5.16: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on MONK's problems dataset

Table 5.17: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Page Blocks Classification dataset

Table 5.18: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Zoo dataset

Table 5.19: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Pima Indians diabetes dataset

Table 5.20: Result of the proposed FLNN-MBF with standard models of FLNN-BP and MLP-BP on Thyroid disease dataset

Table 5.21: The overall improvement of FLNN-MBF

Table 5.22: Classification performance measurements in Balance Scale dataset
Table 5.23: Classification performance measurements in Haberman’s survival dataset

Table 5.24: Classification performance measurements in mammographic mass dataset

Table 5.25: Classification performance measurements in Breast Cancer Wisconsin dataset

Table 5.26: Classification performance measurements in Credit Approval dataset

Table 5.27: Classification performance measurements in ILPD dataset

Table 5.28: Classification performance measurements in Iris dataset

Table 5.29: Classification performance measurements in BUPA dataset

Table 5.30: Classification performance measurements in Lymphographic dataset

Table 5.31: Classification performance measurements in MONK's problems dataset

Table 5.32: Classification performance measurements in Page Blocks Classification dataset

Table 5.33: Classification performance measurements in Zoo dataset

Table 5.34: Classification performance measurements in Pima Indians diabetes dataset

Table 5.35: Classification performance measurements in Thyroid disease dataset

Table 5.36: Rank of each models on different dataset based on classification accuracy

Table 5.37: Post-hoc procedure based on pairwise comparisons method
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The input-output mapping of a classifier</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Neuron and its representation (Samarasinghe, 2006)</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Computational elements in artificial neuron</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>The basic structure of MLP</td>
<td>21</td>
</tr>
<tr>
<td>2.5</td>
<td>Inputs of neuron in first order neural network</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Different architecture of HONNs (Ghazali, 2007)</td>
<td>25</td>
</tr>
<tr>
<td>2.7</td>
<td>Truth table for XOR problem</td>
<td>26</td>
</tr>
<tr>
<td>2.8</td>
<td>Solving XOR problem by using 2nd order HONN</td>
<td>26</td>
</tr>
<tr>
<td>2.9</td>
<td>Linear separation hyperplane for XOR problem using 2nd order HONN</td>
<td>27</td>
</tr>
<tr>
<td>2.10</td>
<td>The architecture of FLNN</td>
<td>28</td>
</tr>
<tr>
<td>2.11</td>
<td>Pi Sigma Neural Network of $K$-th order</td>
<td>29</td>
</tr>
<tr>
<td>2.12</td>
<td>The Ridge Polynomial Neural Network of $k$-th order</td>
<td>30</td>
</tr>
<tr>
<td>2.13</td>
<td>The FLNN of type functional expansion model</td>
<td>32</td>
</tr>
<tr>
<td>2.14</td>
<td>The FLNN of type tensor representation</td>
<td>33</td>
</tr>
<tr>
<td>2.15</td>
<td>The standard ABC pseudo code</td>
<td>43</td>
</tr>
<tr>
<td>2.16</td>
<td>The standard FA pseudo code</td>
<td>48</td>
</tr>
<tr>
<td>2.17</td>
<td>Scenario Leading to the Research Framework</td>
<td>52</td>
</tr>
<tr>
<td>3.1</td>
<td>The research framework</td>
<td>55</td>
</tr>
<tr>
<td>3.2</td>
<td>Pictorial representation of 10-fold cross-validation</td>
<td>64</td>
</tr>
<tr>
<td>3.3</td>
<td>The influence of learning rate on network learning</td>
<td>67</td>
</tr>
<tr>
<td>4.1</td>
<td>The development process of proposed learning algorithm</td>
<td>77</td>
</tr>
<tr>
<td>4.2</td>
<td>Flowchart of ABC learning algorithm for FLNN</td>
<td>80</td>
</tr>
<tr>
<td>4.3</td>
<td>The initialization of food sources (solutions)</td>
<td>81</td>
</tr>
</tbody>
</table>
Figure 4.4: Exploitation process and food positions visualized by the employed bee

Figure 4.5: The greedy selection

Figure 4.6: Exploitation process and food positions visualized by the onlooker bee

Figure 4.7: Exploration by scout bee

Figure 4.8: Pseudo code of the ABC learning algorithm for FLNN

Figure 4.9: Truth table for 3-bit Parity problem

Figure 4.10: Truth table for 4-bit Encoder-Decoder problem

Figure 4.11: Exploitation process of employed and onlooker bee in MABC algorithm

Figure 4.12: Global exploration of scout bee in MABC algorithm

Figure 4.13: Flowchart of MABC learning algorithm for FLNN

Figure 4.14: Pseudo code of the MABC learning algorithm for FLNN

Figure 4.15: Experimental results on binomial/binary classification datasets

Figure 4.16: Experimental results on multiclass classification datasets

Figure 4.17: The main step of the proposed MBF algorithm

Figure 4.18: Pseudo code of the MBF learning algorithm for FLNN

Figure 4.19: Flowchart of MBF learning algorithm for FLNN

Figure 5.1: Best accuracy results from all network models

Figure 5.2: FLNN performance on NMSE with increasing network order

Figure 5.3: MLP performance on NMSE with increasing hidden nodes

Figure 5.4: Classification accuracy of FLNN network trained with different algorithms in Balance Scale dataset
Figure 5.5: Classification accuracy of FLNN network trained with different algorithms in Haberman’s Survival dataset

Figure 5.6: Classification accuracy of FLNN network trained with different algorithms in the mammographic mass dataset

Figure 5.7: Classification accuracy of FLNN network trained with different algorithms in the Breast Cancer Wisconsin dataset

Figure 5.8: Classification accuracy of FLNN network trained with different algorithms in Credit Approval dataset

Figure 5.9: Classification accuracy of FLNN network trained with different algorithms in ILPD dataset

Figure 5.10: Classification accuracy of FLNN network trained with different algorithms in Iris dataset

Figure 5.11: Classification accuracy of FLNN network trained with different algorithms in BUPA dataset

Figure 5.12: Classification accuracy of FLNN network trained with different algorithms in Lymphographic dataset

Figure 5.13: Classification accuracy of FLNN network trained with different algorithms in MONK’s problems dataset

Figure 5.14: Classification accuracy of FLNN network trained with different algorithms in Page Blocks Classification dataset

Figure 5.15: Classification accuracy of FLNN network trained with different algorithms in Zoo dataset

Figure 5.16: Classification accuracy of FLNN network trained with different algorithms in Pima Indians diabetes dataset

Figure 5.17: Classification accuracy of FLNN network trained with different algorithms in Thyroid disease dataset
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w$</td>
<td></td>
<td>Connection weight</td>
</tr>
<tr>
<td>$f(x), \sigma$</td>
<td></td>
<td>Activation function</td>
</tr>
<tr>
<td>$D$</td>
<td></td>
<td>Dimensions</td>
</tr>
<tr>
<td>$E$</td>
<td></td>
<td>Network error</td>
</tr>
<tr>
<td>$x$</td>
<td></td>
<td>The input vector</td>
</tr>
<tr>
<td>$Y$</td>
<td></td>
<td>Network output</td>
</tr>
<tr>
<td>$\hat{Y}$</td>
<td></td>
<td>Target output/ Desired output</td>
</tr>
<tr>
<td>$1 \over 1 + e^{-x}$</td>
<td></td>
<td>Logistic sigmoid function</td>
</tr>
<tr>
<td>tanh(x)</td>
<td></td>
<td>Hyperbolic tangent</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td></td>
<td>Learning rate</td>
</tr>
<tr>
<td>$\mu$</td>
<td></td>
<td>Momentum coefficient</td>
</tr>
<tr>
<td>$\Pi$</td>
<td></td>
<td>product term</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td></td>
<td>summation function</td>
</tr>
<tr>
<td>$LB$</td>
<td></td>
<td>Lower bound</td>
</tr>
<tr>
<td>$UB$</td>
<td></td>
<td>Upper bound</td>
</tr>
<tr>
<td>$SN$</td>
<td></td>
<td>Solutions number</td>
</tr>
<tr>
<td>$TP$</td>
<td></td>
<td>True Positives</td>
</tr>
<tr>
<td>$TN$</td>
<td></td>
<td>True Negatives</td>
</tr>
<tr>
<td>$FP$</td>
<td></td>
<td>False Positives</td>
</tr>
<tr>
<td>$FN$</td>
<td></td>
<td>False Negatives</td>
</tr>
<tr>
<td>$limit$</td>
<td></td>
<td>Local search abandonment limit</td>
</tr>
<tr>
<td>$\beta_0$</td>
<td></td>
<td>Initial attractiveness</td>
</tr>
<tr>
<td>$\gamma$</td>
<td></td>
<td>Light absorption coefficient</td>
</tr>
<tr>
<td>$ABC$</td>
<td></td>
<td>Artificial Bee Colony</td>
</tr>
<tr>
<td>$ANNs$</td>
<td></td>
<td>Artificial Neural Networks</td>
</tr>
<tr>
<td>$BP$</td>
<td></td>
<td>Backpropagation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>Differential evolution</td>
<td></td>
</tr>
<tr>
<td>FA</td>
<td>Firefly Algorithm</td>
<td></td>
</tr>
<tr>
<td>FLNN</td>
<td>Functional Link Neural Network</td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
<td></td>
</tr>
<tr>
<td>HONNs</td>
<td>Higher-order Neural Networks</td>
<td></td>
</tr>
<tr>
<td>MAE</td>
<td>Mean absolute error</td>
<td></td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
<td></td>
</tr>
<tr>
<td>MABC</td>
<td>Modified Artificial Bee Colony</td>
<td></td>
</tr>
<tr>
<td>MBF</td>
<td>Modified Bee-Firefly</td>
<td></td>
</tr>
<tr>
<td>MSE</td>
<td>Mean square error</td>
<td></td>
</tr>
<tr>
<td>NMSE</td>
<td>Normalized mean squared error</td>
<td></td>
</tr>
<tr>
<td>PSNN</td>
<td>Pi-Sigma Neural Network</td>
<td></td>
</tr>
<tr>
<td>PSO</td>
<td>Particle Swarm Optimization</td>
<td></td>
</tr>
<tr>
<td>RMSE</td>
<td>Root mean squared error</td>
<td></td>
</tr>
<tr>
<td>RPNN</td>
<td>Ridge Polynomial Neural Network</td>
<td></td>
</tr>
<tr>
<td>SI</td>
<td>Swarm Intelligence</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
<td></td>
</tr>
</tbody>
</table>
LIST OF PUBLICATIONS

Journals:


(iv) Yana Mazwin Mohmad Hassim and Rozaida Ghazali, “An Improved Functional Link Neural Network Learning Using Artificial Bee Colony Optimisation for Time Series Prediction”, Int. J. Business Intelligence and Data Mining, ISSN(online) 1743-8195 ISSN 1743-8187, Vol. 8, No. 4, 2013 (pp.307–318).
Proceedings:

(i) Yana Mazwin Mohmad Hassim and Rozaida Ghazali (2013). Solving a classification task using Functional Link Neural Networks with modified Artificial Bee Colony. *2013 Ninth International Conference on Natural Computation (ICNC)*, IEEE.


CHAPTER 1

INTRODUCTION

1.1 Background of Research

Classification is a process of categorizing objects which mostly involved in decision-making activity. It is one of the most frequent studies in the area of Artificial Neural Networks (ANNs) (Zhang, 2000; Misra and Dehuri, 2007; Chen et al., 2011; Al-Jarrah and Arafat, 2015; Mason, 2015). Classification is defined as a task of identifying and assigning an object to a predefined group based on a number of observed attributes related to that object. Solving a classification problem required a set of example records which called a training set that needs to be presented to the ANNs network so that the network can “learn” the pattern. Each record in the training set normally consists of several attributes. Within all the attributes in the training set, there is one attribute known as the classifying attribute mainly used to indicate the class to which each record belong. The ANNs builds a classification model (classifier) based on the functional relationship between the classifying attribute with the other attributes of the record in the training set. The classifier is then used to classify new record or unseen record (out of sample record). Several examples of real world application on neural classification tasks include bankruptcy prediction, handwriting recognition, speech recognition, product inspection, fault detection and medical diagnosis.
In the field of Machine Learning (ML), ANNs is known as a family of statistical learning algorithm inspired by the way of human brain process information (Michalski et al., 2013). They are data driven self-adaptive method, in which they can change their structure based on external or internal information that flows through the network to model complex relationships between inputs and outputs. The recent vast research activities in neural classification have established that ANNs are a promising tool and have been widely applied to various real world classification task especially in industry, business and science (Zhang, 2000; Liao and Wen, 2007; Hema et al., 2008; Al-Shayea, 2011; Ghazali et al., 2011; Yeatman et al., 2014; Al-Jarrah and Arafat, 2015). One of the best-known types of ANNs is the Multilayer Perceptron (MLP). The application of MLP in classification tasks has shown better performance in comparison to the statistical method due to their nonlinear nature and training capability (Murtagh, 1991; Walde et al., 2004; Silva et al., 2008; Zabidi et al., 2010).

Despite the development of various types of ANNs, this research work examines the ability of Higher Order Neural Networks (HONNs) which focusing on Functional Link Neural Network (FLNN) for solving classification problems. FLNN is a class of HONNs that can perform nonlinear mapping by using only single layer of units (Giles and Maxwell, 1987). HONNs utilize higher order terms to expand their input space into higher dimensional space to achieve nonlinear separability which reduced the complexity of the network. The single layer property of FLNN also makes the learning algorithm used in the network less complicated as compared to other standard feedforward neural networks (Misra and Dehuri, 2007).

In neural classifications, network training is essential in order to build a classification model. In this procedure, the network model is trained on the basis of data (training data) facilitated by the error correcting learning method where the connection weights of each neuron are adjusted until the network error reaches an acceptable minimum value (Samarasinghe, 2006). The purpose of network training is to find an optimal connection weights values that can minimize the network error and this method can also be viewed as an optimization task (Karaboga and Basturk, 2007). The most widely used error correcting learning method for network training is the Backpropagation (BP) learning algorithm (Rubio et al., 2011; Rojas, 2013; Eberhart, 2014). The BP-learning algorithm is a type of gradient descent optimization method. However, network learning algorithm which is based on
gradient descent optimization method has several drawbacks that can affect the performance of the neural network model (Haring et al., 1997; Sierra et al., 2001; Abu-Mahfouz, 2005; Dehuri et al., 2008). Therefore, this research emphasizes on improving the FLNN network learning algorithm for classification task by using Swarm Intelligence (SI) optimization techniques to overcome such drawback.

Neural networks and swarm intelligence methodologies have been proven effective in solving certain classes of problems. Neural Networks are good at mapping the input vector to outputs while Swarm Intelligence (SI) is very good at optimization task (Kennedy and Eberhart, 2001). It is expected in this work that, the optimization of network learning by using the SI method may overcome the standard gradient-based learning algorithm drawbacks. SI method is concerned with the design of intelligent multi-agent systems inspired by the collective behavior of social insects such as ant or bees as well as an animal society such as a flock birds or a school of fish (Blum and Li, 2008). Numerous well-known methods of SI are; Particle Swarm Optimization (PSO), Artificial Bee Colony Algorithm (ABC), Differential Evolution (DE), Evolutionary Algorithm (EA) and recently Firefly Algorithm (FA). ABC algorithm is known to have good exploration capabilities for their global search strategy (Zhu and Kwong, 2010). On the other hand, FA is renowned for having good ability in local search exploitation strategy (Fister et al., 2013; Bacanin and Tuba, 2014). Hence, this research work came with the interest to integrate ABC and FA algorithms so that the benefit from the advantage of both methods can be utilized to recover the gradient based learning drawbacks for training the FLNN network model.

1.2 Problem Statements

ANNs have been successfully applied to a variety of real-world classification and have been increasingly used as a promising modeling tool over traditional statistical approaches (Zhang, 2000; Cherkassky et al., 2012; Sethi and Jain, 2014). This is related to the fact that they are data-driven self-adaptive methods as they can adjust themselves to the data without any explicit specification of the functional or distributional form (Cybenko, 1989; Hornik, 1991; Zhang, 2000; Cherkassky et al., 2012). Also, the ANNs are universal functional approximators where they can
approximate any function as well as estimating posterior probabilities which provide the basis for establishing classification rule (Richard and Lippmann, 1991; Zhang, 2000).

One of the best-known types of ANNs is the Multilayer Perceptron (MLP). The MLP is a feedforward multilayered structured model and has been successfully applied in a broad class of classification tasks (Murtagh, 1991; Zhang et al., 1999; Zhang, 2000; Silva et al., 2008; Zabidi et al., 2010). The multilayer structure of MLP gives the network the ability to map both linear and non-linear relationship with a condition that, they are provided with a sufficient number of nodes and layers. Beside of it successfulness, MLP adopts computationally intensive training algorithms and introduces many local minima in the error surface (Yu, 2005; Parappa and Singh, 2013). In order to achieve good classification ability, MLP also requires a rather large amount of available measures; as to determine the appropriate number of neurons in layers and to fix a suitable number of hidden layers in its structure. Furthermore, as the number of hidden layers and nodes in MLP structure increases, the network architecture becomes more complex and training the network becomes more challenging. MLP networks with a large number of hidden layers and nodes also mean that they are having a large number of learning parameters. This may also affect the MLP performance as the network with a large number of learning parameters tend to memorize the training data which may lead to over-fitting and poor generalization (Lawrence and Giles, 2000).

To overcome the MLP drawbacks, a Functional Link Neural Network (FLNN) is considered in this study which has the ability to perform nonlinear mapping by using only single layer units (Pao, 1989). FLNN is a type HONNs that utilized a higher combination of it inputs and has certain advantages over MLP. FLNN remove the need of hidden layers and hidden nodes by utilizing a higher order term to expand it input space into higher dimensional space within it single layer units. This simple network architecture reduced the numbers of required learning parameters and thus reduces the learning complexity during network training (Misra and Dehuri, 2007; Ghazali et al., 2008). The standard method for training the FLNN network is using the Backpropagation (BP) learning algorithm (Haring and Kok, 1995; Haring et al., 1997; Sierra et al., 2001; Abu-Mahfouz, 2005; Misra and Dehuri, 2007; Dehuri et al., 2008). The BP-learning is the most well-known method for
network training and widely utilized in conjunction with gradient descent optimization method.

Although BP-learning is the most used algorithm for training the FLNN network, the algorithm however, has several limitations which affect the performance of the FLNN with BP-learning (FLNN-BP) model. FLNN-BP tends to easily get trapped in local minima especially for those with highly non-linearly separable classification problems which is an inherent problem that exists in the BP-learning algorithm. The employment of BP-learning algorithm as learning algorithm has made FLNN-BP model strictly depends on the shape of the error surface and since a common error surface may have many local minima and multimodal, this has typically made the FLNN-BP model prone to stuck in some local minima when moving along the error surface during the training phase. In addition, FLNN-BP model also very dependent on the choices of initial values of the weights as well as the parameters of the algorithm such as the learning rate and momentum which make it not very easy to meet the desired convergence criterion during the network training. Ismail, (2001) detailed that with proper learning algorithm FLNN may possess high learning capabilities that require less memory in terms of weights and nodes when compared to the MLP for similar performance levels. Therefore, further investigations to improve learning algorithm in FLNN are still desired.

Considering the limitations of gradient-based BP-learning algorithm, the intention of utilizing the use of Swarm Intelligence (SI) method which is very good at optimization task (Kennedy and Eberhart, 2001), is considered in this work. The optimization of neural network learning by using SI method may overcome the standard gradient-based learning algorithm drawbacks. The ABC algorithm (Karaboga, 2005) is a family of SI method which is known to have good exploration capabilities for their global search strategy (Zhu and Kwong, 2010). Karaboga and Basturk, (2007) in their work has compared the performance of ABC with Genetic Algorithm (GA), DE and PSO on unconstrained problems and established that ABC is a simple and robust optimization algorithm. As in classification task in data mining, ABC algorithm also provides a good performance in gathering data into classes (Karaboga and Ozturk, 2011). Meanwhile, FA is one of the recent SI methods developed by Yang (2008) and has been successfully applied in solving many optimizations and classification problems. FA is inspired by the flashing behavior of fireflies which is renowned for having good ability in local search exploitation
strategy (Fister et al., 2013; Bacanin and Tuba, 2014). In SI optimization method, both exploration and exploitation are necessary and contradict to each other. In order to achieve good performances, the two abilities should be well balanced. Thus, in conjunction with the benefits of both ABC algorithm (exploration) and firefly algorithm (exploitation), this research intends to propose an improved learning algorithm named: Modified Bee-Firefly (MBF) algorithm to be used for training the FLNN network for the task of classification.

1.3 Aim and Objective of Research

The aim of the research is to design and implement an improved learning algorithm by using an Modified Bee-Firefly algorithm for training the FLNN for the task of classification. In order to achieve the research aim, a few objectives have been set:

(i) to propose an improved learning algorithm for FLNN using ABC optimization algorithm for classification tasks
(ii) to enhance (i) by removing random dimension selection in order to deal with issues in tuning the FLNN weights parameters
(iii) to incorporate Firefly algorithm with (ii) in order to improve its classification accuracy
(iv) to evaluate and compare the out-of-sample performance (iii) with ordinary FLNN and ordinary MLP based on classification performance

1.4 Scope of Research

The study will focus on construction, implementation, and testing the FLNN network trained with Modified Bee-Firefly (FLNN-MBF) for the task of classification. The FLNN input enhancements architecture will be based on tensor model structure. The results later are compared with the Multilayer perceptron (MLP) and Functional Link Neural Network (FLNN) both trained with Backpropagation (BP) learning algorithm respectively addressed as MLP-BP and FLNN-BP models. All networks will be tested on several classification benchmark data and evaluated based on their classification performance.
1.5 **Significance of Research**

FLNN-MBF is important to be used in this research for the task of classification. This research may help in introducing an improved learning algorithm for training the FLNN network that may contribute to better classification performance.

1.6 **Chapter Summary**

There have been many ANNs applications and techniques developed for the task of classifications in the past. Several methods related to ANNs particularly have been investigated and carried out. However, the ordinary feedforward ANNs, which is the MLP, is prone to easily get stuck into local minima. Limitation such as network complexity of the models also makes the existing system less desirable for some applications. Thus, to overcome the drawbacks, this research focuses on utilizing FLNN model for the task of classification. FLNN has a simple architecture that reduces the learning complexity as compared to other feedforward networks. However, during network training FLNN tends to easily get trapped in local minima especially for those non-linearly separable classification problems, which is an inherent problem that exists in the BP-learning algorithm. Thus, an improved learning algorithm for FLNN is proposed in this study. The next chapter will discuss the literature on the existing ANNs and FLNN approaches related to the classification task, the network structure, and the learning algorithm being used for network training.

1.7 **Organization of Thesis**

The rest of the chapters in this thesis are organized as follows: Chapter 2 discusses the relevance background information regarding ANNs and HONNs in classification task in the following order: (1) the overview of ANNs in term of network architecture, (2) the HONNs, the FLNN structure and its advantages, (3) Standard Learning Algorithm for FLNN training, In this chapter, it is also highlighted the drawback of current learning algorithm and argument for an improved learning algorithm for FLNN in classification task. Chapter 3 presents the research framework
of this study and the proposed Modified Bee-Firefly learning algorithm and the algorithms. Chapter 4 discusses the implementation of proposed Modified Bee-Firefly learning algorithm on a classification task and the results of the experiments. Finally, chapter 5 concludes the research and provides suggestions for future works.
CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Artificial Neural Networks (ANNs) are a computational model inspired by biological neural networks that provide a general class of nonlinear models. The term ANNs is used in a broad sense which groups together numerous models of neural system related to their structure and learning method. Different types of ANNs models may have different strength and abilities tailored to their application. Some important criterions of ANNs are the ability to adapt and learn from their environment and the ability to approximate complicated mappings. These intelligence mechanisms have given them an advantage over other conventional mapping models. Despite the development of various types of ANNs, Higher Order Neural Networks (HONNs) have become a renowned method to overcome some of the limitations that exist in ANNs (Dehuri and Cho, 2010). In order to be more certain of this field, this chapter provides the theoretical perspectives of ANNs and HONNs which partly reveal the applications and techniques that have been utilized in the area of classification. This chapter also discussed related works on swarm optimization that in line with the problem under study, the optimization of weights in network learning.
2.2 Classification Task

Classification is the task of assigning objects to one of the several predefined categories and mostly involved in decision-making activity. Some examples of classification tasks include; assigning a given email into "spam" or "non-spam" classes and categorizing cells as malignant or benign based on the results of MRI scans. A classification problem occurs when an object needs to be assigned to a predefined group or class based on a number of observed attributes related to that object (Zhang, 2000). Bandyopadhyay et al. (2004), viewed a classification as an integral part of pattern recognition which involved dealing with a problem of generating appropriate class boundaries that can successfully distinguish various classes in the feature space. A system or classification models that implement classification are known as classifiers. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm that maps input data to its class label.

![Figure 2.1: The input-output mapping of a classifier](image)

Figure 2.1 presents a classification as a task of mapping input attribute $X$ into its class label of $Y$. The input data for a classification task is a collection of records. Each record, also known as an instance or example is characterized by a tuple $\langle X:Y \rangle$, where $X$ is the attribute set and $Y$ is a special attribute, designated as the class label also known as category or target attribute. A tuples $X$ is represented by $n$-dimensional attribute vector, where $X = \{x_1, x_2, ..., x_n\}$, depicting $n$ measurements made on tuple from $n$ database attributes.

In classification problems, a classifier is a systematic method to build classification models from an input data set (Tan et al., 2005). Each method employs a learning algorithm to identify a model that best fits the relationship between the attribute set and the class label of the input data. The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of
records it has not previously seen before. A general approach to solve a classification problem is by presenting a set of example pairs are known as training set which consists of an input attributes and class label to the classifier. The aim is to build a functional relationship between the class label (output) and attributes (input) to capture patterns in data through an iterative learning process devised by a learning algorithm (Rani, 2011). The learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples (test set).

Several methods have been taken toward a classification task which can be categorized into statistical methods and machine learning methods. Statistical methods are generally characterized by having an explicit underlying probability model, which provides a probability of being in each class rather than simply a classification (Michie et al., 1994). Traditional statistical classification methods usually try to find a clear cut boundary to divide the pattern space into some classification regions based on some predefined criterion (Li et al., 2002). Statistical pattern classification approach such as discriminant analysis is built based on Bayesian decision theory (Duda et al., 1995). In these methods, an underlying probability model must be assumed in order to calculate the posterior probability upon which the classification decision is performed.

In traditional statistical pattern classification method, the classifiers construct the model of class-conditional densities and build their decision based on the posteriors probability which is computed using the class-conditional likelihoods (Valous and Sun, 2012). Likelihoods are assumed to either come from a given probability density family, or a mixture of such densities or be written in a completely non-parametric way (Mahmoud et al., 2004). Bayes decision theory then allows choosing the class that minimizes the decision risk. The parameters of the densities are estimated to maximize the likelihood of the given sample for that particular class.

Major limitations of the statistical models are that they work well only when the underlying assumptions are satisfied (Zhang, 2000). Users must have a good knowledge of both data properties and model capabilities before the models can be successfully applied (Dehuri and Cho, 2010). The effectiveness of these methods depends to a large extent on various assumptions or conditions under which the models are developed (Donald et al., 1994). The users also must have a good
knowledge of both data properties and model capabilities before the models can be successfully applied (Dehuri and Cho, 2010).

In order to avoid limitations of classical statistical classification models, some researchers have turned their attention to machine learning methods. Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning aims to generate classifying expressions simple enough to be understood easily by the human (Donald et al., 1994). These methods have emerged as an important tool for classification as they are often capable of solving problems, which are not easily solvable by statistics models (Popelka et al., 2012). In machine learning, classification is considered an instance of supervised learning where a training set of correctly identified observations is available. Several prominent supervised machine learning methods include; decision trees, Support Vector Machine (SVM), and Artificial Neural Network (ANNs).

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree (Zhang and Lee, 2003). Learned trees can also be represented as a set of if-then rules to improve human readability. The goal of decision tree learning is to create a model that predicts the value of a target variable based on several input variables. Decision tree learning methods are among the most popular inductive inference algorithms and have been successfully implemented in a broad range of tasks. However, the method can create over-complex trees that, does not generalize well from the training data. Also, there are certain concepts that are hard to learn and difficult to express when using decision trees representation such as XOR, parity or multiplexer problems. In such cases, the decision tree may become prohibitively large.

Meanwhile, Support Vector Machine (SVM) is another supervised machine learning models. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible (Lei et al., 2012). New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. SVM constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (Cortes and Vapnik, 1995). However, an issue regarding the SVM is that the model only directly
applicable for two-class classification tasks. Therefore, algorithms that can reduce the multi-class task into several binary problems are required for the SVM model.

Another widely used machine learning method is the artificial neural network (ANNs) models. ANNs is inspired by the structure and functional aspects of biological neural networks. The models are presented as systems of interconnected "neurons" which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making the neural nets adaptive to inputs and capable of learning. ANNs are non-linear statistical data modeling tools and are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

According to Zhang (2000), ANNs appears as a promising alternative methodology to various conventional classification methods based on three theoretical aspects; first, they are data-driven self-adaptive methods in which the system can adjust themselves to the data without any explicit specification of functional or distributional form to build the model. Second, they are universal functional approximators that can approximate any function with arbitrary accuracy and have more general and flexible functional forms than the traditional statistical method (Cybenko, 1989; Hornik, 1991). Since a classification task seeks a functional relationship between group membership and the attributes of the object, accurate identification of this function is important and neural networks method is well suited for such tasks (Zhang, 2000). And third, neural networks are a type of nonlinear models, which makes them flexible in modeling real world complex relationships and therefore, provides a robust foundation for establishing classification rules (Richard and Lippmann, 1991).

There are numbers of performance comparisons between neural networks and other conventional classifiers made by numerous studies (Curram and Mingers, 1994; Michie et al., 1994; Zhang, 2000; Misra and Dehuri, 2007). However, while significant progress has been achieved in classification related areas of neural networks, a number of issues in applying neural networks still remain, and have not been solved successfully or completely (Zhang, 2000). Issues such as network computational cost and architectural complexity can have a significant impact which may affect the performance of neural networks. Also, the issue regarding network learning as the network is prone to the problems of local minima trapping, especially
when involving with a highly complex nonlinear problem is still remain actively explored (Chen and Leung, 2004; Ghazali, 2007).

2.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) is a computational information processing models that are inspired by human neural systems. ANNs mimic the structure of the biological central nervous system that could perform “Intelligent” task similar to those performed by the human brain. According to Haykin (2004), the ANNs resembles the brain in two ways; 1) the knowledge is required by the network through a learning process and 2) the interneuron connection strength known as the synaptic weights are used to store the knowledge.

ANNs are generally presented as a system of interconnected neurons which can compute values from inputs and incrementally learn from their environment (data) to capture patterns in data. In the computational form of ANNs, neurons are known as processing units while the connection strengths between neurons are presented as a set of numerical parameters known as adaptive weights. The processing units receive and accumulate weighted input signals and further processes these signals before transporting them to other units. The adaptive weights are tuned by a learning algorithm which is activated during training so that the system can build a model based on their inputs (Bishop, 2006; Karayiannis and Venetsanopoulos, 2013). ANNs are able to learn from data. They are also recognized as data driven self-adaptive models as they can adjust themselves based on external or internal information that flows through the network which giving them the ability to model complex relationships between inputs and outputs or to capture patterns in data (Zhang, 2000).

ANNs have powerful pattern classification and pattern recognition capabilities through learning and generalize from experience (Haykin, 2004). They imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. According to Zaknich (2003), ANNs can often provide suitable solutions for problems that are complex, nonlinear, high-dimensional, noisy, imprecise, imperfect, poorly understood by physical and statistical models, and lack of clearly stated
mathematical solution or algorithm. Until recent, the ANNs have been reported to be successfully apply in many engineering and scientific problems, which include; prediction (Abdual-Salam et al., 2010; Guresen et al., 2011; Zadpoor et al., 2013; Ghazali et al., 2014; Meng et al., 2014), function approximation (Anastassiou, 2011; Zainuddin and Pauline, 2011), pattern recognition and classification (Abu-Mahfouz, 2005; Artyomov and Yadid-Pecht, 2005; Silva et al., 2008; Basu et al., 2010; Zabidi et al., 2010; Al-Shayea, 2011), system identification (Abbas, 2009; Chi-Hsu and Kun-Neng, 2009; Emrani et al., 2010; Patra and Bornand, 2010) and medical analysis (Chien-Cheng et al., 2005; Hema et al., 2008; Hoyt et al., 2010; Liu et al., 2010; Amato et al., 2013). They have attracted much attention due to their adaptive nature and have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming.

2.3.1 Components of Artificial Neural Networks

The idea of ANNs is to model a neuron by building interconnected networks which devised by learning algorithm. In biological wise, the neuron is composed of three main elements, known as dendrites, cell body, and axon. The dendrites received input signals which are weighted by connection strength and channel them to a cell body. A cell body accumulates the weighted input signals and further processes these signals. The axon transmits the output signal to other neurons that are connected to it. The same mechanism and function exist in ANNs. Figure 2.2 shows a neuron and its representation both in biological neuron and an artificial neuron model.
In ANNs, the neuron is the most fundamental element which makes up a neural pathway in its architecture. The neuron is also sometimes called as unit, node, perceptron or processing unit. It received input signals either from an external input or from some other units. In neuron model, input signals are received accumulated or summed ($\Sigma$) and processed further [$f(\Sigma)$] to produce an output. As neurons are interconnected to each other, this output signal becomes an input signal to other neuron and transmitted from one neuron to another neuron through connection between them until the process is completed and a result is generated. Each connection between neurons has different synaptic strength or connection strengths (weights), which must undergo adaptation during the learning process where the neuron learns patterns from the information it received.

According to Sumathi and Paneerselvam (2010) the computational of ANNs consist of six major processing components; weighting factor, summation function, activation function, output function, error function and learning function. Figure 2.3 shows the computational elements in the artificial neuron.
Figure 2.3: Computational elements in artificial neuron

**Weighting factor:** In ANNs, the neurons are connected by links and each link has its own connection strength represented in numerical coefficient (Negnevitsky, 2005). These connections strengths are called synaptic weights and are the most important factor in determining the intensity of input signal received by neuron. They express the strength or the significance of each neuron input in the network. These weights can be adapted in response to various training sets and according to a network’s specific topology or a learning rule (Sumathi and Paneerselvam, 2010). ANNs “learn” through repeated adjustment of these weights until the network reaches a steady state where there are no further significant changes in the synaptic weights.

**Summation function:** The initial step in the operation of processing units is the summation of all weighted inputs. As showed as in Figure 2.3, the inputs and the corresponding weights are vectors that can be represented as \((x_1, x_2, \ldots, x_n)\) and \((w_1, w_2, \ldots, w_n)\). The input signals are the multiplication of each input with their corresponding weight which can be represented as \(w_1x_1 = w_1 \times x_1\), \(w_2x_2 = w_2 \times x_2\) and \(w_nx_n = w_n \times x_n\). The weighted input signals are summed to obtain a linear combination of the input signals \(w_1x_1 + w_2x_2 + \ldots + w_nx_n\) before being processed further by the activation function.

**Activation function:** The result of the summation function is transformed to a working output signal through an algorithmic process known as the activation function. The activation functions are needed in order to introduce nonlinearity into the network and to simplify the network. In the activation function, the summation total can be compared with some threshold to determine the neural output. If the sum
is larger than the threshold value, the processing element generates a signal. If the sum of the input and weight products is lower than the threshold, no signal (or some inhibitory signal) is generated. The activation function employed for ANNs is generally non-linear. A nonlinear activation function limits the amplitude of the output neuron. Typically, activation functions have a “squashing” effect which prevents accelerating growth throughout the network (Cybenko, 1989; Sumathi and Paneerselvam, 2010).

**Output function and error function:** The output from the output function is directly equivalent to the activation function's result. During the network training, the output for each training cycle is known as network output. In most learning networks, the difference between the network output and the desired output is calculated and denoted as the error. This error term is often called the network error. During neural learning, the network error is propagated backward to the previous layer. This back-propagated value, after being scaled by the learning function, is multiplied against each of the incoming connection weights to modify them before the next learning cycle.

**Learning function:** The purpose of the learning function is to modify or update the variable connection weights on the inputs of each processing element according to some neural based algorithm. The process of updating these weights connections is also known as the adaption mode or training mode.

### 2.3.2 Different Models of Neural Networks

ANNs can be classified based on their architecture and the method of training. According to Zaknich (2003) the network topology (architecture) of ANNs can be different for each different type of neural network models in which different types of models have different strength and abilities particular to their application. The neurons in ANNs can be interconnected in many different possible topologies include single layer network topologies and multilayer network topologies. A single layer network is a type of ANNs that has both the input layer and the output layer, whereas a multilayer network can have one or more hidden layers in between the input layer and the output layer.
Generally, there are three types of ANNs architecture: feedforward neural networks, feedback neural networks, and self-organizing neural networks (Deb and Dixit, 2008). In feedforward network, the input signal travel through the network in forward direction, while in feedback network the outputs of some neurons are fed back to some neurons in layers before them, thus the input signal can travel both forward and backward directions. Self-organizing neural networks are distinct from the other two networks. It consists of neurons arranged in the form of a low dimensional grid where each input is connected related to all the output neurons. This type of network employs competitive learning method where each output neurons compete amongst themselves to be activated and the input signals are transmitted by the winning neurons. Among these architectures, the most common and widely used are the feedforward neural networks (Samarasinghe, 2006; Deb and Dixit, 2008; Sumathi and Paneerselvam, 2010).

2.3.3 Feedforward Neural Networks

The feedforward neural networks are the most well-known and commonly used models of ANNs (Epitropakis *et al.*, 2006; Manohar and Reddy, 2008; Chen *et al.*, 2013; Benzer and Benzer, 2015). The Feedforward neural networks structure is organized in a layered architecture, with each layer comprising one or more neurons. Each neuron is connected to one or more other neurons and the connections are only between consecutive layers, all in the same direction. All feedforward neural networks have input layer and output layer. Also, these networks can have numbers of layers and neurons per layer in between the input layer and output layer. These middle layers have no connection with the external world and hence are called hidden layers.

In feedforward network architecture, the input signals are propagated through the network in a forward direction from one layer to the next until they arrive at the outputs. During normal operation; when it serves as a classifier, there are no cycles or loops in the network. A network with only an input layer and an output layer is called a single layer network or a single layer perceptron. While a network with one or more hidden layers in the middle of the input layer and the output layer is known as multilayer feedforward network or multilayer perceptron. The single layer has
limited mapping ability and only capable of learning linearly separable patterns. On the other hand, multilayer perceptron had far greater processing power than the single layer perceptron and is capable of learning nonlinear decision surfaces.

2.3.4 Multilayer Perceptron

Multilayer layer perceptron (MLP) is a feedforward neural network which is formed by a collection of neurons that are connected by their associated weights in a hierarchical structure. The network is also known as first order neural network (Giles et al., 1988). The MLP networks are capable of generating a complex mapping between the input and the output and are capable of solving highly nonlinear problems (Cybenko, 1989). Minsky and Papert (1969) also stated that the MLP network is the most popular machine learning solution as it has the ability to build nonlinear decision surfaces in high dimensional problem space and is capable of providing more computational potential as compared to a single-layer perceptron network. Due to this capability, MLP has been successfully tested in many applications including the classification tasks (Murtagh, 1991; Widrow et al., 1994; Abu-Mahfouz, 2005; Bishop, 2006; Silva et al., 2008; Zabidi et al., 2010).

The MLP structure consists of at least one hidden layer in between input and output layers. The network propagates the input signals in a forward direction. The function of hidden neurons in it architecture is to intervene between the external inputs and the network output in some useful manner (Ghazali, 2007). In MLP structure, as the signals are propagated forward, the output signals from one layer act as an input signals to the next subsequent layer until the output of the network is produced at the output layer (Sumathi and Paneerselvam, 2010). Figure 2.4 illustrates the layout of the basic structure of MLP.
As shown in Figure 2.4, the MLP computes the network output, $Y$ according to the following equation:

$$
Y = \sigma \left( \sum_{j=1}^{J} W_{jk} \sigma \left( \sum_{i=1}^{N} W_{ij} X_i + W_{bj} \right) + W_{bk} \right)
$$

(2.1)

where $X_i$ denotes the input value, $W_{ij}$ is the weights from the input layer to the hidden layer, $W_{jk}$ is the weights from the hidden layer to the output layer, $W_{bj}$ is a bias for hidden neuron, $\sigma$ is a nonlinear activation function, and $Y$ is the network output. The activation function $\sigma$, acts as a squashing function that prevents accelerating growth throughout the network (Cybenko, 1989). The main activation functions used in most MLP applications are the logistic sigmoid $\sigma = \frac{1}{1+e^{-x}}$ and hyperbolic tangent $\sigma = \tanh(x)$. These functions are mostly used because they are substantially outperforms the other activation functions, thus allowing the MLP to model well both strongly and slightly nonlinear mappings (Husaini et al., 2011).

As the MLP network utilize a multilayer structure of neurons in its architecture, the network has the ability to map both linear and non-linear relationship in a condition that, they are provided with a sufficient number of nodes and layers and having a sigmoid function as the nonlinear activation function. However, this complex architecture of MLP tends to introduce many local minima in the error surface (Yu, 2005) which caused the network to make use computationally...
expensive training algorithms and prone to get stuck in local minima. Another
drawback of MLP is that the network needs a lot of training examples to be able to
capture patterns in data which may affect the training process as this can lead to slow
processing and unstable network behavior (Zaknich, 2003). Also, in order to achieve
good classification ability, MLP requires a rather large amount of available measures
such as fixing an appropriate number of neurons in each layer and determining a
suitable number of hidden layers. As the number of hidden layers and nodes in MLP
structure increases, the network architecture becomes more complex and training the
network becomes more challenging. MLP networks with a large number of hidden
layers and nodes also mean that they are having a large number of learning
parameters. This will affect the MLP performance as the network with a large
number of learning parameters tend to memorize the training data which may lead to
over-fitting and poor generalization (Lawrence and Giles, 2000).

2.4 Higher Order Neural Networks

Higher-order Neural Networks (HONNs) expand standard feedforward neural
networks structure by including enhanced nodes at the input layer that provide the
network with a complete understanding of the input patterns and their relations
(Epitropakis et al., 2006). The aim of HONNs is to replace the hidden neurons found
in the first order neural network or commonly known as MLP to reduce the
complexity of their structure (Giles and Maxwell, 1987). Therefore, HONNs
distinguish themselves from ordinary feedforward networks by the presence of high
order terms in the network. In ordinary feedforward networks architecture, neural
inputs are combined using the summing operation only. However, in HONNs
architecture, the network combines input of several neurons in another way than
summation, usually combined with multiplication operation (Hoogerheide, 2006).
The number of inputs that are combined in a non-linear way is the ‘order’ of the
network.

The order of the network is defined by the highest order of the connections in
it. The order of the networks in HONNs is recognized as higher order terms. These
higher order terms are capable of increasing the information capacity in HONNs
which gives the ability to provide a solution with fewer neurons (Giles and Maxwell,
1987; Pao and Takefuji, 1992). The high order terms employed in the network also facilitate in constructing a nonlinear mapping that provides better classification capability as compared to the first order neural network (Guler and Sahin, 1994).

In order to overcome the limitations of conventional ANNs, some researchers have turned their attention to HONNs models (Chang and Cheung, 1992; Yatsuki and Miyajima, 2000; Ming et al., 2002; Shuxiang and Ling, 2008; Fallahnezhad et al., 2011). This is due to its capability to simulate higher frequency, higher-order nonlinear data, and consequently, provide superior simulations compared to the ordinary feedforward networks (Kanaoka et al., 1992; Ming et al., 2002). Several applications that have gain advantage by using HONNs include; prediction (Ming et al., 2002; John et al., 2006; Ghazali, 2007; Husssain et al., 2009; Husaini et al., 2011), function approximation (Ghosh and Shin, 1995; Xu and Zhang, 2002), classification and pattern recognition (Ghosh and Shin, 1995; Kosmatopoulos et al., 1995; Dehuri et al., 2008; Jia-Wei and Jun, 2009).

Even though most neural networks models share a common goal in performing input-output mapping, their ability in handling different type of problems are distinguished by the different type of network architecture. For some tasks, HONNs are needed as ordinary feedforward network like MLP cannot escape from the problems of local minima trapping especially when involving a highly complex nonlinear problem (Chen and Leung, 2004; Ghazali, 2007).

2.4.1 Properties of HONNs

Most ordinary feedforward networks are first order neural networks. Neurons in ordinary feedforward network are also known as first-order neurons. First order neural networks models use a summation function ‘Σ’ which performs a linear weighted sum of its inputs for decision. The inputs of neuron in the first order neural network are represented as in Figure 2.5.
The network is linear in the sense that they can capture only first-order correlations as they provide a linear summation of inputs (Giles and Maxwell, 1987). The non-linearity only occurs by means of application of a nonlinear (activation) function to the weighted sum of inputs in the neuron (Hoogerheide, 2006). They can be represented by:

\[ y_i(x) = \sigma \left[ \sum_{j} w_{ij} x_j \right] \]  

(2.2)

where \( x = \{ x_j \} \) is an input vector, \( w_{ij} \) is a weight parameters, \( N \) is the number of elements in the input vector \( x \), and \( \sigma \) is a sigmoid function (activation function).

HONNs on the other hand, make use of multiplicative function or product term ‘\( \prod \)’ aside from the utilization of summation function ‘\( \Sigma \)’. According to Schmitt (2002), the rationale of applying the multiplication operation in neural networks is to help to increase the computational power of the network. Schmitt (2002) also conveyed an empirical evidence on the existence of exponential and logarithmic dendritic processes in biological neural systems, in which support the application of multiplication and polynomial processing in artificial neural systems. Likewise, Durbin and Rumelhart (1990) also discussed on extending the standard MLP model with multiplicative or product units to increase the computational power in order to be able to work the same way as biological neural networks.

The HONNs’ architecture can be classified into three different groups based on how the product units are utilized in the network. One way is for them to be made available as inputs to the network in addition to the original raw inputs as in Figure 2.6(a). Alternatively, the product units can be used as the final output neurons of the network itself as in Figure 2.6(b). Otherwise, the product units can also be utilized as...
REFERENCES


Samanta, S. and S. Chakraborty (2011). "Parametric optimization of some non-


Shayeghi, H., H. Shayanfar and A. Ghasemi (2011). Artificial bee colony based power system stabilizer design for a turbo-generator in a single-machine power system. *2011 world congress in computer science computer engineering and applied computing (ICAI’11).*


Neural Network. Computer Engineering and Applications (ICCEA), 2010
Second International Conference on.


